А.А.Емельянов
Е.А.Власова
Р.В.Дума

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ЭКОНОМИЧЕСКИХ ПРОЦЕССОВ

Под редакцией
доктора экономических наук
А.А. Емельянова

Рекомендовано
Учебно-методическим объединением вузов РФ
по образованию в области прикладной информатики
в качестве учебного пособия для студентов,
обучающихся по специальности
"Прикладная информатика (по областям)",
а также по другим компьютерным специальностям
и направлениям

МОСКВА
"ФИНАНСЫ И СТАТИСТИКА"
2002

УДК 330.45:004.942(075.8)
ББК 65в6я73
Е60

РЕЦЕНЗЕНТЫ:

кафедра «Информационные системы в экономике»
Уральского государственного экономического университета
(заведующий кафедрой А.Ф. Шориков,
доктор физико-математических наук, профессор);

В.Н. Волкова,
доктор экономических наук,
профессор Санкт-Петербургского государственного
технического университета,
академик Международной академии наук высшей школы

Емельянов А.А. и др.

Е60 Имитационное моделирование экономических процессов: Учеб. пособие / А.А. Емельянов, Е.А. Власова, Р.В. Дума; Под ред. А.А. Емельянова. – М.: Финансы и статистика, 2002. – 368 с.: ил.

ISBN 5-279-02572-0

Представлены современные концепции построения моделирующей системы, формализованные объекты типа материальных, информационных и денежных ресурсов, а также языковые средства создания имитационных моделей, техника их создания, отладки и эксплуатации с использованием CASE-технологии конструирования моделей «без программирования». Показаны особенности моделирования в геопространстве — с привязкой к картам или планам. Описано планирование экстремальных экспериментов.

Для студентов вузов, обучающихся по специальностям «Прикладная информатика (по областям)», «Математическое обеспечение и администрирование информационных систем», а также для других компьютерных специальностей и направлений высшего профессионального образования

Е $\frac{1402020000 - 106}{010(01) - 2002}$ 276 – 2002

УДК 330.45:004.942(075.8)
ББК 65в6я73

ISBN 5-279-02572-0

ПРЕДИСЛОВИЕ

После издания книги Т. Нейлора «Машинные имитационные эксперименты с моделями экономических систем» на русском языке прошло более 25 лет. С тех пор методы имитационного моделирования экономических процессов претерпели существенные изменения. Их применение в экономической деятельности стало иным. Отдельные книги, изданные в последние годы (например, о применении GPSS в технике и технологиях, об алгоритмическом моделировании элементов экономических систем на Visual Basic), повторяют концепции имитационного моделирования 30-летней давности с применением новых программных средств, но не отражают произошедших изменений.

Цель данной книги – всестороннее освещение подходов и способов применения имитационного моделирования в проектной экономической деятельности, появившихся в последние годы, и новых инструментальных средств, предоставляющих экономистам самые различные возможности.

Учебное пособие начинается с описания теоретических основ имитационного моделирования. Далее рассмотрена одна из современных концепций построения моделирующей системы. Приведены языковые средства описания моделей. Описана техника создания, отладки и эксплуатации моделей с использованием CASE-технологии конструирования моделей «без программирования» – с помощью диалогового графического конструктора. Имеется специальная глава, посвященная имитационному моделированию в геопространстве с привязкой к территориям экономических регионов. Рассмотрены вопросы планирования оптимизационных экспериментов – нахождения рациональных параметров процессов с помощью имитационных моделей. Последняя глава содержит набор отлаженных имитационных моделей различного назначения, которые могут быть хорошим подспорьем для различных категорий читателей. Преподавателям они помогут разработать лабораторные работы и задания. Студентам вузов, а также аспирантам и специалистам, самостоятельно изучающим этот вид компьютерного моделирования, они

позволят быстрее перейти к практическому моделированию в своей предметной области.

В конце каждой главы приведены краткие выводы и перечень контрольных вопросов для самопроверки. Краткий словарь терминов и предметный указатель также облегчают усвоение материала книги.

Учебное пособие написано с использованием опыта работы, накопленного авторами в процессе преподавания учебных дисциплин, связанных с имитационным моделированием, управлением рисками, исследованием систем управления, при подготовке и издании в вузах учебных пособий и учебно-методических материалов. В книге нашли отражение результаты авторских научных исследований и разработок.

Труд авторов распределился следующим образом:

А.А. Емельянов, доктор экономических наук, заведующий кафедрой «Общая теория систем и системного анализа» МЭСИ – главы 1 – 3, 6, 7, 8 (разд. 8.1 – 8.3, 8.6, 8.7) и общее редактирование книги.

Е.А. Власова, старший преподаватель кафедры «Общая теория систем и системного анализа» МЭСИ – главы 4 и 8 (разд. 8.4 и 8.5).

Р.В. Дума, кандидат экономических наук, ведущий специалист фирмы «Бизнес-Консоль» – глава 5.

Учебное пособие может быть рекомендовано студентам, обучающимся по компьютерным специальностям и направлениям. Оно может быть полезным при подготовке специалистов-менеджеров и магистров по программам «Мастер делового администрирования» (MBA – Master of Business Administration).

Для самостоятельного изучения книги необходимо предварительное знакомство читателя с информатикой, с основами программирования, высшей математики, теории вероятностей, математической статистики, линейной алгебры, экономической теории и бухгалтерского учета.

Имитационное моделирование (от англ. simulation) – это распространенная разновидность аналогового моделирования, реализуемого с помощью набора математических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования, позволяющих посредством процессов-аналогов провести целенаправленное исследование структуры и функций реального сложного процесса в памяти компьютера в режиме «имитации», выполнить оптимизацию некоторых его параметров.

Имитационной моделью называется специальный программный комплекс, который позволяет имитировать деятельность какого-либо сложного объекта. Он запускает в компьютере параллельные взаимодействующие вычислительные процессы, которые являются по своим временным параметрам (с точностью до масштабов времени и пространства) аналогами исследуемых процессов. В странах, занимающих лидирующее положение в создании новых компьютерных систем и технологий, научное направление Computer Science использует именно такую трактовку имитационного моделирования, а в программах магистерской подготовки по данному направлению имеется соответствующая учебная дисциплина.

Следует отметить, что любое моделирование имеет в своей методологической основе элементы имитации реальности с помощью какой-либо символики (математики) или аналогов. Поэтому иногда в российских вузах имитационным моделированием стали называть целенаправленные серии многовариантных расчетов, выполняемых на компьютере с применением экономико-математических моделей и методов. Однако с точки зрения компьютерных технологий такое моделирование (modelling) – это обычные вычисления, выполняемые с помощью расчетных программ или табличного процессора Excel.

Математические расчеты (в том числе табличные) можно производить и без ЭВМ: используя калькулятор, логарифмическую линейку, правила арифметических действий и вспомогательные таблицы. Но имитационное моделирование – это чисто компьютерная работа, которую невозможно выполнить подручными средствами.

Поэтому часто для этого вида моделирования используется синоним *компьютерное моделирование*.

Имитационную модель нужно создавать. Для этого необходимо специальное программное обеспечение – *система моделирования* (simulation system). Специфика такой системы определяется технологией работы, набором языковых средств, сервисных программ и приемов моделирования.

Имитационная модель должна отражать большое число параметров, логику и закономерности поведения моделируемого объекта во времени (*временная динамика*) и в пространстве (*пространственная динамика*). Моделирование объектов экономики связано с понятием *финансовой динамики* объекта.

С точки зрения специалиста (информатика-экономиста, математика-программиста или экономиста-математика), *имитационное моделирование* контролируемого процесса или управляемого объекта – это высокоуровневая информационная технология, которая обеспечивает два вида действий, выполняемых с помощью компьютера:

1) работы по созданию или модификации имитационной модели;

2) эксплуатацию имитационной модели и интерпретацию результатов.

Имитационное (компьютерное) моделирование экономических процессов обычно применяется в двух случаях:

- для управления сложным бизнес-процессом, когда имитационная модель управляемого экономического объекта используется в качестве инструментального средства в контуре адаптивной системы управления, создаваемой на основе информационных (компьютерных) технологий;
- при проведении экспериментов с дискретно-непрерывными моделями сложных экономических объектов для получения и отслеживания их динамики в экстренных ситуациях, связанных с рисками, натурное моделирование которых нежелательно или невозможно.

Можно выделить следующие типовые задачи, решаемые средствами имитационного моделирования при управлении экономическими объектами:

- моделирование процессов логистики для определения временных и стоимостных параметров;
- управление процессом реализации инвестиционного проекта на различных этапах его жизненного цикла с учетом возможных рисков и тактики выделения денежных сумм;

• анализ клиринговых процессов в работе сети кредитных организаций (в том числе применение к процессам взаимозачетов в условиях российской банковской системы);

• прогнозирование финансовых результатов деятельности предприятия на конкретный период времени (с анализом динамики сальдо на счетах);

• бизнес-реинжиниринг несостоятельного предприятия (изменение структуры и ресурсов предприятия-банкрота, после чего с помощью имитационной модели можно сделать прогноз основных финансовых результатов и дать рекомендации о целесообразности того или иного варианта реконструкции, инвестиций или кредитования производственной деятельности);

• анализ адаптивных свойств и живучести компьютерной региональной банковской информационной системы (например, частично вышедшая из строя в результате природной катастрофы система электронных расчетов и платежей после катастрофического землетрясения 1995 г. на центральных островах Японии продемонстрировала высокую живучесть: операции возобновились через несколько дней);

• оценка параметров надежности и задержек в централизованной экономической информационной системе с коллективным доступом (на примере системы продажи авиабилетов с учетом несовершенства физической организации баз данных и отказов оборудования);

• анализ эксплуатационных параметров распределенной многоуровневой ведомственной информационной управляющей системы с учетом неоднородной структуры, пропускной способности каналов связи и несовершенства физической организации распределенной базы данных в региональных центрах;

• моделирование действий курьерской (фельдъегерьской) вертолетной группы в регионе, пострадавшем в результате природной катастрофы или крупной промышленной аварии;

• анализ сетевой модели PERT (Program Evaluation and Review Technique) для проектов замены и наладки производственного оборудования с учетом возникновения неисправностей;

• анализ работы автотранспортного предприятия, занимающегося коммерческими перевозками грузов, с учетом специфики товарных и денежных потоков в регионе;

• расчет параметров надежности и задержек обработки информации в банковской информационной системе.

Приведенный перечень является неполным и охватывает те примеры использования имитационных моделей, которые описаны в литературе или применялись авторами на практике. Действительная область применения аппарата имитационного моделирования не имеет видимых ограничений. Например, спасение американских астронавтов при возникновении аварийной ситуации на корабле APOLLO стало возможным только благодаря «проигрыванию» различных вариантов спасения на моделях космического комплекса.

Система имитационного моделирования, обеспечивающая создание моделей для решения перечисленных задач, должна обладать следующими свойствами:

- возможностью применения имитационных программ совместно со специальными экономико-математическими моделями и методами, основанными на теории управления;
- инструментальными методами проведения структурного анализа сложного экономического процесса;
- способностью моделирования материальных, денежных и информационных процессов и потоков в рамках единой модели, в общем модельном времени;
- возможностью введения режима постоянного уточнения при получении выходных данных (основных финансовых показателей, временных и пространственных характеристик, параметров рисков и др.) и проведении экстремального эксперимента.

Историческая справка. Имитационное моделирование экономических процессов – разновидность экономико-математического моделирования. Однако этот вид моделирования в значительной степени базируется на компьютерных технологиях. Многие моделирующие системы, идеологически разработанные в 1970–1980-х гг., претерпели эволюцию вместе с компьютерной техникой и операционными системами (например, GPSS – General Purpose Simulation System) и эффективно используются в настоящее время на новых компьютерных платформах. Кроме того, в конце 1990-х гг. появились принципиально новые моделирующие системы, концепции которых не могли возникнуть раньше – при использовании ЭВМ и операционных систем 1970–1980-х гг.

1. Период 1970–1980-х гг. Впервые методы имитационного моделирования для анализа экономических процессов применил Т. Нейлор. На протяжении двух десятилетий попытки использовать этот вид моделирования в реальном управлении экономическими

процессами носили эпизодический характер из-за сложности формализации экономических процессов:

• в математическом обеспечении ЭВМ не было формальной языковой поддержки описания элементарных процессов и их функций в узлах сложной стохастической сети экономических процессов с учетом их иерархической структуры;

• отсутствовали формализованные методы структурного системного анализа, необходимые для иерархического (многослойного) разложения реального моделируемого процесса на элементарные составляющие в модели.

Алгоритмические методы, предлагаемые в течение этих лет для имитационного моделирования, использовались эпизодически по следующим причинам:

• они были трудоемки для создания моделей сложных процессов (требовались весьма существенные затраты на программирование);

• при моделировании простых составляющих процессов они уступали математическим решениям в аналитической форме, получаемым методами теории массового обслуживания. Аналитические модели существенно проще реализовывались в виде компьютерных программ.

Алгоритмический подход и сейчас используется в некоторых вузах для изучения основ моделирования элементов экономических систем.

Сложность реальных экономических процессов и обилие противоречивых условий существования этих процессов (от сотен до тысяч) приводят к следующему результату. Если воспользоваться алгоритмическим подходом при создании имитационной модели с использованием обычных языков программирования (Бейсик, Фортран и др.), то сложность и объем моделирующих программ будут очень велики, а логика модели слишком запутана. Для создания такой имитационной модели требуется значительный период времени (иногда – многие годы). Поэтому имитационное моделирование в основном применялось только в научной деятельности.

Однако в середине 1970-х гг. появились первые достаточно технологичные инструментальные средства имитационного моделирования, обладающие собственными языковыми средствами. Самое мощное из них – система GPSS. Она позволяла создавать модели контролируемых процессов и объектов в основном технического или технологического назначения.

2. Период 1980–1990-х гг. Системы имитационного моделирования более активно стали использоваться в 80-е гг., когда в разных странах применялось более 20 различных систем. Наиболее распространенными были системы GASP-IV, SIMULA-67, GPSS-V и SLAM-II, которые, однако, имели много недостатков.

Система GASP-IV предоставляла пользователю структурированный язык программирования, похожий на Фортран, набор методов событийного моделирования дискретных подсистем модели и моделирования непрерывных подсистем с помощью уравнений переменных состояния, а также датчики псевдослучайных чисел.

Система SIMULA-67 по своим возможностям подобна GASP-IV, но предоставляет пользователю язык структурного программирования, похожий на Алгол-60.

Эффективность моделей, создаваемых с помощью систем GASP-IV и SIMULA-67, в большой степени зависела от искусства разработчика модели. Например, забота о выделении независимых моделируемых процессов полностью возлагалась на разработчика – специалиста, имеющего высокую математическую подготовку. Поэтому данная система в основном использовалась только в научных организациях.

В системах GASP-IV и SIMULA-67 не было средств, пригодных для имитации пространственной динамики моделируемого процесса.

Система GPSS-V предоставила пользователю законченную высокоуровневую информационную технологию создания имитационных моделей. В этой системе имеются средства формализованного описания параллельных дискретных процессов в виде условных графических изображений или с помощью операторов собственного языка. Координация процессов осуществляется автоматически в едином модельном времени. Пользователь в случае необходимости может ввести свои правила синхронизации событий. Имеются средства управления моделью, динамической отладки и автоматизации обработки результатов. Однако эта система имела три основных недостатка:

- разработчик не мог включать непрерывные динамические компоненты в модель, даже используя свои внешние подпрограммы, написанные на PL/1, Фортран или языке Ассемблера;
- отсутствовали средства имитации пространственных процессов;
- система была чисто интерпретирующей, что существенно снижало быстродействие моделей.

Наиболее развитой из указанных систем является SLAM-II, позволяющая создавать сложные модели дискретно-непрерывных процессов. Методология, заложенная в систему SLAM-II, резко расширила область применения имитационного моделирования. Однако и эта система имеет некоторые недостатки: она сложнее GPSS-V в освоении неподготовленным пользователем, в ней нет собственных средств имитации пространственных процессов.

3. Период 1990–2000-х гг. В поколении систем имитационного моделирования 90-х гг. можно выделить следующие распространенные пакеты:

- Process Charter–1.0.2 (компания «Scitor», Менло-Парк, Калифорния, США);
- Powersim–2.01 (компания «Modell Data» AS, Берген, Норвегия);
- Ithink–3.0.61 (компания «High Performance Systems», Ганновер, Нью-Хэмпшир, США);
- Extend+BPR–3.1 (компания «Imagine That!», Сан-Хосе, Калифорния, США);
- ReThink (фирма «Gensym», Кембридж, Массачусетс, США);
- Pilgrim (Россия).

Пакет *Process Charter–1.0.2* имеет «интеллектуальное» средство построения блок-схем моделей. Он ориентирован в основном на дискретное моделирование. Имеет достоинства: удобный и простой в использовании механизм построения модели, самый дешевый из перечисленных продуктов, хорошо приспособлен для решения задач распределения ресурсов. Недостатки пакета: наименее мощный продукт, слабая поддержка моделирования непрерывных компонентов, ограниченный набор средств для анализа чувствительности и построения диаграмм.

Пакет *Powersim–2.01* является хорошим средством создания непрерывных моделей. Имеет достоинства: множество встроенных функций, облегчающих построение моделей, многопользовательский режим для коллективной работы с моделью, средства обработки массивов для упрощения создания моделей со сходными компонентами. Недостатки пакета: сложная специальная система обозначений System Dynamics, ограниченная поддержка дискретного моделирования.

Пакет *Ithink–3.0.61* обеспечивает создание непрерывных и дискретных моделей. Имеет достоинства: встроенные блоки для облегчения создания различных видов моделей, поддержка авторского

моделирования слабо подготовленными пользователями, подробная обучающая программа, развитые средства анализа чувствительности, поддержка множества форматов входных данных. Недостатки пакета: сложная система обозначений System Dynamics, поддержка малого числа функций по сравнению с Powersim–2.01.

Пакет *Extend+BPR–3.1* (BPR – Business Process Reengineering) создан как средство анализа бизнес-процессов, использовался в NASA, поддерживает дискретное и непрерывное моделирование. Имеет достоинства: интуитивно понятная среда построения моделей с помощью блоков, множество встроенных блоков и функций для облегчения создания моделей, поддержка сторонними компаниями (особенно выпускающими приложения для «вертикальных» рынков), гибкие средства анализа чувствительности, средства создания дополнительных функций с помощью встроенного языка. Недостатки пакета: используется в полном объеме только на компьютерах типа Macintosh, имеет высокую стоимость.

Пакет *ReThink* обладает свойствами Extend+BPR–3.1 и в отличие от перечисленных пакетов имеет хороший графический транслятор для создания моделей. Работает под управлением экспертной оболочки G2. Имеет достоинства: все положительные свойства Extend+BPR–3.1 и общее поле данных с экспертной системой реального времени, создаваемой средствами G2.

Пакет *Pilgrim* обладает широким спектром возможностей имитации временной, пространственной и финансовой динамики моделируемых объектов. С его помощью можно создавать дискретно-непрерывные модели. Разрабатываемые модели имеют свойство коллективного управления процессом моделирования. В текст модели можно вставлять любые блоки с помощью стандартного языка C++. Различные версии этой системы работали на IBM-совместимых и DEC-совместимых компьютерах, оснащенных Unix или Windows. Пакет Pilgrim обладает свойством мобильности, т.е. переноса на любую другую платформу при наличии компилятора C++. Модели в системе Pilgrim компилируются и поэтому имеют высокое быстродействие, что очень важно для отработки управленческих решений и адаптивного выбора вариантов в сверхускоренном масштабе времени. Полученный после компиляции объектный код можно встраивать в разрабатываемые программные комплексы или передавать (продавать) заказчику, так как при эксплуатации моделей инструментальные средства пакета Pilgrim не используются. Система имеет сравнительно невысокую стоимость.

Перечисленные выше инструментальные средства имеют общее свойство: возможность графического конструирования модели. В процессе такой инженерной работы удается связать в графическом представлении на одной графической схеме моделируемые процессы с управленческими (административными) или конструктивными особенностями моделируемой системы.

В конце 1990-х гг. в России разработаны новые системы:

- пакет РДО (МГТУ им. Н.Э. Баумана);
- система СИМПАС (МГТУ им. Н.Э. Баумана);
- пятая версия Pilgrim (МЭСИ и несколько компьютерных фирм).

Пакет РДО (РДО – Ресурсы-Действия-Операции) является мощной системой имитационного моделирования для создания продукционных моделей. Обладает развитыми средствами компьютерной графики (вплоть до анимации). Применяется при моделировании сложных технологий и производств.

Система СИМПАС (СИМПАС – СИстема-Моделирования-на-ПАСкале) в качестве основного инструментального средства использует язык программирования Паскаль. Недостаток, связанный со сложностью моделирования на языке общего назначения, компенсируется специальными процедурами и функциями, введенными разработчиками этой системы. Проблемная ориентация системы – это моделирование информационных процессов, компьютеров сложной архитектуры и компьютерных сетей.

Пятая версия Pilgrim – это новый программный продукт, созданный в 2000 г. на объектно-ориентированной основе и учитывающий основные положительные свойства прежних версий. Достоинства этой системы:

- ориентация на совместное моделирование материальных, информационных и «денежных» процессов;
- наличие развитой CASE-оболочки, позволяющей конструировать многоуровневые модели в режиме структурного системного анализа;
- наличие интерфейсов с базами данных;
- возможность для конечного пользователя моделей непосредственно анализировать результаты благодаря формализованной технологии создания функциональных окон наблюдения за моделью с помощью Visual C++, Delphy или других средств;
- возможность управления моделями непосредственно в процессе их выполнения с помощью специальных окон диалога.

В данной книге при рассмотрении практических задач и примеров имитационного моделирования используются концепция, возможности и функциональные средства системы Pilgrim. Это объясняется тем, что в новом поколении Государственных образовательных стандартов высшего профессионального образования, введенном в России с 2000 г., идеология именно этой системы заложена в дидактическое содержание двух компьютерных дисциплин:

- «Имитационное моделирование экономических процессов» – специальность 351400 «Прикладная информатика (по областям)» для областей «экономика» и «менеджмент»;
- «Компьютерное моделирование» – специальность 351500 «Математическое обеспечение и администрирование информационных систем».

Однако читатели, знакомые с имитационным моделированием и применением других пакетов моделирующих программ (например, GPSS), могут убедиться в том, что подходы и методические приемы, используемые в данном пособии, могут быть реализованы с помощью разных моделирующих систем.

Глава 1

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

1.1 ОСНОВНЫЕ ПОНЯТИЯ. РАЗНОВИДНОСТИ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Имитационное моделирование реализуется посредством набора математических инструментальных средств, специальных компьютерных программ и приемов, позволяющих с помощью компьютера провести целенаправленное моделирование в режиме «имитации» структуры и функций сложного процесса и оптимизацию некоторых его параметров. Набор программных средств и приемов моделирования определяет специфику системы моделирования – специального программного обеспечения.

В отличие от других видов и способов математического моделирования с применением ЭВМ имитационное моделирование имеет свою специфику: запуск в компьютере взаимодействующих вычислительных процессов, которые являются по своим временным параметрам – с точностью до масштабов времени и пространства – аналогами исследуемых процессов.

Имитационное моделирование как особая информационная технология состоит из следующих основных этапов.

1. *Структурный анализ процессов.* Проводится формализация структуры сложного реального процесса путем разложения его на подпроцессы, выполняющие определенные функции и имеющие взаимные функциональные связи согласно легенде, разработанной рабочей экспертной группой. Выявленные подпроцессы, в свою очередь, могут разделяться на другие функциональные подпроцессы. Структура общего моделируемого процесса может быть представлена в виде графа, имеющего иерархическую многослойную

структуру. В результате появляется формализованное изображение имитационной модели в графическом виде.

Структурный анализ особенно эффективен при моделировании экономических процессов, где (в отличие от технических) многие составляющие подпроцессы не имеют физической основы и протекают виртуально, поскольку оперируют с информацией, деньгами и логикой (законами) их обработки.

2. *Формализованное описание модели.* Графическое изображение имитационной модели, функции, выполняемые каждым подпроцессом, условия взаимодействия всех подпроцессов и особенности поведения моделируемого процесса (временная, пространственная и финансовая динамика) должны быть описаны на специальном языке для последующей трансляции. Для этого существуют различные способы:

- описание вручную на языке типа GPSS, Pilgrim и даже на Visual Basic. Последний очень прост, на нем можно запрограммировать элементарные модели, но он не подходит для разработки реальных моделей сложных экономических процессов, так как описание модели средствами Pilgrim компактнее аналогичной алгоритмической модели на Visual Basic в десятки-сотни раз;
- автоматизированное описание с помощью компьютерного графического конструктора во время проведения структурного анализа, т.е. с очень незначительными затратами на программирование. Такой конструктор, создающий описание модели, имеется в составе системы моделирования в Pilgrim.

3. *Построение модели* (build). Обычно это трансляция и редактирование связей (сборка модели), верификация (калибровка) параметров.

Трансляция осуществляется в различных режимах:

- в режиме интерпретации, характерном для систем типа GPSS, SLAM-II и ReThink;
- в режиме компиляции (характерен для системы Pilgrim).

Каждый режим имеет свои особенности.

Режим *интерпретации* проще в реализации. Специальная универсальная программа-интерпретатор на основании формализованного описания модели запускает все имитирующие подпрограммы. Данный режим не приводит к получению отдельной моделирующей программы, которую можно было бы передать или продать заказчику (продавать пришлось бы и модель, и систему моделирования, что не всегда возможно).

Режим *компиляции* сложнее реализуется при создании моделирующей системы. Однако это не усложняет процесс разработки модели. В результате можно получить отдельную моделирующую программу, которая работает независимо от системы моделирования в виде отдельного программного продукта.

Верификация (калибровка) параметров модели выполняется в соответствии с легендой, на основании которой построена модель, с помощью специально выбранных тестовых примеров.

4. *Проведение экстремального эксперимента* для оптимизации определенных параметров реального процесса. Планированию таких экспериментов посвящена глава 7.

Примеры, приводимые в данной книге, в основном ориентированы на систему Pilgrim, получившую распространение в экономических вузах России. Однако читатели, владеющие GPSS, SLAM-II или ReThink, без особого труда увидят общие методические приемы, не зависящие от выбранной системы.

Концепция имитационного моделирования требует предварительного знакомства читателя с методом Монте-Карло, с методологией проведения проверок статистических гипотез, с устройством программных датчиков случайных (псевдослучайных) величин и с особенностями законов распределения случайных величин при моделировании экономических процессов, которые не рассматриваются в типовых программах дисциплины «Теория вероятностей».

Кроме того, необходимо рассмотреть специальные стохастические сетевые модели, которые дают представление о временных диаграммах специальных имитационных процессов при выполнении программной модели.

1.2 МЕТОД МОНТЕ-КАРЛО И ПРОВЕРКА СТАТИСТИЧЕСКИХ ГИПОТЕЗ

Статистические испытания по методу Монте-Карло представляют собой простейшее имитационное моделирование при полном отсутствии каких-либо правил поведения. Получение выборок по методу Монте-Карло – основной принцип компьютерного моделирования систем, содержащих стохастические или вероятностные элементы. Зарождение метода связано с работой фон Неймана и Улана в конце 1940-х гг., когда они ввели для него название «Монте-

Карло» и применили его к решению некоторых задач экранирования ядерных излучений. Этот математический метод был известен и ранее, но свое второе рождение нашел в Лос-Аламосе в закрытых работах по ядерной технике, которые велись под кодовым обозначением «Монте-Карло». Применение метода оказалось настолько успешным, что он получил распространение и в других областях, в частности в экономике.

Поэтому многим специалистам термин «метод Монте-Карло» иногда представляется синонимом термина «имитационное моделирование», что в общем случае неверно. Имитационное моделирование – это более широкое понятие, и метод Монте-Карло является важным, но далеко не единственным методическим компонентом имитационного моделирования.

Согласно методу Монте-Карло проектировщик может моделировать работу тысячи сложных систем, управляющих тысячами разновидностей подобных процессов, и исследовать поведение всей группы, обрабатывая статистические данные. Другой способ применения этого метода заключается в том, чтобы моделировать поведение системы управления на очень большом промежутке модельного времени (несколько лет), причем астрономическое время выполнения моделирующей программы на компьютере может составить доли секунды. Рассмотрим метод Монте-Карло подробнее.

В различных задачах, встречающихся при создании сложных систем, могут использоваться величины, значения которых определяются случайным образом. Примерами таких величин являются:

- случайные моменты времени, в которые поступают заказы на фирму;
- загрузка производственных участков или служб объекта экономики;
- внешние воздействия (требования или изменения законов, платежи по штрафам и др.);
- оплата банковских кредитов;
- поступление средств от заказчиков;
- ошибки измерений.

В качестве соответствующих им переменных могут использоваться число, совокупность чисел, вектор или функция. Одной из разновидностей метода Монте-Карло при численном решении задач, включающих случайные переменные, является метод статистических испытаний, который заключается в моделировании случайных событий.

Метод Монте-Карло основан на статистических испытаниях и по природе своей является экстремальным, может применяться для решения полностью детерминированных задач, таких, как обращение матриц, решение дифференциальных уравнений в частных производных, отыскание экстремумов и численное интегрирование. При вычислениях методом Монте-Карло статистические результаты получаются путем повторяющихся испытаний. Вероятность того, что эти результаты отличаются от истинных не более чем на заданную величину, есть функция количества испытаний.

В основе вычислений по методу Монте-Карло лежит случайный выбор чисел из заданного вероятностного распределения. При практических вычислениях эти числа берут из таблиц или получают путем некоторых операций, результатами которых являются псевдослучайные числа с теми же свойствами, что и числа, получаемые путем случайной выборки. Имеется большое число вычислительных алгоритмов, которые позволяют получить длинные последовательности псевдослучайных чисел.

Один из наиболее простых и эффективных вычислительных методов получения последовательности равномерно распределенных случайных чисел r_i с помощью, например, калькулятора или любого другого устройства, работающего в десятичной системе счисления, включает только одну операцию умножения.

Метод заключается в следующем: если $r_1 = 0{,}0040353607$, то $r_{i+1} = \{40353607 r_i\} \bmod 1$, где mod 1 означает операцию извлечения из результата только дробной части после десятичной точки. Как описано в различных литературных источниках, числа r_i начинают повторяться после цикла из 50 миллионов чисел, так что $r_{50000001} = r_1$. Последовательность r_i получается равномерно распределенной на интервале (0, 1). Ниже будут рассмотрены более точные способы получения таких чисел со значительно большими периодами, а также пояснения, как в реальных моделях такие числа становятся практически случайными.

Применение метода Монте-Карло может дать существенный эффект при моделировании развития процессов, натурное наблюдение которых нежелательно или невозможно, а другие математические методы применительно к этим процессам либо не разработаны, либо неприемлемы из-за многочисленных оговорок и допущений, которые могут привести к серьезным погрешностям или неправильным выводам. В связи с этим необходимо не только наблюдать развитие

процесса в нежелательных направлениях, но и оценивать гипотезы о параметрах нежелательных ситуаций, к которым приведет такое развитие, в том числе и параметрах рисков.

Существуют различные методы проверки статистических гипотез. Наиболее широко используются на практике критерии:

- согласия χ^2 (хи-квадрат);
- Крамера–фон Мизеса;
- Колмогорова–Смирнова.

Критерий χ^2 предпочтителен, если объемы выборок N, в отношении которых проводится анализ, велики. Это мощное средство, если $N > 100$ значений. Однако при анализе экономических ситуаций иногда бывает довольно трудно (или невозможно) найти 100 одинаковых процессов, развивающихся с различными исходными данными. Сложность заключается не только в том, что не бывает одинаковых объектов экономики: даже если такие объекты имеются, то к исходным данным относятся не только исходные вероятностные данные и особенности структуры объекта, но и сценарий развития процессов в этом объекте и в тех объектах внешней среды, с которыми он взаимодействует (процессы рынка, указы правительства, принятие новых законов, требования налоговых органов, платежи в бюджеты различных уровней). При относительно малых объемах выборок этот критерий вообще неприменим.

Критерий Крамера–фон Мизеса дает хорошие результаты при малых объемах выборок (при $N < 10$). Однако следует отметить два обстоятельства:

1) при $N < 10$, каким бы методом ни пользоваться, вопрос о доверительной вероятности при проверке статистической гипотезы решается плохо (эта вероятность мала при значительных размерах доверительных интервалов);

2) метод Монте-Карло используется как раз для того, чтобы недостающие данные собрать с помощью специального вычислительного статистического инструментария и компьютера.

Поэтому будем полагать, что реальные объемы выборок, которые можно получить, находятся в пределах $10 \le N < 100$. Как указывают многие исследователи, для указанных пределов хорошие результаты дает критерий Колмогорова–Смирнова. Он применяется в тех случаях, когда проверяемое распределение непрерывно и известны среднее значение и дисперсия проверяемой совокупности. Рассмотрим подробнее методику использования этого критерия на конкретном примере.

Пример 1.1. Предположим, что нужно проверить данные, полученные (или наблюдаемые) при использовании метода Монте-Карло и приведенные в табл. 1.1, на их соответствие распределению Пуассона.

Таблица 1.1

Данные для проверки гипотезы по критерию Колмогорова–Смирнова

Число событий	Наблюдаемая частота	Наблюдаемая вероятность	Теоретическая вероятность	Наблюдаемое распределение	Теоретическое распределение	Абсолютная разность
	1	2	3	4	5	6
0	315	0,619	0,571	0,619	0,571	0,048
1	142	0,279	0,319	0,898	0,890	0,008
2	40	0,078	0,089	0,976	0,979	0,003
3	9	0,018	0,017	0,994	0,996	0,002
4	2	0,004	0,003	0,998	0,999	0,001
5	1	0,002	0,001	1,000	1,000	0,000

Эти данные имеют следующий смысл. На отрезке времени наблюдаем случайные события, число которых равно x. Если это распределение Пуассона, то вероятность $P\{x = n\}$ того, что $x = n$, где n – заданное число, равна

$$P\{x = n\} = \frac{\lambda^n e^{-\lambda}}{n!},$$

где $e = 2{,}71828$;

λ – положительная константа, которая одновременно является и математическим ожиданием, и дисперсией.

Предположим, что $\lambda = 0{,}5577$. Сформулируем гипотезу в следующем виде: Не имеется существенных различий между наблюдаемыми данными во время эксперимента и теми данными, которые должны получаться из распределения Пуассона расчетным путем со средним значением 0,5577 и $N = 509$.

Рассмотрим подробнее табл. 1.1. В ней строки с номерами 0, ... , 5 соответствуют числу n наблюдаемых событий, n=0, ... , 5, столбец 1 содержит наблюдаемую частоту появления n событий, а столбец 2 – наблюдаемую вероятность появления n событий. В столбце 3 представлены рассчитанные по формуле значения вероятности появления n событий.

Прежде всего нужно получить два интегральных распределения (фактически приближения функций распределения). Сначала сделаем это для наблюдаемых данных: с помощью столбца 2 получим столбец 4. Затем – для теоретических данных: с помощью столбца 3 получим столбец 5.

После этих вычислений найдем абсолютные разности для всех групп значений случайной величины и с помощью столбцов 4 и 5 получим столбец 6. В последнем столбце наибольшая абсолютная разность 0,048 получается в группе, соответствующей нулевому числу событий.

Далее необходимо найти так называемое критическое значение D_{extr} для проверки принятой гипотезы. Таблица критических чисел многократно переиздавалась. Критические числа в виде, удобном для выполняемой проверки, приведены в табл. 1.2. Абсолютную разность 0,048 необходимо сравнить с критическим значением, найденным по табл. 1.2.

Таблица 1.2

Критические числа Колмогорова–Смирнова

Степень свободы	Проверка единичной выборки *			Проверка двух выборок **	
N	$D_{0,10}$	$D_{0,05}$	$D_{0,01}$	$D_{0,05}$	$D_{0,01}$
1	0,950	0,975	0,995	–	–
2	0,776	0,842	0,929	–	–
3	0,642	0,708	0,828	–	–
4	0,564	0,624	0,733	1,000	1,000
5	0,510	0,565	0,669	1,000	1,000
6	0,470	0,521	0,618	0,833	1,000
7	0,438	0,486	0,577	0,857	0,857
8	0,411	0,457	0,543	0,750	0,875
9	0,388	0,432	0,514	0,668	0,778
10	0,368	0,410	0,490	0,700	0,800
11	0,352	0,391	0,468	0,636	0,727
12	0,338	0,375	0,450	0,583	0,667
13	0,325	0,361	0,433	0,538	0,692
14	0,314	0,349	0,418	0,571	0,643

Продолжение

Степень свободы	Проверка единичной выборки *			Проверка двух выборок **	
N	$D_{0,10}$	$D_{0,05}$	$D_{0,01}$	$D_{0,05}$	$D_{0,01}$
15	0,304	0,338	0,404	0,533	0,600
16	0,295	0,328	0,392	0,500	0,625
17	0,286	0,318	0,381	0,471	0,588
18	0,278	0,309	0,371	0,500	0,556
19	0,272	0,301	0,363	0,474	0,526
20	0,264	0,294	0,356	0,450	0,550
25	0,240	0,270	0,320	0,400	0,480
30	0,220	0,240	0,290	0,370	0,430
35	0,210	0,230	0,270	0,340	0,390
Более 35	$\frac{1,22}{\sqrt{N}}$	$\frac{1,36}{\sqrt{N}}$	$\frac{1,63}{\sqrt{N}}$	$1,36\sqrt{\frac{N_1+N_2}{N_1N_2}}$	$1,63\sqrt{\frac{N_1+N_2}{N_1N_2}}$

* Применяется для оценки степени близости выборочных значений к теоретическому распределению.

N – объем выборки.

** Применяется для определения принадлежности двух выборок объемами N_1 и N_2 одному и тому же распределению. При малых размерах выборки $N = N_1 = N_2$.

При n=509 и значении индекса критического числа D_α $\alpha = 0,05$ получается критическое значение

$$D_{extr} = \frac{1,36}{\sqrt{N}} = \frac{1,36}{\sqrt{509}} = \frac{1,36}{22,56} = 0,0603.$$

Поскольку наибольшая разность $0,048 < D_{extr}$, то не отказываемся от гипотезы о том, что экспериментальное распределение – пуассоновское. Проверка статистических гипотез о соответствии «событий явлению» и «явлений поведению» дает математический инструмент для оценки «рискованного поведения» исследуемого процесса.

1.3 ИСПОЛЬЗОВАНИЕ ЗАКОНОВ РАСПРЕДЕЛЕНИЯ СЛУЧАЙНЫХ ВЕЛИЧИН ПРИ ИМИТАЦИИ ЭКОНОМИЧЕСКИХ ПРОЦЕССОВ

Рассмотрим приемы, которые автоматически выполняются в современных системах имитационного моделирования *. Однако если читатель захочет самостоятельно реализовать какой-либо прием, то он сможет это сделать, используя приводимые ниже тексты программных модулей.

Равномерное распределение на интервале (0, 1). В литературе приводились описания разных датчиков случайных величин для получения последовательностей чисел, распределенных по какому-то случайному закону. Основная проблема заключалась в программной реализации равномерного распределения на интервале (0, 1). Существуют различные методы получения такого равномерного распределения. Остановимся на программном генераторе, наиболее подходящем для компьютеров с 32-разрядным словом. Период последовательности, получаемой с помощью такого генератора, на несколько порядков превосходит период, получаемый по методу, рассмотренному в разд. 1.2.

Рассмотрим текст функции на языке C++, реализующей этот генератор и возвращающей величину, равномерно распределенную на интервале (0, 1). Данная функция реализует одну из разновидностей мультипликативного конгруентного метода. Генератор предназначен для применения в системе имитационного моделирования, позволяющей параллельное моделирование сложной сети взаимодействующих процессов, причем каждый процесс может иметь свой датчик псевдослучайных величин. Поэтому в качестве глобальной переменной рассматривается указатель k – адрес управляющей структуры такого процесса, имеющего номер next. Ниже следует текст программы:

---

* Все математические приемы и программные датчики псевдослучайных величин, рассмотренные в данном разделе, имеются в составе системы Pilgrim.

```
float rundum(void)
   {
   struct kcb *k;
   k  = addr[next];
   iy = k→bx * 1220703125;
   if (iy < 0)
      iy = iy + 1073741824 + 1073741824;
   rn = iy * 0.4656613e-9;
   k→bx = iy;
   return(rn);
   }
```

Внутри этой структуры расположено 32-разрядное поле bx, которое адресуется как k→bx. Перед моделированием управляющая программа должна занести в это поле длинное нечетное число, например k→bx=(long)2098765431. Следует отметить, что для каждого процесса нужно задать свое число. Этого можно достигнуть, например, применяя для каждого следующего инициируемого процесса начальное значение предыдущего, уменьшенное на 2. Если мы хотим резко повысить эффективность работы этой функции и еще сильнее приблизить псевдослучайную последовательность к случайной, то можно в качестве начального значения использовать битовую последовательность таймера ЭВМ: k→bx=(long)time(NULL). Далее это число проверяется на нечетность, и если оно четное, то его необходимо увеличить на 1.

Описанная процедура в основном применяется для получения более сложных распределений, как дискретных, так и непрерывных. Эти распределения получаются с помощью двух основных приемов:

- обратных функций;
- комбинирования величин, распределенных по другим законам, например по равномерному на интервале (0, 1).

Равномерное распределение на произвольном интервале. Рассмотрим важное и очень простое равномерное распределение на интервале $(m-s,\ m+s)$. Плотность вероятностей этого распределения описывается следующей формулой:

$$p(t)=\begin{cases} 0, & t<m-s; \\ \dfrac{1}{2s}, & m-s\le t\le m+s; \\ 0, & t>m+s, \end{cases}$$

где m – математическое ожидание;

s – максимальное отклонение от математического ожидания.

Такое распределение используется, если об интервалах времени известно только то, что они имеют максимальный разброс, и ничего не известно о распределениях вероятностей этих интервалов.

Программная функция на языке C++ приведена ниже (два ее входных параметра не нуждаются в комментариях):

```
float unifrm(float m, float s)
   {
   float a,b;
   a = m-s;
   b = m+s;
   return(a+(b-a)*rundum());
   }
```

Равномерное распределение можно использовать при расчетах по сетевым графикам работ, в том числе при работе по методу PERT. Это распределение можно применять и при расчетах основных длительностей и времен в военном деле (времени выдвижения воинской части или ее подразделения на исходный рубеж, времени марша, времени подготовки рубежа обороны и др.).

Формула для определения дисперсии получается после получения второго момента:

$$D[T]=\frac{[(m+s)-(m-s)]^2}{12}=\frac{s^2}{3}.$$

Дискретное распределение (общий случай). Предположим, что известны частоты α_i выбора из N объектов на определенном интервале времени, i=1, ..., N. Пример таких частот для N=7 представлен в табл. 1.3. Первая строка таблицы – это номер объекта, а вторая – частота его выбора. Требуется разработать программную функцию, которая должна возвращать значение номера объекта в соответствии с этими частотами.

Таблица 1.3

Значения обратных функций для получения дискретного распределения

i	1	2	3	4	5	6	7
α_i	130	5	11	44	32	2	67
β_i	0,447	0,017	0,038	0,151	0,110	0,007	0,230
γ_i	0,447	0,464	0,502	0,653	0,763	0,770	1,000

Воспользуемся методом обратных функций. Сначала найдем сумму всех частот:

$$\Lambda = \sum_{i=1}^{N} \alpha_i \,.$$

В нашем случае получаем Λ=291. После этого построим таблицу нормированных значений $\beta_i=\alpha_i/\Lambda$ (третья строка табл. 1.3). Далее рассчитаем значения дискретной функции γ_i по формуле

$$\gamma_i = \sum_{k=1}^{i} \beta_k \,.$$

Значения γ_i находятся в четвертой строке табл. 1.3.

Построим график дискретной функции γ_i (рис. 1.1). Далее воспользуемся рассмотренной выше программой получения случайных величин, распределенных равномерно на отрезке (0, 1), и каждый раз будем получать случайную величину p_t. Условимся, что $\gamma_0 = 0$. После этого выбор объекта с номером i осуществляется при выполнении соотношения

$$\gamma_{i-1} < p_t \le \gamma_i \,.$$

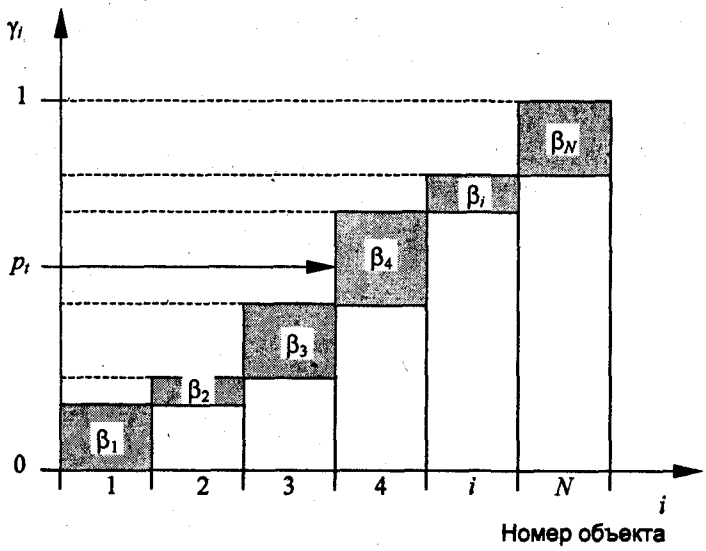


Рис. 1.1. Пример применения метода обратных функций

Нормальное распределение. Нормальное, или гауссово распределение, – это, несомненно, одно из наиболее важных и часто используемых видов непрерывных распределений. Оно симметрично относительно математического ожидания. Сначала остановимся на практическом смысле этого распределения применительно к экономическим задачам и сформулируем центральную предельную теорему теории вероятностей в следующей практической интерпретации.

Непрерывная случайная величина t имеет нормальное распределение вероятностей с параметрами m и $\sigma > 0$, если ее плотность вероятностей имеет вид (рис. 1.2):

$$p(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{t-m}{\sigma}\right)^2\right],$$

где m – математическое ожидание $M[t]$;
σ – среднеквадратичное отклонение.

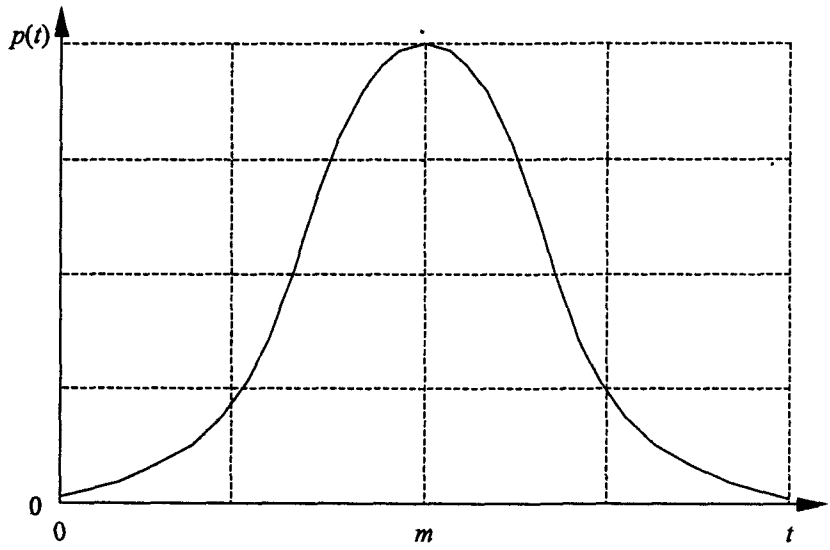


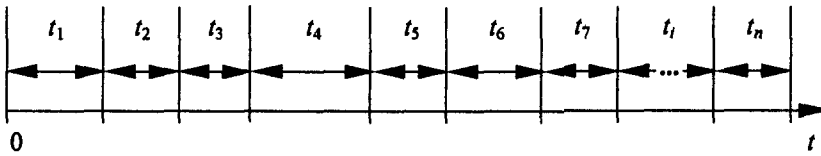
Рис. 1.2. Вид нормального распределения

Причем если $D[t]$ – это дисперсия, то $\sigma = \sqrt{D[t]}$.

Рассмотрим временные диаграммы на рис. 1.3. Предположим, что какой-то случайный процесс состоит из последовательности n

элементарных независимых процессов. Длительность каждого элементарного процесса t_i – это случайная величина, распределенная по неизвестному закону с математическим ожиданием $\bar{t}_i$ и дисперсией σ_i^2. Допустим, что это непрерывное распределение (допущение, справедливое для нормальной жизни общества, где существует природная инертность), причем третий момент должен иметь ограничение по абсолютной величине (это также, естественно, реальное условие). Справедливо соотношение

$$T\{n\} = \sum_{i=1}^{n} t_i \ .$$



$T\{n\} = \sum_{i=1}^{n} t_i$ – случайная величина, являющаяся суммой n независимых случайных величин, распределенных по неизвестному закону и имеющих конечный третий абсолютный момент.

Если сделать предельный переход $n \to \infty$, то распределение величины $t=T\{n\}$ стремится к нормальному с математическим ожиданием $M[t]$, равным сумме математических ожиданий элементарных случайных отрезков t_i, $i=1, ..., n$, и дисперсией $D[t]$, равной сумме дисперсий этих отрезков:

$$M[t] = \sum_{i=1}^{n} \bar{t}_i \quad \text{и} \quad D[t] = \sum_{i=1}^{n} \sigma_i^2 \ .$$

Рис. 1.3. Интерпретация нормального распределения

Теорема 1 (без доказательства). Если сделать предельный переход и устремить $n \to \infty$, то распределение случайной величины $t=T\{n\}$ устремится к нормальному с математическим ожиданием $M[t]$ и дисперсией $D[t]$, определяемыми из следующих соотношений:

$$M[t] = \sum_{i=1}^{n} \bar{t}_i \quad \text{и} \quad D[t] = \sum_{i=1}^{n} \sigma_i^2 \ .$$

В различных математических постановках центральная предельная теорема рассматривается в научной литературе по теории вероятностей и математической статистике.

Практический смысл этой теоремы очень прост. Любые сложные работы на объектах экономики (ввод информации из документа в компьютер, проведение переговоров, ремонт оборудования и др.) состоят из многих коротких последовательных элементарных составляющих работ. Причем количество этих составляющих работ иногда настолько велико, что требования в приведенной выше теореме о независимости и одинаковом распределении становятся излишними. Поэтому при оценках трудозатрат всегда справедливо предположение о том, что их продолжительность – это случайная величина, которая распределена по нормальному закону.

Ниже приведена функция на C++, возвращающая нормально распределенную случайную величину:

```
float normal(float m, float s)
   {
   float a;
   int   i;
   a=0.0;
   for (i=0; i<12; i++)
      a += rundum();
   return(m+(a-6.0)*s);
   }
```

Работа функции основана на применении центральной предельной теоремы. При получении одного числа используются 12 равномерно распределенных на отрезке (0, 1) величин, которые суммируются. Последовательность чисел, распределенных равномерно на отрезке (0, 1), имеет математическое ожидание 1/2 и дисперсию 1/12. С учетом центральной предельной теоремы сумма таких 12 чисел имеет математическое ожидание 6 и дисперсию 1. После суммирования выполняются необходимые действия для обеспечения параметров нормального распределения: математического ожидания m и дисперсии σ^2.

Входными параметрами этой функции являются:

- m – математическое ожидание $M[t]$;
- s – среднеквадратичное отклонение $\sigma = \sqrt{D[t]}$.

Преимущество этой функции – высокое быстродействие. Недостатком является игнорирование «хвостов» нормального распределения, которые могут уходить в обе стороны от величины m на расстояние, превышающее 6σ. Поэтому при проведении особо точных экспериментов применяются другие – более точные (но более медленные) функции. В современных системах имитационного моде-

лирования обычно используются не менее двух программных датчиков случайных величин, распределенных по нормальному закону (их выбор осуществляется автоматически управляющей программой).

Экспоненциальное распределение. Оно также занимает очень важное место при проведении системного анализа экономической деятельности. Этому закону распределения подчиняются многие явления, например:

- время поступления заказа на предприятие;
- посещение покупателями магазина-супермаркета;
- телефонные разговоры;
- срок службы деталей и узлов в компьютере, установленном, например, в бухгалтерии.

Рассмотрим это распределение подробнее. Если вероятность наступления события на малом интервале времени Δt очень мала и не зависит от наступления других событий, то интервалы времени между последовательностями событий распределяются по экспоненциальному закону с плотностью вероятностей

$$p(t) = \begin{cases} \lambda e^{-\lambda t} & , \ t \geq 0 \ ; \\ 0 & , \ t < 0 \ . \end{cases}$$

Особенностью этого распределения являются его параметры:

- математическое ожидание $M[t] = 1/\lambda$;
- дисперсия $D[t] = \sigma^2 = (1/\lambda)^2$.

Математическое ожидание равно среднеквадратичному отклонению, что является одним из основных свойств экспоненциального распределения.

На первый взгляд, это распределение является надуманным. Поэтому рассмотрим предельную теорему о суперпозиции потоков. Предположим, что можно наблюдать k независимых потоков событий (рис. 1.4). В каждом таком потоке можно наблюдать m_j элементарных событий, $j=1, \ldots, k$. Интервалы времени между событиями – это независимые случайные величины, распределенные по неизвестному закону с математическим ожиданием $1/\lambda_j$. Спроектируем моменты всех событий на общую ось времени и рассмотрим случайный интервал времени $t=T\{k\}$ между двумя событиями полученного суммарного потока, состоящего из n событий, где

$$n = \sum_{j=1}^{k} m_j \, .$$

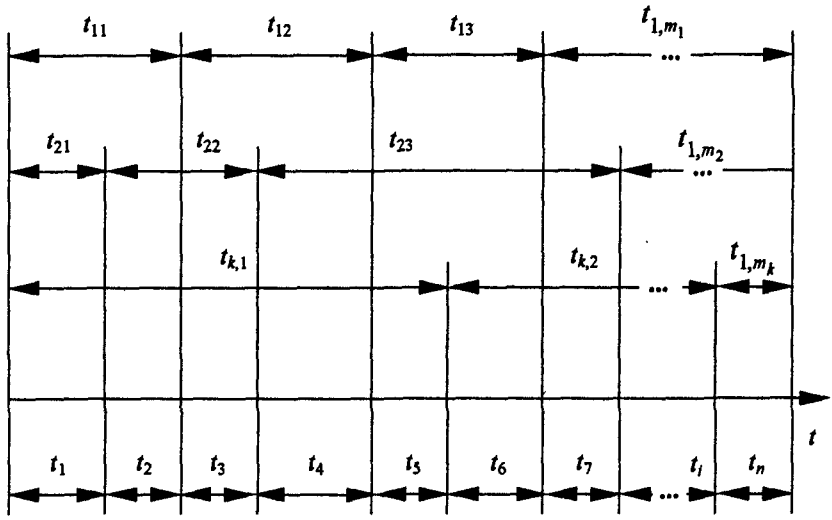


Рис. 1.4. Интерпретация предельной теоремы о суперпозиции потоков событий

Теорема 2 (без доказательства). Если сделать предельный переход и устремить $n \to \infty$, то распределение случайной величины интервала $t=T\{k\}$ в суммарном потоке событий, состоящем из k элементарных потоков, устремится к экспоненциальному с математическим ожиданием

$$M[t] = \frac{1}{\sum_{j=1}^{k} \lambda_j}.$$

Следствие (без доказательства). Поток заявок, интервал поступления которых в некую систему имеет экспоненциальное распределение, является простейшим.

Прокомментируем практический смысл этой теоремы.

Пример 1.2. Допустим, что имеется некая крупная фирма. Клиенты фирмы – это физические и юридические лица. Каждый из них может иметь набор планов и расписанных дел на значительном интервале времени. Однако если рассмотреть суммарный поток обращений этих клиентов к служащим фирмы по разным вопросам, то интервал времени между двумя последовательными обращениями в

соответствии с рассмотренной теоремой является случайной величиной, распределенной по экспоненциальному закону.

Перейдем к рассмотрению функции, позволяющей получить псевдослучайную последовательность, распределенную по экспоненциальному закону. Текст соответствующей программы на C++:

```
float expont(float m)
   {
   float r;
   r=log(rundum());
   return(m*(-r));
   }
```

Единственный входной параметр программы – математическое ожидание m= $M[t]$. В этой программе использован метод обратных функций.

Обобщенное распределение Эрланга. Обычно распределение Эрланга используется в случаях, когда длительность какого-либо процесса можно представить как сумму k элементарных последовательных составляющих, распределенных по экспоненциальному закону. Если обозначить математическое ожидание длительности всего процесса как $M[t]=1/\lambda$, среднюю длительность элементарной составляющей как $1/\lambda$, то плотность вероятностей распределения Эрланга представляется следующей формулой:

$$p(t)=\begin{cases}\lambda k\dfrac{(\lambda k t)^{k-1}}{(k-1)!}e^{-\lambda k t}, & t\geq 0\ ;\\ 0, & t<0\ .\end{cases}$$

Дисперсия такого распределения $D[t]=\dfrac{1}{\lambda^2 k}$.

Очевидно, что при k=1 – это экспоненциальное распределение. Разновидности этого распределения для разных $k>0$ представлены на рис. 1.5.

Предположим, что в распределении Эрланга имеется не строго фиксированное число экспоненциально распределенных отрезков k, а переменное, с вероятными изменениями в пределах одного интервала. Тогда можно говорить лишь о средней величине s таких отрезков, где s – число с плавающей точкой. После такого перехода от дискретных к непрерывным величинам появляется возможность работы и со значениями в пределах $0<s<1$.

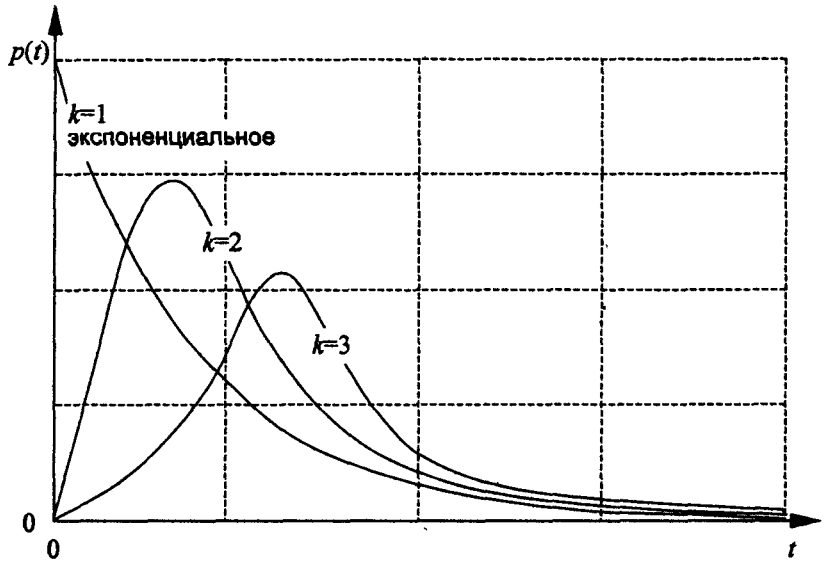


Рис. 1.5. Вид распределения Эрланга при различных значениях *k*:
1 – экспоненциальное; 2, 3 – распределения второго и третьего порядков

Рассмотрим программную функцию, реализующую такое распределение:

```
float erlang(float m, float s)
   {
   float a, b;
   int   i, k;
   a = 0.0;
   k = s;
   b = s - k;
   for( i=0;i<k;i++ )
      a += expont(m);
   if( rundum() < b )
      a += expont(m);
   return(a);
   }
```

Эта функция имеет два входных параметра:

- m – математическое ожидание элементарного интервала времени, причем $m=1/\lambda k$;
- s – среднее число элементарных отрезков в общей длительности процесса.

Видно, что при значениях $s \geq 1$ (в том числе целых) получаем обычное распределение Эрланга с параметрами: $M[t] = ms = 1/\lambda$ и $D[t] = m^2s=1/\lambda^2 k$. Однако при $0 < s < 0$ это распределение меняется коренным образом: фактически мы получаем процесс испытаний Бернулли. В результате этих испытаний «успехом» считается получение элементарного отрезка, распределенного по экспоненциальному закону с математическим ожиданием m (вероятность успеха равна s), а неудачей с вероятностью 1– s является получение элементарного отрезка с нулевой длиной. Если по такому правилу будет работать какой-то генератор заявок, то он будет создавать группы заявок. Причем средний размер группы будет равен $\bar{n} = 1/s$, а средний интервал времени между двумя последовательными группами равен m.

Математическое ожидание интервала между двумя последовательными заявками и в этом случае определяется выражением $M[t] = ms = 1/\lambda$. Что касается дисперсии, то она существенно меняется и определяется по формуле:

$$D[t] = m^2 s^2 \left(\frac{2}{s} - 1\right).$$

Одно из свойств групповых потоков заключается в том, что среднеквадратичное отклонение интервала между заявками превосходит математическое ожидание этого интервала.

Если имеется возможность собрать статистику по групповому потоку на практике или получить такой поток с помощью рассмотренной выше программной функции, то можно определить коэффициент вариации c и связь этого коэффициента со средним размером группы заявок по формуле

$$\bar{n} = \frac{1}{2}\left(1 + c^2\right) .$$

Это соотношение позволяет отслеживать появление групповых потоков в реальных системах или в их имитационных моделях.

Обобщенное распределение Эрланга применяется при создании как чисто математических, так и имитационных моделей в двух случаях.

Во-первых, его удобно применять вместо нормального распределения, если модель можно свести к чисто математической задаче, применяя аппарат марковских или полумарковских процессов либо

используя метод Кендалла. Однако такие модели далеко не всегда адекватны реальным процессам.

Во-вторых, в реальной жизни существует объективная вероятность возникновения групп заявок в качестве реакции на какие-то действия, поэтому возникают групповые потоки. Применение чисто математических методов для исследования в моделях эффектов от таких групповых потоков либо невозможно из-за отсутствия способа получения аналитического выражения, либо затруднено, так как аналитические выражения содержат большую систематическую погрешность из-за многочисленных допущений, благодаря которым исследователь смог получить эти выражения. Для описания одной из разновидностей группового потока можно применить обобщенное распределение Эрланга, которое рассмотрим ниже. Внешне похожее на гамма-распределение, оно имеет свои математические особенности.

Появление групповых потоков в сложных экономических системах приводит к резкому увеличению средних длительностей различных задержек (заказов в очередях, задержек платежей и др.), а также к увеличению вероятностей рисковых событий или страховых случаев.

Треугольное распределение. Применимость такого распределения рассмотрим на примере, связанном с динамическими характеристиками системы управления базами данных (СУБД) в экономической информационной системе.

Пример 1.3. Предположим, что база данных находится на компьютере, не входящем в состав какой-либо вычислительной сети. Поэтому пользователь, работающий с этой базой, имеет во время работы монопольный доступ к ней. Известны структуры и частоты запросов пользователей к этой базе данных. Рассмотрим три случая физической организации базы данных (рис. 1.6).

Первый случай. Допустим, что администратор базы данных (системный программист) осуществил физическую организацию данных, которая обладает следующими свойствами:

- наиболее вероятное время ответа на запрос близко к 0 с;
- минимальное вероятное время ответа не менее 0 с;
- максимальное вероятное время ответа не превышает 15 с;
- распределение вероятностей представлено линией 1 на рис. 1.6.

Этот системный программист обеспечил минимальное время для наиболее вероятных запросов за счет увеличения времени для менее вероятных. Среднее время получения ответа в этом случае $\bar{t} = 5$ с.

Второй случай. Администратор базы данных по просьбе пользователей решил уменьшить время ответа на те запросы, которые редко возникают. Для этого он переделал физическую организацию данных и получил следующие ее свойства:

- наиболее вероятное время ответа на запрос равно 5 с;
- минимальное вероятное время ответа не менее 0 с;
- максимальное вероятное время ответа не превышает 10 с;
- распределение вероятностей показано линией 2 на рис. 1.6.

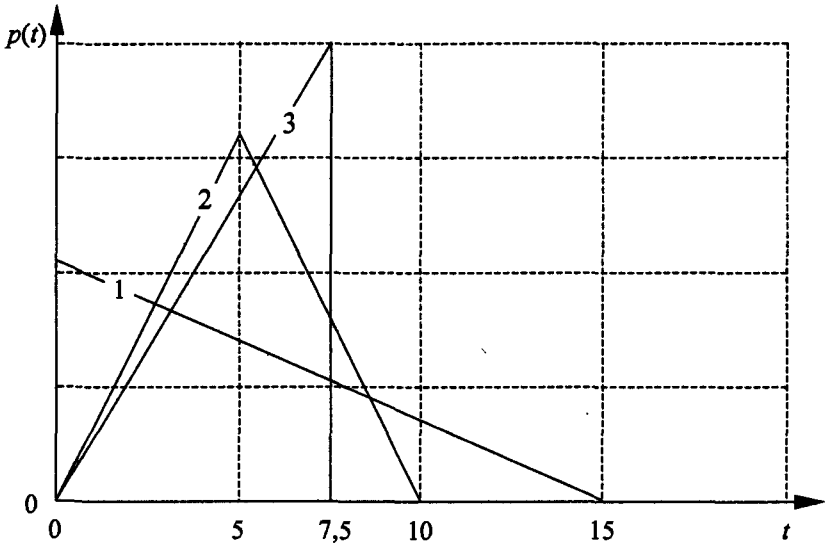


Рис. 1.6. График плотности вероятностей для треугольного распределения: 1 – максимум слева; 2 – максимум в центре; 3 – максимум справа

Таким образом, системный программист обеспечил снижение времени ответа для менее вероятных запросов за счет увеличения времени ответа для наиболее вероятных. Среднее время получения ответа осталось тем же: $\bar{t} = 5$ с.

Третий случай. Администратор базы данных решил еще более уменьшить время ответа на менее вероятные запросы. Для этого он опять переделал физическую организацию данных и получил следующие свойства:

- наиболее вероятное время ответа на запрос равно 7,5 с;
- минимальное вероятное время ответа не менее 0 с;

- максимальное вероятное время ответа не превышает 7,5;
- распределение вероятностей изображено линией 3 на рис. 1.6.

Этот системный программист обеспечил дальнейшее снижение времени ответа для менее вероятных запросов за счет увеличения времени ответа для более вероятных. Среднее время получения ответа и в этом случае не изменилось: $\bar{t} = 5$ с.

Возникает естественный вопрос: «Какая физическая организация лучше?». Если отбросить факторы, определяющие большую или меньшую важность запросов, и вспомнить, что база данных не имеет множественного доступа из вычислительной сети, то можно утверждать, что все три способа организации данных одинаковы, так как пользователи этой базы имеют одно и то же среднее время ответа.

Однако если подключить компьютер с нашей базой к локальной вычислительной сети и разрешить доступ к базе данных большому числу пользователей этой сети из рабочих компьютеров этих пользователей, то необходимо учитывать возникновение очереди запросов к базе данных при ее монопольном использовании. Предположим, что число пользователей довольно велико и выполняются условия предельной теоремы о суперпозиции потоков событий (в нашем случае возникновение запроса к базе данных – это событие). Тогда поток запросов к базе простейший (экспоненциальное распределение интервала поступления). Поэтому выполняются условия, при которых справедлива следующая формула для оценки средней задержки запросов в очереди (формула Поллачека–Хинчина):

$$t_q = \frac{t_s \rho \left(1 + c_s^2\right)}{2(1-\rho)},$$

где t_q – искомая средняя задержка в очереди;
t_s – среднее время обслуживания;
ρ – загрузка обслуживающего узла ($\rho = t_s / t_q \leq 1$);
c_s – коэффициент вариации времени обслуживания.

Если известно среднеквадратичное отклонение времени обслуживания σ_s, то $c_s = \sigma_s / t_s$. В трех рассмотренных случаях $t_s = \bar{t} = 5$ с. Загрузка не изменяется, так как поток запросов к базе данных тот же самый. Однако разброс значений в первом случае примерно в 3 раза

больше, чем в третьем. Соответственно c_s^2 может быть больше приблизительно в 9 раз (т.е. на порядок!), а это часть множителя в числителе формулы.

После этого можно сделать вывод, что задержка в очереди в первом случае будет значительно больше, чем в третьем. Во втором случае задержка в очереди также будет превосходить задержку, возникающую в третьем случае. Поэтому наиболее рациональным относительно возникающих задержек является третий способ организации базы данных.

Выражения для определения математического ожидания $M[t]$ и дисперсии $D[t]$ получаются интегрированием с использованием определений первого и второго моментов:

$$M[t]=\frac{a+m+b}{3} \text{ и } D[t]=\frac{(b-a)^2+(b-m)^2+(m-a)^2}{36}.$$

Ниже приведен текст программной функции на C++, возвращающей случайную величину, распределенную по треугольному закону:

```
float triplex(float a, float m, float b)
   {
   float x, r;
   r=rundum();
   if( r <= (m-a)/(b-a) )
      x = a + sqrt( r*(m-a)*(b-a) );
   else
      x = b - sqrt( (1.0-r)*(b-m)*(b-a) );
   return(x);
   }
```

Эта программа использует метод обратных функций. Она имеет три входных параметра:

- a – минимально возможное значение интервала времени;
- b – максимально возможное значение интервала времени;
- m – наиболее вероятное значение интервала времени (максимум плотности вероятностей).

Естественно, входные параметры должны удовлетворять следующим условиям: $a \le m \le b$.

1.4
НЕТРАДИЦИОННЫЕ СЕТЕВЫЕ МОДЕЛИ И ВРЕМЕННЫЕ ДИАГРАММЫ ИНТЕРВАЛОВ АКТИВНОСТИ

Основа концепции имитационного инструментария, с помощью которого можно проводить структурный анализ и имитационное моделирование, заключается в механизмах, позволяющих агрегировать элементарные процессы и устанавливать между ними функциональные связи (причинно-следственные, информационные, финансовые и иные). Ниже предлагается сетевая концепция, существенно отличающаяся от аналитического аппарата, рассмотренного в литературе по теории массового обслуживания, использующая удачные результаты теории стохастических сетей и численные методы, основанные на диффузной аппроксимации процессов массового обслуживания.

Эта концепция разработана, в первую очередь, для последующей реализации имитационных механизмов в рамках специального пакета имитационного моделирования. Она предназначена для верификации работоспособности пакета, для оценочных расчетов при отладке имитационных моделей, но не предназначена для практических расчетов показателей риска по аналитическим формулам.

Идею предлагаемой концепции рассмотрим на примере из конкретного проекта «Открытое образование» («*e*-образование»), реализуемого под патронажем Международной академии открытого образования (МАОО).

Учебные процессы в открытом образовании. Учебный процесс – это понятие, охватывающее всю учебную деятельность классического университета. Учебный процесс состоит из многих компонентов: процесса обучения студента по конкретной специальности в течение пяти лет, семестрового учебного процесса на потоке, процесса изучения дисциплины. Классический университет имеет жесткий избыточный набор ресурсов, который позволяет реализовать учебный процесс в любой его интерпретации. Однако такой фиксированный набор приводит к издержкам планирования, к удорожанию обучения студента без гарантий высокого качества.

В открытом образовании работают специалисты, имеющие квалификацию не ниже, чем в классическом университете. Единственное, что их отличает, – это различные образовательные технологии (классическая и комплексная).

Процессом* изучения дисциплины (далее – процессом) в распределенном институте назовем неделимую функцию освоения дисциплины студентом по утвержденной программе. Для реализации процесса необходимы различные ресурсы.

В системе открытого образования ресурсы используются в распределенном режиме. Ресурсы распределенного института можно поделить на два типа: интеллектуальный ресурс (учитель) и учебный ресурс (далее – просто ресурс).

Учителя – это преподаватели кафедр, тренеры учебно-тренировочных фирм, тьюторы-консультанты. Учителя предметно относятся к разным распределенным кафедрам через механизм «аттестации».

Ресурсы – это комплекты учебно-практических пособий, студии (если есть дистанционная фаза типа телеконференции), режимы Интернет-доступа, аудитории (если есть очная фаза) и другие, без которых обучение студента может не состояться.

Процесс запущен, если возникла необходимость изучения дисциплины и распределенный институт имеет для этого ресурсы. Запуск процесса не означает, что в любой момент времени будет хотя бы один студент, изучающий эту дисциплину. Соответственно процесс может быть снят (или отменен).

Далее будем полагать, что распределенный институт ориентирован прежде всего на индивидуализацию обучения студента. Поэтому с учетом случайных явлений, не зависящих от распределенного института, при массовом обслуживании студентов возможны технологические задержки: очереди к учителям и задержки из-за временной нехватки ресурсов. Возникает задача определения такого числа ресурсов, при котором процесс обучения по конкретной специальности имел бы продолжительность не хуже заданной с учетом технологических задержек.

При реализации обучения по специальности процессы могут иметь причинно-следственные связи. Поэтому можно говорить о том, что они образуют направленный граф (рис. 1.7). Применение методов сетевого планирования и управления невозможно; основная трудность – это циклы. Циклы возникают по двум причинам: студенты обучаются не по жесткому учебному плану (возможны различные индивидуальные планы), для отстающих студентов органи-

---

* В университетах используется более подходящее слово «курс». Однако в интересах поддержки общности с экономикой используется термин «процесс».

зуется повторное обучение (возврат к пройденной ранее, но не защищенной дисциплине для ее более глубокого изучения). Относительно пути студента по графу в каждый момент времени он находится в определенном текущем процессе – в узле графа. Процесс, который передал студента в текущий процесс, назовем *производителем*, а процесс, который примет студента после завершения текущего, назовем *потребителем*.

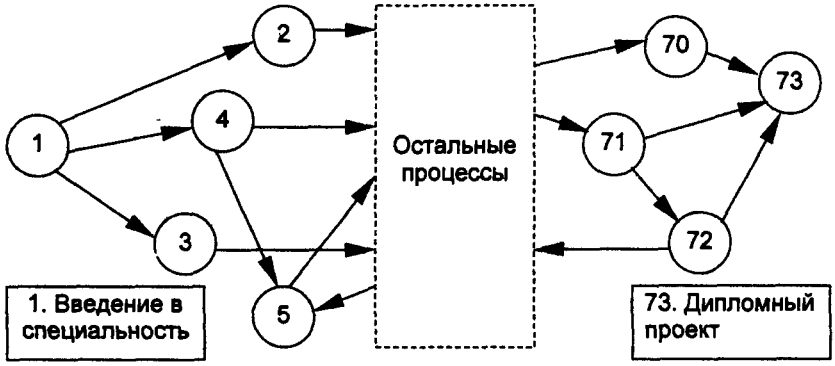


Рис. 1.7. Сеть процессов (курсов) при обучении по специальности

Рассмотрим возможные диаграммы состояний процесса (рис. 1.8). Если мощности ресурсов бесконечны либо каждый процесс используется вместе с постоянно закрепленными за ним ресурсами, то возможны два состояния (рис.1.8,*а*): ожидание студентов (ЖС) и выполнение процесса изучения дисциплины (ПУ). В таких ситуациях не возникает необходимости в незапланированных ресурсах: у студента есть учебный план. В состояние ПУ процесс попадает, получив студента от процесса-производителя. После изучения дисциплины студент переходит к процессу-потребителю и попадает в состояние ЖС, если какой-либо производитель не подготовил следующего студента.

В более реальном случае (рис. 1.8,*б*) при конечных мощностях глобальных ресурсов появляется состояние ожидания ресурса, когда процессу (точнее, студенту в процессе изучения дисциплины) нужны ресурсы (НР).

В условиях реального университета, когда обучение контролируется, а выделение ресурсов и их возвращение осуществляется с помощью процессов планирования и распределения ресурсов, вво-

дятся еще два состояния (рис.1.8,*в*): подготовка к выполнению (ГВ) и завершение выполнения – контрольные мероприятия, экзамены, зачеты (ЭЗ).

Когда возникает потребность в незапланированных ресурсах, то возможны обратные переходы типа ПУ→НР (рис. 1.8,*б*, 1.8,*в*). Такие переходы могут привести к блокировкам, которые можно разрешить с помощью известных решений задачи взаимного исключения.

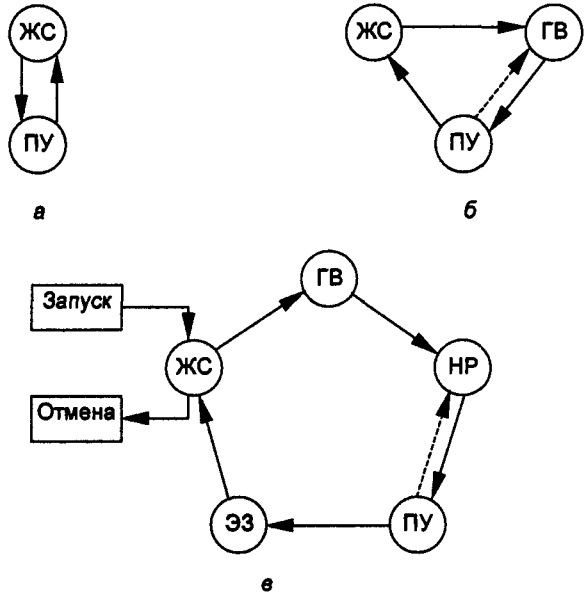


Рис. 1.8. Диаграммы состояний процессов (курсов):
а – мощности ресурсов бесконечны; *б* – мощности ресурсов конечны;
в – мощности ресурсов конечны, есть накладные расходы времени;
ЖС – ждет студентов; ГВ – готовится к выполнению; НР – нужны ресурсы;
ПУ – учеба по дисциплине; ЭЗ – экзамен, зачет

Во время подготовки к выполнению (ГВ) осуществляется планирование ресурсов, а после завершения (ЭЗ) – возврат ресурсов в распоряжение планирующих и распределяющих процессов.

Организация и взаимосвязь различных компонентов системы открытого образования может быть рассмотрена относительно управления процессами в следующих подразделениях университета (рис. 1.9):

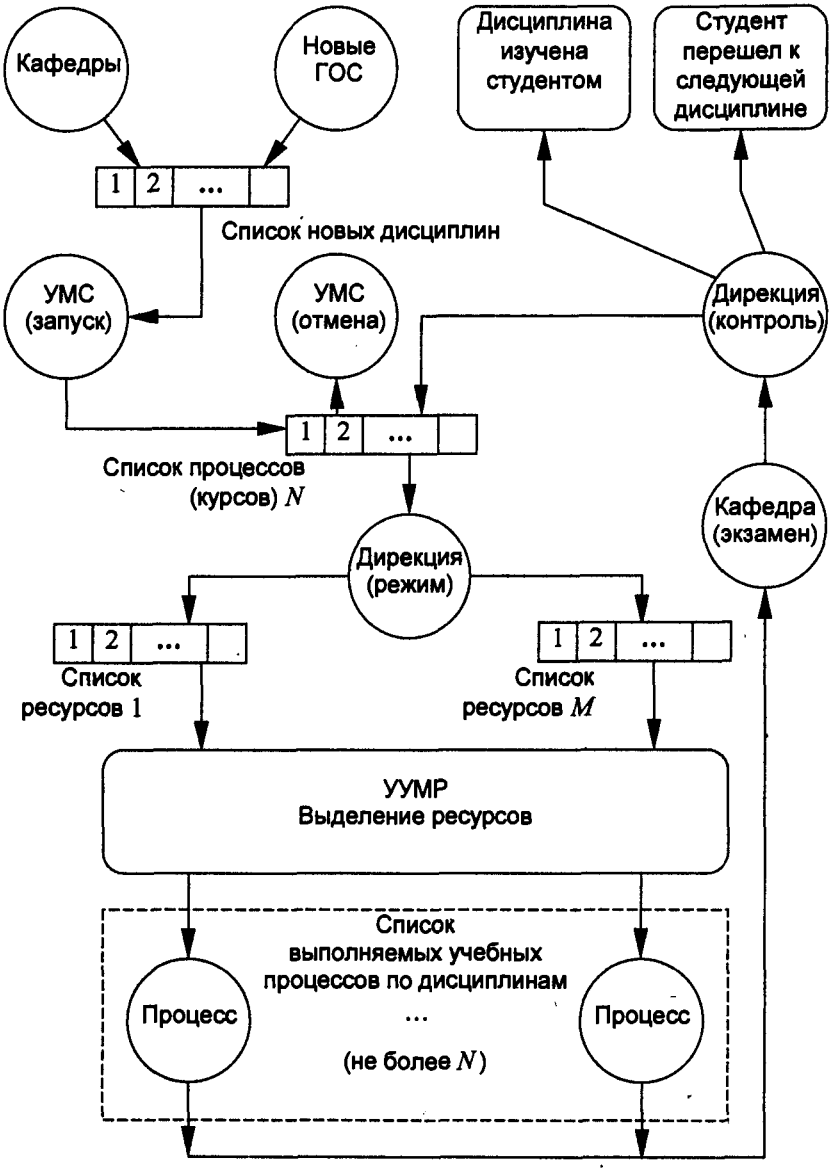


Рис. 1.9. Функциональные взаимосвязи организующих объектов в учебном процессе распределенного института

• управления учебной и учебно-методической работой (УУМР), которое ведает всеми ресурсами, относящимися к учебному процессу;

• дирекции распределенного университета, которая совместно с территориально-распределенными филиалами университета (партнерами) отвечает за реализацию учебного плана;

• учебно-методологического совета (УМС), который работает при дирекции в качестве коллегиального совещательного органа для постоянного совершенствования государственных образовательных стандартов (ГОС), изменяющихся примерно раз в пять лет, и учебного плана, который корректируется ежегодно (в рамках действующего ГОС);

• кафедр (распределенных кафедр открытого образования). Кафедры – это обладатели интеллектуального ресурса (профессорско-преподавательского состава, тренеров (тьюторов-консультантов), аспирантов, докторантов и других преподавателей).

Далее перейдем к оценке времени изучения студентами дисциплин учебного плана. Время прохождения всех дисциплин учебного плана студентом – это время пребывания заявки в стохастической сети (*см.* рис. 1.7). Заявки в такой сети будем называть «транзактами», чтобы отличать от других элементарных заявок.

Транзакт, попадая из одного узла сети (процесс-производитель) в другой узел (процесс-потребитель), свидетельствует о необходимости изучения студентом следующей дисциплины учебного плана. После этого процесс-потребитель выводится из состояния ЖС и попадает в состояние ГВ. После выделения ресуроов (НР), выполнения функции (ПУ) и завершения выполнения контрольных мероприятий (ЭЗ) транзакт появляется на выходе узла-производителя, а процесс возвращается в состояние ЖС. Случайный интервал времени, ограниченный моментом выхода процесса из состояния ЖС в начале изучения дисциплины и ближайшим моментом попадания в это состояние, назовем *интервалом активности* процесса. Длительность пребывания транзакта в соответствующем узле – это интервал активности. Для оценки времени реакции системы открытого образования, реализуемой в рамках распределенного института по учебному плану, необходимо уметь рассчитывать значения интервалов активности всех процессов, входящих в состав сети.

Оценка нитервала активности процесса. Далее построим итерационную процедуру, позволяющую провести соответствующие оценки. Обозначим А – начало очередной итерации; Б – конец очередной итерации.

А. Начало итерации. Пронумеруем N узлов стохастической сети, характеризующих конкретный план, индексами j и предположим справедливость допущений:

1) известны средние значения всех интервалов активности t_{ij}, j=1,2, ... , N (только в каком-то приближении, так как их необходимо рассчитать);

2) известны вероятности поступления транзакта из каждого процесса-производителя в любой процесс-потребитель π_{jn} , j, n = 1, 2, ... , N; эти вероятности определяются исходя из параметров набора учебных планов, индивидуальных схем обучения и числа студентов, проходящих через систему открытого образования;

3) сеть, отображающая конкретный учебный план, является полнодоступной с матрицей передач П=[π_{jn}], причем вероятность поступления транзакта в процесс-потребитель n в течение интервала времени (t, t+Δ_t) является линейной комбинацией с постоянными коэффициентами π_{jn} вероятностей появления заявок на выходах вершин-производителей с номерами j, j=1,2, ... , N.

Такие допущения могут быть в какой-то степени справедливы, если учебный процесс находится в стационарном режиме (если переходные процессы и были, то они завершились). Поэтому при их выполнении можно получить среднее время изучения всех дисциплин учебного плана по формуле для замкнутых сетей:

$$и = \frac{1}{\lambda} \sum_{j=1}^{N} \lambda_j \; t_{ij},$$

где t_{ij} – средняя длительность интервала активности;
λ_j – интенсивность запросов на курс с номером j;
λ – интенсивность поступления потока студентов, желающих обучаться по данному учебному плану.

Поэтому для оценки времени выполнения учебного плана необходимо знать средние значения интервалов активности всех процессов.

Для анализа интервала активности необходимо рассмотреть «нетрадиционную модель» массового обслуживания. Введем условные обозначения параметров временных интервалов:

$t_ж$ – процесс ждет студентов (состояние ЖС);
$t_г$ – готовится к выполнению (состояние ГВ);
$t_н$ – процессу нужны ресурсы (состояние НР);

$t_п$ – процесс учебы (состояние ПУ);
$d_п$ – дисперсия процесса учебы;
$t_э$ – экзамен, зачет, контрольное мероприятие (состояние ЭЗ);
$t_р$ – длительность ожидания запроса в очереди к ресурсу;
$t_и$ – длительность интервала активности процесса;
$t_с$ – основная составляющая интервала активности $t_с = t_и - t_г$;
$t_о$ – длительность ожидания первого элемента ресурса;
t_r – длительность ожидания какого-либо ресурса;
t_q – длительность ожидания обслуживания в очереди к ресурсу;
t_s – длительность обслуживания в очереди к ресурсу;
t_v – время ожидания из-за нехватки элементов ресурса;
t_w – время ожидания запрошенного элемента ресурса;
c_r – коэффициент вариации времени ожидания ресурса;
c_s – коэффициент вариации времени обслуживания в очереди к ресурсу;
c_a – коэффициент вариации интервала запросов к ресурсу.

На интервале активности процесс может находиться в состояниях ГВ, НР, ПУ и ЭЗ. Будем считать, что ресурсы выделяются во время пребывания в состоянии НР (интервал $t_{нj}$), а освобождаются все сразу – в конце интервала пребывания в состоянии ЭЗ (по истечении $t_{эj}$); оба эти интервала – детерминированные величины. Интервал активности равен:

$$t_{иj}^{(t)} = t_{гj} + t_{нj}^{(t)} + t_{пj}^{(t)} + t_{эj},$$

где $t_{нj}^{(t)}$ – длительность пребывания в состоянии НР;

$t_{пj}^{(t)}$ – длительность пребывания в состоянии ПУ;

j – номер процесса, j=1,2, ... , N.

(t) – индекс, показывающий случайный характер индексируемой величины.

Будем считать, что величины $t_{гj}$ и $t_{эj}$ известны, а интервал $t_{пj}^{(t)}$ задан с помощью математического ожидания $t_п$ и дисперсии $d_п$. Интервал $t_{пj}^{(t)}$ можно определять с помощью одного из трех возможных способов:

1) исходя из характеристик распределенного института (головного университета, осуществляющего открытое образование);

2) с помощью хронометрирования;

3) если институт осуществляет приоритетное обслуживание для некоторых категорий учащихся, то $t_{пj}^{(t)}$ – это цикл обслуживания; методика определения цикла обслуживания для потоков типа пуассоновского или группового и формулы для расчетов приведены в работе [19].

Предположим, что в распоряжении института имеется M глобальных ресурсов, используемых при обучении. Мощность каждого ресурса – S_i элементов, а для выполнения процесса (изучения курса) j предварительно необходимо выделить R_{ij} элементов каждого ресурса. Причем $0 \le R_{ij} \le S_i$, $i=1,2, \ldots, M, j=1,2, \ldots, N$.

Поставим в соответствие началу интервала активности момент появления транзакта на входе модельной системы, изображенной на рис. 1.10,*а*. Этот транзакт попадает на вход генератора, на каждом i-м выходе которого через время t_{ij} появятся порции R_{ij} заявок, которые распределяются по S_i очередям. Длительность обслуживания в каждой очереди $t_{ri}^{(t)}$ – это интервал времени, начинающийся в момент выделения процессу первого элемента ресурса i из набора свободных ресурсов и заканчивающийся моментом возвращения всех R_{ij} элементов в этот набор (каждому ресурсу соответствует свой менеджер обслуживания, контролирующий очередь).

Через какой-то интервал времени

$$t_{oij}^{(t)} = \min_{R_{ij}} \left\{ t_{wij}^{(t)} \right\}$$

на входе счетчика появится первая удовлетворенная заявка очереди i-го ресурса. После этого проходит еще $R_{ij}-1$ случайных интервалов, пока не появятся остальные заявки (каждая соответствует одному выделенному элементу). Если считать, что интенсивность освобождения процессами элементов стационарна, то каждый из $R_{ij}-1$ интервалов в среднем равен t_{ri} / S_i, где $t_{ri} = M[t_{ri}^{(t)}]$.

Через время

$$t_{vij}^{(t)} = \max_{R_{ij}} \left\{ t_{wij}^{(t)} \right\}$$

появится последняя заявка, соответствующая выделению последнего из запрошенных r_{ij} элементов i-го ресурса, после чего учебный процесс выполняется (за время $t_{uj}^{(t)}$) и завершается контрольными мероприятиями (за время $t_{\text{э}j}$). Поток заявок, поступающий на вход рассматриваемой модели, – неординарный с интенсивностью λ_j. Мы не рассматриваем классический учебный процесс классического университета, строящийся по принципу объединения: специальность⇔учебный план⇔учебное расписание⇔контингент (группы студентов).

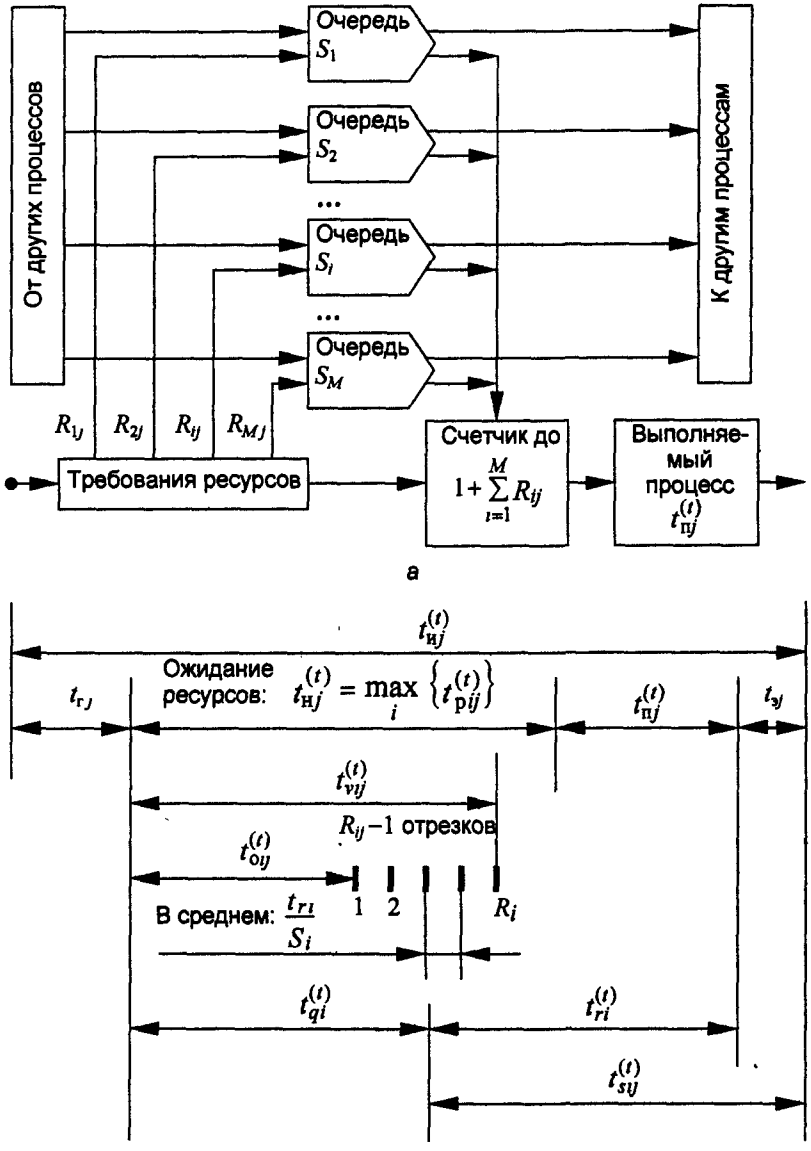


Рис. 1.10. Модель анализа интервала активности процесса:
а – схема массового обслуживания в пределах интервала активности;
б – временные диаграммы элементарных процессов

Однако есть две причины, позволяющие предположить возникновение в модели режима открытого образования случайных групп в потоках транзактов:

- один студент может привести несколько студентов (а в модели один транзакт может породить группу других транзактов);
- в учебном плане могут быть циклы.

Сеть процессов, образующих учебный план, – довольно сложная, полнодоступная. Поэтому в практических расчетах будем считать, что поток групп – пуассоновский, а размер группы распределен по закону обобщенного распределения Эрланга.

Одно из свойств групповых потоков заключается в том, что σ превосходит математическое ожидание интервала между заявками, поэтому коэффициент вариации $c > 1$. Формула для оценки среднего размера группы заявок при обобщенном распределении Эрланга имеет вид

$$\bar{n} = \frac{1}{2}\left(1 + c^2\right).$$

Это соотношение позволяет отслеживать появление групповых потоков в реальных системах или в их имитационных моделях. Особенность обобщенного распределения Эрланга заключается в том, что его применение позволяет выполнить расчет на худший случай (при перегрузках).

Далее воспользуемся свойствами полученного распределения. Поэтому применим гипотезу Л. Клейнрока о независимости не к потокам заявок, а к потокам групп заявок. Коэффициент вариации интервала поступления c_j в групповом потоке такого типа не меньше единицы. Исходя из свойств рассмотренного распределения, средний размер случайной группы $\bar{n}_j$ связан с коэффициентом вариации соотношениями:

$$\bar{n}_j = \frac{1}{2}\left(1 + c_j^2\right) \text{ и, наоборот, } c_j^2 = 2\bar{n}_j - 1.$$

Определим интенсивность потока, поступающего к одному из S_i элементов i-го ресурса

$$\lambda_{ri} = \frac{1}{S_i}\sum_{j=1}^{N}\lambda_j R_{ij}.$$

Далее введем вероятность того, что запрос на выделение элемента i-го ресурса поступил от j-го процесса:

$$P_{ij} = \frac{\lambda_j R_{ij}}{S_i \lambda_{ri}}.$$

Размер группы заявок к ресурсу i от модуля j равен $\overline{n}_{rij} = \overline{n}_j R_{ij}$. Средний размер группы, поступающей в одну очередь, равен:

$$\overline{n}_{ai} = 1 + \frac{1}{S_i}\left[\left(\sum_{j=1}^{N} P_{ij}\overline{n}_{rij}\right) - 1\right],$$

откуда можно получить квадрат коэффициента вариации для потока транзактов, поступающих в i-ю очередь:

$$c_{ai}^2 = 2\overline{n}_{ai} - 1.$$

Составляющая интервала активности $t_{cj}^{(t)} = t_{нj}^{(t)} + t_{пj}^{(t)} + t_{эj} + t_{иj}^{(t)} - t_{гj}$ может быть разбита на два слагаемых: интервал пребывания в очереди из-за отсутствия элементов ресурса $t_{qi}^{(t)}$ и интервал времени $t_{sij}^{(t)}$, начавшийся в момент выделения элемента i-го ресурса j-му модулю и заканчивающийся в момент возвращения в общий набор: $t_{cj}^{(t)} = t_{qi}^{(t)} + t_{sij}^{(t)}$ (рис. 1.10,*б*). Предположим, что известна вероятность ненулевой задержки в очереди ρ_i^* (в режиме перегрузок) и среднее значение $t_{cj} = t_{иj} + t_{гj}$. По правилу расчетов для объединения процессов с учетом формул диффузной аппроксимации получим среднее время обслуживания очереди:

$$t_{rj} = \sum_{j=1}^{N} P_{ij}\, t_{sij} = (1 - \rho_i^*) \sum_{j=1}^{N} P_{ij}(t_{иj} - t_{гj}).$$

Допустим, что известна средняя задержка в очереди к ресурсу t_{qi}. Тогда можно аппроксимировать функцию распределения времени пребывания в очереди $h_{qi}(t)$ выражением

$$h_{qi}(t) = 1 - \rho_i^* \exp\left\{\frac{-\rho_i^* t}{t_{qi}}\right\}.$$

Такая аппроксимация позволяет получать дисперсии d_{qi} с погрешностью, находящейся в пределах 15 % даже при эрланговских потоках и постоянных временах обслуживания. Однако при перегрузках (при значениях ρ_i^*, существенно больших 0 и близких к 1) это выражение становится практически точным, и можно показать, что распределение числа заявок в очереди при перегрузке – экспоненциальное. Поэтому дисперсия времени пребывания в очереди равна

$$d_{qi} = \frac{t_{qi}^2}{\rho_i^*}\left(2 - \rho_i^*\right),$$

а средняя длительность задержки в очереди определяется из соотношения, получаемого при аппроксимации процесса изменения длины очереди при перегрузках процессом диффузии,

$$t_{qi} = \frac{t_{ri}\rho_i^*}{1-\rho_i^*}.$$

Дисперсии величин $d_{qi}^{(t)}$ и $d_{ui}^{(t)}$ связаны очевидным соотношением: $d_{wj}=d_{qi}+d_{sij}$, так как t_{sj} – константа. По определению второго момента с использованием правила получения дисперсии объединенного процесса получим дисперсию

$$d_{ri} = \sum_{j=1}^{N} P_{ij}\left[d_{wj} + d_{qi} + \left(t_{wj} - t_{rj} - t_{qi}\right)^2\right] - t_{ri}^2.$$

Далее уточняем величину загрузки элемента ρ_i, которую считали заданной при начале итерации, $\rho_i=\lambda_{ri}t_{ri}$ и определяем квадрат коэффициента вариации длительности обслуживания $c_{si}^2 = d_{ri}/t_{ri}^2$. При расчете на худший случай, естественно, нас беспокоит задержка в очереди при значениях ρ_i, приближающихся к единице, т.е. в режиме перегрузки.

По поводу методических погрешностей, возникающих в данной сетевой модели, можно отметить следующее. Дж. Кингман доказал справедливость использования процесса диффузии для анализа режима перегрузки. Х. Кобаяши попытался эвристически применить формулы, характеризующие решение задачи математической физики «процесс диффузий с отражающим экраном при наличии течения», к расчету параметров стохастических сетей при режимах, далеких от перегрузки, и получил результаты, очень близко совпадающие с

результатами статистического моделирования. Е. Геленбе доказал, что применение формул диффузного процесса при расчете средней длины очереди дает погрешность, приблизительно равную $\varepsilon_i = c_{si}^2/2$, где c_{si}^2 – квадрат коэффициента вариации длительности обслуживания, т.е. эти формулы дают результаты с удовлетворительной погрешностью. Формула для оценки ρ_i^*, характерная для больших значений ρ_i (обычно при $0{,}5 \le \rho_i < 1$), имеет следующий вид:

$$\rho_i^* = \exp\left\{-\frac{2(1-\rho_i)}{c_{si}^2 + \rho_i c_{ai}^2}\right\}.$$

Благодаря стационарности потока возвращаемых элементов ресурсов интервал $t_{sij}^{(t)}$ в среднем начинается на середине отрезка, состоящего из $R_{ij} - 1$ интервалов выделения запрошенных элементов. Поэтому среднее время ожидания R_{ij} запрошенных элементов

$$t_{vij} = M\left[\max_i \left\{t_{wij}^{(t)}\right\}\right] = t_{qi} + \frac{R_{ij}-1}{2S_i} t_{ri}.$$

Вероятность того, что величина $tv_{ij}^{(t)}$ отлична от нуля, равна

$$\rho_{vij} = sign\{R_{ij}\}\left[1-(1-\rho_i^*)^{S_i}\right].$$

Функцию распределения времени ожидания выделения всех элементов i-го ресурса j-му процессу представим в виде

$$h_{ij}(t) = 1 - \rho_{vij} \exp\left\{-\frac{\rho_{vij}}{t_{vij}} t\right\}.$$

Погрешность при такой аппроксимации также невелика из-за наличия перегрузки и суперпозиции потоков заявок, проходящих через каждую очередь.

Нас интересует функция распределения $H_j(t)$ задержки процесса j:

$$\max_i \left\{ t_{vij}^{(t)} \right\}.$$

Обозначим $F_{ij}(t) = 1 - h_{ij}(t)$. Тогда справедливы следующие соотношения.

1. При $M=1$ и индексе $i=1$ имеет место формула

$$H_j(t)=1-F_{1j}(t).$$

Далее используем определение функции распределения для $M=1, 2, \ldots, i$.

2. При $M=2$ и индексах $i=1,2$ справедливо равенство

$$H_j(t)=1-[F_{1j}(t)+F_{2j}(t)]+F_{1j}(t)F_{2j}(t).$$

3. При $M=3$ и индексах $i=1, 2, 3$ получаем выражение

$$H_j(t)=1-[F_{1j}(t)+F_{2j}(t))+F_{3j}(t)]+[F_{1j}(t)F_{2j}(t))+F_{1j}(t)F_{3j}(t))+F_{2j}(t)F_{2j}(t))]-$$
$$-F_{1j}(t)F_{2j}(t)F_{3j}(t).$$

4. При произвольном M используем метод полной математической индукции и получаем следующее выражение $H_j(t)$ для любых $M\geq 1$:

$$H_j(t)=1-\sum_{n=1}^{M}\left\langle(-1)^{n+1}\left\{\sum_{i_1=1}^{M-n+1}F_{i_1j}(t)\sum_{i_2=i_1+1}^{M-n+2}F_{i_2j}(t)\ldots\sum_{i_n=i_{n-1}+1}^{M}F_{i_nj}(t)\right\}\right\rangle.$$

Причем существуют M ограничений на вспомогательные переменные i_n:

- $1\leq i_n\leq M$ при $n=1$;
- $i_{n-1}+1\leq i_n\leq M$ при $n=2, 3, \ldots, M$.

Эти ограничения влияют на количество знаков Σ в фигурных скобках.

Используя $H_j(t)$ и определения математического ожидания и дисперсии, применяя метод полной математической индукции, получим среднее время ожидания всех ресурсов $t_{нj}$ и его дисперсию $d_{нj}$:

$$t_{нj}=\sum_{n=1}^{M}\left\langle(-1)^{n+1}\left\{\sum_{i_1=1}^{M-n+1}\sum_{i_2=i_1+1}^{M-n+2}\ldots\sum_{i_n=i_{n-1}+1}^{M}\left[\frac{\alpha_{nj}}{\beta_{nj}}\right]\right\}\right\rangle;$$

$$d_{нj}=\sum_{n=1}^{M}\left\langle(-1)^{n+1}\left\{\sum_{i_1=1}^{M-n+1}\sum_{i_2=i_1+1}^{M-n+2}\ldots\sum_{i_n=i_{n-1}+1}^{M}\left[\frac{\alpha_{nj}(2-\alpha_{nj})}{\beta_{nj}^2}\right]\right\}\right\rangle,$$

где для разгрузки формул введены вспомогательные переменные

$$\alpha_{nj}=\prod_{m=1}^{n}\rho v_{i_mj}\ \text{ и }\ \beta_{nj}=\sum_{m=1}^{n}\frac{\rho v_{i_mj}}{tv_{i_mj}}.$$

Б. Конец итерации. Из временной диаграммы на рис. 1.10,*б* следует:

$$t_{ну} = t_{гу} + t_{ну} + t_{пу} + t_{эу}, \qquad d_{ну} = d_{нц} + d_{пу}.$$

Уточнив значения, известные в начале итерации с какой-то погрешностью, итерацию можно повторить, пока процесс не сойдется к результатам с приемлемой точностью. Таким образом, методом последовательных приближений можно получить интересующие нас параметры интервала активности. Расчеты выполняются методом последовательных приближений с использованием рекурсивной функции, написанной на языке C++.

Предлагаемый аппарат описания процессов в узлах стохастической сети распространяется на другую предметную область, более широкую по сравнению с классическими моделями*. Предложенные выше формулы предназначены только для предварительных оценок средних величин в установившемся режиме. Они не пригодны для расчетов в режиме переходного процесса: сложный переходный процесс обычно изучается с помощью имитационной модели.

Однако рассмотренные выше временные диаграммы обладают высокой универсальностью. С точки зрения Computer Science они полностью совпадают с временными диаграммами совокупности параллельных вычислительных процессов, взаимодействующих через общие ресурсы в памяти компьютера. Поэтому они составили идеологическую основу рассматриваемой далее системы Pilgrim.

Выводы

1. Основные составляющие технологии имитационного моделирования:
   - структурный анализ сложного процесса;
   - формализованное описание модели;
   - построение модели;
   - проведение экстремального эксперимента.

2. Метод статистических испытаний (Монте-Карло), основанный на использовании датчиков псевдослучайных величин при многочисленных реализациях вариантов поведения сложной экономиче-

---

* Клейнрок Л. Коммуникационные сети. Стохастические потоки и задержки сообщений. – М.: Наука, 1970. – 255 с.

ской системы (или сложного процесса) и аппарата проверки статистических гипотез, полезен для предварительного анализа последствий принимаемых решений. Являясь бесспорно мощным средством при исследовании систем, этот метод вынуждает разрабатывать моделирующую программу. Такое обстоятельство не позволяет применять в чистом виде метод Монте-Карло для решения экономических задач. С учетом отмеченных особенностей данный метод включается в состав многих моделирующих систем, но только для статистических испытаний с возможностью проверки гипотез.

3. Для реализации имитационных моделей экономических процессов необходимы датчики псевдослучайных величин и соответствующие моделирующие функции. Обобщенное распределение Эрланга и треугольное распределение дают возможность проведения экспресс-оценок причин и следствий возникновения групповых потоков, резких увеличений случайных задержек: в очередях, при получении ресурсов, при осуществлении денежных операций (платежей).

4. Нетрадиционная модель стохастической сети, где потоки транзактов нестационарны (стационарность можно наблюдать в отношении процессов восстановления ресурсов), неординарны (транзакт может породить группу транзактов) и есть последействие (процессы в узлах сети взаимозависимы через ресурсы), дает формулы, которые можно применить для приближенного расчета средних значений. Но временные диаграммы этой модели справедливы и во время переходных режимов, и при взаимном опосредованном влиянии процессов в узлах сети через общие параметры; поэтому их можно взять за основу при создании системы имитационного моделирования.

Вопросы для самопроверки

1. Что такое имитационное моделирование?
2. Зачем нужна имитационная модель?
3. Какова роль структурного анализа при проведении имитационного моделирования?
4. Для чего применяется имитационное моделирование экономических процессов?
5. Является ли метод Монте-Карло в сочетании с проверкой статистических гипотез имитационным моделированием?

6. Какие типовые задачи решаются средствами имитационного моделирования при управлении экономическими объектами?
7. Какими свойствами обладает распределение, равномерное на интервале?
8. Что такое нормальное распределение (дать экономическую трактовку)?
9. Как получается на практике экспоненциальное распределение (дать интерпретацию применительно к экономическим процессам)?
10. Для чего используется обобщенное распределение Эрланга?
11. Какие случайные процессы удобно описывать с помощью треугольного распределения?
12. Что такое интервал активности?
13. Какие процессы можно изобразить с помощью временных диаграмм интервалов активности?
14. Какие свойства имеет режим интерпретации модели?
15. Что дает режим компиляции модели?
16. Для чего нужна калибровка модели?
17. Каково назначение датчика случайных величин?
18. Действительно ли при моделировании экономических процессов программные датчики дают случайные числа? Если нет, то почему?
19. Нужно ли проверять статистические гипотезы?
20. Для чего используется критерий согласия χ^2 (хи-квадрат)?
21. В чем преимущество критерия Крамера–фон Мизеса по сравнению с критерием хи-квадрат?
22. Когда необходимо применять критерий Колмогорова–Смирнова?
23. Почему для анализа временных параметров сложного процесса трудно применить теорию стохастических сетей?
24. Какие достоинства имеют временные диаграммы интервалов активности нетрадиционной сетевой модели?
25. Учитывают ли формулы нетрадиционной сетевой модели наличие переходных процессов?

Глава 2

КОНЦЕПЦИЯ И ВОЗМОЖНОСТИ ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ МОДЕЛИРУЮЩЕЙ СИСТЕМЫ

2.1 ОСНОВНЫЕ ОБЪЕКТЫ МОДЕЛИ

Моделирующая система выполняет следующие основные функции:

1) предоставляет разработчику средства для формализованного описания дискретных компонентов, дисциплин выполнения различных работ, для задания структуры графа и привязки объектов модели к координатной сетке общего информационного поля;

2) осуществляет координацию событий, определение путей прохождения транзактов, изменение состояний узлов и передачу управления моделям непрерывных компонентов.

Такая система позволяет передавать результаты моделирования, используемые для принятия управленческих решений, из модели в базы данных экономической информационной системы (например, через интерфейс ODBC – Open Data Base Connectivity, если моделирование проводится в среде Windows) либо «подкачивать» актуализируемые во времени параметры в модель из баз данных.

В рассмотренной на рис. 1.10,*б* временной диаграмме длительность выполнения функции процессом $t_{пj}^{(t)}$ не зависит от типов ресурсов и характера их использования. Диаграмма обладает универсальностью, позволяет работать и с опосредованными ресурсами. Эта диаграмма использована при создании теоретических основ, концепции и алгоритмов специального программного инструментария – объектно-ориентированной системы имитационного моделирования Pilgrim, имеющей возможность агрегирования экономических объектов. Аналогичные диаграммы автоматически получаются при управлении модельным временем.

Существуют шесть основных понятий, на которых базируется концепция моделирующей системы.

1.*Граф модели.* Все процессы, независимо от количества уровней структурного анализа, объединяются в виде направленного графа. Пример изображения модели в виде многослойного иерархического графа, полученного при структурном анализе процесса, показан на рис. 2.1.

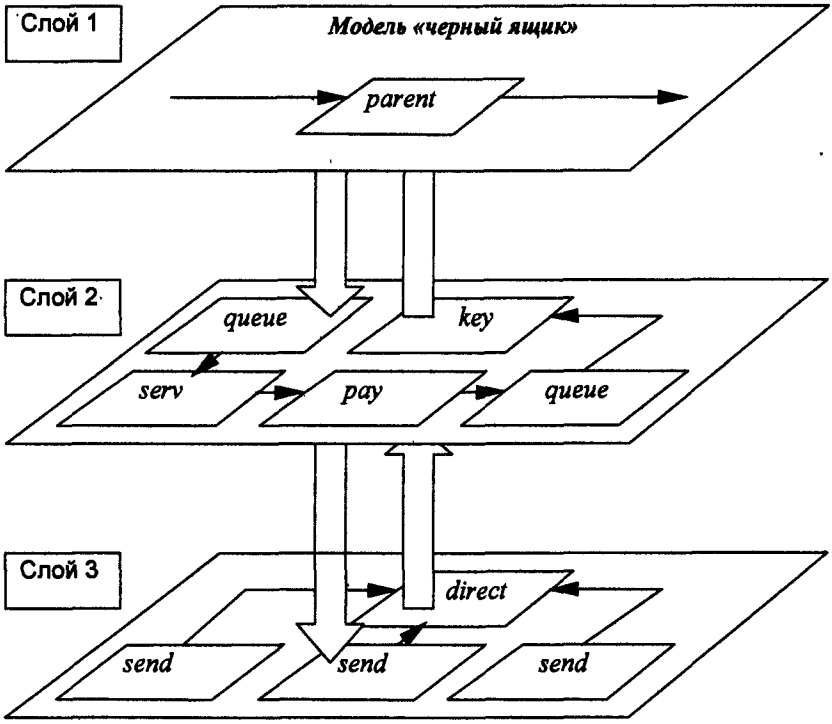


Рис. 2.1. Многослойный граф

2.*Транзакт* – это формальный запрос на какое-либо обслуживание. Транзакт в отличие от обычных заявок, которые рассматриваются при анализе моделей массового обслуживания, имеет набор динамически изменяющихся особых свойств и параметров. Пути миграции транзактов по графу стохастической сети определяются логикой функционирования компонентов модели в узлах сети.

Транзакт является динамической единицей любой модели, работающей под управлением имитатора.

Транзакт может выполнять следующие действия:

- порождать группы (семейства) других транзактов;
- поглощать другие транзакты конкретного семейства;
- захватывать ресурсы и использовать их некоторое время, а затем – освобождать;
- определять времена обслуживания, накапливать информацию о пройденном пути и иметь информацию о своем дальнейшем пути и о путях других транзактов.

Основные параметры транзактов:

- уникальный идентификатор транзакта;
- идентификатор (номер) семейства, к которому принадлежит транзакт;
- наборы различных ресурсов, которые транзакт может захватывать и использовать какое-то время;
- время жизни транзакта;
- приоритет – неотрицательное число; чем больше приоритет, тем приоритетнее транзакт (например, в очереди);
- параметры обслуживания в каком-либо обслуживающем устройстве (включая вероятностные характеристики).

Примеры транзактов:

- требование на перечисление денег;
- заказ на выполнение работ в фирме;
- телеграмма, поступающая на узел коммутации сообщений;
- сигнал о загрязнении какого-либо пункта местности;
- приказ руководства;
- покупатель в магазине;
- пассажир самолета;
- проба загрязненной почвы, ожидающая соответствующего анализа.

3. *Узлы* графа сети представляют собой центры обслуживания транзактов (но необязательно массового обслуживания). В узлах транзакты могут задерживаться, обслуживаться, порождать семейства новых транзактов, уничтожать другие транзакты. С точки зрения вычислительных процессов в каждом узле порождается независимый процесс. Вычислительные процессы выполняются параллельно и координируют друг друга. Они реализуются в едином модельном времени, в одном пространстве, учитывают временную, пространственную и финансовую динамику.

Нумерация и присвоение имен узлам стохастической сети производится разработчиком модели. Следует учесть, что транзакт всегда принадлежит одному из узлов графа и независимо от этого относится к определенной точке пространства или местности, координаты которой могут изменяться.

Примеры узлов:

- счет бухгалтерского учета;
- бухгалтерия;
- производственный (ремонтный) участок;
- генератор или размножитель транзактов;
- транспортное средство, которое перемещает ресурсы из одной точки пространства в другую;
- передвижная лаборатория;
- компьютерный центр коммутации сообщений (или пакетов сообщений);
- склад ресурсов.

4.*Событием* называется факт выхода из узла одного транзакта. События всегда происходят в определенные моменты времени. Они могут быть связаны и с точкой пространства. Интервалы между двумя соседними событиями в модели – это, как правило, случайные величины. Предположим, что в момент времени t произошло какое-то событие, а в момент времени $t+d$ должно произойти ближайшее следующее, но не обязательно в этом же узле. Если в модель включены непрерывные компоненты, то очевидно, что передать управление таким компонентам модели можно только на время в пределах интервала $(t, t+d)$.

Разработчик модели практически не может управлять событиями вручную (например, из программы). Поэтому функция управления событиями отдана специальной управляющей программе – координатору, автоматически внедряемому в состав модели.

5.*Ресурс* независимо от его природы в процессе моделирования может характеризоваться тремя общими параметрами: мощностью, остатком и дефицитом. Мощность ресурса – это максимальное число ресурсных единиц, которые можно использовать для различных целей. Остаток ресурса – число незанятых на данный момент единиц, которые можно использовать для удовлетворения транзактов. Дефицит ресурса – количество единиц ресурса в суммарном запросе транзактов, стоящих в очереди к данному ресурсу.

При решении задач динамического управления ресурсами можно выделить три основных типа: материальные, информационные и денежные ресурсы.

6.*Пространство* – географическое, декартова плоскость (можно ввести и другие). Узлы, транзакты и ресурсы могут быть привязаны к точкам пространства и мигрировать в нем.

Внутренняя реализация модели использует объектно-ориентированный способ представления экономических процессов. Транзакты, узлы, события и ресурсы – основные объекты имитационной модели. Взаимодействие таких объектов показано на рис. 2.2, где обозначены следующие моделирующие функции: ag, key, queue, dynam, proc, term, e1 и e2. Функциональное назначение этих и других средств моделирующей системы рассматривается в соответствующих разделах.

В различных моделирующих системах имеются разные способы представления узлов графа. Это связано с отличительными свойствами таких систем. Например, в системе GPSS узлы называются блоками; причем количество различных типов блоков более сотни, что затрудняет восприятие графа модели. В системе Pilgrim имеется всего 17 типов узлов, которые функционально перекрывают все возможности блоков GPSS и предоставляют дополнительные средства, которые в GPSS отсутствуют:

- возможность работы с непрерывными процессами;
- моделирование пространственной динамики;
- работу с ресурсами, представляющими собой деньги и материальные ценности, счета бухгалтерского учета, банковские счета.

Имеется система обозначений узлов, помогающая «читать» граф модели. Полный перечень изображений узлов Pilgrim приведен на рис. 2.3. Каждый узел имеет графическое обозначение, функциональное наименование, произвольный уникальный номер и произвольное название (например: наименование – serv, номер – 123, название – «Мастерская»). Пути транзактов обозначаются дугами – сплошными линиями со сплошной стрелкой на одном конце. Возможны информационные воздействия из одних узлов на другие; направления таких воздействий изображаются пунктирными линиями со сплошной стрелкой на одном конце. Если моделируются бухгалтерские проводки или перечисления денег, то пути денежных сумм со счета на счет показываются пунктирными линиями с штриховой стрелкой.

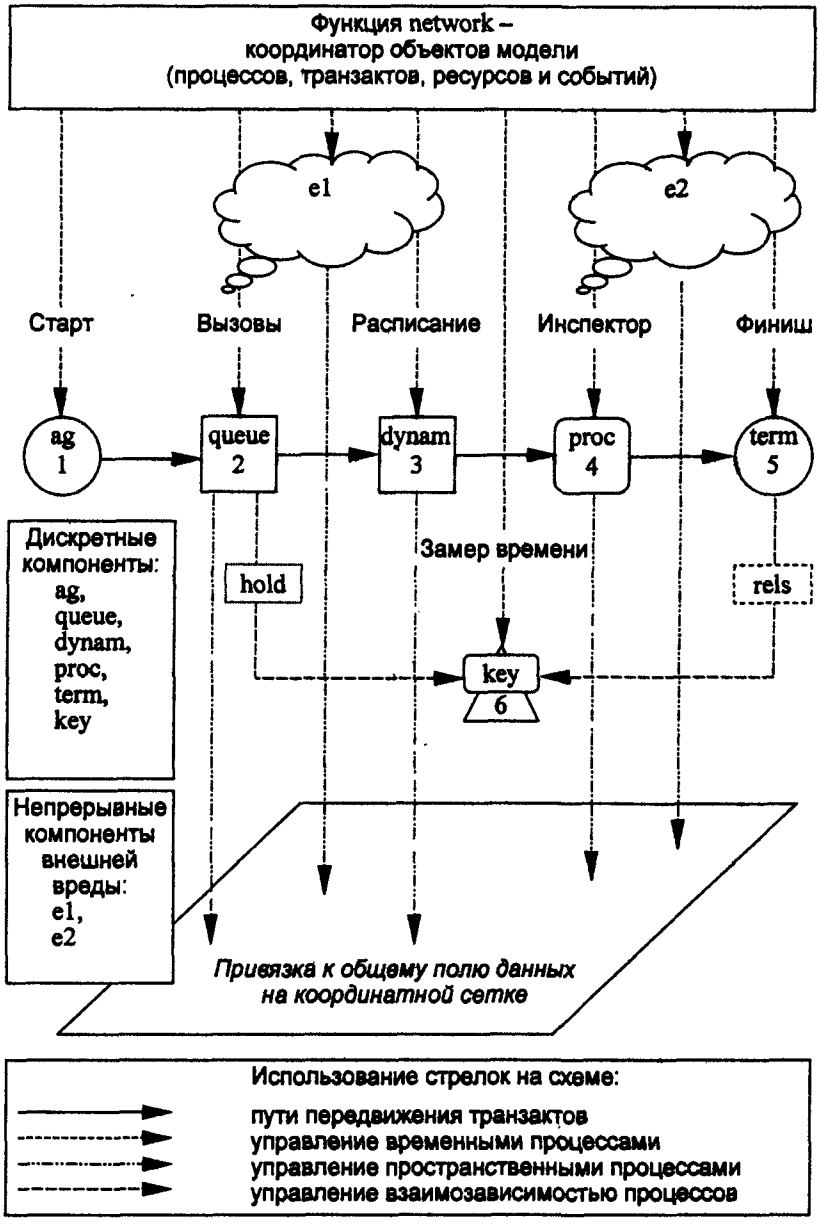


Рис. 2.2. Пример взаимодействия объектов имитационной модели

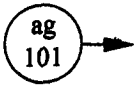
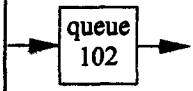
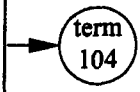
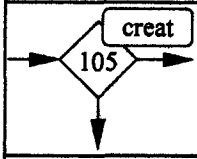
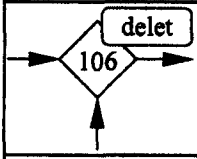
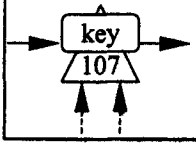
Изображение	Функциональное назначение узла
	Генератор транзактов (с бесконечной емкостью)
	Очередь (с приоритетами или без приоритетов)
	Узел обслуживания с многими параллельными каналами
	Терминатор, убирающий транзакты из модели
	Управляемый генератор (размножитель) транзактов
	Управляемый терминатор транзактов
	Клапан, перекрывающий путь транзактам

Рис. 2.3. Графические обозначения на схемах моделей (начало)

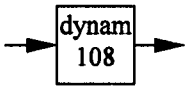
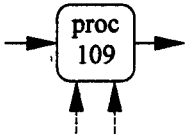
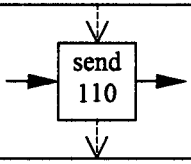
Изображение	Функциональное назначение узла
	Очередь с пространственно-зависимыми приоритетами
	Управляемый процесс (непрерывный или пространственный)
	Счет бухгалтерского учета (операция типа «проводка»)
	Распорядитель финансов (главный бухгалтер)
	Склад перемещаемых ресурсов
	Менеджер (или распорядитель) ресурсов
	Структурный узел финансово-хозяйственных платежей

Рис. 2.3. Продолжение

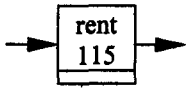
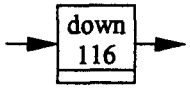
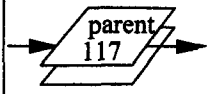
Изображение	Функциональное назначение узла
	Структурный узел выделения ресурсов
	Произвольный структурный узел
	Виртуальный структурный узел

Рис. 2.3. Окончание

Рассмотрим основные процессы и действия, которые могут выполняться в различных узлах модели. Подробные описания таких действий на уровне языка моделирования и соответствующие правила будут приведены в последующих разделах.

Генератор транзактов (с бесконечной емкостью) имеет наименование ag. Узлы-генераторы создают новые транзакты и передают их в другие узлы модели. Параметры генератора в случае необходимости можно изменить посредством информационного воздействия из другого узла с помощью сигнала cheg (здесь и далее *сигнал* – это специальная функция, выполненная транзактом, находящимся в одном узле, в отношении другого узла).

Очередь (с относительными приоритетами или без приоритетов) имеет наименование queue. Если приоритеты не учитываются, то транзакты упорядочиваются в очереди в порядке поступления. Когда приоритеты учитываются, транзакт попадает не в «хвост» очереди, а в конец своей приоритетной группы. Приоритетные группы упорядочиваются от «головы» очереди к «хвосту» в порядке уменьшения приоритета. Если транзакт попадает в очередь и не имеет своей при-

оритетной группы, то группа с таким приоритетом сразу возникнет: в ней будет один вновь поступивший транзакт.

Узел обслуживания с многими параллельными каналами имеет наименование serv. Обслуживание может быть в порядке поступления транзакта в освободившийся канал либо по правилу абсолютных приоритетов. Если такое правило задано и возникает ситуация, при которой в «голове» очереди на обслуживание находится транзакт с ненулевым приоритетом, все каналы заняты, причем в одном из каналов на обслуживании находится транзакт с более низким приоритетом, то выполняется следующее:

- обслуживание неприоритетного транзакта прерывается;
- неприоритетный транзакт удаляется из канала в стек временного хранения;
- канал занимает более приоритетный транзакт.

После освобождения канала прерванный транзакт возвращается в канал и дообслуживается столько времени, сколько оставалось на момент прерывания.

Реально возможны прерывания в прерываниях, когда на вход узла поступают все более приоритетные транзакты, а обслуживание происходит медленно. Поэтому глубина стека временного хранения не ограничена.

Терминатор, убирающий транзакты из модели, имеет наименование term. Транзакт, поступающий в терминатор, уничтожается. В терминаторе фиксируется время жизни транзакта.

Управляемый генератор (размножитель) транзактов имеет наименование creat. Он позволяет создавать новые семейства транзактов. Дело в том, что транзакты, создаваемые обычными генераторами ag, принадлежат семейству с номером 0 (номер семейства – один из параметров транзакта). Если возникает необходимость создать новое семейство с ненулевым номером, то соответствующее требование содержится в порождающем транзакте, поступающем на вход creat. Далее за нулевое модельное время происходит следующее:

- порождающий транзакт выходит из узла creat;
- из этого же узла выходит группа новых транзактов, принадлежащих семейству с заданным номером.

Управляемый терминатор транзактов имеет наименование delet. Иногда в модели возникает необходимость уничтожить (поглотить) заданное число транзактов, принадлежащих конкретному семейству. Требование на такое действие содержится в уничтожающем транзакте, поступающем на вход узла delet. Этот транзакт ждет

поступления в узел транзактов указанного семейства и уничтожает их; время жизни при этом регистрируется. После поглощения заданного количества транзактов (или по специальному сигналу freed из другого узла) уничтожающий транзакт покидает узел.

Клапан, перекрывающий путь транзактам, имеет наименование key. Если на клапан воздействовать сигналом hold из какого-либо узла, то клапан перекрывается и транзакты не могут через него проходить. Сигнал rels из другого узла открывает клапан. Транзакты проходят через этот узел без задержки – за нулевое модельное время. Часто этот узел используется для целей синхронизации или для моделирования работы с информационными ресурсами.

Очередь с пространственно-зависимыми приоритетами имеет наименование dynam. Транзакты, попадающие в такую очередь, привязаны к точкам пространства. Очередь обслуживается специальным узлом proc, работающим в режиме пространственных перемещений. Смысл обслуживания транзактов заключается в том, чтобы посетить все точки пространства, с которыми связаны (или из которых поступили) транзакты. При поступлении каждого нового транзакта, если он не единственный в очереди, происходит переупорядочение очереди таким образом, чтобы суммарный путь посещения точек был минимальным. Не следует считать, что при этом решается задача коммивояжера: для решения такой задачи в нулевой момент времени имеется вся информация о точках пространства. В данном же случае информация о новых точках поступает во время движения, когда некоторые точки уже посещены. Рассмотренное правило работы узла dynam в литературе называется «алгоритмом скорой помощи».

Управляемый процесс (*непрерывный или пространственный*) имеет наименование proc. Этот узел работает в трех взаимно исключающих режимах:

1) моделирование управляемого непрерывного процесса (например, процесса в химическом реакторе);

2) моделирование доступа к оперативным информационным ресурсам;

3) моделирование пространственных перемещений (например, вертолета или корабля по поверхности Земли).

В первом режиме после входа транзакта в узел запускается непрерывная модель, являющаяся функцией на языке C++, имеющая параметр «время». Такой моделью могут быть математическая формула или разностное уравнение, или другое. Эта модель синхрони-

зирована с другими узлами имитационной модели. Выполнением (активностью) непрерывной модели можно управлять из других узлов. По сигналу passiv транзакт вытесняется из узла proc в стек, после чего очередные элементарные интервалы времени d перестают поступать в непрерывную модель, а расчет по формуле или интегрирование разностного уравнения прекращается. Сигнал activ возвращает транзакт в узел и восстанавливает расчет по непрерывной модели. После выхода транзакта из узла выполнение непрерывной модели прекращается. Чистое время пребывания транзакта без учета вытеснения его в стек – это и время обслуживания транзакта, и время выполнения непрерывной модели.

Второй режим отличается от предыдущего только тем, что непрерывные процессы в узле не моделируются, так как они не нужны для моделирования доступа к информационным ресурсам.

В третьем режиме обслуживание каждого нового транзакта заключается в имитации перемещения узла proc в новую точку пространства, координаты которой – это параметры транзакта. Перемещение осуществляется с заданной скоростью.

Счет бухгалтерского учета (*операция типа «проводка»*) имеет наименование send. Транзакт, который входит в такой узел, является запросом на перечисление денег со счета на счет или на бухгалтерскую проводку. Правильность работы со счетами регулируется специальным узлом direct, который имитирует работу бухгалтерии. Транзакт, вошедший в узел send, далее может перейти только в узел direct. Если в узле send остаток денег достаточен, чтобы выполнить перечисление на другой счет (в другой узел send), то узел direct выполняет перечисление и выпускает обслуженный транзакт. В противном случае в узле send возникает дефицит средств и соответственно очередь необслуженных транзактов.

Распорядитель финансов (*«главный бухгалтер»*) имеет наименование direct. Он управляет работой узлов типа send. Причем для правильной работы модели достаточно одного узла direct; он обслужит все счета без нарушения логики модели. Однако не будет ошибкой, если каждый счет send будет обслуживаться отдельным бухгалтером direct. Поэтому, чтобы разделить статистику по разным участкам моделируемой бухгалтерии, можно использовать несколько узлов direct.

Если при обслуживании какого-либо счета возникает дефицит ресурсов, то в этом случае возможны различные дисциплины обслуживания: в хронологическом порядке поступления транзактов, по

жестко задаваемым приоритетам, по днамическим приоритетам (чем меньшую сумму нужно перечислить, тем приоритетнее транзакт).

Склад перемещаемых ресурсов имеет наименование attach. Это хранилище какого-то количества однотипного ресурса (например, гаражное хозяйство, имеющее 25 грузовиков). Единицы ресурсов в нужном количестве выделяются транзактам, поступающим в узел attach, если остаток (количество единиц, имеющихся в наличии) позволяет выполнить такое обслуживание. В противном случае возникает очередь необслуженных транзактов и соответственно дефицит ресурса. Транзакты, получившие ресурсы, вместе с ними мигрируют по графу во время выполнения модели и возвращают по мере необходимости разными способами: либо все единицы вместе, либо небольшими партиями, либо поштучно. На один и тот же склад можно обращаться несколько раз, не возвращая ранее полученные с этого склада ресурсы. Корректность работы склада обеспечивает менеджер – специальный узел manage.

Менеджер (или распорядитель) ресурсов имеет наименование manage. Он управляет работой узлов типа attach. Для правильной работы модели достаточно иметь один узел-менеджер; он обслужит все склады без нарушения логики модели. Однако не будет ошибкой, если склад будет обслуживаться отдельным менеджером. Поэтому, чтобы разделить статистику по разным складам перемещаемых ресурсов, можно использовать несколько узлов-менеджеров.

Если при обслуживании какого-либо склада возникает дефицит ресурсов, то в этом случае возможны те же дисциплины обслуживания, которые использовались в узле direct.

Структурный узел финансово-хозяйственных платежей имеет наименование pay. Он предназначен для упрощения той части имитационной модели, которая связана с работой бухгалтерии. Если позволить обращения к счетам бухгалтерского учета из всех частей модели, где возникают требования на проводки или перечисления, то граф станет запутанным. Описание условий прохождения транзактов по путям будет очень длинным и сложным. Условия – это логические выражения с многочисленными if, else, switch, case и while. Очень сложные условия увеличивают модель и порождают семантические ошибки, которые могут быть обнаружены только после длительного тестирования модели. Поэтому вся работа бухгалтерии собирается на одном структурном слое модели (слой 3, *см.* рис. 2.1). Обращения на этот слой в нужные входы-узлы происходят

с других слоев из узла pay автоматически, без графического объединения этих слоев с помощью дуг.

Структурный узел выделения ресурсов имеет наименование rent. Он применяется для упрощения графа и всей модели при работе со многими складами с различных уровней структурной схемы точно так же, как узел pay.

Произвольный структурный узел имеет наименование down. Он бывает необходим для упрощения очень сложного слоя модели, заключающегося в «развязывании» сложной запутанной схемы, находящейся на одном слое, по двум разным уровням (или слоям). Его польза точно такая же, как полезность узлов pay и rent.

Виртуальный структурный узел имеет наименование parent. Узел parent – мощное средство структурного анализа при создании модели. Узел виртуален. В тексте модели он отсутствует. Этот узел позволяет объединить некоторое множество любых узлов модели и поместить их на более низкий слой, оставив на исходном слое только графический значок parent. Работа с такими узлами возможна только в режиме CASE-технологии создания имитационных моделей при использовании графического конструктора. Такая технология будет рассмотрена в последующих разделах.

2.2 МОДЕЛИРОВАНИЕ РАБОТЫ С МАТЕРИАЛЬНЫМИ РЕСУРСАМИ

Материальные ресурсы подразделяются на две разновидности: неперемещаемые и перемещаемые. Неперемещаемый ресурс выделяется в определенном месте (как в реальности, так и в модели). Например, мастер в парикмахерской – это один элемент ресурса, выделяемый клиенту для обслуживания (стрижки и бритья). Этот элемент не может перемещаться вместе с клиентом (транзактом). После обслуживания одного клиента он либо приступит к обслуживанию следующего, если есть очередь, либо будет отдыхать.

Перемещаемый ресурс выделяется клиенту, после чего клиент использует его в других местах и возвращает только при отсутствии необходимости дальнейшего использования. Например, ресурс – это гараж; клиенту можно выделить три грузовика для использования в работах, проводимых в других местах (естественно, не в гараже).

Неперемещаемый ресурс (рис. 2.4) представляет собой «базу», на которой расположены (или к которой приписаны) какие-то ресурсные единицы; их можно использовать только на базе. Поток транзактов поступает в очередь к ресурсу.

Неперемещаемый ресурс имитируется в виде многоканального обслуживающего прибора. Каждой ресурсной единице соответствует один канал обслуживания. Канал принимает в себя транзакт (или захватывается транзактом) на время, которое может зависеть от атрибутов узла, транзакта и других параметров. Очередь в имитационной модели описывается в виде узла queue, а многоканальный обслуживающий прибор – узлом типа serv.

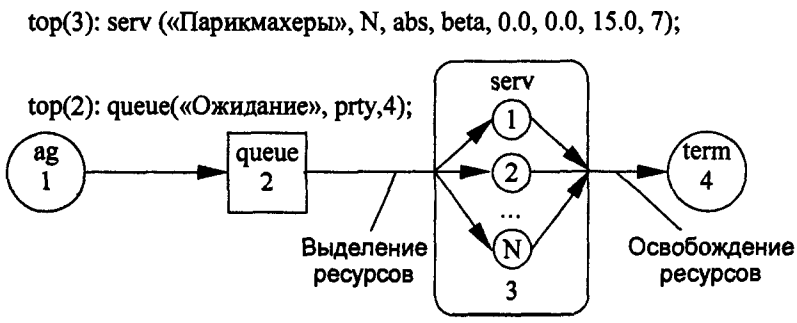


Рис. 2.4. Схема получения и освобождения транзактами элементов неперемещаемого ресурса – каналов узла serv: N – число каналов обслуживания (парикмахеров)

По истечении времени обслуживания канал (элемент ресурса) безусловно освобождается, а транзакт переходит в следующий узел. Очередь может быть как с приоритетами, так и без приоритетов. Каналы могут работать в режиме прерывания обслуживания менее приоритетных транзактов более приоритетными.

В моделях автоматически определяются задержка в очереди и загрузка неперемещаемого ресурса. Число свободных каналов в serv – это остаток ресурса, а количество транзактов в очереди queue – это дефицит ресурса. Мощность базируемого ресурса N – величина постоянная.

Перемещаемый ресурс представляет собой «склад» единиц ресурса, количество которых известно. Число таких складов не регла-

ментировано. Транзакт попадает в очередь к складу и требует выделения определенного числа единиц ресурса. Склад ресурсов описывается в имитационной модели в виде узла attach. В узле attach образуется очередь транзактов, которая может быть организована в хронологическом порядке или по приоритетным правилам:

- по заранее заданным приоритетам транзактов;
- при равенстве приоритетов транзактов происходит их дополнительное ранжирование – чем меньше транзакт запрашивает единиц, тем он более приоритетный.

Обслуживанием транзактов занимается узел типа «менеджер» – manage. Таких узлов в модели может быть несколько. Пример взаимодействия узлов attach и manage показан на рис. 2.5.

Рис. 2.5. Общая схема распределения мобильных материальных ресурсов

Обслуживание транзакта заключается в выделении ему требуемого числа единиц ресурса. Обслуженный транзакт проходит узел manage и «путешествует» с захваченными единицами по графу модели до тех пор, пока в соответствии с определенными условиями он не вернет все (или часть) единицы ресурса с помощью функции detach. Транзакт может несколько раз становиться в очередь к одному и тому же ресурсу, получая дополнительные единицы.

Существует интересная особенность при работе с перемещаемыми ресурсами: транзакт может отдать какие-либо единицы ресурса не только на тот склад, на котором он их получил, но и на другой.

При таком перераспределении (или «похищении») на этих двух складах произойдет изменение мощностей: на одном она уменьшится, а на другом – увеличится. Данная особенность реализуется с помощью сигнальной функции carryoff.

В моделях автоматически определяются задержка в очереди attach, загрузка ресурса, остаток и дефицит. Начальная мощность задается при инициализации модели функцией supply.

2.3 ИМИТАЦИЯ ИНФОРМАЦИОННЫХ РЕСУРСОВ

Информационные ресурсы – это необходимые сведения, оперативная информация (например, биржевая информация из сайтов Интернета), временно предоставляемые права на что-либо, документация и иные нематериальные ценности, без которых невозможно выполнение важной функции. Эти ресурсы подразделяются на две разновидности:

- стартовый информационный ресурс, без которого нельзя начинать выполнение функции (например, право или разрешение на ее выполнение, инструкция по сборке принципиально нового устройства);
- оперативный информационный ресурс, постоянно необходимый при выполнении функции (например, оперативная диспетчерская информация, отсутствие которой делает невозможной посадку самолета на аэродром).

Стартовый информационный ресурс дает возможность отправить заявку на выполнение какой-либо функции, т.е. поместить транзакт в очередь на обслуживание. На рис. 2.6 показана схема получения такого ресурса. Для выполнения основной функции нужны только два узла: первый (очередь queue) и седьмой – последний (обслуживающий процесс serv). Узлы 2 – 6 предназначены для имитации получения информации из N источников. Эти источники – каналы в узле обслуживания 5 (serv). В данном случае предполагается, что ко всем каналам или источникам информации доступ осуществляется через общую очередь 4 (queue). Если необходимо смоделировать отдельные механизмы доступа к каждому уникальному источнику информации, то данную схему нужно усложнить: это будет N очередей к N одноканальным узлам обслуживания.

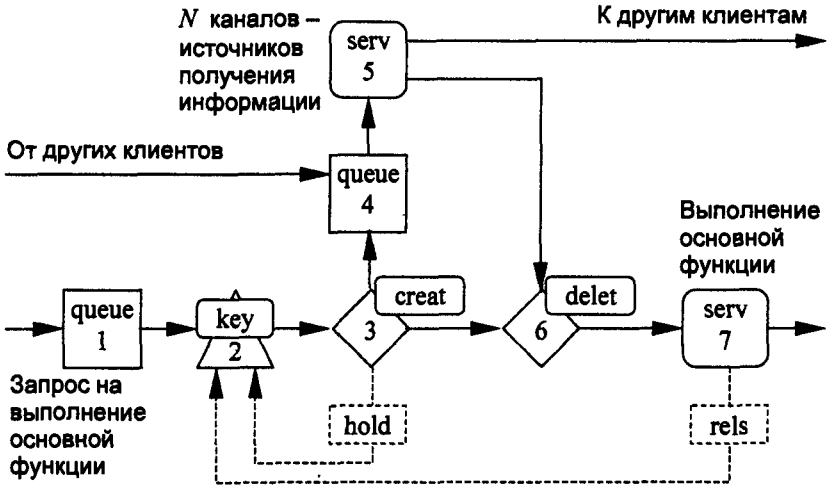


Рис. 2.6. Схема получения информационного ресурса для выполнения основной функции

Рассмотрим логику моделирования. Запросы на выполнение основной функции поступают в очередь с номером 1. Первый же запрос проходит через открытый клапан 2 (key) и далее поступает в управляемый генератор 3 (creat); при входе в него выполняется сигнальная функция hold, которая закрывает клапан, чтобы преградить путь следующим транзактам.

Узел creat создает новое семейство транзактов (от 1 до N). Каждый из них – это запрос, который поступает в очередь к источникам информации. Время получения информации (оно не равно нулю) можно сделать уникальным для каждого транзакта, поместив значение временного интервала в один из его параметров. После обслуживания каждый такой транзакт поступит в узел delet.

Основной (порождающий) транзакт за нулевое время проходит узел creat и поступает в узел delet, где он становится уничтожающим для только что созданного семейства. Если порожденный транзакт достигает узла delet, то это означает получение необходимой информации из очередного источника. Далее он становится ненужным и поглощается основным транзактом.

После получения всей необходимой информации все дополнительные транзакты поглощены, и основной транзакт переходит к

отработке основной функции в узле 7 (serv). При входе в этот узел выполняется сигнальная функция rels, открывающая клапан для прохождения других транзактов.

Следует отметить, что параллельно с обслуживанием запросов на выполнение основной функции в данной схеме моделируется обслуживание потока запросов от других клиентов. Такой поток обычно называется фоновым; если работать без приоритетов, то он приводит к увеличению задержек в очереди с номером 4.

Оперативный информационный ресурс может быть получен двумя способами:

- предварительно, вместе со стартовыми;
- во время выполнения транзактом основной функции.

На самом деле неважно, как получен ресурс; важно иметь доступ к этому ресурсу по возможности постоянно, так как прекращение доступа повлечет за собой приостановку выполнения основной функции. Моделирование механизма таких приостановок показано на рис. 2.7.

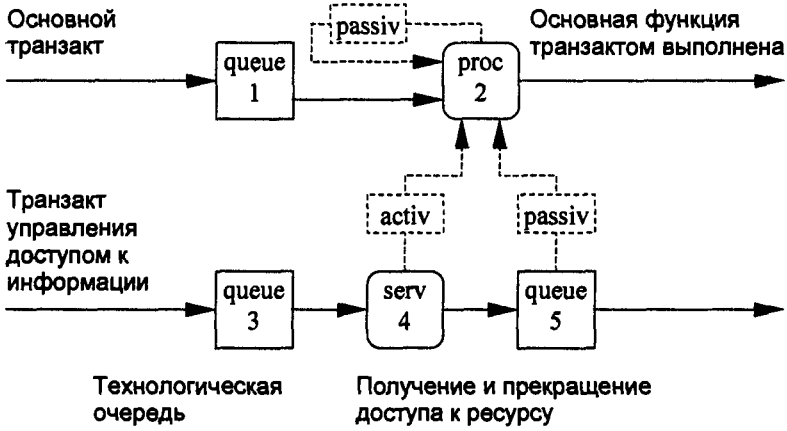


Рис. 2.7. Схема выполнения функции основным транзактом при наличии доступа к информации

Основной транзакт – это запрос на выполнение основной функции. Он поступает в очередь queue с номером 1. Выполнение основной функции имитируется в данном случае не узлом обслуживания serv, а с помощью узла-процесса 2 (proc). Узел proc отрабатывает

только время обслуживания, и непрерывный компонент ему не нужен.

Доступ к оперативной информации осуществляется специальной службой, которая моделируется с помощью узлов 3, 4 и 5. Для определенности считаем, что в узел proc поступает основной транзакт, который сразу попадает в пассивное состояние и не обслуживается, если нет доступа к информации. Управление доступом выполняет другой транзакт, который поступает из очереди 3 (queue) в узел обслуживания 4 (serv). При входе в этот узел управляющий транзакт разрешает доступ: он посылает сигнал activ, по которому основной транзакт переходит в активное состояние, и в узле proc выполняется работа по его обслуживанию. После пребывания управляющего транзакта в узле serv в течение определенного времени – времени разрешенного (или оплаченного) доступа к оперативной информации, он поступит в узел 5 и выполнит сигнальную функцию перевода узла 2 в пассивное состояние. Если время разрешенного доступа выбрано так, что оно не меньше длительности выполнения основной функции, а моменты входов управляющего транзакта в узел 4 и основного транзакта в узел 2 совпадают, то выполнение основной функции произойдет без прерываний. В противном случае возможны прерывания основной функции, и она будет выполняться за большее время. Так моделируется влияние наличия или отсутствия оперативной информации на работоспособность, например, диспетчерских служб или консалтинговых агентств.

2.4 ДЕНЕЖНЫЕ РЕСУРСЫ

Денежный ресурс представляет собой «емкость», в которой содержится определенное количество ресурса, измеряемого числом с плавающей точкой. Обычно эту емкость отождествляют со счетом бухгалтерского или банковского учета. Этот счет описывается с помощью узла типа send (пересылка). В узле send образуется очередь транзактов, в которых содержится запрос на перевод денежных средств с данного счета send на какой-либо другой. Эта очередь может быть организована по приоритетному принципу: чем меньше денег требует транзакт перевести с данного счета, тем он приоритетнее. Можно устанавливать приоритеты и по-другому, например по такой приоритетной таблице: сначала налоги, затем – зарплата, а после этого – все остальные платежи.

Непосредственно проводками средств с одного счета на другой занимается узел типа direct (рис. 2.8). Этот узел имитирует работу бухгалтера. Достаточно иметь один узел direct на всю модель (однако их можно сделать и несколько, имитируя разделение функций при работе бухгалтерии). Обслуживание в узле direct заключается в следующем: если запрос транзакта может быть удовлетворен, то транзакт проходит через узел direct, перечисляя требуемую сумму с данного счета-узла send на другой за нулевое модельное время, уменьшая остаток на счете. Начальные значения средств на некоторых счетах задаются при инициализации модели с помощью функции assign. Вид денежной единицы не имеет значения. Например, целая часть суммы – это рубли, а два знака после десятичной точки – это копейки. В модели автоматически определяются задержка в очереди send, остаток (положительное сальдо) и дефицит (отрицательное сальдо).

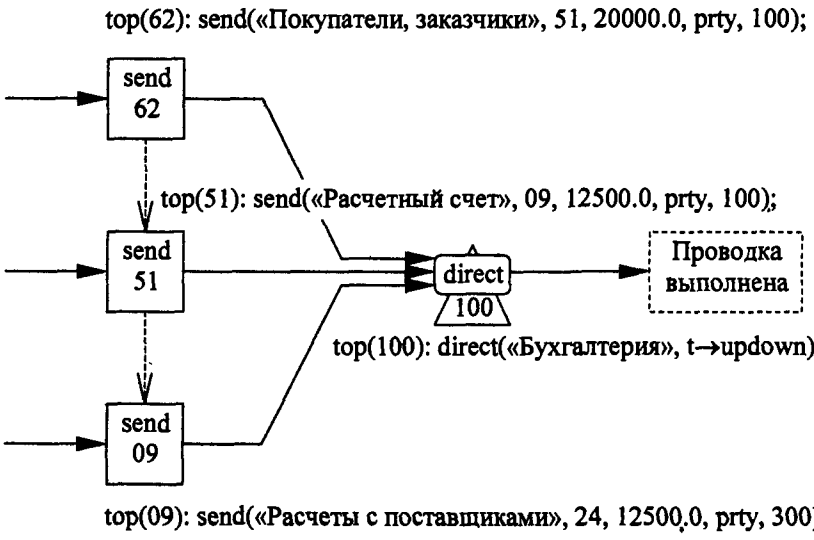


Рис. 2.8. Схема распределения денежных ресурсов (бухгалтерские проводки)

Динамика задержек платежей при расчетах с поставщиками при анализе одного из возможных вариантов организации бизнес-процесса предприятия с помощью имитационной модели представ-

лена на рис. 2.9 (график получен автоматически на мониторе компьютера средствами Pilgrim). График показывает, что исследуемый вариант бизнес-процесса неприемлем, так как на конец I квартала модель выдает прогноз существенных задержек платежей, которые создадут риск банкротства. Здесь же отражена полученная из модели вероятность того, что банк не предоставит ссуду: она равна 0,777.

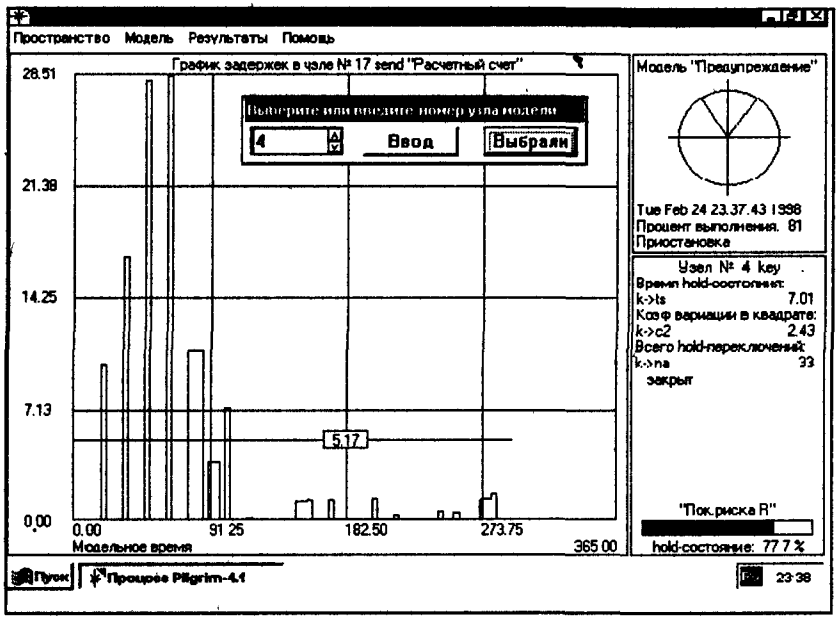


Рис. 2.9. Окно оценки возможных задержек платежей с помощью модели

2.5 МОДЕЛИРОВАНИЕ ПРОСТРАНСТВЕННОЙ ДИНАМИКИ

Поведение исследуемой системы в пространстве моделируется с помощью узлов типа creat, delet, proc и dynam.

Логика узла creat такова: он получает координаты порождающего транзакта, в результате происходит имитация перемещения в пространстве. Узел delet получает координаты каждого уничтожаемого транзакта, т.е. он перемещается по координатной сетке в процессе нахождения в нем поглощающего транзакта.

Для моделирования пространственных перемещений, связанных с поставкой товаров во многие пункты местности, используется узел proc.

Пример 2.1 (рис. 2.10). Часть модели, состоящая из узлов queue и proc, предназначена для моделирования движения транспортного средства. Имеется специальный массив, предварительно загруженный координатами M пунктов региона из файла или базы данных средствами моделирующей системы. На вход этой части модели в разные моменты времени поступают M транзактов, причем каждый из них «читает» в свои внутренние параметры координаты очередной точки. Узел proc, моделирующий транспортное средство, в качестве одного из параметров получает скорость перемещения, которая может быть изменяемой. При поступлении каждого следующего транзакта из очереди в узел proc с помощью функции geoway автоматически определяется расстояние по поверхности Земли от предыдущего пункта до следующего. Время обслуживания транзакта – это расстояние, деленное на скорость. По истечении времени обслуживания узел получает новые координаты того пункта, в который он попал. Порядок посещения пунктов узлом proc – хронологический (в порядке поступления транзактов в очередь) или в соответствии с приоритетами транзактов.

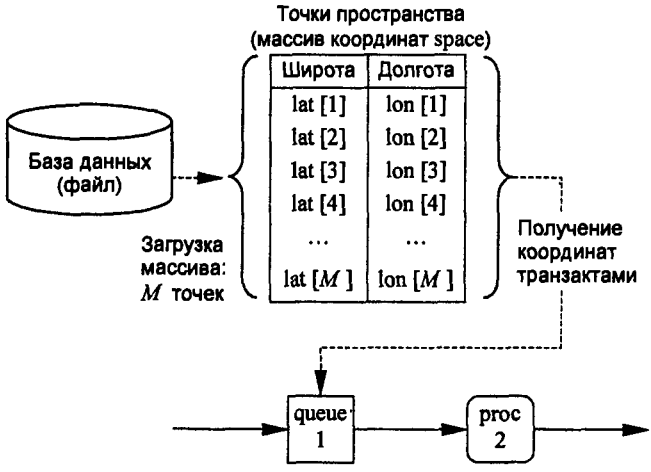


Рис. 2.10. Имитация перемещения в пространстве по координатам вызывающих транзактов: queue – очередь транзактов из точек пространства; proc – имитация перемещения (транспортировки)

Узел dynam предназначен для моделирования управляемой очереди обслуживания транзактов с динамическими пространственно-зависимыми приоритетами.

Задача оптимального расписания для обслуживания транзактов с пространственно-зависимыми приоритетами может возникнуть, например, при моделировании следующих сложных процессов и объектов:

- участка гибкого автоматизированного производства с роботизированными тележками, путешествующими по цеху;
- местности, подверженной какому-то бедствию, в процессе ее обследования специальной командой на вертолете и др.

Пример 2.2. Использование вертолета «скорой помощи». Допустим, во время дежурства на вертолет поступают радиограммы из различных пунктов. Это радиограммы вызова к больному. Если во время работы скопилось несколько радиограмм, то путь вертолета должен быть минимальным, чтобы уменьшить среднее время ожидания больного. Более того, если полет из одного пункта к другому уже начался, то при появлении новой радиограммы из пункта, находящегося не очень далеко от курса вертолета, маршрут может быть изменен в пользу такого вызова, если суммарный путь будет минимален.

В такой интерпретации узел proc – это вертолет, радиограммы – транзакты, имеющие координаты населенных пунктов, а узел dynam – оптимизатор курса, специальная программа в бортовом компьютере вертолета (соединение узлов показано на рис. 2.11,*а*). Узел dynam управляет очередью транзактов по правилу динамических пространственно-зависимых приоритетов. Порядок обслуживания транзактов в этой очереди пересматривается каждый раз при поступлении в нее нового транзакта или при выходе транзакта из «головы» очереди в узел proc.

Узел dynam всегда «заглядывает» в узел proc и анализирует, не следует ли прервать обслуживание находящегося в нем транзакта. Если такое решение будет принято, то вычисляется местонахождение текущей точки пространства, в которой находится узел proc; далее dynam «извлекает» из узла proc транзакт обратно в свою очередь и посылает в него транзакт, более «выгодный» относительно оптимизации маршрута.

В системе Pilgrim реализован быстродействующий алгоритм оптимизации динамического расписания – правила построения динамической очереди dynam для обслуживания транзактов. Рассмотрим упрощенное описание этого алгоритма.

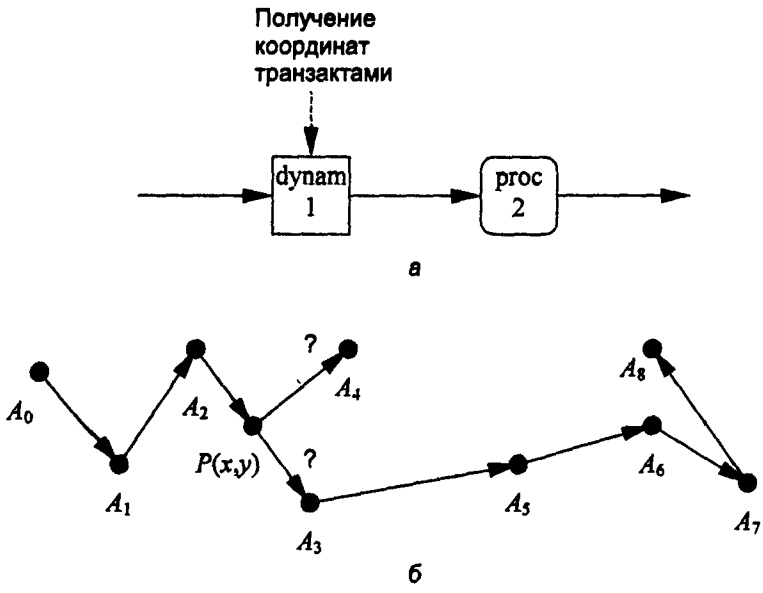


Рис. 2.11. Пример очереди с динамическими приоритетами для оптимизации перемещений: *а* – применение узла dynam; *б* – альтернативы при определении маршрута

Предположим, что каждый населенный пункт может выдать только один сигнал (или сообщение) о необходимости его срочного обследования. Это допущение не слишком грубое, так как можно все последующие сигналы из одного пункта считать дополнением первого.

Введем обозначения:

A_i – название населенного пункта;

i – порядковый номер во времени сигнала из пункта A_i, т.е. номер населенного пункта, присваиваемый в хронологическом порядке;

A_0 – аэродром базирования вертолета, откуда он вылетает по первому вызову;

$P(x,y)$ – точка с координатами x и y, в которой вертолет находится в момент времени t;

L_{ij} – расстояние от A_i до A_j по прямой линии.

Допустим, что из пунктов A_1 и A_2 поступили сигналы – вызовы к больному (рис. 2.11,*б*). По этим сигналам вертолет последовательно

посетил оба пункта. Предположим, что во время обследования пункта A_2 пришел сигнал из пункта A_3, после чего вертолет направился к A_3. Далее в точке $P(x, y)$ поступил сигнал из A_4, а затем сигналы придут из A_5, A_6, A_7 и A_8. Обратим внимание на два обстоятельства:

1) в точке $P(x, y)$ необходимо принять решение либо о продолжении полета к намеченному пункту (т.е. к A_3), либо изменить маршрут в целях сокращения суммарного пути и лететь к новому пункту (т.е. к A_4);

2) если уже имеются в наличии и другие сигналы (т.е. из $A_5 - A_8$), то дополнительно необходимо установить порядок их посещения с целью минимизации длины пути, т.е. нарушить хронологический порядок посещения. В этом случае число анализируемых вариантов – это число перестановок сигналов в группе, объем которой меняется случайным образом.

Предположим, что удалось учесть оба обстоятельства, принять оптимальное решение и найти минимум длины пути L_{min}. Обозначим точку $P(x, y)$ как A_{i_0}. Поскольку начиная с момента поступления сигнала из A_4 в точке A_{i_0} до момента поступления сигнала из A_8 в очереди находятся шесть транзактов, справедливо следующее выражение:

$$L_{\min} = \sum_{n=0}^{6} L_{i_{n-1}, j_n},$$

где i, j – номера сигналов;

L_{i_{n-1}, j_n} – расстояние по прямой от пункта A_i до пункта A_j при условии, что в группе сигналов сообщениям из этих пунктов присвоены номера $n-1$ и n после оптимизации (посещение будет производиться в порядке возрастания n, причем необязательно, чтобы $j = i+1$).

Оптимизация достигается за счет изменения порядка посещения. Легко показать, что если устанавливать оптимальные порядки в случайных группах и всякий раз получать L_{min}, то после завершения всего обследования весь пройденный путь не будет оптимальным по следующим двум причинам.

1. В полученный путь, длина которого L_{min}, не входит уже пройденный путь $A_0 \rightarrow A_1 \rightarrow A_2 \rightarrow P(x, y)$. Поэтому нельзя утверждать, что выражение

$$L_{0,1} + L_{1,2} + L_{2,P} + L_{\min}$$

определяет возможный минимум, так как если сигналы идут редко и не образуют случайных групп, то и нечего оптимизировать: вертолет успевает посещать пункты в хронологическом порядке по неоптимальному пути.

2. Через некоторое время появятся сигналы из новых пунктов, пока неизвестных (координаты их тоже заранее неизвестны). Поэтому можно сделать вывод о том, что оперативное расписание, составляемое каждый раз при получении нового сигнала в момент времени t_k из пункта A_k, дает только локальный минимум в момент времени t_k для случайного набора в очереди, состоящего из транзактов N_k пунктов. Причем минимизации подлежит выражение

$$L_q = \sum_{n=1}^{N_k} S_{i_{n-1}, j_n}. \qquad (2.1)$$

Для минимизации функционала (2.1) можно применять различные методы, например метод динамического программирования, дающий строгий минимум L_q. Однако строгий минимум не гарантирует лучшего решения для всего маршрутного пути.

Эффективность методов составления маршрутных расписаний можно оценить либо после завершения полетов, либо с помощью проигрывания на имитационной модели большого числа различных вариантов. Второй способ предпочтительнее, если имеются средства для быстрой подготовки и проведения таких имитационных экспериментов (их предоставляет Pilgrim).

В любом случае для оценки эффективности метода необходимо его сравнивать с простейшим методом, работающим по правилу fifo (first in – first out) – «первым пришел – первым обслужен», применение которого всегда дает для выражения (2.1) следующие соотношения:

$$i = k - N_k + n, \quad n = 1, ..., N_k;$$
$$j = k - N_k + 1 + n, \quad n = 0, ..., N_k,$$

причем $A_{(k-N_k)_0}$ – это точка $P(x, y)$.

Весь путь, пройденный при использовании этого метода, всегда определяется по формуле

$$L_{\text{fifo}} = \sum_{n=1}^{M} L_{i-1, i},$$

где M – общее число пунктов, из которых поступили сигналы до и во время облета района вертолетом «скорой помощи», т.е. до завершения всего полета в момент времени T.

Рассмотрим метод построения оперативного расписания. Будем использовать для описания метода следующую терминологию. Район, в котором возникают сигналы-вызовы – это генератор транзактов (сигналов, сообщений, заявок), который имеет заранее неизвестную конечную мощность M.

Транзакт в нашем случае – это формализованный сигнал, который может иметь такие параметры:

i – хронологический номер транзакта и, соответственно, пункта;

t_i – момент времени получения сигнала из пункта i экипажем вертолета;

x_i, y_i – координаты пункта на местности, из которого пришел сигнал с хронологическим номером i.

Далее считаем, что транзакты поступают в очередь для обслуживания. Обслуживание транзакта i будет заключаться в полете к пункту A_i за время, которое зависит от порядка посещения (от двух рассмотренных ранее обстоятельств).

Введем в рассмотрение приоритет транзакта p_i: если приоритеты всех транзактов равны, то порядок транзактов в очереди (в случайной группе сигналов величиной N_k) определен порядковым номером i (первым пришел – первым обслужен).

Чем больше i, тем ближе к «хвосту» очереди находится транзакт (т.е. всегда вновь поступивший транзакт будет последним). Если же приоритеты всех транзактов различны, то очередь всегда упорядочивается специальными диспетчерскими средствами от «головы» к «хвосту» в порядке уменьшения p_i, а при совпадении приоритетов двух расположенных рядом в очереди транзактов ближе к хвосту будет находиться тот, у которого больше номер i.

Кроме того, введем в рассмотрение текущую длину пройденного пути L_g, которая до начала полета равна 0, и перейдем к описанию метода составления оперативного расписания, который реализуется автоматически.

Пополнение очереди. Допустим, что вертолет находится в точке $P(x, y)$, а в очереди находятся N_k транзактов. Если A_{j_1} – первый в очереди транзакт, относящийся к пункту j, то длительность дообслуживания очередного транзакта из пункта i определяется как

$$t_{s_{0,1}} = \frac{1}{v} L_{i_0, j_1},$$

где v – скорость вертолета, которую можно изменять после подлета к каждому пункту.

Каждый последующий в очереди транзакт будет обслуживаться за время

$$t_{s_{n-1,n}} = \frac{1}{v} L_{i_{n-1}, j_n},$$

где i, j – номера соответствующих транзактов;
n – номер транзакта (или место) в очереди (n=1,...,N_k).

Очередь упорядочена по приоритетам, которые присвоены так, чтобы расставить транзакты для получения минимального пути между N_k пунктами, попавшими в случайную группу.

Допустим, что в очередь поступает транзакт из пункта A_{k+1}. Тогда увеличим N_k: N_k=N_k+1. Далее проделаем N_k раз одну и ту же процедуру: условно поставим новый транзакт в очередь перед каждым, расположенным на месте n (n =1,...,N_k), и рассчитаем путь по формуле (2.1) для нового набора транзактов, запоминая расстановку и результат после каждого расчета. Далее выбираем вариант с минимальным результатом и принимаем следующие решения (в зависимости от условий):

1) если наилучшим оказался вариант, когда транзакт нужно поставить на место с номером n, то ему присваивается приоритет транзакта, расположенного на месте n +1, увеличенный на 1, а для транзактов, расположенных на местах 1, ... , n–1, приоритеты просто увеличиваются на 1;

2) если n=N_k, то это означает, что транзакт нужно поставить на последнее место;

3) если n=1, то необходимо в точке $P(x, y)$ изменить маршрут и лететь к новому пункту A_{k+1}, так как он попал на первое место в очереди.

Далее увеличиваем величину пройденного пути L_g на расстояние $L_{i_0, P}$. После этого при появлении нового транзакта опять можно повторить рассмотренную процедуру пополнения очереди (заметим, что A_{i_0} – это всегда последний обследованный пункт).

Освобождение очереди. После того как вертолет долетит до очередного пункта, длину пройденного пути L_g необходимо увеличить на расстояние L_{P, i_1}.

Если в очереди есть хотя бы один транзакт, то при взлете после обследования очередного пункта следующий населенный пункт определяется транзактом j, который находится на первом месте в очереди.

После взлета этот транзакт исключается из очереди (т.е. считается, что он попал на место с номером 0), а размер группы уменьшается на 1: $N_k=N_k-1$.

Если же очередь пуста, то в зависимости от ситуации можно, например, перейти к патрулированию местности по заранее намеченному маршруту.

Для определения эффективности данного метода проводилось имитационное моделирование большого числа возможных ситуаций в Брянской области с помощью Pilgrim-моделей. Была использована реальная информация из банка данных по Гордеевскому, Клинцовскому, Красногорскому, Новозыбковскому и Злынковскому районам. Максимальный выигрыш от применения этого алгоритма в реальной работе вертолетных групп (т.е. не только при использовании для реализации функции dynam) находится в пределах 30–50%. Причем, чем меньше средний интервал между заявками и чем больше густота населенных пунктов, тем больше выигрыш.

В примере 2.2 рассмотрена реальная ситуация, при которой сигналы поступают в процессе полета. Однако возможен крайний случай, когда по некоторым причинам уже в момент вылета вертолета известны все M пунктов. Такая вырожденная ситуация порождает задачу коммивояжера, а рассмотренный метод превращается в разновидность известного алгоритма «ближайшего непосещенного города». Он позволяет сократить путь, но не дает глобального минимума: расписания, составляемые с его помощью, хуже оптимального относительно длины маршрутного пути. Более удачный способ решения задачи коммивояжера рассмотрен в главе 6.

2.6 УПРАВЛЕНИЕ МОДЕЛЬНЫМ ВРЕМЕНЕМ

События модели происходят в некотором модельном времени. *Модельное время* – это виртуальное время, в котором автоматически упорядочиваются все события, причем не обязательно пропорционально реальному времени, в котором развивается моделируемый процесс. Например:

- реальное время развития процесса – 3 года;
- модель процесса, охватывающая эти 3 года, выполняется на компьютере за 1 с;
- все события при выполнении модели выстроены в нужном порядке, и все статистические данные в результате ее выполнения замерены.

Масштаб времени – это число, которое задает длительность моделирования одной единицы модельного времени, пересчитанной в секунды, в секундах астрономического реального времени при выполнении модели. Относительный масштаб времени – это дробь, показывающая, сколько единиц модельного времени помещается в одной единице процессорного времени при выполнении модели в компьютере.

Можно выделить четыре разновидности масштаба времени:

1.*Реальный* масштаб времени – вводится значение выбранной единицы измерения модельного времени, выраженное в секундах. Например, если в качестве единицы модельного времени выбран 1 ч, а в качестве масштаба задать число 3600, то модель будет выполняться со скоростью реального процесса, а интервалы времени между событиями в модели будут равны интервалам времени между реальными событиями в моделируемом объекте (с точностью до поправок на погрешности при задании исходных данных). Относительный масштаб в этом случае равен 1:1.

2.*Максимально ускоренный* масштаб времени – задается число 0. В этом случае время моделирования определяется чисто процессорным временем выполнения модели. Например, если в модели произошли три события, причем длительность модельного времени между первым и вторым событиями составляет 1 мин, а между вторым и третьим интервал модельного времени равен 24 ч, то в компьютере соответствующие интервалы астрономического времени – это длительность выполнения управляющих программ имитатора, т.е. оба интервала приблизительно равны. Они зависят от используемого процессора ЭВМ и могут измеряться малыми долями секунды. Это обстоятельство позволяет достигнуть максимального быстродействия модели и автоматически исключать из процесса моделирования непроизводительные отрезки модельного времени (например, в ночное время фирма не работает). Относительный масштаб в этом случае практически трудно определить.

3.*Пропорционально ускоренный* масштаб времени – вводится значение выбранной единицы измерения модельного времени, вы-

раженное в секундах. Причем это значение меньше выбранной единицы. Например, если в качестве единицы модельного времени выбран 1 ч, а в качестве масштаба задать число 0,1, то модель будет выполняться быстрее реального процесса. Причем 1 ч реального процесса будет моделироваться в ЭВМ в течение 0,1 с (с учетом погрешностей), т.е. примерно в 36 000 раз быстрее. Относительный масштаб равен 1:36 000.

4. *Замедленный* масштаб времени – вводится значение выбранной единицы измерения модельного времени, выраженное в секундах. Причем это значение меньше выбранной единицы. Например, если в качестве единицы модельного времени выбран 1 ч, а в качестве масштаба задать число 7 200, то модель будет выполняться медленнее реального процесса. Причем 1 ч реального процесса будет моделироваться в ЭВМ в течение 2 ч, т.е. примерно в 2 раза медленнее. Относительный масштаб равен 2:1. Замедленный масштаб не представляет интереса для проведения исследований с моделями. Однако замедленная работа необходима при исследовании самого имитатора и характеристик его координатора (например, при калибровке общего модельного таймера).

Механизм планирования событий и модельный таймер. Рассмотрим механизм планирования событий. В процессе моделирования образуются управляющие структуры данных. На фазе инициализации для каждого узла в памяти ЭВМ выделяется блок управления узлом kcb.

Эти блоки уничтожаются при завершении моделирования. Если в процессе прогона модели появляется новый транзакт, то на все время его существования образуется блок управления транзактом tcb. При входе транзакта в узел возникает блок управления событием ecb, который уничтожается после выхода транзакта из этого узла. Если транзакт захватывает какое-то количество единиц ресурса определенного типа, то к нему присоединяется блок управления ресурсом rcb с идентификатором этого ресурса; в этом блоке отмечается используемое количество единиц. Если ресурс полностью освобожден, то rcb уничтожается.

В действительности tcb, ecb и rcb не уничтожаются; они переводятся в соответствующие списки отработанных структур, откуда будут извлечены вновь при возникновении новых транзактов и событий. По окончании одного прогона модели все kcb, tcb, ecb и rcb, включая отработанные, будут уничтожены специальной программой – «мусорщиком».

Число управляющих структур tcb и ecb в любой конкретный момент случайно. Конфигурация взаимосвязей kcb, tcb и ecb также изменяется от события к событию. Алгоритм планирования интервала времени (t, $t+d$) между двумя ближайшими событиями можно пояснить на примере фрагмента конкретной конфигурации (рис. 2.12).

Транзакт, вошедший в узел, получает (или уже имеет) информацию о том, в какой следующий узел он должен перейти. При этом образуется ecb, в котором проставляется значение интервала времени задержки транзакта в этом узле (переменная ct типа float, $ct \geq 0$). Все ecb планируемых событий сцеплены в список, где они упорядочены по возрастанию величины ct.

События, у которых ct=0, готовы к своему завершению, если, кроме этого, выполнены и другие условия: например, если узел, в который транзакт должен перейти (узел-приемник), не занят.

В узле типа queue (очередь) может произойти только одно событие после того, как все транзакты покинут очередь. При входе *нового транзакта в пустую очередь будет образован новый ecb.*

Узел типа serv (обслуживающий прибор) может принять в себя столько транзактов, сколько он имеет обслуживающих каналов. Если такому узлу дать возможность приоритетного обслуживания, то более приоритетные транзакты будут входить в каналы, вытесняя менее приоритетные в стек (пару ecb→tcb), который динамически создается для такого узла. После обслуживания приоритетного транзакта пара ecb→tcb возвращается в список планируемых ecb с соблюдением его упорядоченности.

Очередь на рис. 2.12 содержит три транзакта (ecb 2 и tcb 1, tcb 2, tcb 3), а обслуживающий прибор, имеющий два канала (nc=2), содержит два транзакта с приоритетами pr=4 и pr=3, в то время как в стек вытеснены три транзакта с прерыванием их обслуживания (pr=2, pr=2 и pr=0).

Программный имитатор имеет в своем составе специальную функцию – координатор network. Координатор использует следующие правила для определения транзакта, который надо перевести из одного узла в другой, а также для завершения соответствующего события и активизации на время d непрерывного компонента модели объекта.

1. Начиная с первого ecb в списке планируемых событий выбирается первый транзакт, который можно продвинуть в следующий узел; в соответствующем ecb этого транзакта ct=0. Когда такой транзакт найден, определено и очередное событие.

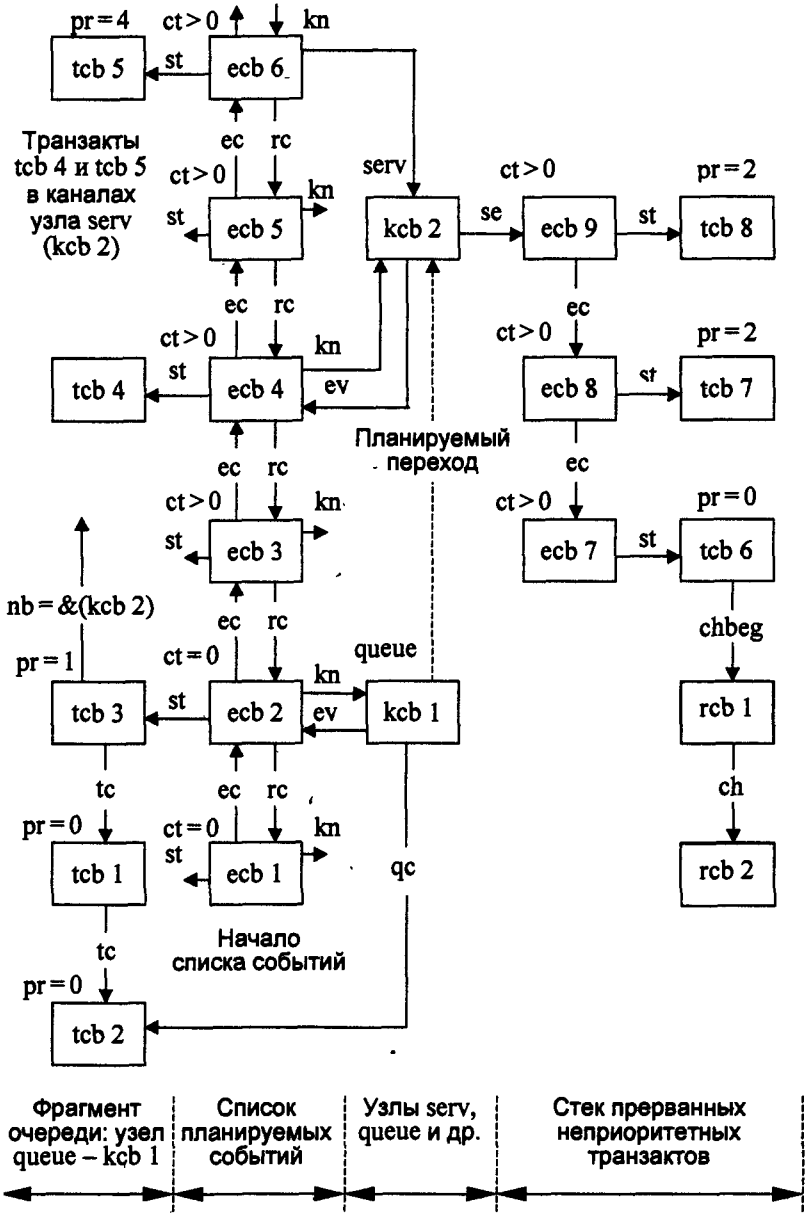


Рис. 2.12. Фрагмент случайной конфигурации управляющих структур

2. Если среди всех ecb, имеющих ct=0, не нашлось транзакта, который можно продвинуть дальше, то в списке планируемых событий выбирается первый ecb, у которого ct > 0. Далее величине *d* присваивается значение ct, величина ct в данном ecb получает нулевое значение, а у всех последующих в списке ecb значения их ct уменьшаются на величину *d*. После этого производятся обращения к моделям непрерывных компонентов с передачей им параметра *d*, а показание общего модельного таймера увеличивается на величину *d*. Потом делается попытка продвинуть транзакт, связанный с этим ecb, в следующий узел. Пункт 2 выполняется до тех пор, пока не будет выбран транзакт для продвижения либо пока координатор network не сделает вывод о том, что по какой-то причине все узлы заблокированы и моделирование не может быть продолжено.

Дело в том, что в моделях в результате неправильно проведенного структурного анализа может возникать блокировка. Имитатор предоставляет специальные приемы для борьбы с этим явлением. Разработчик моделей имеет возможность автоматической проверки состояний интересующих его узлов и в случае необходимости разблокирования их.

Рассмотренный алгоритм только поясняет специфику автоматического управления событиями; в реальной системе работает другой – скоростной алгоритм управления временем.

Способы реализации непрерывных моделей. Непрерывные компоненты модели представляют процессы перемещения товаров транспортными средствами с учетом дорожной сети, химическое производство или окружающую среду вокруг объекта экономики. Непрерывные компоненты, если они необходимы, могут быть представлены:

- разностными уравнениями;
- расчетными формулами, реализующими конкретный математический метод.

При реализации непрерывных компонентов очередной интервал (или шаг интегрирования) – это отрезок времени между двумя ближайшими событиями в стохастической сети. В данной системе моделирования обеспечиваются два способа реализации моделей непрерывных компонентов: пассивный и транзактно-управляемый. Пассивные непрерывные модели (например, модель процесса в природной среде, который можно только наблюдать, не имея возможности управления) запускаются сразу координатором network в нулевой момент модельного времени.

Транзактно-управляемые непрерывные модели запускаются по прибытии транзактов в узлы типа proc. Эти модели запускаются на определенное время активности – время обслуживания транзакта в таком узле. Активностью процесса выполнения непрерывной модели можно управлять из других узлов: с помощью сигнальной функции passiv выполнение непрерывной модели в узле proc приостанавливается, а с помощью другой функции activ – возобновляется. Такие свойства удобны при моделировании систем управления непрерывным производством. Для моделирования многоуровневых прерываний в вычислительной системе удобнее использовать узел serv.

Выводы

1. Рассмотренная концепция имитационного моделирования экономических процессов, основанная на специальном аппарате формального манипулирования узлами, транзактами, событиями и ресурсами, является довольно универсальной для применения в риск-менеджменте. На ее основе создана система моделирования Pilgrim. Данная концепция использует следующие математические методы:

- аппарат стохастических сетей для построения структурной схемы моделируемого процесса (не обязательно экономического);
- метод Монте-Карло для статистических испытаний и проверки гипотез;
- специально созданный набор датчиков псевдослучайных величин для решения экономических и иных задач;
- методы планирования экстремальных экспериментов.

2. В процессе создания модели в виде многоуровневой стохастической сети экономисту-исследователю не всегда понятно, каким образом выделять и детализировать процессы, включаемые в качестве узлов в состав модели. Поэтому Pilgrim имеет специальный инструментарий для структурного системного анализа моделируемых экономических объектов и систем, который выполняет две основные функции:

- создает графическую схему модели методами структурной послойной декомпозиции объекта экономики;
- генерирует программный код имитационной модели на языке Pilgrim в процессе диалога и последовательной декомпозиции, что позволяет применять используемую методологию экономистами–непрофессионалами в области программирования.

Вопросы для самопроверки

1. Какие основные функции выполняет моделирующая система?
2. Чему в реальной действительности соответствует граф модели?
3. Как определяется транзакт?
4. Что такое событие?
5. Какие бывают ресурсы (дать определение и указать разновидности)?
6. Как связаны узлы модели с ее графом?
7. Что такое пространственное моделирование?
8. Чем отличается генератор транзактов с бесконечной емкостью от узлов других типов?
9. Какой процесс можно назвать очередью (с приоритетами или без приоритетов)?
10. Как функционирует узел обслуживания с многими параллельными каналами?
11. Какие две основные функции выполняет терминатор, убирающий транзакты из модели?
12. Как работает управляемый генератор (размножитель) транзактов?
13. Чем отличается управляемый терминатор транзактов от обычного терминатора?
14. Как может перекрывать путь транзактам управляемый клапан?
15. Что такое очередь с пространственно-зависимыми приоритетами?
16. Какие основные функции может выполнять управляемый процесс (непрерывный или пространственный)?
17. Какие процессы возникают в имитационной модели в связи со счетами бухгалтерского учета?
18. Какие правила заложены в основу работы узла типа распорядитель финансов (или «главный бухгалтер») в имитационной модели?
19. Какой ресурс представляет собой склад перемещаемых ресурсов?
20. Какие функции выполняет менеджер (или распорядитель) ресурсов?
21. Для чего нужен структурный узел финансово-хозяйственных платежей?
22. В чем заключается различие между перемещаемыми и неперемещаемыми материальными ресурсами в имитационных моделях?

23. Что такое стартовый и оперативный информационные ресурсы?
24. Как моделируется работа с денежными ресурсами?
25. Какие узлы реализуют пространственную имитацию?
26. В чем смысл алгоритма «скорой помощи»?
27. Зачем нужен структурный узел выделения ресурсов (материальных или денежных)?
28. Для чего нужен произвольный структурный узел?
29. Зачем используется виртуальный структурный узел в качестве графа модели?
30. Чем принципиально отличается модельное время от астрономического?
31. Какие бывают масштабы времени при имитационном моделировании (указать разновидности)?

Глава 3

ОСНОВНЫЕ ПРАВИЛА МОДЕЛИРОВАНИЯ. МОДЕЛИРУЮЩИЕ ФУНКЦИИ

3.1 ЯЗЫКОВЫЕ СРЕДСТВА

Все узлы имитационных моделей являются процессами в системе Pilgrim. Стохастическая сеть, в виде которой представляется модель, не является вычислительным алгоритмом. Попытки представить имитационную модель в виде набора алгоритмов приводят к написанию больших (и сложных) моделирующих программ. Такой подход называется алгоритмическим моделированием; он не всегда доступен экономисту, даже имеющему подготовку в области программирования. Например, запрограммированная на языке Visual Basic алгоритмическая модель, состоящая из очереди и обслуживающего прибора (всего два узла), занимает несколько страниц текста на этом языке программирования.

Реальная модель экономического процесса может состоять из десятков (сотен) узлов. Поэтому нужны особые языковые средства, не являющиеся языком программирования, которые бы позволили в лаконичном (по сравнению с текстом компьютерной программы) виде описать модель. Эти средства должны учитывать особенности узлов, жизненных циклов транзактов, условий прохождения транзактов по дугам, их размножения и гибели, а также функциональные особенности взаимодействия процессов, связанных с финансовыми, материальными и информационными ресурсами.

Ниже рассмотрен набор языковых взаимосвязанных средств, предназначенных для описания имитационных моделей экономических процессов. Эти средства подразделяются на пять связанных частей:

• формализация запуска имитационной модели: инициализация объектов и структур данных;

• описание узлов с помощью общих операторов управления транзактами, событиями и узлами модели;

• функциональное описание процессов управления материальными и денежными ресурсами;

• формализация структурного анализа – управление переходами между слоями модели при многоуровневой декомпозиции;

• описание сигнальных управляющих функций.

Данные средства по форме записи являются функциями, через параметры которых реализуются синтаксические связи между объектами (узлами, транзактами, ресурсами и событиями) имитационной модели. Форма записи различных условий и условных действий соответствует языку C++.

Рассмотрение моделирующих функций имитатора, их аргументов или параметров при реализации конкретной модели на ЭВМ требует указывать типы соответствующих переменных. Эти типы интересны профессиональным математикам–программистам (но их необязательно знать экономистам, впервые начавшим заниматься имитационным моделированием):

• int – целое значение (обычно 16 разрядов);

• long – длинное целое значение (обычно 32 разряда);

• float – переменная с плавающей точкой (32 разряда);

• double – переменная с плавающей точкой двойной точности (64 разряда);

• char – символьная переменная или строка символов (1 байт).

3.2 ИНИЦИАЛИЗАЦИЯ ОБЪЕКТОВ И СТРУКТУР ДАННЫХ ДЛЯ ЗАПУСКА ИМИТАЦИОННОЙ МОДЕЛИ

Каждому типу узла имитационной модели соответствует определенная функция. Полный перечень условных обозначений узлов, которые дают название соответствующим операторам (функциям), и толкование процесса приведены на рис. 2.3.

Использование предложенной в главе 2 концепции имитационного моделирования Pilgrim позволяет разрабатывать два типа моде-

лей: разомкнутые и замкнутые (так же, как и в теории стохастических сетей). Разомкнутые модели позволяют сравнительно легко реализовать исследование внутренних процессов в фирме, но они не учитывают взаимосвязи с объектами внешней среды: рынком, госбюджетом, населением и другими. Замкнутые модели выглядят сложнее (в смысле графа стохастической сети), но позволяют учесть влияние внешней среды и исследовать связь объекта экономики с другими объектами, финансово-кредитными учреждениями, рынком.

Модель состоит из двух характерных частей: секции инициализации и блока описания стохастической сети. Все рассматриваемые ниже операторы – это программные функции, управляемые соответствующими аргументами.

Рассмотрим подробно структуру секции инициализации (рис. 3.1). Она состоит из 14 компонентов, причем необязательные компоненты заключены в квадратные скобки.

Секция начинается с макрооперации #include <Pilgrim.h>, которая подключает моделирующую среду имитатора к модели. Далее следуют описания глобальных переменных и дополнительных функций, используемых в модели. Например, это могут быть функции, описывающие непрерывные компоненты, или вызовы этих функций.

Собственно модель начинается с оператора forward. За ним, если это необходимо, указываются локальные переменные разработчика модели.

Оператор первоначальной настройки. Первоначальную настройку моделирующих программ и инициализацию (разметку) в памяти ЭВМ графа модели осуществляет оператор modbeg(p_1,p_2,p_3, p_4,p_5,p_6,p_7,p_8,p_9). Это одна из первых операций в основном блоке модели (после forward).

Аргументы этой функции имеют следующий смысл.

Параметр p_1 – символическое имя модели (char): строка длиной не более 14 символов.

Параметр p_2 – максимальный номер узла модели (int), причем справедливо соотношение $2 \le p_2 \le m_{max}$, где m_{max} – некоторое граничное значение, которое задается при установке имитатора на ЭВМ (обычно $m_{max} = 1024$).

Параметр p_3 – модельное время, в течение которого необходимо производить моделирование (типа float).

№ п/п	Оператор	Необходимость применения в модели
1	#include <Pilgrim.h>	Всегда необходим
2	[Глобальные переменные и функции]	Только в сложных моделях
3	forward	Всегда необходим
4	{	
5	[Локальные переменные модели]	Часто необходимы
6	modbeg(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9);	Всегда необходим
7	ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8); ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8); ... ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8);	Всегда необходимы
8	[Сигнальные функции]	В ресурсных моделях
9	network(p_1,p_2)	Всегда необходим
10	{	Всегда необходим
Вставляется в секцию	Блок описания узлов модели (стохастическая многоуровневая сеть)	Всегда необходим
11	}	Всегда необходим
12	modend(p_1,p_2,p_3,p_4);	Всегда необходим
13	return 0;	Всегда необходим
14	}	Всегда необходим

Рис. 3.1. Структура секции инициализации модели

Параметр p_4 – произвольное целое число, используемое для настройки датчиков псевдослучайных величин (long). В каждом узле есть свой независимый датчик. В качестве этого числа полезно использовать значение таймера ЭВМ, обращение к которому имеет следующий вид: p_4 = (long)time(NULL). В этом случае результаты разных прогонов модели будут разными, имеющими случайные отклонения. При отладке лучше использовать постоянную комбинацию цифр, например p_4 = (long)2013456789.

Параметр p_5 – признак режима пространственной имитации (int):

• earth – поверхность Земли, т.е. сферические географические координаты, широта и долгота;

• plane – декартова плоскость, т.е. прямоугольная система координат;

• cosmos – произвольное пространство (ответственность за правильность его представления возлагается на разработчика моделей);

• none – если пространственная имитация в экономической модели не используется.

Параметр p_6 – номер одной из очередей (узла типа queue, attach или send), которую необходимо контролировать во времени для анализа динамики задержек с графическим отображением результатов (int). Например, если указать число 3 и в модели действительно имеется очередь с таким номером, то можно получать изображение динамики этой очереди непосредственно в процессе моделирования.

Параметр p_7 – номер (int) одного из процессов (узла типа proc), который необходимо контролировать как в пространстве, так и во времени с графическим отображением результатов. Пространственная имитация будет рассмотрена в разд. 3.3, описывающем функцию proc (не все версии имитатора имеют это свойство). Если нет необходимости в графической интерпретации пространственной имитации, то указывается none.

Параметр p_8 – номер (int) терминатора (узла типа term), на входе которого необходимо наблюдать интенсивность потока транзактов во время моделирования. Если такой необходимости нет, то указывается none.

Параметр p_9 – точность (int):

• если p_9=1 – 6, то имитатор будет использовать от 1 до 6 знаков после десятичной точки при формировании замеренных результатов (временных интервалов);

• если p_9=none, то имитатор будет представлять результаты округленными до целых значений.

За оператором modbeg следуют описания узлов – генераторов ag. Это единственный тип узла, который должен быть описан в секции инициализации. Эти генераторы не зависят от состояния других узлов сети и должны работать до того, как узлы других типов будут приведены в рабочее состояние.

Генератор транзактов. Генератор с бесконечной емкостью – это функция ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8). Каждый генератор задается одним оператором ag. Обычно все ag записываются подряд (если их несколько) после modbeg. Аргументы ag имеют следующий смысл.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – номер узла–генератора (int).

Параметр p_3 – приоритет (int), назначаемый каждому сгенерированному транзакту, – число в диапазоне 1 – 32 767; если приоритет не нужен, то p_3=none.

Параметр p_4 – тип функции распределения интервала времени между двумя последовательно сгенерированными транзактами (int), имеющий значения:

- norm – нормальное распределение;
- unif – равномерное распределение;
- expo – экспоненциальное распределение;
- erln – обобщенное распределение Эрланга;
- beta – треугольное распределение;
- none – если интервал между транзактами является детерминированной величиной.

Полная информация об этих распределениях и соответствующих программных датчиках, используемых в модели, содержится в главе 1.

Параметр p_5 – величина, зависимая от типа функции распределения (float):

- математическое ожидание интервала времени между двумя последовательно сгенерированными транзактами (p_4 = norm, unif, expo);
- математическое ожидание одного слагаемого этого интервала (p_4 = erln);
- минимальное значение интервала (p_4 = beta);
- постоянная величина этого интервала (p_4 = none).

Параметр p_6 – величина, зависимая от типа функции распределения (float):

- среднеквадратичное отклонение (p_4 = norm);
- максимальное отклонение от среднего (p_4 = unif);
- значение zero (p_4 = expo, none);
- число слагаемых, входящих в случайный интервал и распределенных по экспоненциальному закону (если p_4 = erln, то $p_6 > 0$);
- наиболее вероятное значение интервала времени между двумя последовательно сгенерированными транзактами (p_4 = beta).

Параметр p_7 – величина, также зависимая от типа функции распределения (float):

• максимально возможное значение интервала времени между двумя последовательно сгенерированными транзактами (p_4 = beta);

• значение zero (p_4 = norm, unif, expo, erln, none).

Параметр p_8 – номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

Координатор сети процессов. Координатор network (p_1,p_2) осуществляет диспетчеризацию транзактов в узлах (процессах) модели, планирует события в едином модельном времени и активизирует дискретные или непрерывные компоненты модели, имитирующие внешнюю среду. Координатор выполняет всю модель либо до первой ошибки, либо пока не истечет время, заданное аргументом p_3 в функции modbeg.

Относительно синтаксиса языка программирования эта функция является структурным оператором, за которым следует фигурная скобка { , открывающая блок описания стохастической сети. Этот блок заканчивается закрывающей скобкой }. Обращение к этой функции делается только после того, как выполнены оператор modbeg и все операторы ag.

Аргументы p_1 и p_2 – имена (адреса) соответствующих программных функций моделирования внешней среды, производящих интегрирование, решение разностных уравнений, вычисление по формулам и т.д. Эти функции возвращают значение типа float. Им координатор передает единственный аргумент: float d – интервал времени, в течение которого они получают управление. Понятие внешней среды может быть не только из экономики, например:

• в экологии $p_1(d)$ и $p_2(d)$ могут быть произвольными функциями моделирования процессов переноса загрязняющих веществ в природной среде (в атмосфере – $p_1(d)$, а в воде – $p_2(d)$);

• в химии или энергетике $p_1(d)$ – уравнения, описывающие процессы, происходящие в реакторе, причем дискретные компоненты модели реактора – это его система управления, включающая управляющую ЭВМ, систему опроса датчиков и другие компоненты (в данном случае $p_2(d)$ не нужна);

• при моделировании вычислительной системы дискретные компоненты модели – это очереди в операционной системе, процессор, канал, диски и др., а функции p_1 и p_2 не нужны (если только эта вычислительная система не управляет, например, процессом плавки в доменной печи).

Аргумент d в функциях p_1 и p_2 – это глобальная переменная, содержащая интервал времени между двумя ближайшими событиями.

Разработчик модели лишен возможности самостоятельно (т.е. напрямую из модели) вызывать p_1 и p_2 во избежание непредсказуемых или ложных результатов, а переменная d может изменяться только координатором network.

Функции float p_1(d) и float p_2(d), если они необходимы, пишутся пользователем. Они должны производить либо интегрирование, либо вычисления по формулам, либо решение разностных уравнений на отрезке времени d, который каждый раз передается в качестве параметра типа float. В общем случае, кроме C++, для этого может использоваться язык Паскаль (однако на C++ вычисления идут быстрее) или пакет математических программ.

Выбор очередного узла, в который необходимо «продвинуть» транзакт, осуществляется с помощью вызова network(p_1,p_2), записываемого только один раз перед блоком описания узлов стохастической сети. Если процессы p_1(d) или p_2(d) не моделируются, то в качестве p_1 и p_2 необходимо соответственно указать слово dummy.

Оператор завершения моделирования. Оператор завершения модели modend(p_1,p_2,p_3,p_4) должен выполняться после того, как будет нарушено внутреннее условие noerr, определяющее, что в модели либо зафиксирована грубая ошибка, либо истекло время моделирования, указанное в операторе modbeg. Он удаляет все созданные в процессе моделирования управляющие структуры из памяти ЭВМ – как образованные заранее с помощью функции modbeg блоки kcb, так и блоки типа ecb и tcb из модели или цепочек свободных блоков, образуемых после их логического уничтожения в процессе моделирования. Далее этот оператор позволяет просмотреть на экране монитора графические результаты и выводит итоговые табличные результаты в файл-отчет на жестком диске. Значения аргументов следующие.

Параметр p_1 – символическое имя специального файла-отчета, в который записываются стандартные характеристики модели, полученные и замеренные в процессе ее выполнения. Это имя (char) является строкой и оформляется по системным соглашениям (например, имя Results1.doc).

Параметр p_2 – номер первой страницы отчета (int).

Параметр p_3 – число строк (int) на каждой странице.

Параметр p_4 – имеет два значения (int): либо page, если в файле-отчете необходимо проставить символ перевода страницы, либо none.

3.3 ОБЩИЕ ФУНКЦИИ УПРАВЛЕНИЯ УЗЛАМИ, ТРАНЗАКТАМИ И СОБЫТИЯМИ В МОДЕЛИ

Описание узлов графа, условий прохождения транзактов и моделирование дискретных компонентов производятся с помощью независимых программных ветвей, активностью которых управляет координатор network. Каждый узел имеет следующую типовую структуру (рис. 3.2). Узел состоит из шести типовых компонентов; необязательные компоненты заключены в квадратные скобки.

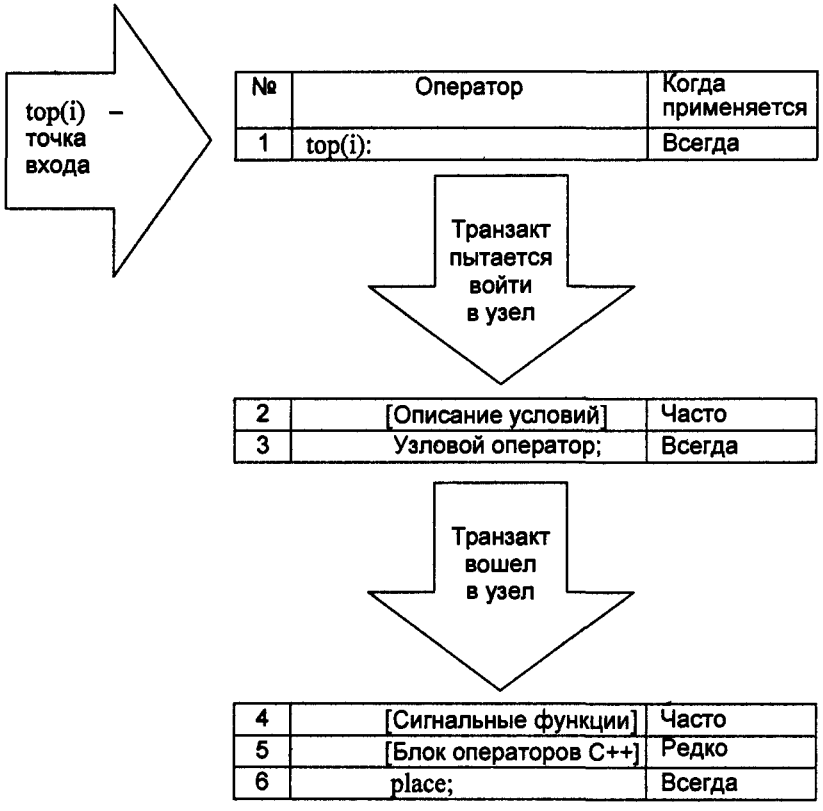


Рис. 3.2. Типовая структура описания узла

Относительно топологии можно рассматривать два типа структурных схем моделей: разомкнутые и замкнутые. Пример разомкнутой модели «Эффективность компьютеров в АРМ бухгалтерии» (ветвление с использованием параметров транзактов) приведен в главе 8. Простая (но эффективная) замкнутая модель «Минимизация затрат производства», отражающая схему зарядки модели, также рассмотрена в главе 8.

Описание узла начинается с особой «метки» – функции top(i), где i – номер этого узла, а заканчивается оператором place. После top(i) ставится двоеточие. Начиная с метки top(i) транзакт проходит все операторы узла i без задержек (в смысле модельного времени) и попадает в оператор place, где находится до тех пор, пока не появятся условия для перехода в следующий узел.

В некоторых узлах одновременно могут находиться несколько транзактов. Все они содержатся в одном и том же операторе place. Поэтому не следует путать поток транзактов в модели с потоком управлений в обычной программе. Задержка транзакта в place – это и есть время пребывания в узле.

Операторы анализа условий. После метки top(i) можно анализировать условия продвижения транзактов по графу модели, при этом используются операторы if или switch. Если необходимо менять направления путей транзактов, законы распределения, значения времени обслуживания и другие параметры, то можно использовать операцию присваивания. Например, имеются локальные переменные int a, b. Если в узле 3 мы хотим изменять значение переменной b в зависимости от значения a, то должны записать:

```
top(3): if (a < 0)
           b =4;
        else
           b =3*a+5;
        key("Переключатель", b);
        place;
```

где key – оператор, определяющий узел типа «клапан» (*см.* рис. 2.3).

При анализе условий запрещено использовать рекурсию типа x=f(x), x=x+1 и т.д., а также нежелательно выполнять ресурсоемкие вычисления, требующие существенных затрат времени процессора ЭВМ.

Узловой оператор. Основной обязательный оператор после двоеточия метки-функции – это оператор определения типа узла (далее – узловой оператор). Этот оператор имеет условное наименование, совпадающее с типом узла. Причем могут использоваться все типы, перечисленные на рис. 2.3, кроме узлов ag и parent. Узловые операторы – это функции: queue, serv, term, creat, delet, key, dynam, proc, send, direct, attach, manage, pay, rent и down. В описании каждого узла должен быть только один вызов любой из этих функций.

Сигнальные управляющие функции и блок операторов C++ используются не во всех узлах. Они, как необязательные элементы узла, рассмотрены в конце данной главы.

Оператор place – последний оператор узла.

Описание узла иногда состоит только из метки-функции top(i), узлового оператора и оператора place. Например:

```
top(i): key("Переключатель",j);
        place;
```

В самом конце блока описания стохастической сети полезно записать функцию fault(err), которой управление будет передано в том случае, если из-за невнимательности пользователя произойдет попытка перехода транзакта в несуществующий узел. После такой попытки моделирование прекратится и будет завершено с кодом возврата err, где err – произвольно задаваемое пользователем число, обычно в пределах от 101 до 32 767.

Рассмотрим основные процессы, образующие узлы модели.

Очередь (с приоритетами или без приоритетов). Функция queue(p_1,p_2,p_3) определяет узел, моделирующий очередь транзактов. Эта очередь строится по одному из двух правил: либо транзакты упорядочены в порядке поступления, либо вновь поступающие транзакты поступают в конец своей приоритетной группы (более приоритетные транзакты находятся ближе к началу очереди, а менее приоритетные – к концу). Чем больше численное значение приоритета транзакта, тем он приоритетнее. Аргументы (параметры) имеют следующие значения.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – тип организации очереди (int): либо p_2 = prty, если очередь с приоритетами, либо p_2 = none, если очередь без приоритетов.

Параметр p_3 – номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

График динамики изменения очереди получается автоматически, если в modbeg подставить в качестве параметра p_6 номер контролируемой очереди.

Узел обслуживания с *N* параллельными каналами. Моделирующая функция serv(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8) описывает узел, осуществляющий какое–либо обслуживание транзактов в течение модельного времени, отличного от нуля. В данном случае узел – это одно- или многоканальный обслуживающий прибор, работающий по правилам абсолютных приоритетов или без них и имеющий стек для прерванных транзактов (правило относительных приоритетов реализуется в узле типа queue – очередь). Аргументы, используемые в функции serv, имеют следующий смысл.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – число обслуживающих каналов(int), причем $1 \le p_1 \le 32\,767$.

Параметр p_3 – дисциплина обслуживания:

- если p_3 = abs, то используется приоритетная дисциплина, с прерыванием обслуживания менее приоритетного транзакта более приоритетным (будет рассмотрена ниже);
- если p_3 = none, то используется бесприоритетная дисциплина.

Параметр p_4 – тип функции распределения интервала времени обслуживания транзакта в канале узла serv (int), имеющий значения: norm – нормальное распределение; unif – равномерное распределение; expo – экспоненциальное распределение; erln – обобщенное распределение Эрланга; beta – треугольное распределение; none – если интервал обслуживания является детерминированной величиной.

Параметр p_5 – величина, зависящая от типа функции распределения (float): математическое ожидание интервала времени обслуживания транзакта (при p_4= norm, unif, expo); математическое ожидание одного слагаемого этого интервала (при p_4 = erln); минимальное значение интервала (при p_4 = beta); постоянная величина этого интервала (при p_4 = none).

Параметр p_6 – величина, зависящая от типа функции распределения (float): среднеквадратичное отклонение времени обслуживания (при p_4 = norm); максимальное отклонение от среднего времени обслуживания (при p_4 = unif); значение zero (при p_4 = expo, none); число

слагаемых, распределенных по экспоненциальному закону и входящих в случайный интервал обслуживания (если p_4 = erln, то $p_6 > 0$); наиболее вероятное значение интервала времени обслуживания транзакта (при p_4=beta).

Параметр p_7 – величина, также зависимая от типа функции распределения (float): максимально возможное значение интервала времени обслуживания транзакта (p_4 = beta); значение zero (p_4 = norm, unif, expo, erln, none).

Параметр p_8 – номер узла (int), в который передается обслуженный транзакт (узел-приемник).

Рассмотрим подробнее прерывания обслуживания неприоритетных транзактов. Если задать p_3 = abs, то имеются две возможности для работы с прерванными неприоритетными транзактами:

- они дообслуживаются после ухода приоритетного транзакта, причем сохраняется чистое заданное время обслуживания, без учета времени нахождения прерванного транзакта в специальном стеке имитатора;
- характер прерывания обслуживания транзакта был таков, что нужно возобновить обслуживание заново.

Пример 3.1. Повторный ввод информации при отсутствии контрольных точек. Оператор ЭВМ в течение нескольких часов вводит большой массив информации, причем он не позаботился о периодическом сохранении информации в памяти на жестком диске HDD. Если произойдет случайный сбой или бросок электропитания, то информация, расположенная в оперативной памяти RAM, пропадет; после этого оператор будет вынужден возобновить многочасовую работу заново.

Однако если перед функцией serv транзакт пройдет через оператор присваивания типа t→ga=again, то он при прохождении через serv получает признак обслуживания заново. После выхода из узла serv этот признак теряется.

Следует отметить, что среднее время обслуживания транзактов в узле serv и среднее время задержки в узле – это разные времена. При прерываниях обслуживания транзакта (с его дообслуживанием) чистое время его обработки будет меньше времени задержки в узле на длительность обслуживания приоритетных транзактов.

Терминатор, убирающий транзакты из модели. Функция term(p_1) описывает узел–терминатор, назначение которого заключается в следующем: он удаляет из модели входящий в него транзакт и фиксирует время его существования начиная с момента выхода это-

го транзакта из генератора. Единственный параметр p_1 – это символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Существуют средства автоматического построения графика изменения потока транзактов, поступающих на вход терминатора. График динамики потока таких транзактов получается автоматически, если номер этого терминатора задан параметром p_8 в функции modbeg, причем время жизни каждого транзакта измеряется с момента его генерации и до момента уничтожения.

Когда пользователю необходимо иметь инструмент для анализа динамики потока транзактов по какой-либо ветви графа, причем принимающий узел не является терминатором, то необходимо модель дополнить двумя узлами:

- завести дополнительный терминатор и указать его номер в качестве параметра p_8 в функции modbeg;
- в ветвь перед исследуемым узлом вставить узел creat (будет рассмотрен ниже), который в момент прохождения каждого транзакта по ветви будет генерировать один дополнительный, направляемый в такой дополнительный терминатор, а основной транзакт будет входить в узел, на входе которого проводятся измерения.

Транзактно-управляемый генератор. Функция creat (p_1,p_2,p_3,p_4,p_5,p_6) предназначена для создания нового семейства транзактов. Следует отметить, что все транзакты принадлежат какому-либо семейству. Транзакты, выходящие из обычного генератора ag, принадлежат к семейству с номером 0. Узел creat в отличие от ag – это управляемый генератор. Назначение его самое различное. В замкнутых моделях он применяется для схемы зарядки. Аргументы этого оператора следующие.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – идентификатор (int) порождаемого семейства транзактов.

Параметр p_3 – число порождаемых транзактов (int).

Параметр p_4 – имеет следующие значения (int): либо copy – для тиражирования параметров порождающего транзакта каждому порожденному (включая время жизни), либо none – для присвоения каждому порожденному транзакту в качестве параметров нулевых значений.

Параметр p_5 – номер узла (int), в который направляются порожденные транзакты.

Параметр p_6 – номер узла (int), в который направляется порождающий транзакт.

Логика функционирования узла creat такова:

1) через узел проходит порождающий транзакт, который принадлежит семейству f_1, и поступает в узел p_6;

2) одновременно с этим в узле генерируются p_3 новых транзактов, принадлежащих семейству с номером $f_2=p_2$, которые будут направлены в узел p_5. В общем случае p_5 и p_6 – это любые узлы (кроме ag), в частности, это может быть один и тот же узел. Номера семейств f_1 и f_2 в общем случае могут совпадать;

3) после прохождения порождающего транзакта узел creat получает его координаты, т.е. перемещается;

4) семейству f_2 могут передаваться или не передаваться все свойства (параметры) порождающего транзакта.

Для решения проблем, связаных с регулированием численности семейств транзактов или группировки нескольких транзактов в один, используется узел delet, функционально противоположный узлу creat.

Транзактно-управляемый терминатор. Узловой оператор delet(p_1,p_2,p_3,p_4,p_5,p_6) предназначен для уничтожения группы транзактов семейств, номера которых относятся к диапазону, задаваемому параметрами p_2 и p_3. В отличие от терминатора term он управляется специальным транзактом, который называется поглощающим. Если, например, обозначить номер семейства транзакта как Numer, то транзакт будет уничтожен при входе в узел delet при выполнении двух условий:

- в узел зашел поглощающий транзакт, принадлежащий семейству p_4;
- выполнено соотношение $p_2 \le$ Numer $\le p_3$.

Рассмотрим параметры узла.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – начало (int) диапазона номеров семейств уничтожаемых транзактов.

Параметр p_3 – конец (int) диапазона номеров семейств уничтожаемых транзактов.

Параметр p_4 – идентификатор (int) семейства, к которому принадлежит поглощающий транзакт.

Параметр p_5 – число уничтожаемых транзактов семейства p_2 (int).

Параметр p_6 – номер узла (int), в который направляется уничтожающий транзакт.

Логика функционирования этого узла такова: в узел входит транзакт семейства p_4 и находится там до тех пор, пока в него не поступят p_5 транзактов семейства $p_2 \le \text{Number} \le p_3$, которые он должен мгновенно уничтожить (поглотить). Время существования этих транзактов фиксируется в узле delet. Узел получает координаты каждого уничтожаемого транзакта, т.е. он перемещается по координатной сетке. В общем случае номера семейств p_1 и p_2 могут совпадать с номером p_3 (но с методической точки зрения такие совпадения нежелательны).

Среднее время задержки в данном случае – это среднее время ожидания всех уничтожаемых транзактов или части таких транзактов, если из состояния ожидания узел delet выводится принудительно с помощью функции freed. Следует учесть, что обслуживание транзактов в таком узле – это их уничтожение.

Отметим, что если в такой узел так и не поступят p_5 транзактов, то уничтожающий транзакт будет все время находиться в нем, блокируя поступление в него других уничтожающих транзактов. Поэтому имеется средство для *изгнания* уничтожающего транзакта из узла delet – функция freed(i). Эта функция является сигнальной. Она рассмотрена в разд. 5.5.

В данном случае имеется возможность автоматического подсчета:

1) среднего времени жизни уничтожаемых транзактов (или части транзактов, если из состояния ожидания узел delet выводится принудительно – выполнением функции freed);

2) числа уничтоженных транзактов.

Клапан на пути транзактов. Функция key(p_1,p_2) описывает прохождение транзакта через некий клапан. Когда клапан закрыт, транзакт не может в него войти из другого узла. Если же клапан открыт, то транзакт проходит через него в узел n без задержки. Среднее время пребывания такого узла в закрытом состоянии подсчитывается автоматически. Для управления этим клапаном или ключом существуют вспомогательные функции hold и rels.

Среднее время задержки – это среднее время пребывания ключа в закрытом состоянии. Число обслуженных транзактов – это число переключений ключа из закрытого состояния в открытое.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

Транзактно-управляемый процесс. Специальная суперфункция моделирования транзактно-управляемого непрерывного процесса

$$proc(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8)$$

объединяет в себе возможности имитации:

1) обслуживание в узле подобно serv с одним каналом;

2) перемещение узла по общему полю данных на координатной сетке;

3) запуск на время активности функции типа

$$float\ p_2(d)\ ,$$

где d – элементарный интервал активности (float), который определяется системой Pilgrim в процессе моделирования и зависит от параметров p_3, p_4, p_5, p_6.

Интервал требуемого обслуживания транзакта может быть меньше времени пребывания транзакта в этом узле, так как процесс может быть переведен в состояние «пассивен» или «активен» каким-либо транзактом из другого узла с помощью функций activ и passiv.

Если процесс пассивен, то обслуживание транзакта приостанавливается, а выполнение функции p_2 прерывается до тех пор, пока процесс не будет переведен в активное состояние.

Аргументы имеют следующий смысл.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – имя (адрес) программы, написанной пользователем, или слово dummy, если такой программы нет. Эта программа может моделировать процесс с помощью формул, дифференциального уравнения и т.д.

Параметр p_3 – номер исходной точки, в которую устанавливается узел proc перед началом моделирования (int).

Параметр p_4 – тип функции распределения интервала активности процесса либо возможность работы с координатным пространством (int). Функции распределения обозначаются так: norm – нормальное распределение, unif – равномерное, expo – экспоненциальное, erln – обобщенное распределение Эрланга, beta – треугольное распределение, none – интервал, являющийся детерминированной величиной. Возможность работы с пространством задается одним из трех способов: earth – поверхность Земли (географические координаты широта

и долгота), plane – декартова плоскость (прямоугольная система координат), cosmos – произвольное пространство, none – если нет распределений или режим пространственной имитации в данном узле не нужен.

Параметр p_5 – математическое ожидание интервала времени (float) активности (p_4 = norm, unif, expo), либо математическое ожидание одного слагаемого этого интервала (p_4=erln), либо минимальное значение интервала обслуживания (p_4=beta), либо постоянная величина этого интервала (p_4=none), либо интервал «непрерывного» нахождения этого узла в точке на координатной сетке (если p_4 = earth, plane или cosmos).

Параметр p_6 – параметр интервала активности обслуживания (float), который зависит от значения параметра p_4. Это либо среднеквадратичное отклонение (p_4=norm), либо максимальное отклонение от среднего (p_4= unif), либо значение zero (p_4=expo, none), либо число слагаемых интервала обслуживания (если p_4=erln, то $p_6 > 0$), либо наиболее вероятное значение интервала времени обслуживания транзактов (p_4=beta), либо скорость перемещения узла от одной точки пространства к другой (если p_4 = earth, plane или cosmos).

Параметр p_7 – величина, также зависящая от типа функции распределения (float) интервала активности: либо значение zero (p_4=norm, unif, expo, erln, none), либо максимальное значение интервала времени обслуживания (p_4=beta);

Параметр p_8 – номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

Приведенный выше набор свойств и параметров узла типа proc делает эту суперфункцию чрезвычайно удобной для имитации таких реальных процессов, как:

- запуск процесса (реакции) в реакторе и управление активностью этого процесса;
- облет на вертолете местности, подвергшейся какому-либо бедствию;
- перемещение позиционера, доступ и чтение информации с накопителя на магнитном диске в вычислительной системе;
- оперативное управление вертолетами «скорой помощи» с учетом динамически меняющейся ситуации.

Значения параметра p_4 обеспечивают дополнительные возможности имитации поведения узла proc в пространстве.

Перед входом в узел очередного транзакта координаты узла – это координаты какой-либо точки A пространства, номер которой находится в параметре этого узла k→kx.

Транзакт, который входит в узел, имеет другие координаты – координаты точки B (номер которой помещен в параметр транзакта t→tx). После выхода транзакта из узла автоматически вычисляется длина l отрезка AB, которая автоматически прибавляется к суммарному значению пути, пройденного этим узлом. Время обслуживания транзакта определяется в этом случае по формуле

$$t_{\text{обсл}} = \frac{l}{p_6} + p_5 .$$

Значения и смысл параметров p_5 и p_6 можно пояснить на следующем примере.

Пример 3.2. Допустим, что имеется вертолет «скорой помощи», который вылетает по заявкам-радиограммам в различные населенные пункты. Во время полета на борт вертолета могут поступать новые радиограммы. Здесь радиограммы из населенных пунктов – это транзакты, а вертолет – узел proc. Время полета из одного пункта в другой равно

$$t_{\text{полета}} = \frac{l}{p_6} ,$$

а время посещения больного равно p_5. При управлении вертолетом можно оптимизировать обслуживание транзактов (уже полученных радиограмм) с помощью оптимизации маршрута и, более того, можно прервать полет к одному пункту для залета в другой, если это приведет к снижению суммарного пути.

Другими словами, полет, длительность которого равна $t_{\text{полета}}$, подвержен возможному прерыванию, а обследование больного (время p_5) – это непрерываемое нахождение в конкретной точке на координатной сетке. Значения p_5 и p_6 в данном случае – конкретные числа. Поэтому, если необходимо все-таки подставить случайные значения, можно использовать датчики псевдослучайных величин, работа с которыми рассматривается в разд. 4.1.

Отметим, что в вырожденном случае узел proc эквивалентен узлу serv. Например, следующие две функции эквивалентны:

1) serv("Обслуживание", 1,none,expo,10.0,zero,zero,n);

2) proc("Обслуживание", dummy,none,expo,10.0,zero,zero,n).

Они обеспечивают моделирование одноканальной системы массового обслуживания без приоритетов с временем обслуживания, распределенным по экспоненциальному закону с математическим ожиданием 10 единиц.

Практическое применение этот узел нашел в моделях, предназначенных для оперативного управления работами при ликвидации последствий аварии на Чернобыльской АЭС на территории Брянской области.

Очередь с пространственно-зависимыми приоритетами. Функция dynam(p_1,p_2) предназначена для моделирования обслуживания транзактов в очереди с динамическими пространственно-зависимыми приоритетами. Эта функция моделирует оптимально-управляемую очередь (очередь типа «скорая помощь»), которая находится на входе узла типа proc. Узел имеет два параметра.

Параметр p_1 – символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – номер узла (int) типа proc, в который передается сгенерированный транзакт (узел-приемник); proc обязательно должен использовать свой параметр p_4= earth, plane или cosmos.

Задача оптимального расписания для обслуживания транзактов с пространственно-зависимыми приоритетами может возникнуть, например, при моделировании следующих сложных процессов и объектов:

- участка гибкого автоматизированного производства (ГАП) с роботизированными тележками, «путешествующими» по цеху;
- местности, подверженной какому-либо бедствию, в процессе ее обследования специальной командой на вертолете и др.

Пример 3.3. Оперативное планирование маршрута вертолета «скорой помощи». Допустим, во время дежурства на вертолет поступают радиограммы из различных пунктов. Эти радиограммы – вызовы к больному. Если во время работы скопилось несколько радиограмм, то путь вертолета должен быть минимальным, чтобы уменьшить среднее время ожидания больного. Более того, если полет из одного пункта к другому уже начался, то при появлении новой радиограммы из пункта, находящегося не очень далеко от курса вертолета, маршрут может быть изменен в пользу этого вызова, если суммарный путь будет минимален.

В такой интерпретации функция proc – это вертолет, радиограммы – транзакты, имеющие координаты населенных пунктов, а функция dynam – оптимизатор курса. В модели же оператор dynam – это

очередь, обслуживающая входящие транзакты по правилу динамических пространственно-зависимых приоритетов. Порядок обслуживания транзактов в этой очереди пересматривается каждый раз при поступлении в нее (в хвост очереди) нового транзакта и при переходе первого транзакта (из головы очереди) в узел proc.

Функция dynam всегда «заглядывает» в узел proc и анализирует, не следует ли прервать обслуживание находящегося в нем транзакта. Если такое решение будет принято, то вычисляется местонахождение текущей точки пространства, в которой находится узел proc; функция dynam извлекает из этого узла транзакт обратно в свою очередь и посылает в него более выгодный относительно оптимизации транзакт.

Оригинальный алгоритм оптимизации динамического расписания – правила построения динамической очереди dynam – описан в главе 2. Он обладает высоким быстродействием и имеет практическое применение.

3.4 УПРАВЛЕНИЕ МАТЕРИАЛЬНЫМИ И ДЕНЕЖНЫМИ РЕСУРСАМИ

Моделирование материальных и денежных ресурсов учитывает подобие их основных характеристик: остаток ресурса похож на положительное сальдо, дефицит подобен отрицательному сальдо. Есть и другие аналогии (кроме перечисления денег и бухгалтерских проводок).

Функция запроса ресурсов со склада. Каждый склад ресурсов описывается в имитационной модели в виде узла типа attach. В узле attach образуется очередь транзактов, которая может быть организована по приоритетному принципу: чем меньше транзакт запрашивает единиц, тем более он приоритетен. Соответствующая функция имеет вид:

$$\text{attach}(p_1,p_2,p_3,p_4) \ .$$

Эта функция включает 4 параметра.

Параметр p_1 – символическое имя узла-ресурса: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – требуемое число элементов ресурса (long).

Параметр p_3 – работа с приоритетами: prty или none. Если указано prty, то требования на ресурс в случае отсутствия необходимого

числа элементов образуют очередь в узле attach, причем соответствующие транзакты располагаются в порядке убывания значения приоритета (ближе к голове очереди находится самая приоритетная группа транзактов). Внутри приоритетной группы транзакты расположены в следующем порядке: чем меньше элементов необходимо транзакту, тем ближе транзакт находится к голове своей приоритетной группы. Если же требования на число элементов одинаковы, то транзакты расположены в хронологическом порядке (правило fifo): чем раньше транзакт пришел в очередь, тем раньше он обслужен. Когда указано значение none, работает только правило fifo.

Параметр p_4 – номер узла-приемника (int). Таким узлом может быть только узел-менеджер (manage).

Функция имитации менеджера ресурсов. Обслуживанием транзактов занимается узел типа «менеджер» – manage (*см.* рис. 2.5). Обслуженный транзакт проходит узел manage и «путешествует» с захваченными единицами по графу модели. Транзакт может несколько раз становиться в очередь к одному и тому же ресурсу, получая дополнительные единицы. Соответствующая функция имеет вид

$$\text{manage}(p_1,p_2)\ ,$$

включая два параметра.

Параметр p_1 – символическое имя узла-менеджера: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – номер узла-приемника (int). Таким узлом может быть любой узел модели, кроме manage.

Функция имитации бухгалтерской проводки. Основные объекты системы Pilgrim (узел, транзакт, событие) очень хорошо подходят для описания финансовой динамики на счетах бухгалтерского учета предприятия (фирмы). Узлом считается счет (субсчет) бухгалтерского учета; предположим, что номер этого узла i. Транзакт, вошедший в узел i, – это запрос на проводку со счета i определенной суммы на какой-то другой счет. Для осуществления проводки необходимо, чтобы на счете i (т.е. в узле i) была сумма не менее требуемой. При отсутствии такой суммы транзакт становится в ожидание момента поступления на счет i достаточных средств. Другими словами, узел с номером i, который формирует запрос на бухгалтерскую проводку, – это специальная очередь транзактов. Описание узла-счета i:

$$\text{send}(p_1,p_2,p_3,p_4,p_5)\ .$$

Этот узел имеет пять параметров.

Параметр p_1 – символическое имя узла-ресурса: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – узел-счет, на который необходимо перевести заданную сумму(int);

Параметр p_3 – размер заданной суммы (double). Единицы измерения финансовых средств – любые (рубли, доллары и т.д.). После точки обязательно необходимо указывать одно или два числа – доли используемых единиц измерений. Например: 1000000.00 (*Один миллион руб. 00 коп.*).

Параметр p_4 – возможность работы с приоритетами: prty или none. Если указано prty, то требования на перечисление денег со счета i в случае отсутствия необходимой суммы образуют очередь в узле send, причем соответствующие транзакты располагаются в порядке убывания значения приоритета (ближе к голове очереди находится самая приоритетная группа транзактов). Внутри приоритетной группы транзакты расположены в следующем порядке: чем меньше требуемая сумма, тем ближе транзакт находится к голове своей приоритетной группы. Если же суммы одинаковы, то транзакты расположены в хронологическом порядке (правило fifo): чем раньше транзакт пришел в очередь, тем раньше он обслужен. Когда указано значение none, работает только правило fifo.

Параметр p_5 – номер узла типа «финансовый директор» (узла direct), который осуществляет финансовый менеджмент и выполняет проводки по мере необходимости.

Событием в узле типа send является факт выполнения проводки со счета i на счет p_2. Момент времени такого события – это момент времени проводки, определяемый выводом транзакта из узла send.

В каждом узле типа send имеется внутренний атрибут saldo, который отражает остаток средств на счете i. Дефицит средств на счетах бухгалтерского учета содержится в другом атрибуте – defic. Если атрибут saldo в узле i имеет нулевое значение и в этом узле имеются транзакты (один или несколько), запросившие проводки, то суммарный дефицит затребованных этими транзактами сумм автоматически отражается в атрибуте defic.

Имитация работы бухгалтера. Обслуживание очередей типа send возможно с помощью одного или нескольких узлов типа «финансовый директор». Описание такого узла:

direct(p_1,p_2).

Эта функция имеет два параметра.

Параметр p_1 – символическое имя узла-ресурса: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – это узел-приемник транзакта, выполнившего проводку. Этот узел может быть любого типа, кроме direct. При моделировании бизнес-процесса небольшого предприятия достаточно одного узла типа direct. Однако можно имитировать одновременную работу нескольких бухгалтеров, каждый из которых отвечает за свою группу бухгалтерских операций.

Пример работы с материальными и денежными ресурсами изложен в главе 8.

3.5 СТРУКТУРНЫЙ АНАЛИЗ: УПРАВЛЕНИЕ ПЕРЕХОДАМИ МЕЖДУ СЛОЯМИ МОДЕЛИ ПРИ МНОГОУРОВНЕВОЙ ДЕКОМПОЗИЦИИ

Процесс построения графа имитационной модели сопровождается структурным анализом исследуемого процесса. При структурном анализе возникает задача перехода между слоями: нижние слои модели содержат декомпозицию узлов, расположенных выше. Декомпозиция – это детализация одного узла с помощью совокупности других узлов.

Существуют четыре разновидности декомпозиции процессов:

1) общий случай декомпозиции сложного процесса с помощью узлов типа down;

2) декомпозиция процессов перечисления денег (платежей, бухгалтерских проводок и др.) с помощью узлов типа pay;

3) декомпозиция процессов выделения ресурсов с помощью узлов типа rent;

4) абстрактное объединение группы процессов в один псевдопроцесс с помощью виртуального (мнимого, не существующего в реальности) узла parent без образования нового узла.

Общая схема взаимодействия между слоями модели показана на рис. 2.1. Здесь рассматривается декомпозиция реальных узлов pay, rent и down. Управление переходами между слоями модели при многоуровневой декомпозиции основано на применении виртуального узла parent, который будет рассмотрен в главе 5, так как этот узел – атрибут диалогового CASE-конструктора, позволяющего проводить структурный анализ и создавать модели в графическом виде.

Функция имитации перечисления денежной суммы. Имитация перечисления денежной суммы с помощью узла pay выглядит гораздо понятнее, чем это делается с помощью запутанных цепочек send⇒direct, рассмотренных выше. Однако узел pay подлежит детализации на более низком уровне с помощью все тех же узлов send и direct. Функция pay имеет следующий вид:

$$pay(p_1,p_2,p_3,p_4,p_5,p_6,p_7)\ .$$

Для этой функции задаются семь аргументов.

Параметр p_1 – символическое имя узла pay: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – номер узла-счета типа send, на который переводится денежная сумма (int).

Параметр p_3 – значение денежной суммы (или стоимость). Это переменная типа double.

Параметр p_4 – номер узла-счета типа send, с которого переводится денежная сумма (int).

Параметр p_5 – признак работы с приоритетами (prty или none).

Параметр p_6 – номер узла-приемника на нижнем слое (int).

Параметр p_7 – номер узла возврата на данном слое модели, где расположен узел pay (int).

Фуикция имитации получения ресурса со склада. Имитация получения ресурсов со склада внешне похожа на работу с узлом pay. Это делается с помощью узла rent. Но сам узел rent подлежит декомпозиции на более низком уровне с помощью attach и manage. Функция rent имеет следующий вид:

$$rent(p_1,p_2,p_3,p_4,p_5,p_6)\ .$$

Для этой функции существуют шесть аргументов, подобных аргументам узла pay.

Параметр p_1 – символическое имя узла rent: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – требование на число элементов ресурса (long).

Параметр p_3 – номер узла–склада ресурсов attach, с которого необходимо получить ресурсы (int).

Параметр p_4 – признак работы с приоритетами (prty или none).

Параметр p_5 – номер узла-приемника на нижнем слое модели (int).

Параметр p_6 – номер узла возврата на данном слое, где расположен узел rent (int).

Функция перехода на нижерасположенный слой. Иногда бывает полезно большую группу узлов специально объединить в один общий узел, который находится на одном из слоев модели. Затем этот узел подвергается декомпозиции на слоях модели, которые расположены ниже. Функция, описывающая такой узел, называется down:

$$\text{down}(p_1,p_2,p_3) \ .$$

Эта функция характеризуется тремя аргументами.

Параметр p_1 – символическое имя узла down: строка длиной до 14 символов, включая пробелы (char).

Параметр p_2 – номер узла-приемника на нижнем слое модели (int).

Параметр p_3 – номер узла возврата на данном слое, где расположен узел down (int).

3.6 СИГНАЛЬНЫЕ УПРАВЛЯЮЩИЕ ФУНКЦИИ

Сигнальные управляющие функции используются не во всех узлах, а только там, где необходима специальная логика или должны использоваться функциональные уточнения особенностей узла. Сигнальные функции (табл. 3.1) можно выполнять в любом узле, кроме ag.

Функция interrupt для прерывания модели. Модель, которая, с точки зрения пользователя, выглядит отлаженной, на самом деле может таковой не быть. Она не обязательно сохранит свое быстродействие и логику работы, если экспериментатор существенно изменит входные данные и подправит под них текст модели. Поэтому, независимо от опыта пользователя, создавшего очень сложную модель, необходимо иметь возможность временной приостановки модели и просмотра промежуточных результатов, после чего возобновить счет. Если результаты окажутся подозрительно неправдоподобными, то следует прекратить моделирование или перейти в режим отладки средствами «трассировки».

Таблица 3.1

Сигнальные управляющие функции

№ п/п	Наименование функции	Назначение сигнальной функции в узле стохастической сети (кроме узла ag)
1	interrupt	Выполнить прерывание модели
2	cheg	Изменить параметры генератора транзактов ag
3	rels	Открыть клапан key
4	hold	Закрыть клапан key
5	activ	Возобновить активность обслуживания в узле proc
6	passiv	Приостановить активность обслуживания в узле proc
7	supply	Присвоить начальное значение мощности ресурса attach
8	assign	Задать начальное значение на счете бухгалтерского учета send
9	freed	Изгнать уничтожающий транзакт из узла delet
10	sewt	Перенести транзакт в другую точку пространства
11	sewk	Перенести узел в другую точку пространства
12	geoway	Определить расстояние между точками *x* и *y* на поверхности Земли по географическим координатам
13	dekart	Определить расстояние на декартовой плоскости
14	change	Заменить узел обслуживания очереди
15	clcode	Включить блок операторов языка C++

Прерывания осуществляются двумя способами:

• в процессе диалога с моделью во время ее выполнения (клавиша Esc);

• программно – с помощью функции interrupt, которую можно по какому-либо условию выполнить в любой ветви графа модели. Эта функция не имеет параметров и не возвращает никаких результатов. Она записывается следующим образом: interrupt() .

После выполнения этой функции в каком-либо узле работа модели прекращается, а экспериментатор может исследовать полученные результаты.

Функция cheg для перенастройки генератора транзактов ag. Часто бывает необходимо перенастроить генератор транзактов

ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8). Номер генератора – это параметр p_2. Функция cheg(p_2,p_3,p_4,p_5,p_6,p_7,p_8) предназначена для изменения параметров генератора ag, кроме первых двух. Номер перенастраиваемого генератора содержится в параметре p_2. Такая перенастройка произойдет за нулевое модельное время.Смысл аргументов p_3 – p_8 – тот же самый, какой имеют параметры генератора с номером p_2.

Функции rels и hold для управления клапаном key. Узел типа «клапан» управляется из других узлов. Функции rels(i) и hold(i) предназначены для управления клапаном с номером i из любых других узлов.

После выполнения rels клапан принимает состояние «открыт», если до этого он был закрыт. Соответственно после hold клапан перейдет в состояние «закрыт», если до этого он был в открытом состоянии. Обычно клапаны помещают на выходах каких-либо очередей (но могут быть и другие варианты их использования).

Функции activ и passiv для управление активностью процесса proc. Функция activ(i) переводит процесс (узел типа proc) с номером i в активное состояние, если он был пассивен. При этом возобновляются обслуживание транзакта (т.е. отсчет активного времени) и выполнение модели процесса (программы float ep). Если процесс уже был активен либо вообще узел i был пустым (нет ни транзакта, ни выполнения модели процесса), то никаких действий не осуществляется. Функция activ в общем случае выполняется в другом узле, номер которого не равен i.

Функция passiv(i) переводит процесс (узел типа proc) в пассивное состояние, если он был активен. При этом транзакт и соответствующее событие исключаются из списка планируемых событий и переводятся в стек прерванных транзактов. Прекращается и выполнение модели процесса (программы float ep). Если процесс уже был пассивен либо узел i был пустым, то никаких действий не осуществляется.

Функция supply для обеспечения начальной мощности ресурса. При работе со складом ресурсов attach в какой-то момент необходимо принудительно либо установить начальное значение, либо его изменить на заданную величину. Это делается с помощью функции supply:

supply(p_1,p_2,p_3) .

Функция имеет следующие параметры.

Параметр p_1 – номер узла attach (int).

Параметр p_2 – признак add (добавить) или none (безусловно установить).

Параметр p_3 – количество единиц ресурса (long).

Функция assign для ассигнования на счет денежной суммы. При работе со счетами send в какой-то момент необходимо принудительно либо установить начальное значение остатка денежных средств, либо его изменить на заданную величину. Это делается с помощью функции assign:

$$\text{assign}(p_1,p_2,p_3)\ .$$

Функция имеет следующие параметры.

Параметр p_1 – номер счета – узла send (int).

Параметр p_2 – признак add (добавить) или none (безусловно установить).

Параметр p_3 – значение денежной суммы (double).

Функция freed для изгнания уничтожающего транзакта из узла delet. Рассмотрим узел delet(p_1,p_2,p_3,p_4,p_5,p_6). Если в такой узел не поступят p_5 транзактов, то уничтожающий транзакт будет все время находиться в нем, блокируя его для поступления других уничтожающих транзактов. Поэтому существует задача изгнания уничтожающего транзакта из узла delet. Для изгнания используется оператор freed(i), который работает в соответствии со следующей логикой. Допустим, что в узле delet с номером i застрял транзакт. Текущий транзакт, который проходит через функцию freed(i) в каком-то другом узле, с ее помощью генерирует вспомогательный транзакт и мгновенно направляет его в узел i. Этот вспомогательный транзакт выталкивает застрявший, приводит delet в нормальное состояние, а сам погибает. Застрявший транзакт будет направлен в узел p_6, определенный в узловом операторе delet.

Факт посещения вспомогательным транзактом фиксируется в узле delet и отражается на статистических результатах, но время его жизни равно нулю.

Функции sewt и sewk для привязки транзактов и узлов к точкам пространства. Функция sewt(x) помещает текущий транзакт в точку пространства, имеющую номер x (т.е. приписывает ему координаты этой точки путем занесения значения x в параметр транзакта t→tx). Функция sewk(x,i) помещает узел с номером i в точку пространства, имеющую номер x (т.е. приписывает узлу координаты этой точки, записывая значение x в параметр узла k→kx).

Функции geoway и dekart для определения расстояния между точками пространства.

Функция geoway(latA,lonA,latB,lonB) служит для определения расстояния между точками А и В по их географическим координатам, измеряемым в радианах, причем

latA, lonA – широта и долгота точки А;

latB, lonB – координаты точки В.

Функция geoway используется имитатором автоматически внутри узла dynam, однако ее можно вызывать из любой программы пользователя. Метод расчета расстояний, используемый в ней, описан в главе 6. Расстояние между двумя пунктами определяется по основным формулам сферической тригонометрии. Радиусы Земли для разных широт вычисляются по эллипсоиду Красовского.

Функция dekart(xA,yA,xB,yB) аналогична функции geoway. Она служит для определения расстояний на декартовой плоскости в прямоугольных координатах по теореме Пифагора. Обозначим координаты точки А как xA и yA, а координаты точки В – как xB и yB. В этом случае функция dekart возвращает расстояние между точками по прямой:

$$L_{AB} = \sqrt{(xA - xB)^2 + (yA - yB)^2} \ .$$

Функция change для замены узла обслуживания очереди. Данная сигнальная функция чаще всего используется при моделировании клиринговых процессов. Она имеет вид

change (p_1,p_2).

Предположим, что узел-очередь имеет номер p_1. Этот узел относится к одному из трех типов: 1) send – счет; 2) attach – склад ресурсов; 3) queue – очередь. Очередь обслуживается каким-либо узлом. Если состав или количество транзактов в очереди перестают удовлетворять каким-то требованиям, то все транзакты из очереди необходимо направить в узел с номером p_2 и разгрузить эту очередь.

Параметр p_1 – номер узла-очереди (int).

Параметр p_2 – номер узла, в который необходимо перенаправить транзакты (int).

Функция clcode для включения блока операторов языка C++. В блоке описания узлов иногда разработчик модели может помещать свою программу (хотя чаще всего достаточно собственных средств имитатора): программные блоки на языке C++, обращения к функциям, написанным на языке C++ или Паскаль, обращения к систем-

ным вызовам. Однако это нужно делать при соблюдении определенных правил.

Во-первых, если необходимо производить сложные или длительные по времени ресурсоемкие вычисления, использовать рекурсии с некоторыми переменными (например, $x=f(x)$ или $x++$), работать с системными вызовами, обращаться к ресурсоемким вычислительным функциям, то это делается между функцией определения типа узла и соответствующим оператором place с помощью специального средства clcode.

Во-вторых, нельзя использовать операцию языка C++ типа goto не только внутри одного узла (между меткой-функцией top и соответствующим оператором place), но и для перехода в другой узел. Для принудительного завершения моделирования вместо goto можно использовать только операцию interrupt. После такой операции управление будет отобрано у координатора и передано оператору, следующему за блоком описания графа (например, функции modend).

После узлового оператора можно записать блок любых операторов языка C++, оформленный в виде

```
clcode
      {
      Группа операторов языка C++
      }
```

Если необходимо использовать блок из одного оператора, то это записывается так:

```
clcode
      Оператор языка C++ ;
```

В целом одно из назначений рассмотренных выше программных функций заключается в посылке сигналов (передаче и получении информации) из одних узлов в другие непосредственно, а не с помощью транзактов.

Выводы

1. Рассмотрен набор языковых средств описания структуры имитационной модели, логики ее работы и функциональных особенностей моделируемых процессов. Эти средства представляют собой

специализированный язык. Имеется возможность подключения пользовательских программ на C++ в любой узел, если такая необходимость возникнет.

2. Узловые функции, реализующие дискретные процессы в имитационных моделях, в целом выполняют следующие операции:

- структурную декомпозицию в модели, переходы с одного структурного слоя на другой;
- запуск и координацию процессов управления транзактами, событиями и узлами модели;
- управление материальными, информационными и денежными ресурсами;
- имитацию получения, использования и возврата материальных, информационных и денежных ресурсов.

При этом учитывается, что пути перемещения транзактов по графу модели и пути перемещения денежных ресурсов (между узлами типа «счет») – это разные маршруты.

3. В моделях используется специальный набор сигнальных функций. Эти функции не являются обязательными, однако без их помощи невозможно выполнять следующие действия:

- посылать сигналы, передавать и получать информацию из одних узлов в другие непосредственно, а не с помощью транзактов;
- управлять из одних узлов другими узлами и транзактами;
- защищать модель от «клинча» (явления блокировки), когда в модели нет ошибок, с точки зрения экономиста-пользователя, но складывается случайная ситуация, приводящая к «зависанию» компьютера в связи с замыканием нескольких процессов через совместно используемые ресурсы.

Вопросы для самопроверки

1. Что такое разомкнутые модели?
2. Как создаются замкнутые модели?
3. Для чего нужна секция инициализации модели?
4. Какой оператор является первым оператором модели?
5. Зачем нужен оператор первоначальной настройки?
6. В какой части модели помещается генератор транзактов?
7. Что такое координатор сети процессов?
8. Какие действия выполняет оператор завершения моделирования?

9. В какую часть модели помещается блок описания стохастической сети?
10. Как реализуются и какие действия выполняют операторы анализа условий?
11. Каковы основные функции узлового оператора?
12. Что такое оператор place и каковы его функции?
13. Какие возможности существуют для проведения многослойного структурного анализа?
14. Для чего нужен структурный оператор pay?
15. Каково назначение оператора rent?
16. Какую функцию выполняет структурный оператор down?
17. Какие преимущества имеет оператор parent по сравнению с pay, rent и down?
18. Для чего нужны сигнальные управляющие функции?
19. Зачем необходимо прерывание модели и как его выполнить?
20. Как и когда можно изменить настройки генератора транзактов?
21. Каким образом можно открыть или закрыть клапан key?
22. Как можно присвоить начальные значения мощности ресурса?
23. С помощью каких средств можно задавать начальные значения на счетах бухгалтерского учета?
24. Можно ли освободить уничтожающий транзакт из «зависшего» узла delet?
25. Какими средствами можно перенести узел или транзакт в другую точку пространства?
26. Какая функция позволяет определить расстояние между точками на поверхности Земли по географическим координатам с большой точностью?
27. Как определить расстояние между точками на декартовой плоскости?
28. Каким образом можно включить блок операторов C++ в состав узла модели?

Глава 4

ПРИЕМЫ ПРОГРАММИРОВАНИЯ И ОТЛАДКИ МОДЕЛЕЙ

4.1 ИСПОЛЬЗОВАНИЕ ПАРАМЕТРОВ ТРАНЗАКТОВ И УЗЛОВ

Повышение эффективности создаваемых моделей невозможно без знания некоторых параметров транзактов, узлов, глобальных переменных моделирующей системы и полезных функций, значения которых можно использовать. Аргументами рассмотренных функций системы Pilgrim могут быть числа (как целые, так и с плавающей точкой – тип определяется по смыслу), параметры транзактов и узлов, некоторые глобальные переменные и любые переменные, вводимые разработчиком модели.

Отсчет модельного времени всегда начинается с нуля, а его значения находятся в глобальной переменной timer. Единицы времени условны. Интервалы времени – это переменные с плавающей точкой.

Транзакты имеют параметры, некоторые из которых всегда доступны пользователю-разработчику модели:

t→iu0, t→iu1, t→iu2, t→iu3 – произвольные целочисленные параметры, которые могут использоваться для различных целей, например чтобы пометить транзакт;

t→ru0, t→ru1, t→ru2, t→ru3 – произвольные параметры, имеющие вид переменной с плавающей точкой;

t→ga – признак обслуживания транзакта вновь, который устанавливается операцией присваивания ему глобальной константы again, как это было показано при рассмотрении функции serv;

t→pr – приоритет транзакта;

t→ft – номер семейства, к которому принадлежит транзакт;

t→tx – параметр запоминает номер (индекс) точки пространства.

Номер точки позволяет определить ее координаты lat и lon в массиве space.

Изменять, не опасаясь непредсказуемого поведения модели, можно только параметры t→iu0, t→iu1, t→iu2, t→iu3, t→ru0, t→ru1, t→ru2, t→ru3, t→ga и иногда – параметр t→pr. Изменение других параметров следует доверить только системе Pilgrim.

Во время прохождения транзакта через узел номер этого узла находится в глобальной переменной next. Перечисленные ниже параметры узла доступны пользователю для анализа (но не для их изменения):

addr[next]→nc – число каналов в узле;

addr[next]→na – число транзактов, прошедших через узел на данный момент модельного времени;

addr[next]→tn – число транзактов, находящихся в узле в данный момент;

addr[next]→ts – среднее время обслуживания, подсчитанное на данный момент;

addr[next]→op – признак состояния узла типа key. Целочисленная переменная, принимающая значение true, если клапан открыт, или false – если он закрыт;

addr[next]→se – признак состояния узла типа delet. Целочисленная переменная, принимающая значение nil, если в узле нет уничтожающего транзакта;

addr[next]→kx – номер (индекс) точки, в которой находится узел типа creat, delet или proc. По номеру точки определяются ее координаты lat и lon.

Для анализа параметров любого узла можно использовать его номер n. Например, если транзакту необходимо проанализировать состояние узла типа delet, имеющего номер 5, и в случае отсутствия уничтожающего транзакта выполнить какой-либо оператор, он должен пройти через следующее выражение:

```
if (addr[5]→se == nil)
      оператор;
```

причем сам транзакт в это время находится в произвольном узле. Следует отметить, что возможны пробные обращения координатора network к ветви top(i) для анализа состояния узла, но не для ввода транзакта в этот узел. Факт входа транзакта фиксируется в глобальной переменной go и автоматически анализируется всеми функция-

ми Pilgrim. Поэтому еще раз необходимо обратить внимание на то, что поток управлений в программе и поток транзактов в модели – это разные явления.

Параметры узлов можно наблюдать в процессе выполнения имитационной модели.

Все необычные ситуации, мешающие нормальному продолжению моделирования, фиксируются в глобальной переменной error. Моделирование может выполняться только тогда, когда error равна нулю. Значения этой переменной от 1 до 100 зарезервированы для нужд Pilgrim.

Пользователь сам может останавливать модель, если при выполнении каких-то условий в модели этой переменной присвоить значение любой константы начиная с 101.

Например, если нет уверенности в том, что используются допустимые значения для определения номеров узлов при переходе транзактов из узла в узел, то самой последней строкой описания графа модели может быть строка fault(параметр), где параметр – любое число, например 123. В этом случае при возникновении ошибки координатор остановит модель, выведет накопленную статистику и сообщит код причины остановки модели – число 123.

Часто необходимо получить случайную величину в формате float, распределенную по какому-то закону. В системе Pilgrim есть стандартные 32-разрядные датчики псевдослучайных величин. В каждом узле имеется свой датчик, независимый от датчиков других узлов.

Введем в рассмотрение переменную пользователя типа float v. Связь с датчиками осуществляется с помощью следующих функций:

v = normal(m,s) – нормальный закон распределения;
v = expont(m) – экспоненциальный закон;
v = unifrm(m,r) – равномерный закон на отрезке [m–r,m+r];
v = rundum() – равномерный закон на отрезке [0,1];
v = erlang(e,z) – обобщенный закон Эрланга;
v = triplex(a,b,c)– треугольный закон распределения.

В данном случае использованы следующие обозначения входных параметров типа float:

m – математическое ожидание (в случаях normal, expont, unifrm);
s – среднеквадратичное отклонение (в случае normal);

e – матожидание величины одного элемента (в случае erlang);
r – максимальное отклонение (в случае unifrm);
z – количество отрезков z > 0 (в случае erlang);
a – минимальное значение (в случае triplex);
b – наиболее вероятное значение (в случае triplex);
c – максимальное значение (для triplex).

При обращении к одной из этих функций используется датчик того узла, в котором находится транзакт.

4.2 ОТЛАДКА МОДЕЛЕЙ В ПРОЦЕССЕ ИХ ВЫПОЛНЕНИЯ

Часто синтаксически правильная модель выдает неправильные результаты, что свидетельствует либо о неадекватности модели, либо о логической (семантической) ошибке в ее описании. Для отладки моделей совместно используются два вида выходной информации: итоговая таблица с результатами моделирования и динамически изменяющиеся отладочные таблицы, появляющиеся в специальном окне в режиме трассировки модели.

Итоговая таблица состоит из 12 столбцов, каждый из которых несет определенную смысловую нагрузку (табл. 4.1). Если в выходном документе меньше 12 столбцов, то их взаимное расположение не изменяется.

Таблица 4.1

Трактовка итоговой таблицы результатов моделирования

№ п/п	Наименование столбца	Содержание столбца
1	№ узла	Номер узла
2	Наименование узла	Смысловое название узла в конкретной модели
3	Тип узла	Один из типов: ag, serv, key, queue, term, creat, delet, proc, dynam, send, direct, attach, manage, pay, rent, down или parent

Продолжение

№ п/п	Наименование столбца	Содержание столбца
4	Точка	Номер последней точки пространства, в которой находится узел типа creat, delet или proc в момент окончания моделирования
5	Загрузка (%=)	Загрузка (коэффициент использования транзактами) узлов типа serv или proc в процентах. Для узла key – доля времени пребывания в закрытом состоянии
6	Путь, км	Если производятся пространственные перемещения узлов типа proc, creat или delet, то подсчитывается пройденный путь. Для пространств типа geo путь считается в километрах
7	M [t] среднее время	Среднее значение времени задержки транзакта в узле или иной интервал времени, зависящий от типа узла: 1) для serv – это среднее время пребывания в узле (оно может быть больше времени обслуживания у неприоритетных транзактов при p_3=abs, т.е. при наличии приоритетных транзактов и правила абсолютных приоритетов); 2) для queue – среднее время задержки в очереди; 3) для ag – среднее время между двумя сгенерированными транзактами; 4) для term или delet – среднее время существования транзакта; 5) для key – среднее время пребывания в закрытом состоянии; 6) для creat и dynam – всегда нулевое значение; 7) для proc при p_4=none, p_4=norm, p_4=expo или p_4=unif –среднее время пребывания в узле (оно может быть больше времени обслуживания транзакта при переводе узла в пассивное состояние); 8) для proc при p_4=earth, p_4=plane или p_4=cosmos –суммарное время пребывания транзакта в узлах dynam и proc с учетом возможных возвратов транзактов из proc в dynam
8	C^2 [t] квадрат коэффициента вариации	Отношение дисперсии временного интервала к квадрату его среднего значения – коэффициент вариации, возведенный в квадрат

Продолжение

№ п/п	Наименование столбца	Содержание столбца
9	Счетчик входов и hold	Число транзактов: • прошедших через узел; • сгенерированных транзактов (для ag или creat); • уничтоженных (для term или delet); • выполнивших операцию hold из другого узла в отношении узла key
10	Количество каналов	Число каналов в узле
11	Осталось транзактов	Количество транзактов, которые остались в узле на момент завершения моделирования
12	Состояние узла в этот момент	Состояние узла в момент окончания прогона модели: узел может быть открыт (свободен), закрыт для входа очередного транзакта, активен или пассивен (узел proc). В узле типа send может быть положительное сальдо на момент завершения модели (денежная сумма с буквой S) и отрицательное сальдо (сумма с буквой D). В узле типа attach может быть остаток ресурса на момент завершения модели (денежная сумма с буквой S) и дефицит (сумма с буквой D). В узлах типа pay и rent указывается количество переходов транзактов на нижние уровни – слои модели (на момент завершения модели)

В режим трассировки модель можно перевести, используя меню модели или панель ее инструментов. Можно запустить модель сразу в режиме трассировки. Окно фрагмента трассировки показано на рис. 4.1. Содержимое окна по умолчанию изменяется после каждого события. Однако число событий в модели может быть очень велико, поэтому существуют следующие режимы трассировки, позволяющие ускорить процесс отладки:

- выйти в режим трассировки после конкретного события;
- перейти в трассировку, если какой-то транзакт входит в определенный узел или выходит из него;
- отслеживать путь конкретного транзакта по сложному графу модели;
- выйти в режим трассировки по показанию модельного таймера.

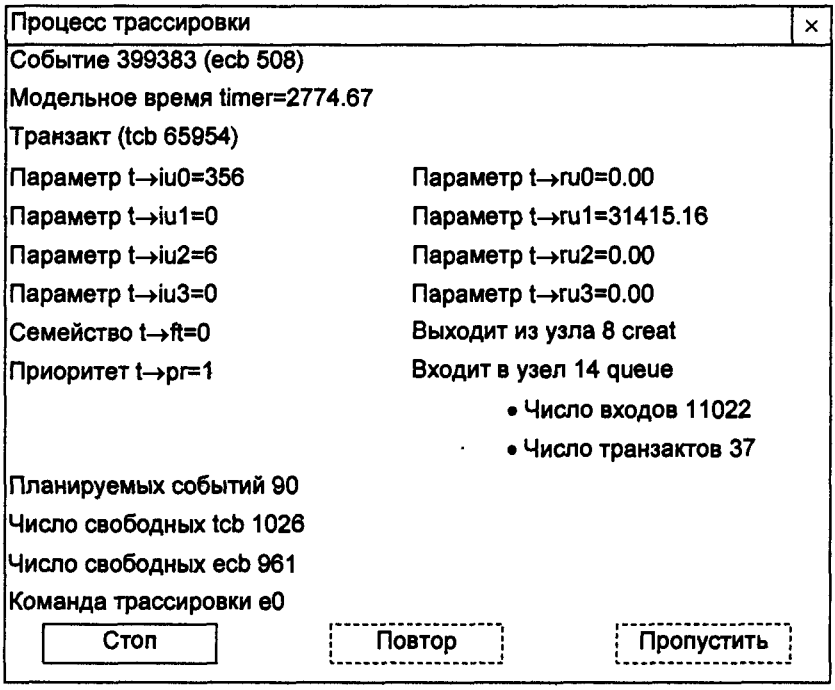


Рис. 4.1. Диалоговое окно трассировки

Управление трассировкой осуществляется из общего меню модельного окна.

4.3 ПРОГРАММИРОВАНИЕ УСЛОВИЙ ПРОХОЖДЕНИЯ ТРАНЗАКТА ПО ГРАФУ МОДЕЛИ

Рассмотрим случайный выбор из класса узлов. При построении моделей часто может встретиться ситуация, когда какой-либо объект имеет несколько подчиненных объектов, объединенных общим именем класса, в который они попадают, и требуется решить, в какой из подчиненных узлов направлять транзакт. В этом случае можно поступить, например, следующим образом: транзакт тем чаще направляется в узел, чем больший вес он имеет. Под *весом* в данном случае

понимается любая количественная характеристика объекта, по которой может производиться сравнение. На рис. 4.2 приведен пример фрагмента графа модели.

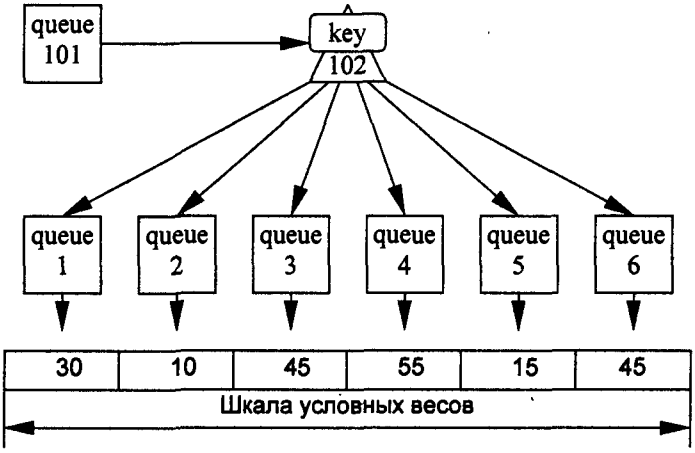


Рис. 4.2. Фрагмент графа модели

Условие выбора узла с номером k выглядит следующим образом:

$$\sum_{i=1}^{k-1} \frac{\text{Вес}_i}{\text{Сумма\_весов}} < p \le \sum_{i=1}^{k} \frac{\text{Вес}_i}{\text{Сумма\_весов}},$$

где p – случайная величина, равномерно распределенная на отрезке [0 , 1].

Условие нормирования записывается так:

$$\text{Сумма\_весов} = \sum_{i=1}^{k} \text{Вес}_i .$$

Следует учесть, что

$$\sum_{i=1}^{0} \frac{\text{Вес}_i}{\text{Сумма\_весов}} = 0 .$$

Приведенное соотношение следует из рис. 4.3, на котором показано, как реализовано дискретное распределение вероятностей. Заштрихованные прямоугольники изображают отношение веса каждо-

го объекта к общей сумме весов (весовую долю); пунктирные линии определяют границы интервала, при попадании в который величины p в данном примере будет выбран узел 3. Вероятность выбора узла 3 самая высокая, так как он имеет наибольший вес.

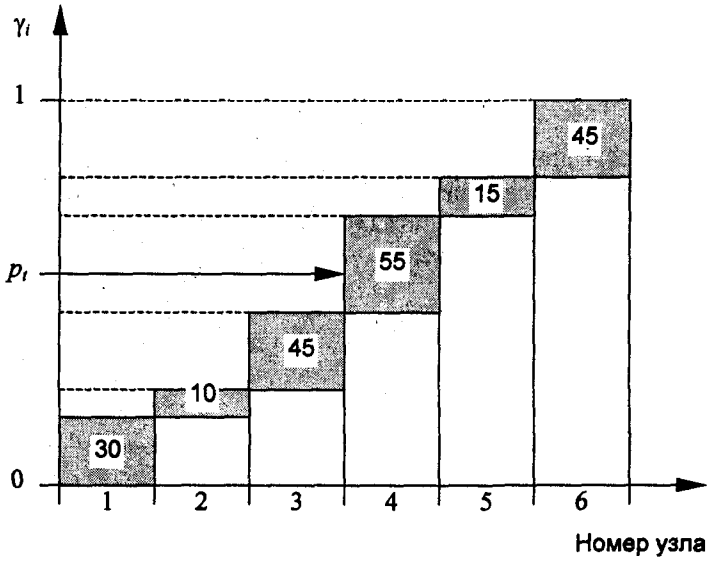


Рис. 4.3. Схема выбора узлов по их условным весам

Функция Number, определяющая номер выбираемого узла, показана ниже. Входными параметрами являются массив весов, число узлов и суммарный вес. Функция возвращает номер элемента массива весов.

Накапливаемое значение p0 соответствует суммам площадей прямоугольников на рис. 4.3. Оно будет накапливаться до тех пор, пока функция Number не доберется до нужного столбика.

```
int Number (float *weight, int n, float sum)
   {
   int i, k;
   float p, p0;
   p0=0.0;
   p=rundum();    // Датчик случайных чисел Pilgrim
                  // на отрезке [0, 1]
```

```
for (i=0; i<n; i++)
   {
   k=i;
   if (( p >= p0) && ( p <= p0+weight[i]/sum ))
      break;
   else
      p0 += weight[i] / sum;
   }
return (k+1);
}
```

На рис. 4.4 представлена схема выбора дальнейшего пути продвижения транзакта в зависимости от значения некоторого его параметра. В этом случае блок выбора номера следующего узла с использованием параметра транзакта t→iu0 может выглядеть следующим образом:

Рис. 4.4. Схема выбора пути в соответствии с параметром транзакта

```
top( Номер_узла_ветвления ):
   switch( Параметр_транзакта )
      {
      case(значение_1):t→iu0=(Номер_узла_1);break;
      case(значение_2):t→iu0=(Номер_узла_2);break;
      …
      case(значение_m):t→iu0=(Номер_узла_m);break;
      }
   Тип_узла_ветвления(Параметры_узла, t→iu0);
   place;
```

Существуют следующие рекомендации по нумерации узлов составных объектов одного класса. Если в модели имеется несколько принадлежащих к одному классу объектов, каждый из которых представлен несколькими узлами, то целесообразно нумеровать узлы, относящиеся к каждому объекту, таким образом, чтобы номера соответствующих узлов разных объектов были функцией от номера объекта данного класса (рис. 4.5). В этом случае типовые блоки обработки транзактов в одних и тех же узлах разных объектов можно оформить в виде процедур, вызываемых с параметром «Номер объекта», вместо того чтобы переписывать одни и те же операторы обработки, изменяя лишь номера узлов. Таким образом, для каждого класса объектов, входящих в модель, могут быть составлены наборы стандартных модулей обработки транзактов, что значительно упростит программирование модели.

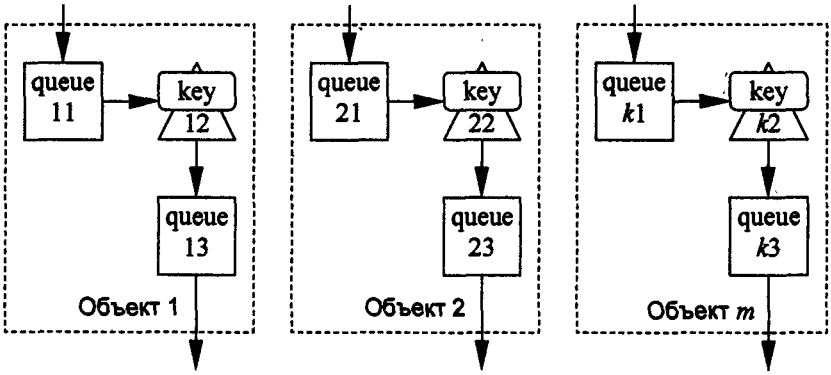


Рис. 4.5. Пример объектно-зависимой нумерации узлов:
$k1=10m+1$; $k2=10m+2$; $k3=10m+3$

4.4
ОСОБЕННОСТИ ЗАМКНУТЫХ МОДЕЛЕЙ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Под *замкнутой моделью* корпоративной информационной системы (КИС) будем понимать модель системы, работающей в режиме «запрос-ответ», в которой транзакты, единожды сгенерированные, циркулируют в пределах графа модели, не погибая в терминаторах. Например, это может быть работа группы пользователей с компьютерной сетью в диалоговом режиме. Транзакт – это запрос пользователя. Выйдя из генератора, транзакт проходит по графу модели и постепенно, по мере работы КИС, превращается в ответ и возвращается к пользователю, после чего вновь начинает играть роль запроса того же пользователя.

Моделировать новый запрос новым сгенерированным транзактом не представляется возможным, так как нельзя рассчитать заранее время обработки запроса системой (это случайная величина) и, следовательно, нельзя задать частоту генератора. В данном случае модель может быть построена следующим образом:

- пользователи (или группы пользователей, в зависимости от сложности моделируемой системы) представляются одно- или многоканальными узлами типа serv (серверы);
- число каналов сервера соответствует числу пользователей, время обработки транзакта сервером соответствует времени подготовки пользователями запроса;
- приписываемая транзакту роль запроса или ответа обозначается в одном из его параметров;
- для зарядки транзактами серверов пользователей, принадлежащих к одному классу, используется единственный генератор, порождающий всего один транзакт. Далее транзакты размножаются с помощью узлов типа creat.

Ниже представлены варианты зарядки модели транзактами.

1. *Зарядка одного многоканального сервера* (рис. 4.6). Описание узлов имеет следующий вид:

```
ag("Один транзакт",1,none,none,1.0,zero,zero,2);
...
top(2): creat("Размножение",4,Users,none,4,3);
        place;
top(3): term("Выключение");
```

```
        cheg(1,none,none,Modtime,zero,zero,3);
        place;
top(4): serv("Пользователи",Users,none,expo,
              Think_time,zero,zero,узел_приемник);
        clcode
           t→iu0=QUERY;
        place;
```

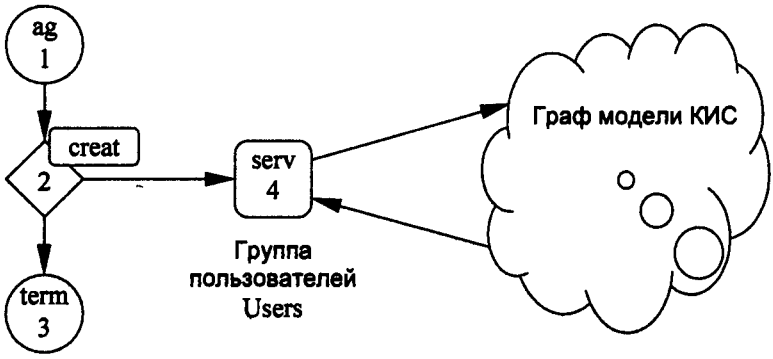


Рис. 4.6. Модель корпоративной системы с одним многоканальным сервером для моделирования поведения клиентов

Генератор (узел 1) в течение единицы модельного времени генерирует один транзакт, который попадает в узел creat (узел 2) в качестве порождающего. Узел 2 генерирует (порождает) семейство транзактов количеством Users (модельная константа – число пользователей) и с номером семейства 4 (номер узла, моделирующего работу пользователей). Порождающий транзакт погибает в терминаторе (узел 3). В этот же момент происходит перенастройка генератора (оператор cheg): интервал между генерацией транзактов устанавливается равным времени моделирования Modtime (это же время задано в modbeg). После этого генератор «выключается», так как за время моделирования больше не успевает выпустить ни одного транзакта.

Порожденные узлом creat транзакты попадают в узел serv (узел 4), моделирующий работу пользователей. Число каналов в нем равно числу вошедших в него транзактов (константа Users), а среднее время обслуживания транзакта равно среднему времени обдумывания пользователем запроса (модельная константа Think-time).

Далее – перед выходом транзакта из узла – ему присваивается начальная метка «запрос» (в параметр t→iu0 заносится модельная константа QUERY).

Таким образом, по графу модели начинает циркулировать столько запросов, сколько имеется пользователей. Адрес же возврата запроса можно определить по значению параметра t→ft «номер семейства». В случае с одним сервером пользователей эта проблема не столь актуальна, тогда как в двух описанных далее случаях этот способ очень удобен.

2. *Зарядка нескольких одноканальных серверов* (рис.4.7). Описание узлов имеет следующий вид:

```
ag("Один транзакт",1,none,none,1.0,zero,zero,2);
...
top(2): creat("Размножение",none,Users,none,4,3);
        place;
top(3): term("Выключение");
        cheg(1,none,none,modtime,zero,zero,3);
        place;
top(4): t→ft  = addr[4]→na + 5;
        t→ru0 = Think_time;
        t→iu0 = QUERY;
        queue("Прием",none,t→ft);
        place;
```

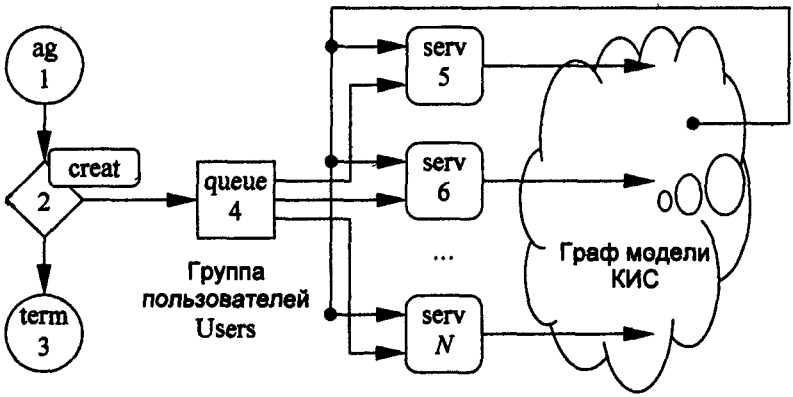


Рис. 4.7. Модель корпоративной системы с несколькими одноканальными серверами для моделирования поведения клиентов: N = Users + 4

Отличие этой схемы от варианта 1 состоит в введении дополнительного узла queue (узел 4). Узел 2 creat генерирует Users транзактов (по числу пользователей), не присваивая им номер семейства. Далее эти транзакты попадают в очередь (узел 4), где каждому из них присваивается свой номер семейства, равный номеру сервера, в который этот транзакт поступит. Также в параметр транзакта t→ru0 заносится среднее время задержки в сервере, моделирующем работу пользователя. Этот параметр используется при описании серверов пользователей (узлов с номерами из диапазона [5, Users + 4], где Users – число серверов).

3. *Зарядка нескольких многоканальных серверов.* Этот вариант (рис. 4.8) отличается от варианта 2 введением второго узла creat (узел 5). Он получает последовательно из очереди (узел 4) порождающие транзакты, уже обладающие необходимыми параметрами для каждого сервера, моделирующего работу группы пользователей (*см.* описание узла 4 в варианте 2). Эти транзакты становятся образцами для порождаемых семейств. Количество транзактов, генерируемых дополнительно для каждого сервера, равно числу моделируемых им пользователей минус 1, так как порождающий транзакт-образец тоже используется в качестве запроса и поступает в тот же сервер, что и порожденные.

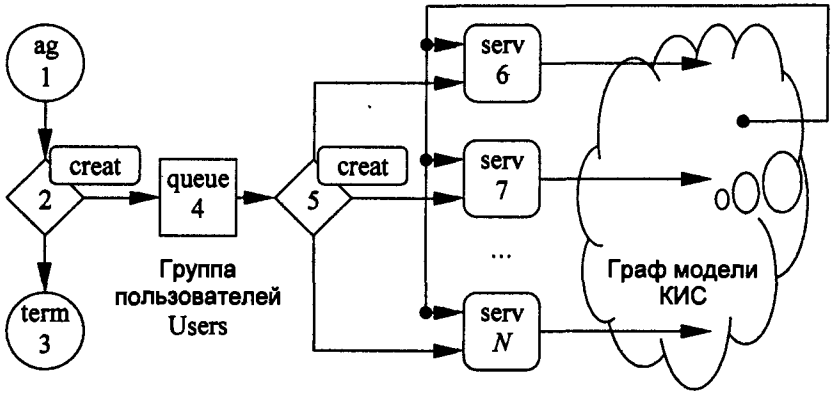


Рис. 4.8. Модель корпоративной системы
с несколькими многоканальными серверами
для моделирования поведения клиентов: N = Users + 5

В данном случае Users – это число групп пользователей. Описание второго узла creat имеет вид:

```
top(5): creat("Размножение",t→ft,
              Число_каналов-1,copy,t→ft,t→ft);
        place;
```

В операторе queue нужно также заменить параметр «номер семейства» t→ft на номер 5, так как из очереди все транзакты идут в узел 5 creat. Для описания серверов число пользователей можно задать в некотором массиве, доступ к которому осуществляется через значение параметра транзакта «номер семейства», равный номеру соответствующего сервера.

Далее рассмотрим возможности определения времени ответа на запрос в КИС. При моделировании работы корпоративных информационных систем может быть поставлен вопрос, сколько времени в среднем тратит система на обработку одного запроса пользователя. Существуют три способа решения этой задачи:

1) добавить в модель дополнительный узел key и с его помощью проводить замеры времени реакции;

2) получить с помощью модели средние времена пребывания транзактов во всех узлах имитационной модели КИС и использовать эти данные для расчета времени реакции;

3) использовать только среднее время подготовки конкретным пользователем запроса (или задания) для КИС, полученное с помощью модели, и загрузку узла serv, имитирующего этого пользователя.

Рассмотрим эти способы подробнее.

1. Через дополнительный узел key (рис. 4.9) транзакты не проходят; он служит только для измерений. Каждый пользователь имитируется одним сервером. Дополнительно каждому пользователю приписывается узел key. На рис. 4.9 показана работа одного пользователя. Транзакт – задание для КИС – предварительно проходит через key и запирает его. Далее, после обработки запроса, этот транзакт возвращается из модели КИС в этот же сервер, но при входе в узел serv он открывает key. В модели автоматически измеряется интервал закрытого состояния узла key и определяются математическое ожидание и среднеквадратичное отклонение.

Достоинство способа: измерения очень точные при большом числе транзактов. *Недостаток*: необходимо для каждого пользователя ввести в модель и описать дополнительный узел key, что усложняет модель.

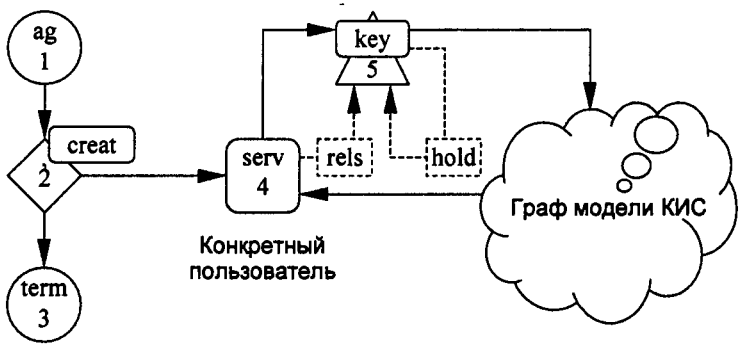


Рис. 4.9. Пример измерения среднего времени реакции системы с помощью дополнительного узла key

2. Можно получить с помощью модели средние времена пребывания транзактов во всех узлах имитационной модели КИС. Далее необходимо определить средние значения вероятностей переходов для всех узлов в модели (это несложно). Затем, используя нетрадиционные методы теории стохастических сетей (*см.* главу 1), определим среднее время пребывания транзакта в графе модели КИС.

Достоинство способа: применяется известная расчетная методика.

Недостатки: а) метод приближенный, так как вероятности могут меняться во времени; б) расчетная методика довольно сложная.

3. Среднее время подготовки конкретным пользователем запроса (или задания) для КИС и загрузка соответствующего узла serv, имитирующего этого пользователя, определяются в модели автоматически. Причем эти измерения всегда точны: и в случае моделирования группы типовых пользователей одним многоканальным узлом serv, и в случае моделирования каждого уникального пользователя отдельным одноканальным узлом serv. Далее опишем способ измерения времени реакции КИС подробнее.

Рассмотрим временную диаграмму работы уникального пользователя с КИС (рис. 4.10). На достаточно длинном интервале времени моделирования t_m происходит n обращений к КИС. Среднее время t_u подготовки пользователем каждого задания КИС автоматически определяется по формуле

$$t_u = \frac{1}{n}\left(\sum_{i=1}^{n} t_i\right).$$

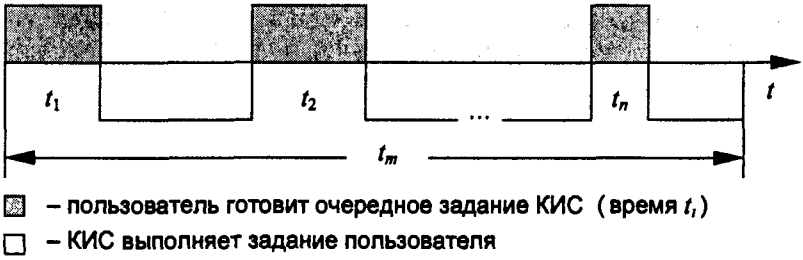


Рис. 4.10. Временная диаграмма работы пользователя и системы

Загрузка ρ узла serv транзактом – это доля всего времени t_m, которое тратится пользователем на подготовку каждого нового задания КИС, причем справедлива формула

$$\rho = \frac{1}{t_m}\left(\sum_{i=1}^{n} t_i\right).$$

Нетрудно заметить, что доля времени моделирования, в течение которого КИС обрабатывала n запросов, равна 1 – ρ. Поэтому справедливо соотношение

$$\rho t_m + (1-\rho)t_m = t_m.$$

Разделим левую и правую части этого равенства на n и получим следующее выражение:

$$t_u + \frac{(1-\rho)t_m}{n} = t_c,$$

где t_c – среднее время одного цикла «запрос-ответ» при выполнении всех n заданий пользователя:

$$t_c = \frac{t_m}{n}.$$

Время цикла состоит из двух слагаемых: времени подготовки задания и времени обработки этого задания – времени реакции КИС. Поэтому запишем

$$t_u + t_r = t_c,$$

где t_r – искомое время реакции КИС.

В результате моделирования нам известны t_u и ρ. В связи с вышеизложенным запишем систему уравнений

$$\begin{cases} \dfrac{t_u}{\rho} = t_c; \\ \dfrac{t_r}{(1-\rho)} = t_c, \end{cases}$$

из которой получим время реакции КИС:

$$t_r = \frac{1-\rho}{\rho} t_u \ .$$

Достоинства способа: а) он имеет такую же высокую точность, как при работе с дополнительным узлом key; б) этот способ значительно проще, чем способы 1 и 2: действительно, расчет по формуле значительно проще, чем включение *n* дополнительных узлов key или решение системы из *n* нелинейных уравнений методом итераций.

Недостаток: не определяется среднеквадратичное отклонение.

4.5 СОСТАВ ПРОЕКТА И ПРИМЕНЕНИЕ ОБОЛОЧКИ DEVELOPER STUDIO

Обычно имитационная модель создается в среде Developer Studio с применением средств Visual C++. Проект относительно разработки программ – это приложение Windows. Предположим, что среда Developer Studio расположена на жестком диске C в папке Msdev (путь c:\Msdev), а файл с текстом модели в терминах Pilgrim находится в папке с именем ModelDir на жестком диске D (путь d:\ModelDir). Текст модели – это файл, созданный текстовым редактором Developer Studio или Notepad, имеющий cpp-окончание (суффикс), например ModelText.cpp. Для определенности назовем наш проект ModelPro.

Рассмотрим типовые технологические последовательности действий, необходимые при создании, модернизации и выполнении моделей. Во всех режимах используются три клавиши меню Developer Studio: File – файл, Insert – вставить и Build – построить.

Создание моделей. Создание новой модели обычно проходит с использованием следующего пути по меню и подменю: File→ New → Project WorkSpace → режим Application → выбор папки ModelDir (можно через режим Browse) → указание имени проекта ModelPro. В результате будет создан файл управления проектом ModelPro.mdf.

Дальнейшие действия различаются в зависимости от типа проекта:

- типовой проект модели;
- проект модели с диалоговым окном для управления параметрами при ее запуске или во время выполнения;
- проект модели с функциональным окном для конечного пользователя (вывод информации в понятном для потребителя виде).

Типовой проект модели. Создание типового проекта не требует от разработчика знаний языка C++. Проект состоит из четырех файлов:

- текст модели ModelText.cpp. Находится в папке d:\ModelDir;
- общая библиотека Comctl32.lib, содержащая средства Windows и Pilgrim. Ее состав слабо изменяется при разработке новых версий моделирующей системы Pilgrim. Она находится в папке c:\Msdev\Lib;
- библиотека Pilgrim.lib. Существенно изменяется в процессе развития системы Pilgrim. Ее можно найти в папке c:\Msdev\Lib;
- файл ресурсов Pilgrim.res. Также изменяется в процессе развития системы Pilgrim. Его можно найти в папке c:\Msdev\Projects.

Все вышеперечисленные файлы подключаются к проекту с использованием клавиши Insert общего меню Developer Studio. Обычно в меню и подменю выполняются действия: Insert → Files into Project → выбор суффикса (cpp, lib, res) → определение пути к файлу → подключение.

Далее выполняются компиляция и сборка модели: выбирается клавиша Build общего меню Developer Studio и режим Rebuild All. В результате создается выполняемая модель: файл ModelPro.exe.

Запуск готовой модели из общего меню можно выполнить так: Build → Execute ModelPro.exe.

Проект модели с диалоговым окном для управления параметрами. Создание такого проекта требует от разработчика некоторых знаний Visual C++.

Сначала необходимо скопировать из папки c:\Msdev\Projects в папку d:\ModelDir следующие пять файлов: Parametr.cpp,

Palette.bmp, Pilgrim.ico, UserHid.h, UserRes.rc. Проект имеет следующий состав:

- текст модели ModelText.cpp. Находится в папке d:\ModelDir;
- типовая программа Parametr.cpp. Программа написана на языке C++; в зависимости от количества регулируемых параметров ее необходимо настраивать на модель. Она также находится в папке d:\ModelDir;
- общая библиотека Comctl32.lib, содержащая средства Windows и Pilgrim. Находится в папке c:\Msdev\Lib;
- библиотека Pilgrim.lib. Она существенно изменяется в процессе развития системы Pilgrim. Находится в папке c:\Msdev\Lib;
- файл ресурсов UserRes.rc. Находится в папке d:\ModelDir. Написан на языке создания Windows-ресурсов. В зависимости от количества регулируемых параметров этот файл необходимо настраивать на модель.

Прежде чем компилировать и собирать модель, рассмотрим отдельно файл ресурсов UserRes.rc и типовую программу Parametr.cpp.

Файл ресурсов имеет две части: неизменяемую (стандартную) и изменяемую. Макет файла – это UserRes.rc. Текст изменяемой части на языке описания ресурсов Vsual C++ приведен ниже (*см.* приложение 1).

```
//-----------------------------------------------
//            ИЗМЕНЯЕМАЯ ЧАСТЬ МАКЕТА
//-----------------------------------------------
PARMBOX DIALOG DISCARDABLE  1, 28, 315, 202
STYLE WS_POPUP | WS_CAPTION
CAPTION "Диалоговая настройка модели"
BEGIN
 GROUPBOX "Параметры бизнес-процесса",
          100,5,3,245,30,WS_TABSTOP
 LTEXT    "Фирма:",    -1, 10, 20,30,8,NOT WS_GROUP
 EDITTEXT DM_TITLE,40,18,200,12
 LTEXT    "Единица времени",        -1, 78,38,80,8,
          NOT WS_GROUP
 LTEXT    "Время моделирования",    -1, 63,53,95,8,
          NOT WS_GROUP
 LTEXT    "Прибытие пассажиров",    -1, 63,68,95,8,
          NOT WS_GROUP
 LTEXT    "Интервал подхода такси",-1, 63,83,95,8,
          NOT WS_GROUP
 LTEXT    "Период времени",         -1,229,38,80,8,
          NOT WS_GROUP
 LTEXT    "Окошко  4",-1, 63, 98,95,8,NOT WS_GROUP
```

```
 LTEXT     "Окошко  5",-1,  63,113,95,8,NOT WS_GROUP
 LTEXT     "Окошко  6",-1,  63,128,95,8,NOT WS_GROUP
 LTEXT     "Окошко  7",-1,  63,143,95,8,NOT WS_GROUP
 LTEXT     "Окошко  8",-1,  63,158,95,8,NOT WS_GROUP
 LTEXT     "Окошко  9",-1,  63,173,95,8,NOT WS_GROUP
 LTEXT     "Окошко 10",-1,  63,188,95,8,NOT WS_GROUP
 LTEXT     "Окошко 11",-1,214,  53,95,8,NOT WS_GROUP
 LTEXT     "Окошко 12",-1,214,  68,95,8,NOT WS_GROUP
 LTEXT     "Окошко 13",-1,214,  83 95,8,NOT WS_GROUP
 LTEXT     "Окошко 14",-1,214,  98,95,8,NOT WS_GROUP
 LTEXT     "Окошко 15",-1,214,113,95,8,NOT WS_GROUP
 LTEXT     "Окошко 16",-1,214,128,95,8,NOT WS_GROUP
 LTEXT     "Окошко 17",-1,214,143,95,8,NOT WS_GROUP
 LTEXT     "Окошко 18",-1,214,158,95,8,NOT WS_GROUP
 LTEXT     "Окошко 19",-1,214,173,95,8,NOT WS_GROUP
 LTEXT     "Окошко 20",-1,214,188,95,8,NOT WS_GROUP
 EDITTEXT    DM_T1,    5, 36,70,12
 EDITTEXT    DM_P1,    5, 51,55,12
 EDITTEXT    DM_P2,    5, 66,55,12
 EDITTEXT    DM_P3,    5, 81,55,12
 EDITTEXT    DM_P4,    5, 96,55,12
 EDITTEXT    DM_P5,    5,111,55,12
 EDITTEXT    DM_P6,    5,126,55,12
 EDITTEXT    DM_P7,    5,141,55,12
 EDITTEXT    DM_P8,    5,156,55,12
 EDITTEXT    DM_P9,    5,171,55,12
 EDITTEXT    DM_P10,   5,186,55,12
 EDITTEXT    DM_T2,  156, 36,70,12
 EDITTEXT    DM_P11,156, 51,55,12
 EDITTEXT    DM_P12,156, 66,55,12
 EDITTEXT    DM_P13,156, 81,55,12
 EDITTEXT    DM_P14,156, 96,55,12
 EDITTEXT    DM_P15,156,111,55,12
 EDITTEXT    DM_P16,156,126,55,12
 EDITTEXT    DM_P17,156,141,55,12
 EDITTEXT    DM_P18,156,156,55,12
 EDITTEXT    DM_P19,156,171,55,12
 EDITTEXT    DM_P20,156,186,55,12
 PUSHBUTTON "Готово",IDOK,     265, 7,34,12
 PUSHBUTTON "Отмена",IDCANCEL,265,21,34,12
END
```

Этот файл обеспечивает вывод диалогового окна для управления 20 параметрами модели (рис. 4.11). Если такое количество велико, то можно изменить (сократить) текст файла UserRes.rc для сокращения размеров этого окна и уменьшения количества наблюдаемых или управляемых параметров модели. Кроме того, можно задать совсем другое расположение внутренних окошек для наблюдения за параметрами.

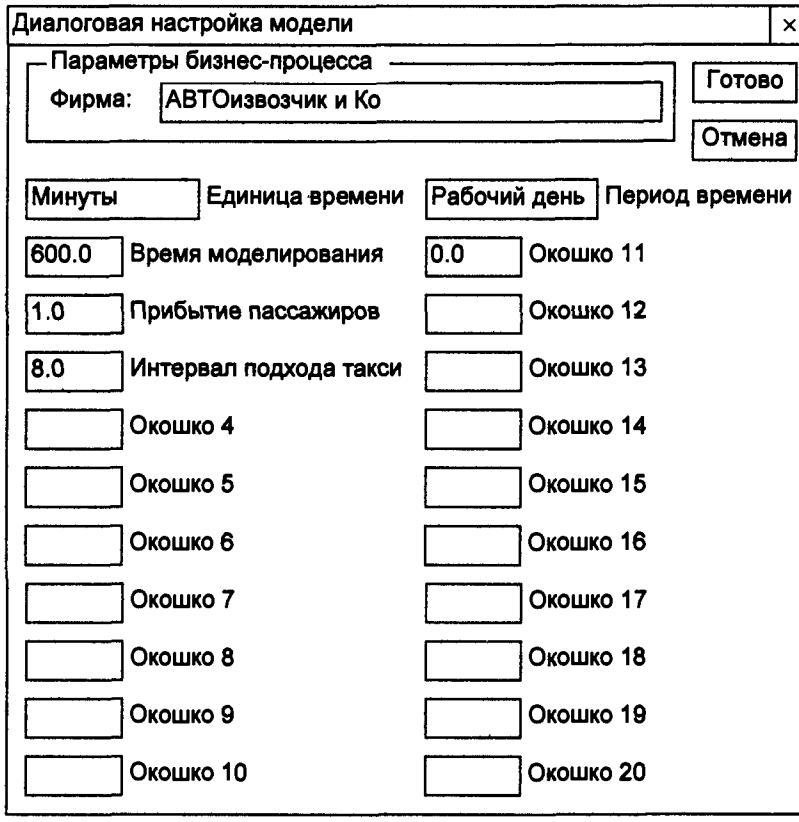


Рис. 4.11. Диалоговое окно для управления параметрами модели

В данном примере количество окошек избыточно, задействовано только 3 из 20 для отображения и управления тремя параметрами модели: 1) float win01 «Время моделирования»; 2) float win02 «Прибытие пассажиров»; 3) float win03 «Интервал подхода такси». Зарезервировано одно окошко (с номером 11) для отображения параметра float win11, который в модели не используется.

Далее рассмотрим текст программы, которая выводит численные значения параметров модели и позволяет их корректировать. Обращение к этой программе производится с помощью вызова Parametr из любого места (или любого узла) модели. Описание макета такой программы приведено ниже.

```
#include <Pilgrim.h>
#include "UserHid.h"
extern char  commtext[];
extern char  lefttext[];
extern float win01;
extern float win02;
extern float win03;
extern char  rightext[];
extern float win11;
//-------------------------------------------------
//     Диалог настройки параметров процесса
//-------------------------------------------------
BOOL CALLBACK ParametrFunc(HWND   hdwnd,
                           UINT   message,
                           WPARAM wParam,
                           LPARAM lParam)
 {
 switch(message)
  {
  case WM_INITDIALOG:
   SetDlgItemText(hdwnd,DM_TITLE,commtext);
   SetDlgItemText(hdwnd,DM_T1, lefttext);
   sprintf(str,"%4.2f",win01);
   SetDlgItemText(hdwnd,DM_P1, str     );
   sprintf(str,"%4.2f",win02);
   SetDlgItemText(hdwnd,DM_P2, str     );
   sprintf(str,"%4.2f",win03);
   SetDlgItemText(hdwnd,DM_P3, str     );
   SetDlgItemText(hdwnd,DM_T2, rightext);
   sprintf(str,"%4.2f",win11);
   SetDlgItemText(hdwnd,DM_P11,str     );
   return FALSE;
  case WM_COMMAND:
   switch(wParam)
    {
    case IDOK:
     GetDlgItemText(hdwnd,DM_TITLE,commtext,80);
     GetDlgItemText(hdwnd,DM_T1,   lefttext,80);
     GetDlgItemText(hdwnd,DM_P1,   str,     80);
     sscanf(str,"%f",&win01);
     GetDlgItemText(hdwnd,DM_P2,   str,     80);
     sscanf(str,"%f",&win02);
     GetDlgItemText(hdwnd,DM_P3,   str,     80);
     sscanf(str,"%f",&win03);
     GetDlgItemText(hdwnd,DM_T2,   rightext,80);
     GetDlgItemText(hdwnd,DM_P11,  str,     80);
     sscanf(str,"%f",&win11);
     EndDialog(hdwnd,TRUE);
     break;
```

```
    case IDCANCEL:
     EndDialog(hdwnd,FALSE);
     break;
     default:
     return FALSE;
    }
   break;
  default:
  return FALSE;
  }
 return TRUE;
 }
```

Начальные значения переменных win01, win02, win03, win11, а также заголовки и надписи задаются в головной части текста модели:

```
#include <Pilgrim.h>              // Модель с диалогом
char commtext[81]="АВТОизвозчик и Ко"; // Название
char lefttext[81] ="Минуты";   // Время в минутах
float win01         =   600.0; // Моделируем 10 час
float win02         =     1.0; // Приход такси
float win03         =     8.0; // Приход пассажиров
char rightext[81]="Рабочий день";          // Период
float win11=0.0;              // Резерв (окно 11)
forward
   {
   Parametr;                  // Диалог настройки
   modbeg("Стоянка такси",8,
          win01,(long)time(NULL),none,1,none,4,2);
   …
   modend("Res_taxi.doc",1,12,page);
   return 0;
   }
```

После возможной корректировки файлов UserRes.rc и Parametr.cpp выполняется компиляция и сборка модели, для чего используется клавиша Build общего меню Developer Studio и режим Rebuild All. В результате создается выполняемая модель ModelPro.exe. Запуск модели: Build → Execute ModelPro.exe.

Проект модели с функциональным окном для конечного пользователя. Часто бывает необходимо показать конечному пользователю информацию в понятном ему проблемно-ориентированном виде. Это могут быть изменяющиеся таблички, графики, перемещающиеся по экрану изображения (например, автомобили). Такой пользователь не будет изучать моделирующую систему. Поэтому для него средствами Visual C++ создается специальная функциональная программа

funcwindow, которая помещается в динамически вызываемую библиотеку Windows (dll-библиотеку). Обращение к такой библиотеке производится после каждого события в модели. После первого события она загружается в оперативную память. Типовые операторы Visual C++ , используемые при создании такого окна, приведены в приложении 2.

В качестве примера можно рассмотреть полезную программу – «часы моделирования». Изображение, появляющееся в правом верхнем углу такого окна, показано на рис. 4.12. Как только очередной интервал моделирования превзойдет 1% модельного времени, эта программа переформировывает содержание функционального окна. В результате можно наблюдать эффект анимации.

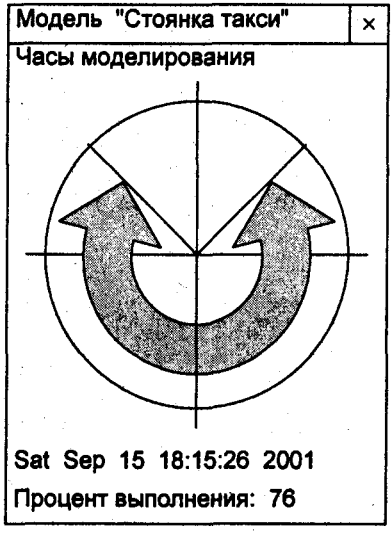


Рис. 4.12. Функциональное окно Часы моделирования

Программа функционального окна получает в качестве параметра область памяти struct fwcb, в которой размещены все необходимые оперативные данные на момент последнего события, которое произошло в модели. Адрес этой области памяти содержится в параметре s, передаваемом в программу funcwindow. Ее структура показана ниже:

```
struct fwcb // БЛОК УПРАВЛЕНИЯ ФУНКЦИОНАЛЬНЫМ ОКНОМ
 {
 void *pointer;    // Указатель произвольной области
 LONG  mywndproc;      // Адрес оконной процедуры
 HWND  hwnd;           // Дескриптор окна
 HDC   hdc;            // DC устройства
 HDC   memdc;          // DC виртуального окна
 HFONT hOldf;          // Старый шрифт
 HFONT hNewf;          // Новый шрифт
 HBITMAP hbit;         // Растр - виртуальное окно
 HBRUSH hbrush;        // Рабочая кисть
 HBRUSH hOldbrush;     // Дескриптор прежней кисти
 HBRUSH hHollowbrush; // Прозрачная кисть
 HBRUSH hWhitebrush;   // Белая кисть
 HBRUSH hBlackbrush;   // Черная кисть
 HBRUSH hRedbrush;     // Красная кисть
 HBRUSH hGreenbrush;   // Зеленая кисть
 HBRUSH hBluebrush;    // Синяя кисть
 HBRUSH hYellowbrush; // Желтая кисть
 HBRUSH hMagentabrush;// Фиолетовая кисть
 HBRUSH hCyanbrush;    // Бирюзовая кисть
 HPEN hOldpen;         // Дескриптор прежнего пера
 HPEN hWhitepen;       // Белое перо
 HPEN hBlackpen;       // Черное перо
 HPEN hRedpen;         // Красное перо
 HPEN hGreenpen;       // Зеленое перо
 HPEN hBluepen;        // Синее перо
 HPEN hYellowpen;      // Желтое перо
 HPEN hMagentapen;     // Фиолетовое перо
 HPEN hCyanpen;        // Бирюзовое перо
 double timer;         // Модельный таймер
 float  tmax;          // Время моделирования
 long   nd;            // Число событий в модели
 struct kcb *addr[pool];// Массив указателей узлов
 struct tcb *t;    // Адрес продвигаемого транзакта
 struct ecb *e;    // Адрес вновь образованного ecb
 struct ecb *w;    // Адрес спланированного события
 int next;             // Номер «текущего» узла
 int error;            // Фатальная ошибка в модели
 int maxn;             // Число узлов модели
 int presise;      // Знаков после десятичной точки
 int maxX;             // Размеры экрана X
 int maxY;             // Размеры экрана Y
 int percent;          // Процент выполнения модели
 int stop0;            // Признак выполнения модели
 int stop1;            // Признак приостановки
 int stop2;            // Признак трассировки
 int stop3;            // Изменить масштаб времени
 char modname[16];     // Название модели
 char filename[256];   // Имя найденного файла
 };
```

Рассмотренная область описана в системном h-файле Simulate.h. В соответствии с ее структурой ниже приведен текст программы, реализующей окно типа «часы моделирования». Следует отметить, что в данной программе значительное место занимают операторы, необходимые для автоматизации размещения текстов и изображений на заданном прямоугольнике. Если это окно сделать без такого сервиса, то программа будет значительно короче (и понятнее):

```
#include <Windows.h>
#include <Simulate.h>
void  funcwindow(struct fwcb*);

BOOL WINAPI DllEntryPoint(HINSTANCE hDLL,
                          DWORD dwReason,
                          LPWORD Reserved)
  {
  switch(dwReason)
    {
    case DLL_PROCESS_ATTACH:
      {
      break;
      }
    case DLL_PROCESS_DETACH:
      {
      break;
      }
    }
  return TRUE;
  }
void  funcwindow(struct fwcb *s)
  {
  TEXTMETRIC tm;          // Метрики текста
  SIZE       size;        // Структура для метрик
  time_t     ct;          // Переменная для таймера
  char       str[3000];   // Буфер для строки вывода
  char       comwork[96];// Строка для измерений
                          // длин текстов
  struct     tm *newtime;// Адрес структуры таймера
  struct     tcb *k;      // Адрес узла модели
  double     ap;          // Текущий угол
  double     hp;          // Шаг угла
  float      p;           // Проценты
  int        xL;          // X левого верхнего угла
  int        yL;          // Y левого верхнего угла
  int        wE;          // Ширина эллипса
  int        hE;          // Высота эллипса
  int        xLp;         // Левый край индикатора
  int        xRp;         // Правый край индикатора
  int        yHp;         // Верхний край индикатора
```

```
int         yLp;          // Нижний край индикатора
int         wX;           // Текущие координаты X
int         wY;           // Текущие координаты Y
int         xS1;          // Рабочая координата
int         xS2;          // Рабочая координата
int         yS;           // Рабочая координата
int         iP;           // Рабочая для процентов
int         i;            // Рабочая переменная
s->hOldf=(HFONT)SelectObject(s->memdc,s->hNewf);
                          // Шрифт ANSI
if(!SetWindowLong(s->hwnd,GWL_WNDPROC,
                  s->mywndproc))
  MessageBox(s->hwnd,"","Не подключена программа
                        MyWindowProcedure",MB_OK);
hp=pi/100;
p=s->timer/s->tmax*100;
iP=p;
if(iP>100)
  iP=100;
if(iP>s->percent || s->percent ==100 || s->percent
   && ! s->stop0 && ! s->stop1 && ! s->stop2
   && ! s->stop3))
  {
  MessageBeep(1);
  ap=hp*p+0.0001;
  if(ap>pi)
    ap=pi;
  xRp=s->maxX-2;
  yHp=2;
  GetTextMetrics(s->memdc,&tm); //Метрики текста
  sprintf(comwork,"Процент выполнения: 100   ");
  GetTextExtentPoint32(s->memdc,comwork,
                       strlen(comwork),&size);
  i=xRp-size.cx;
  xLp=i;
  sprintf(comwork,"Модель \"%s\"",s->modname);
  GetTextExtentPoint32(s->memdc,comwork,
                       strlen(comwork),&size);
  i=xRp-size.cx;
  if(i < xLp )
    xLp=i;
  ct=time(NULL);        // Получить новое время
  newtime=localtime(&ct);
  strcpy(str,asctime(newtime)); // Новое время
  str[strlen(str)-1]='\0';      // Удалить \r\n
  GetTextExtentPoint32(s->memdc,str,
                       strlen(str),&size);
  i=xRp-size.cx-10;
  if(i < xLp )
    xLp=i;
```

```
xLp -= 9;
yLp = yHp + 3 + 12*(tm.tmHeight +
                    tm.tmExternalLeading) + 3;
wX=xLp+3;                    // Текущий X текста
wY=yHp+3;                    // Текущий Y текста
TextOut(s->memdc,wX,wY,comwork,
        strlen(comwork)); // Название модели
wY = wY + 9*(tm.tmHeight +
             tm.tmExternalLeading);
TextOut(s->memdc,wX,wY,str,strlen(str));// Время
wY = wY + (tm.tmHeight + tm.tmExternalLeading);
sprintf(str,"Процент выполнения: %3d          ",
        iP);
TextOut(s->memdc,wX,wY,str,strlen(str));
wY = wY + (tm.tmHeight + tm.tmExternalLeading);
if (iP == 100)
  {
  sprintf(str,"Выполнена...       ");
  TextOut(s->memdc,wX,wY,str,strlen(str));
  }
else if (s->nd == 0)
  {
  sprintf(str,"Моделирование        ");
  TextOut(s->memdc,wX,wY,str,strlen(str));
  }
xL=(xLp+xRp)/2;          // X и Y центра эллипса
hE=6*(tm.tmHeight + tm.tmExternalLeading);
                         // Высота эллипса
wE=hE;                   // Ширина эллипса
xL=xL-hE/2.0;            // X левого верхнего угла
yL=yHp+3+2*(tm.tmHeight+tm.tmExternalLeading);
                         //Y левого верхнего угла
SelectObject(s->memdc,s->hRedpen);
                         // Выбрать красное перо
SelectObject(s->memdc,s->>hWhitebrush);
                         // Выбрать белую кисть
Ellipse(s->memdc,xL,yL,xL+wE,yL+hE); // Эллипс
xS1=xL+wE*(1-sin(ap))/2;
xS2=xL+wE*(1+sin(ap))/2;
yS =yL+hE*(1+cos(ap))/2;
SelectObject(s->memdc,s->hCyanbrush);
                         // Бирюзовая кисть
Pie(s->memdc,xL,yL,xL+wE,yL+hE,xS1,yS,xS2,yS);
                         // Сектор
SelectObject(s->memdc,s->hBluepen);
                         // Выбрать синее перо
MoveToEx(s->memdc,xL+wE/2-1,yL-s->maxY/60,NULL);
LineTo(s->memdc,xL+wE/2-1,yL+hE+s->maxY/60);
MoveToEx(s->memdc,xL-s->maxX/80,yL+hE/2-1,NULL);
LineTo(s->memdc,xL+wE+s->maxX/80,yL+hE/2-1);
```

```
    SelectObject(s->memdc,s->hHollowbrush);
                          // Прозрачная кисть
    SelectObject(s->memdc,s->hGreenpen);
                          // Выбрать зеленое перо
    Rectangle(s->memdc,xLp,yLp,xRp,yHp);
                          // Прямоугольник
    s->percent=iP;
    InvalidateRect(s->hwnd,NULL,1);
    }
  return;
  }
```

Модернизация моделей. Все элементы проекта автоматически сохраняются средствами Developer Studio. Для того чтобы прервать разработку, необходимо выполнить следующее:

- на всякий случай принудительно сохранить последний модифицированный текст модели (если модификации были), выполнив действия File → Close через главное меню Developer Studio или нажав значок «дискета» на панели инструментов;
- затем либо просто выйти из Developer Studio (если нет других проектов), либо в главном меню выполнить действия: File → CloseWorkspace (если нужно перейти к другому проекту).

Модернизация моделей возможна всегда. Чтобы выполнить модернизацию, через меню Developer Studio открывается существующий проект: File → OpenWorkspace.

Далее автоматически подключается проводник для поиска файла управления проектом с суффиксом mdf (например, ModelPro.mdf). Этот файл автоматически показывается в папке с проектом. После двойного щелчка по имени этого файла проект восстановится в том виде, в котором он был в последний раз (повторно его собирать не нужно). Можно сразу перейти к изменению любых текстов и состава модели.

Выполнение моделей. По умолчанию создается exe-файл в папке Debug, вложенной в папку с проектом. Это и есть компьютерная модель. В exe-файле есть все необходимое для отладки модели. Окончательный вариант отлаженной модели обычно берется из папки Release: он работает быстрее и меньше по объему.

В процессе отладки модель запускается из папки Debug одним из двух способов:

- нажатием комбинации клавиш Ctrl+F5;
- через главное меню Build → Execute ModelPro.exe.

Файл с моделью можно переписать в любую другую папку и на любой другой компьютер. Он всегда будет работать.

Замечание. Если модель содержит функциональное окно для конечного пользователя, то соответствующий dll-файл должен быть помещен в одну из папок:

- в ту же папку, где находится exe-файл (этот способ универсальный, самый простой и не зависим от типа операционной системы);
- в папку System, если модель должна выполняться в системе Windows 98;
- в папку System32, если модель должна выполняться в системе Windows NT (для этого необходимо получить режим администратора). В системе Windows 2000 необходимо выполнить аналогичные действия.

Выводы

1. Рассмотрены основные приемы программирования и отладки моделей. Программисту предоставляется возможность получать, задавать и использовать значения параметров транзактов, узлов и глобальных переменных моделирующей системы, а также непосредственно обращаться к датчикам псевдослучайных чисел. Для отладки моделей используются итоговая таблица с результатами моделирования и динамически изменяющиеся отладочные таблицы.

2. Имитационная модель создается в среде Developer Studio с применением средств Visual C++. Приведены типовые технологические последовательности действий, необходимые при создании, модернизации и выполнении моделей. Программист может создать типовой проект, проект с диалоговым окном для управления параметрами при запуске модели или во время ее выполнения, а также проект модели с функциональным окном для конечного пользователя.

Вопросы для самопроверки

1. Как осуществляется отсчет модельного времени?
2. В какой переменной находится значение модельного времени?
3. Какие параметры транзактов всегда доступны разработчику модели?

4. Какая переменная содержит номер узла?
5. Какие параметры узла доступны разработчику модели для анализа?
6. Для чего используется глобальная переменная *error*?
7. Как непосредственно обратиться к датчикам псевдослучайных чисел?
8. Какие виды выходной информации используются для отладки моделей?
9. Что входит в итоговую таблицу результатов моделирования?
10. Какие существуют режимы трассировки, позволяющие ускорить процесс отладки?
11. Как осуществляется случайный выбор из класса узлов?
12. Как задать схему выбора дальнейшего пути продвижения транзакта в зависимости от значения некоторого его параметра?
13. Какие модели относятся к замкнутым корпоративным информационным системам (КИС) ?
14. Как производится зарядка одного многоканального сервера?
15. Каким образом осуществляется зарядка нескольких одноканальных серверов?
16. Как выполняется зарядка нескольких многоканальных серверов?
17. Как определить время ответа на запрос в КИС?
18. В какой среде обычно создается имитационная модель?
19. Как создать типовой проект модели?
20. Из каких файлов состоит типовой проект модели?
21. Как выполнить проект модели с диалоговым окном для управления параметрами? Из каких файлов состоит такой проект? Какие файлы необходимо откорректировать при создании данного проекта?
22. Как создать проект модели с функциональным окном для конечного пользователя?
23. Что входит в блок управления функциональным окном?

Глава 5

СОЗДАНИЕ МНОГОСЛОЙНЫХ МОДЕЛЕЙ С ПОМОЩЬЮ ГРАФИЧЕСКОГО КОНСТРУКТОРА

5.1 CASE-ТЕХНОЛОГИЯ МНОГОСЛОЙНОГО ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

В последние годы все большее распространение получают CASE-средства, позволяющие автоматизировать процессы проектирования, разработки и поддержки программных приложений:

- компьютерных экономико-математических моделей;
- экономических информационных систем;
- вычислительных программ прикладной математики экономического назначения.

CASE-средства активно используют методологию структурного анализа, предусматривающую наглядное и эффективное проектирование системы путем выделения ее составляющих и их последовательного рассмотрения. Описание системы начинается с общего обзора и выделения основных ее компонентов или процессов. Для визуального представления создается первый уровень или слой, на котором отображаются выделенные процессы и их взаимосвязи. Далее для ряда процессов может быть проведена детализация, в свою очередь выделяющая новые процессы в их структуре. Так, последовательным усложнением описания объекта и его процессов разработчик достигает необходимой детализации. Глубина детализации определяется как необходимой точностью, так и набором исходных данных. В процессе структурного анализа выявляется иерархическая структура модели.

Рассмотренный ниже декомпозиционный подход реализуется в программных CASE-пакетах в различных вариациях, поскольку су-

ществует достаточно широкий круг задач, для которых схожие методы могут быть применены. Однако все CASE-пакеты предоставляют пользователю инструментарий работы с проектом, опирающийся на мощные современные графические средства отображения информации в виде графов, диаграмм, схем и таблиц.

Одним из достаточно интересных и полезных применений CASE-средств является не только их интеграция в процессы проектирования, разработки и поддержки структуры программного проекта, но и автоматизация процесса создания или генерации программного кода. Использование CASE-средств, дополненных такой возможностью, имеет ряд несомненных преимуществ перед простым кодированием, поскольку позволяет:

- отвлечься от кодирования данных и обратить большее внимание на структуру разрабатываемой системы;
- избежать некоторых ошибок за счет автоматического контроля;
- ускорить процесс проектирования и разработки проекта.

Теперь рассмотрим CASE-технологии применительно к системе имитационного моделирования. Для создания имитационной модели в отсутствие CASE-средств разработчику приходится писать программный код, использующий языковые средства системы моделирования Pilgrim. Модель имеет стандартную структуру. Внутри текста модели содержатся обращения к функциям Pilgrim, но может быть и произвольный C++ код.

Учитывая, что текст модели обрабатывается препроцессором и стандартным компилятором C++ (Microsoft, Borland и др.), можно выделить ряд проблем, возникающих перед пользователем при описании модели в операторах Pilgrim, а именно:

- необходимо знать элементы языка C++;
- нужно иметь отчетливое представление о структуре программы, опирающейся на библиотеку Pilgrim;
- требуется знать функции описания узлов и их параметров;
- имеется вероятность появления ошибки в порядке перечисления позиционных параметров, причем ошибка может быть не замечена компилятором C++, в результате чего модель будет выполняться, но иметь на выходе неправильные результаты. Обнаружение такой ошибки тем сложнее, чем большее количество узлов имеет модель;

• сложность описания больших моделей. Поскольку модель любого размера выглядит как простое линейное перечисление узлов и условий переходов между ними, то, чем больше текст модели, тем он сложнее воспринимается пользователем.

Конструктор моделей Pilgrim (далее – конструктор) позволяет автоматизировать процесс создания графа модели и автоматически генерировать код Pilgrim-программы. Тем самым снимаются отмеченные выше проблемы, возникающие при ручном кодировании модели в виде Pilgrim-файла:

• автоматическая генерация программного кода позволяет пользователю не задумываться о структуре и синтаксисе программы, уделяя все внимание структуре и параметрам самой модели и ее узлов;

• генерация функций описания узлов конструктором исключает ошибки, связанные с неправильной последовательностью указания позиционных параметров или пропуском некоторых из них;

• анализируя модель, конструктор не позволяет пользователю выполнять заведомо неверные действия, а также предупреждает о возможных ошибках;

• поддержка конструктором множества плоскостей обеспечивает создание иерархических моделей, что может быть очень удобно при выполнении моделей с большим количеством узлов. Действительно, не только восприятие, но и отображение больших моделей в виде плоского одноуровневого графа на бумаге или экране монитора достаточно затруднено.

5.2 ОСОБЕННОСТИ РЕАЛИЗАЦИИ КОНСТРУКТОРА МОДЕЛЕЙ GEM

Конструктор создан для работы под управлением Windows 95/98/NT и использует удобные диалоговые средства, предоставленные графическим интерфейсом этих операционных систем.

Модель с точки зрения конструктора Pilgrim можно представить как набор следующих компонентов:

• граф модели;

• параметры инициализации модели;

• переменные модели;

• включенные в модель фрагменты программного кода на языке C++.

Кратко рассмотрим каждый из перечисленных компонентов.

Вершины *графа модели* представляют собой узлы – пункты обработки транзактов. Для всех типов узлов имеются условные обозначения, упрощенные по сравнению с графическими изображениями (*см.* рис. 2.3). Перечень таких обозначений приведен на рис. 5.1.

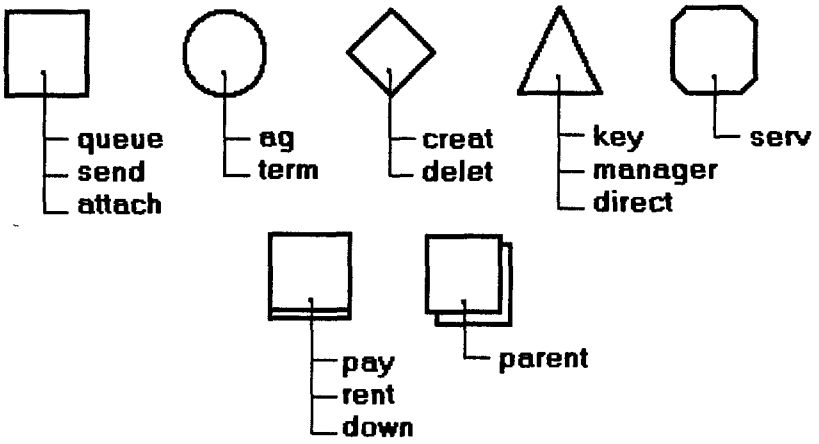


Рис. 5.1. Условные обозначения узлов Pilgrim при работе с графическим конструктором

Внутри любого узла (кроме виртуального узла parent) происходит обработка транзакта, определяемая спецификой его типа. Дуги графа представляют собой пути миграции транзактов по графу модели и имеют направленность. Возможны ситуации, когда один узел имеет несколько выходов, тогда путь транзакта определяется условиями, заданными в узле-источнике.

Каждый узел модели имеет собственные параметры, как общие для всех типов узлов (имя, номер, условия ссылок), так и специфические, определяемые типом узла. Такие специфические параметры также можно назвать параметрами функции узла.

Параметры инициализации модели – это набор переменных, включенных в функции инициализации модели modbeg и завершения модели modend.

Конструктор также позволяет определять *переменные модели* – глобальные переменные, инициализируемые в начале текста программы. Доступ к ним возможен из любого узла в любой момент выполнения программы.

Иногда пользователю необходимо включение в программу *собственных программных вставок*. Поэтому конструктор позволяет пользователю вводить произвольный C++ код. Программный текст может размещаться в начале программы, выполняя подготовку каких-либо данных, и внутри любого из узлов.

Важнейшим моментом создания модели является построение графа, описывающего зависимости между процессами моделируемой системы. Для того чтобы изначально не зайти в тупик, пытаясь построить некорректный, трудно воспринимаемый или неудобный для дальнейшей работы граф, необходимо четко представлять методы, средства и ограничения конструктора по построению графа.

Основное *достоинство* конструктора Pilgrim заключается в том, что он позволяет:

- проводить многоуровневую (т.е. многослойную) иерархическую декомпозицию глобального процесса, разлагая его на составляющие компоненты (проводить структурный системный анализ);
- представлять каждый уровень структурной детализации в виде графического слоя;
- автоматически генерировать программный текст модели.

С помощью конструктора удобно строить многоуровневые модели, организуя иерархию плоскостей построения модели.

Конструктор позволяет создавать множество плоскостей, на которых расположены слои модели с любым количеством узлов и ссылок. Иерархические модели устроены следующим образом: верхний уровень модели содержит ряд узлов, среди которых есть такие, которые детализируются на нижних уровнях. При этом, создавая детализирующий уровень, пользователь опять может поместить на нем узлы, которые могут быть детализированы еще ниже; так граф модели примет иерархическую структуру. Число уровней вложенности не ограничено конструктором, т.е. пользователь может сколько угодно подробно детализировать и обобщать процессы модели.

При создании новой модели пользователь начинает работу с корневого уровня, и поэтому некоторые процессы модели могут быть представлены здесь в общем виде, с целью дальнейшей детализации. Для выполнения структурной декомпозиции к набору узлов, определяемому системой Pilgrim, конструктор добавляет новый осо-

бый тип – parent, использующийся исключительно как средство создания наглядных многоуровневых моделей. Узел типа parent содержит ссылку на плоскость, его детализирующую, т.е. используется для описания процессов, которые можно рассмотреть на отдельной плоскости. Узел parent не выполняет никаких действий по обработке транзакта и при генерации программного кода просто заменяется своей декомпозицией, т.е. порождаемым им слоем. Например, если пользователь создал узел типа parent с номером 4, то в программном файле узел с таким номером сгенерирован не будет. С точки зрения генератора программного файла узел parent представляет собой простой маршрутизатор, который может иметь любое количество входов, ссылку на детализирующий уровень и единственный выход.

Рассмотрим пример на рис. 5.2, где в плоскости 1 имеется последовательность узлов Клапан_1→Действие_1→Очередь_1.

Узел Действие_1 является узлом типа parent и содержит детализирующую плоскость 11. Плоскость 11 содержит цепочку Очередь_2→Сервер_2.

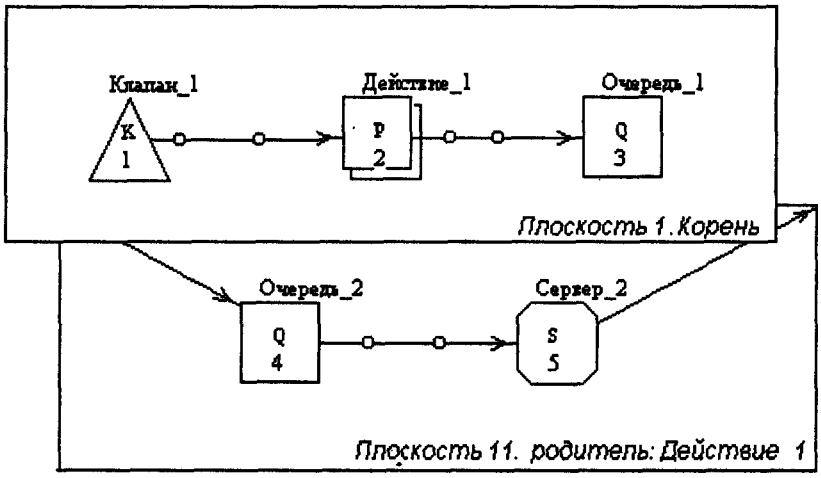


Рис. 5.2. Детализация узла типа parent

В плоскости 11 стрелка, направленная из левого верхнего угла экрана в узел Очередь_2, обозначает, что этот узел является входом на плоскость, а стрелка из узла Сервер_2 в правый верхний угол – что это выход с плоскости. При запуске генерации программного

файла получим следующую цепочку: Клапан_1→Очередь_2→ Сервер_2→Очередь_1, т. е. узел Действие_1 является лишь средством реализации многоуровневости и будет обработан генератором как узел Pilgrim.

Иногда при построении модели может возникнуть необходимость выделения некоторых типовых действий по обработке данных. Это могут быть запросы на выполнение бухгалтерской проводки, требования выделения моделируемого ресурса или какие-либо другие действия. При возникновении такой задачи удобно обозначить подпрограммы, обращение к которым было бы возможно из любого места модели. Для этого используются узлы типа pay, rent, down. Такие узлы, так же как и parent, содержат переход на более низкий уровень модели, однако имеют несколько иной механизм действия и область применения. Если с помощью узла типа parent можно создавать иерархические модели, имеющие на любой сколько угодно глубоко вложенной плоскости новые узлы parent, то с помощью узлов типа pay, rent, down возможно лишь реализовать подпрограммы на двух слоях модели и невозможно построить общую иерархию уровней.

Рассмотрим принцип работы таких узлов на примере узла pay (рис. 5.3). На плоскости 1 находится узел типа pay, содержащий обращение к подпрограмме, расположенной в плоскости 12. Входом плоскости 12 является узел с названием «Расчетный счет фирмы», а выходом – узел с именем «Бухгалтерия». При генерации программного файла в узле «Бухгалтерия» в качестве параметра, определяющего номер следующего узла, на который переходит транзакт, будет указано не конкретное значение, а специальный параметр транзакта updown. При этом предполагается, что каждый транзакт, попадающий в выходной узел плоскости, содержит в параметре updown номер узла, на который следует выполнить переход. Параметр транзакта updown инициализируется в узле типа pay, т.е. в нашем случае в узле с названием «Плата поставщику».

Аналогичным образом реализуется переход на подпрограмму с использованием узлов типа rent и down. Они также инициализируют переменную транзакта updown. Из вышеизложенного можно сделать вывод о невозможности использовать узлы типа pay, rent и down для реализации иерархических моделей по следующей причине: на уровне подпрограммы узла одного из перечисленных типов нельзя размещать никакой из узлов типа pay, rent, down, так как каждый из этих узлов заново выполнит инициализацию параметра транзакта updown, т.е. заменит значение updown, установленное уровнем вы-

ше, организуя циклическую ссылку (это приведет к семантической ошибке).

Чтобы защитить пользователя от совершения таких ошибок, конструктор не позволяет создавать в текущей плоскости узлы типа pay, rent, down, если управление передано с более высокого уровня через узел перечисленного типа. Однако узлы обращения к подпрограмме имеют одно важное преимущество перед узлом parent. Оно состоит в том, что транзакт сам «помнит», куда ему необходимо вернуться; поэтому из нескольких узлов (или слоев) можно обращаться к общей плоскости, содержащей детальное выполнение типового действия.

Обратимся еще раз к примеру, приведенному на рис. 5.3. Предположим, что имеется некоторая организация, имеющая собственный расчетный счет и выполняющая ряд операций по перечислению средств; часть операций необходимо смоделировать. В плоскости 12 создана подпрограмма «Плата поставщику», выполняющая бухгалтерскую проводку по перечислению средств со счета фирмы. Если возникает необходимость смоделировать аналогичную ситуацию с перечислением средств (например, возврат кредита), то можно создать новый узел типа pay и указать ему в качестве подпрограммы плоскость 12.

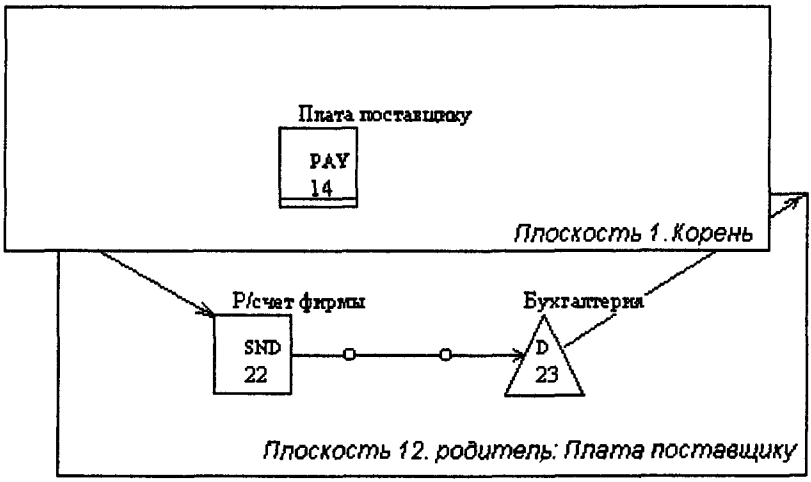


Рис. 5.3. Пример использования узла типа pay в качестве средства реализации подпрограмм

В процессе работы с конструктором, при создании нового узла обращения к подпрограмме, появляется диалоговое окно с просьбой указать номер плоскости, на которой должна размещаться подпрограмма. По умолчанию это номер новой пустой плоскости, однако пользователь самостоятельно может задать номер уже существующей плоскости.

Иногда в модели можно выделить ряд процессов, независимых на уровне транзактов, но пересекающихся на уровне моделируемых ресурсов. Например, это может быть моделирование процесса производства с участием нескольких независимых торговых компаний, приобретающих товар у производителя и имеющих свой финансовый резерв и рынок сбыта. Тогда можно выделить как независимые процессы торговых компаний и процесс функционирования производителя. Для разработчика модели было бы неудобно воспринимать в одной корневой плоскости несколько параллельных процессов, поэтому конструктор предоставляет возможность содержать несколько корневых плоскостей с целью размещения в них непересекающихся на уровне транзактов процессов.

5.3 РАБОТА С ГРАФИЧЕСКИМ КОНСТРУКТОРОМ В СИСТЕМЕ PILGRIM

Конструктор моделей состоит из программного файла gem.exe, файлов настроек с расширением ini, файла помощи и примеров имитационных моделей. При запуске gem.exe на выполнение перед пользователем появляется основное окно, содержащее меню, панель «горячих кнопок», панель инструментов, информационную строку (рис. 5.4). Область построения графа модели пуста, для редактирования необходимо создать новую модель либо загрузить ранее сохраненную. Рассмотрим сначала выполняемые конструктором операции с файлами.

Всю информацию о модели конструктор сохраняет в файле с расширением «pgf» (Pilgrim graph file). При создании законченной версии имитационной модели пользователь может генерировать программный файл Pilgrim с расширением «cpp» (c plus plus). Этот файл далее компилируется в среде Visual C++ с подключением необходимых библиотек и ресурсов Pilgrim. Создаваемый конструктором программный файл с расширением «cpp» при своей генерации

теряет связь с моделью, т.е. его редактирование никоим образом не отразится на модели, редактируемой с помощью конструктора. Таким образом, создание модели необходимо целиком выполнять с помощью конструктора (исключение составляют пользовательские библиотеки, используемые для расширения возможностей модели). При работе с моделью рекомендуется периодично сохранять результаты работы, оставляя копии рабочих версий модели.

Создание новой модели, сохранение и загрузка версий выполняются выбором соответствующих пунктов раздела «Файл» основного меню программы или через «горячие кнопки».

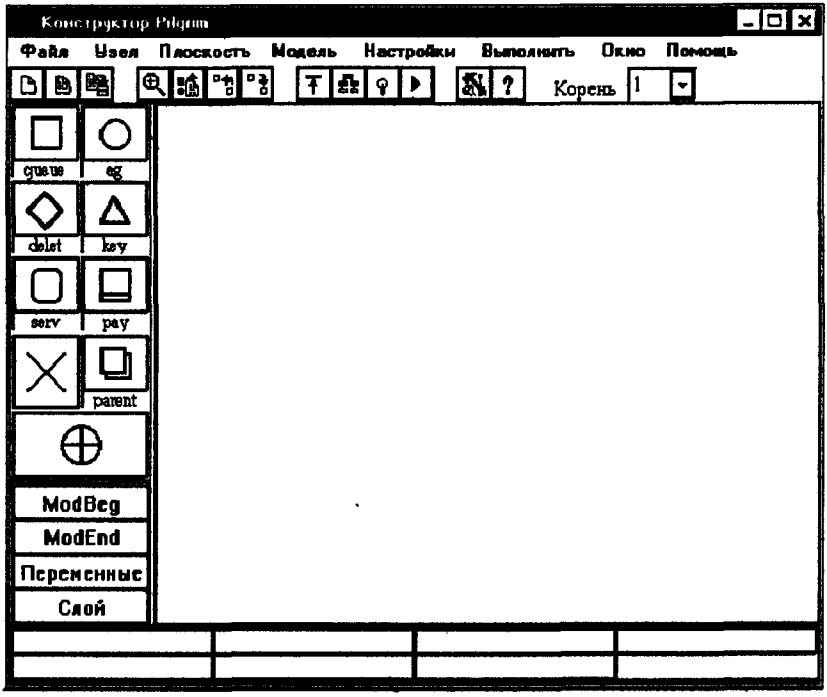


Рис. 5.4. Основное окно конструктора моделей с пустой областью построения графа

Редактирование графа модели. В основе любой имитационной модели лежит граф взаимодействия ее процессов. Выделим типовые действия редактирования графа модели. Это добавление новых узлов, их перемещение в области построения, удаление существую-

щих, определение маршрутов переходов транзактов, или ссылок. В конструкторе все перечисленные действия выполняются «перетягиванием» объекта, обозначающего тип требуемого узла или действия, из панели инструментов в область построения графа. Панель инструментов в левой части окна редактора содержит значки, обозначающие узлы системы.

В панели инструментов находится набор значков всех типов узлов системы (*см.* рис. 5.1). Следует заметить, что для нескольких различных типов узлов может существовать одно и то же обозначение. Например, квадратом обозначаются узлы типа queue, send и attach. Поэтому под каждым значком с изображением узла находится кнопка с текущим типом создаваемых узлов данного обозначения. Для изменения типа достаточно щелкать мышью по кнопке с именем типа до появления необходимого.

Местоположение узла в плоскости построения графа не несет никакой смысловой нагрузки с точки зрения логики модели. Поэтому конструктор позволяет свободно перемещать узлы так, чтобы граф модели был удобен для восприятия. При перемещении узлов также используется технология перетягивания объектов.

Аналогичным образом выполняется удаление узла: в панели инструментов необходимо захватить значок удаления узла и перетащить его на узел, который требуется удалить. Узел будет удален вместе со всеми входящими и исходящими ссылками. Дополнительные условия возникают при удалении узлов, содержащих переход на нижнюю плоскость. Например, если требуется удалить узел типа parent, то конструктором проверяется условие пустой детализирующей плоскости. Алгоритм конструктора не позволит удалить узел типа parent, пока на детализирующей его плоскости находится хотя бы один узел, так как это приведет к его потере. Чтобы упростить удаление множества узлов на плоскости, к которой ссылается, к примеру, parent, в меню «плоскость» существует специальный пункт «удалить все узлы текущей плоскости».

Несколько по-иному, чем для parent, реализовано разрешение на удаление узлов обращения к подпрограмме, т. е. pay, rent, down. Если в модели имеется плоскость-подпрограмма для единственного узла pay, rent или down, то его нельзя удалить, пока плоскость не пуста (так же как в случае с parent). Если же к подпрограмме обращаются несколько узлов типа pay, rent или down, то конструктор позволяет удалить любой из них, так как в этом случае плоскость-подпрограмма не теряется, поскольку продолжают существовать

ссылки, на нее указывающие. Таким образом, смысл условий удаления узлов, порождающих плоскости, крайне прост: необходимо, чтобы в модели не был потерян ни один узел.

Создание ссылок, или путей переходов транзактов, происходит следующим образом: в панели инструментов захватывается значок направленной в экран стрелки (перекрестие, заключенное в круг) и перемещается на узел-источник транзакта. При отпускании кнопки мыши за курсором потянется стрелка, обозначающая ссылку с невыбранным узлом-приемником транзакта. Для выбора узла-приемника необходимо щелкнуть по нему мышью, в результате чего создастся новая ссылка. Созданные ссылки отображаются на экране в виде направленных стрелок, имеющих три звена. Звенья не несут никакой смысловой нагрузки и служат для удобства отображения графа на плоскости. Две промежуточные точки между звеньями выделены небольшими кругами, захватывая и перемещая которые можно придать стрелке нужную форму.

Следует помнить, что некоторые типы узлов не могут иметь входящие или, напротив, исходящие ссылки. Конструктором постоянно выполняется проверка корректности действий пользователя, запрещающая заведомо недопустимые преобразования графа модели, в частности создание неправильных ссылок.

Определение параметров узла. Каждый узел модели характеризуется множеством параметров: типом, порядковым номером, именем, принадлежностью к плоскости, ссылками и условиями переходов, встроенным программным текстом, а также непосредственно параметрами, определяемыми спецификой типа узла, такими, как закон распределения для узла типа serv, приоритет для queue и т.п. Для просмотра или редактирования параметров узла необходимо дважды щелкнуть по нему левой кнопкой мыши либо один раз щелкнуть по узлу правой кнопкой, в результате чего отобразится всплывающее меню, и выбрать в нем пункт «параметры узла». Появится диалоговое окно, определяющее параметры. На рис. 5.5 показано окно параметров узла типа serv, номер 101, имеющего имя «Производство». Необходимо пояснить некоторые компоненты окна и способы работы с ними.

Номер узла присваивается конструктором при его создании. Обычно первые 100 номеров зарезервированы для узлов send – бухгалтерских счетов. Нумерация создаваемых узлов осуществляется последовательно, с номера 101. Смена номера узла возможна с помощью соседнего с номером диалогового поля, но не рекомендуется

для узлов, созданных ранее. Дело в том, что в тексте модели могут существовать привязки к номеру узла, созданные вручную. Конструктор не может их отследить, поэтому при перенумерации логика модели рушится.

Поле «Имя» содержит имя узла, отображаемое на схеме и при выполнении модели. Поле доступно для редактирования. Не рекомендуется использовать имена, не умещающиеся в поле редактирования.

Класс узла может быть выбран из списка. В списке приводятся только те типы узлов, которые имеют одинаковое обозначение. Например, узел типа send можно сменить на attach (но при этом изменяется набор и смысл параметров). Поэтому функция смены типа полезна и имеет смысл только при создании нового узла.

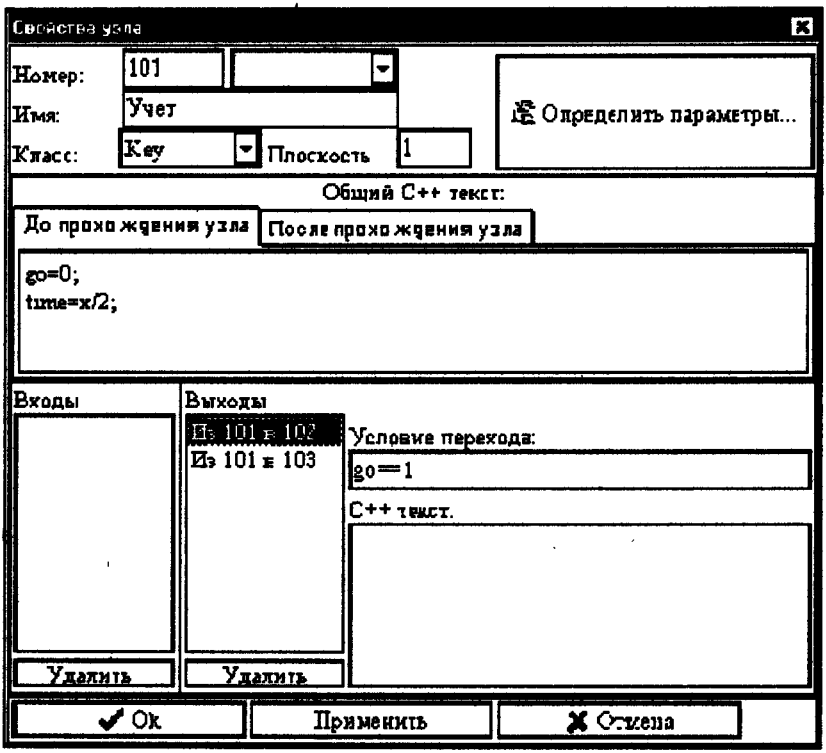


Рис. 5.5. Окно определения параметров узла

Поле «Плоскость» показывает, к какой плоскости принадлежит узел, и доступно только для просмотра.

Модель получает дополнительную гибкость за счет использования вставок C++ кода. Панель «Общий C++ текст» позволяет пользователю включить в процедуру обработки узла произвольный текст на языке C++. Текст делится на две части: одна выполняется до вызова функции узла, другая – после нее. Смысл такого разбиения заключается в том, что программный текст, выполняющийся до вызова функции узла, может подготавливать какие-либо переменные, которые функцией будут использованы. Так, например, может быть подсчитано время обслуживания транзакта перед выполнением функции узла типа serv. Программный текст, следующий после вызова функции узла, на ее выполнение уже никак не влияет и может использоваться для обработки параметров выполненной функции или подготовки параметров для других функций модели.

Узел может содержать несколько исходящих ссылок, по которым способен переместиться транзакт. Выбор маршрута должен осуществляться по условиям. Поэтому в правой нижней части окна определения параметров транзакта имеются поля, определяющие эти условия, а также выполнение соответствующих им индивидуальных фрагментов программного кода. Выбор исходящей ссылки осуществляется щелчком мыши в диалоговом поле «Выходы». Как видно из рис. 5.5, любой вход и выход можно удалить, нажав кнопку внизу списка.

И наконец, необходимым компонентом представленного окна является кнопка «Определить параметры», нажатие на которую вызывает окно определения параметров самой функции обработки узла. Вид появляющегося диалогового окна зависит от типа узла. Пример окна для узла типа serv приведен на рис. 5.6.

Итак, рассмотрена схема определения параметров узла типа serv. Аналогичным образом определяются параметры для узлов любого типа, но окно определения параметров функции узла (рис. 5.6) имеет различный вид. Например, для узла типа queue окно содержит настройку единственного параметра – признака приоритета прохождения транзактов, а для узлов типа term параметров функции узла и исходящих ссылок не существует вовсе.

Определение параметров инициализации/завершения модели. Модель имеет параметры инициализации и завершения, задаваемые функциями modbeg и modend. Определение этих параметров производится через диалоговые окна, вызываемые нажатием кнопок

«Modbeg» и «Modend» в основном окне редактора либо выбором подпунктов основного меню в разделе «Модель».

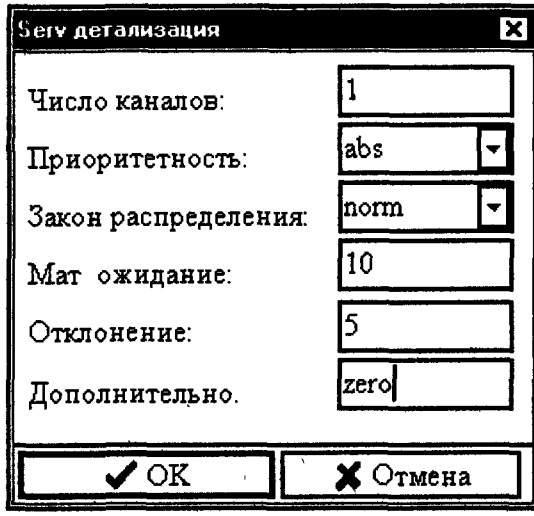


Рис. 5.6. Окно определения параметров функции для узла типа serv

Окно определения параметров функции modbeg приведено на рис. 5.7. В его правой верхней части записывается какой-либо начальный текст на C++, если он необходим. Программный текст делится на две части: начальный C++ текст используется для подключения внешних библиотек и настройки глобальных параметров; текст инициализации ресурсов подготавливает параметры конкретных узлов типов attach и send. Другие поля окна позволяют редактировать переменные, стандартные для функции modbeg.

Редактирование переменных функции modend осуществляется через диалоговое окно, приведенное на рис. 5.8.

Работа в плоскостях модели. При работе с большой моделью удобно пользоваться набором плоскостей построения. Для этого конструктор предлагает набор плоскостей с номерами 1 – 9, фрагменты графов которых не пересекаются на уровне маршрутов транзактов. В каждой из плоскостей могут находиться узлы типа parent, pay, rent, или down, в свою очередь порождающие новые плоскости. Порождаемые плоскости имеют номера, начинающиеся с 10.

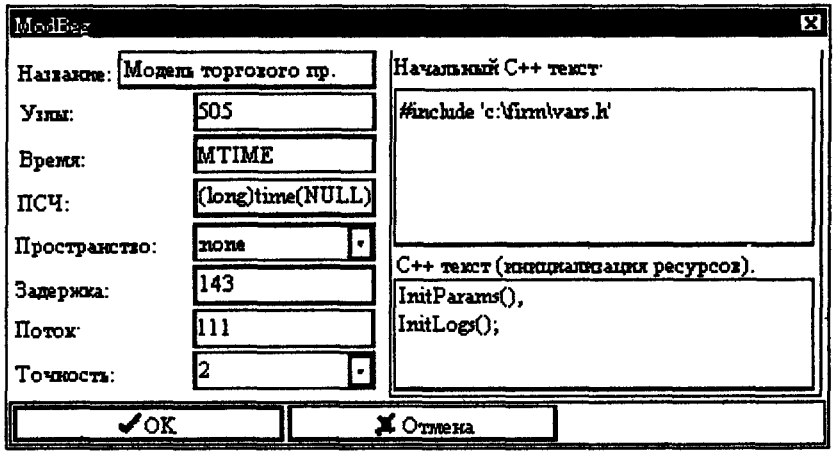


Рис. 5.7. Окно определения параметров инициализации модели

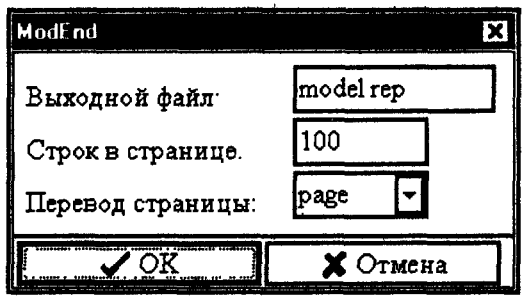


Рис. 5.8. Окно определения переменных функции modend

Возможны следующие операции перехода с уровня на уровень: переход на детализирующую плоскость, подъем на плоскость узла-родителя и переключение между корневыми плоскостями. Для каждой открытой плоскости конструктор предоставляет собственное окно построения.

Переключение между корневыми плоскостями осуществляется через верхнюю панель управления. В правой части панели расположено специальное выпадающее меню, позволяющее выбирать номер корневой плоскости. При этом в информационной строке появится надпись «Корень» и номер плоскости.

Для перехода на плоскость, детализирующую узел, необходимо дважды щелкнуть мышью по узлу-родителю. При этом в рабочей области экрана будет активизирована детализирующая плоскость, в информационной строке на панели «имя плоскости» появятся имя узла-родителя, а также номер текущей плоскости. Для плоскости необходимо задать вход и выход, как номера узлов, в которые попадает транзакт при переходе на плоскость из порождающего узла и из которого происходит возврат на верхнюю плоскость. Узел-вход и узел-выход обозначены на графе входящей стрелкой из левого верхнего угла рабочей области и исходящей в правый верхний угол области соответственно. До тех пор пока вход и выход плоскости не определены, в информационной строке панели красным цветом будет выведено «Вход не назначен» и «Выход не назначен». Чтобы назначить вход и выход, необходимо на левой панели нажать кнопку «Слой». При этом будет выведено диалоговое окно (рис. 5.9).

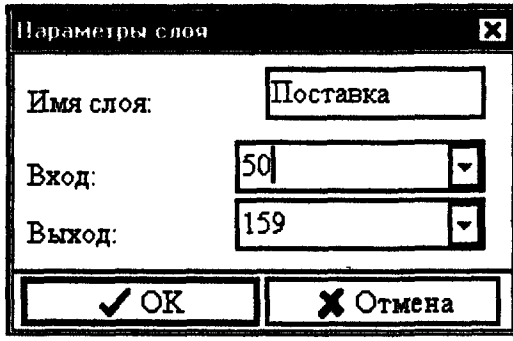


Рис. 5.9. Окно определения параметров плоскости

Возврат на плоскость, которой принадлежит порождающий узел (т.е. на уровень выше), выполняется нажатием кнопки «подняться на уровень вверх» в панели «горячих кнопок» или выбором аналогичного подпункта основного меню из раздела «Уровень».

Для удобства навигации в многослойной модели предусмотрено диалоговое окно, отображающее иерархическую зависимость между плоскостями или узлами модели (рис. 5.10).

Определение переменных модели. Конструктор обеспечивает пользователя простым блоком диалога, который позволяет задавать имя, тип и начальное значение переменных (рис. 5.11).

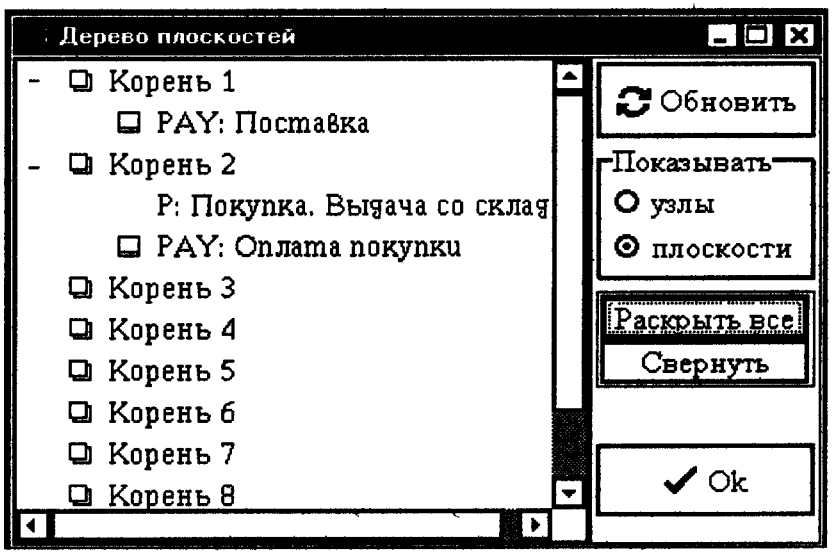


Рис. 5.10. Окно определения иерархии плоскостей модели

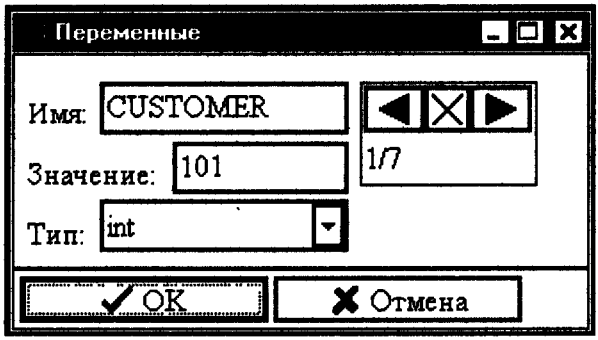


Рис. 5.11. Окно описания переменных

При вводе новой переменной обязательными параметрами являются имя и тип, значение пользователь может указывать по своему усмотрению.

Дополнительные функции. Конструктор содержит ряд функций, позволяющих сделать работу пользователя более удобной и простой. Ниже приведен перечень сервисных функций, реализованных в конструкторе.

Изменения настроек экрана. Изменение настроек экрана выполняется через диалоговое окно, изображенное на рис. 5.12 (основное меню, пункт «Настройки»).

Рис. 5.12. Окно настройки экрана

Пользователь–разработчик модели может по своему усмотрению изменять масштаб, цвет фигур и фона, показать или скрыть панель инструментов и установить признак прямых стрелок. Можно работать на мониторах с различным разрешением и размером диагонали, поэтому очевидна необходимость изменения масштаба в области построения графа (т.е. размеров фигур, обозначающих узлы, стрелок, надписей и других изображений). Разработчик может разместить на одной плоскости большое количество узлов; чтобы их увидеть на одном экране, необходимо уменьшить масштаб. Механизм масштабирования дополняют линейки прокрутки, располагающиеся по бокам рабочей области построения графа.

Если признак прямых стрелок выключен, то при перемещении узла внутри плоскости промежуточные звеньевые точки стрелок, привязанных к нему, будут оставаться на месте, а при включенном признаке стрелки будут постоянно иметь прямой вид.

Проверка корректности модели. Конструктор проводит проверку двумя независимыми способами:

• не позволяет осуществлять заведомо ложные действия при редактировании пользователем модели;

• имеет возможность проверки графа в целом.

Результат проверки графа выдается в окне, приведенном на рис. 5.13, как список ошибок возможных и ошибок явных, не позволяющих генерировать программный Pilgrim-файл. К явным ошибкам относятся отсутствие выходов или входов узла и неопределенные вход/выход плоскости.

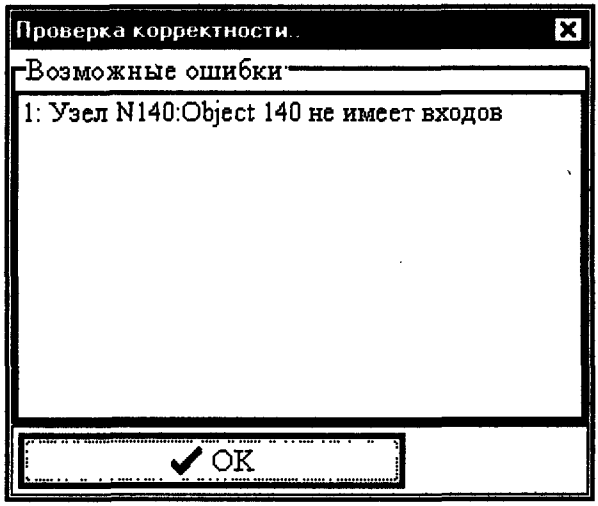


Рис. 5.13. Окно проверки корректности графа модели

Процедура проверки корректности графа вызывается автоматически при запросе пользователя сгенерировать программный Pilgrim-файл, а также в любой момент времени из пункта «Выполнить» основного меню. При этом осуществляется проверка модели целиком, независимо от того, в какой плоскости работает пользователь.

План бухгалтерских счетов. При моделировании бухгалтерских проводок необходимо знать номер счета-приемника. Поскольку номер любого счета – это номер узла, его моделирующего, то для того, чтобы проще ориентироваться в их нумерации, существует информационное окно, отображающее номера и названия счетов. Это

окно может быть вызвано выбором подпункта «План счетов» пункта «Модель» основного меню программы.

Поиск узла. Функция вызывается через меню «Узел» либо через «горячую кнопку» на верхней панели (рис. 5.14). Поиск может осуществляться по номеру или по параметрам узла, таким, как имя или тип. Результаты поиска выводятся в виде списка, откуда можно перейти на любой узел, редактировать его параметры или скопировать узел в буфер.

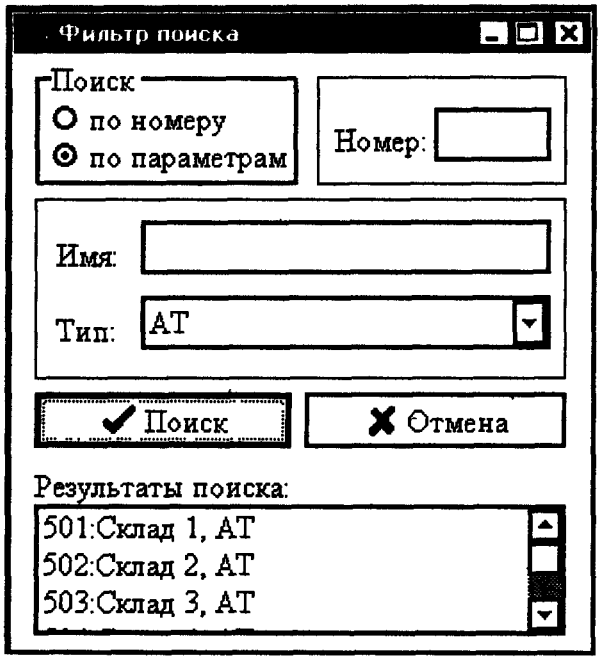


Рис. 5.14. Окно поиска узла

Копирование/вставка узла. Для упрощения создания набора узлов с рядом совпадающих параметров в конструкторе предусмотрена функция копирования/вставки узла. Чтобы скопировать узел в буфер, необходимо выделить его щелчком мыши и выбрать пункт «Копировать» из меню «Узел». Вставка в текущую плоскость осуществляется выбором пункта «Вставить» меню «Узел». При этом будет создан узел того же типа, что и прототип, и с аналогичными параметрами.

Очистить плоскость. Функция удаляет все узлы текущей плоскости. Вызывается через меню «Плоскость». Операция недопустима и выдаст сообщение об ошибке в случае, если хотя бы один из узлов плоскости содержит декомпозицию.

5.4 ПРИМЕР СОЗДАНИЯ ИМИТАЦИОННОЙ МОДЕЛИ ЭКОНОМИЧЕСКОГО ПРОЦЕССА

В качестве примера, иллюстрирующего создание моделей с помощью конструктора, рассмотрим следующий объект. Имеется предприятие, принимающее заказы на изготовление продукции типов А и В. Заказы на продукцию типа А поступают в среднем раз в 30 дней, а на продукцию типа В – раз в 5 дней. Для производства продукции типа А необходимо 10 дней, а для продукции типа В – 2 дня. Одновременно может идти производство только по одному заказу. При этом заказы на продукцию А являются первоочередными. Все сведения о выполнении заказов типов А и В фиксируются в отдельных документах. Модель строится для исследования очереди задержек заказов.

Необходимо заметить, что модель является достаточно простой и легко может быть построена в одной плоскости конструктора. Но для демонстрации механизма построения многоуровневых моделей создадим двухуровневую модель.

Опишем создание имитационной модели с помощью графического конструктора как последовательность технологических шагов.

Ш а г I. Создание графа модели.

1. Рассмотрим в качестве автономных процессов:

- производство и фиксацию выполненных заказов;
- формирование заказов и очередь заказов на производство.

2. Определим единицу модельного времени для последующего формирования параметров модели. Очевидно, в данном примере за единицу времени удобно взять один рабочий день.

3. Запустим конструктор на выполнение.

4. В основном меню выберем «Файл» → «Создать». В результате, в рабочей области экрана появится пустое окно с именем «Плоскость 1». Это пустая корневая плоскость.

5. Из панели узлов в левой части экрана перетащим в плоскость построения узлы типов parent, serv. В панели узлов также найдем узел типа ag и щелкнем мышью по узлу ag. Надпись на панели сменится на term. Добавим в плоскость построения два узла типа term. Расположим узлы в рабочей области так, как показано на рис. 5.15.

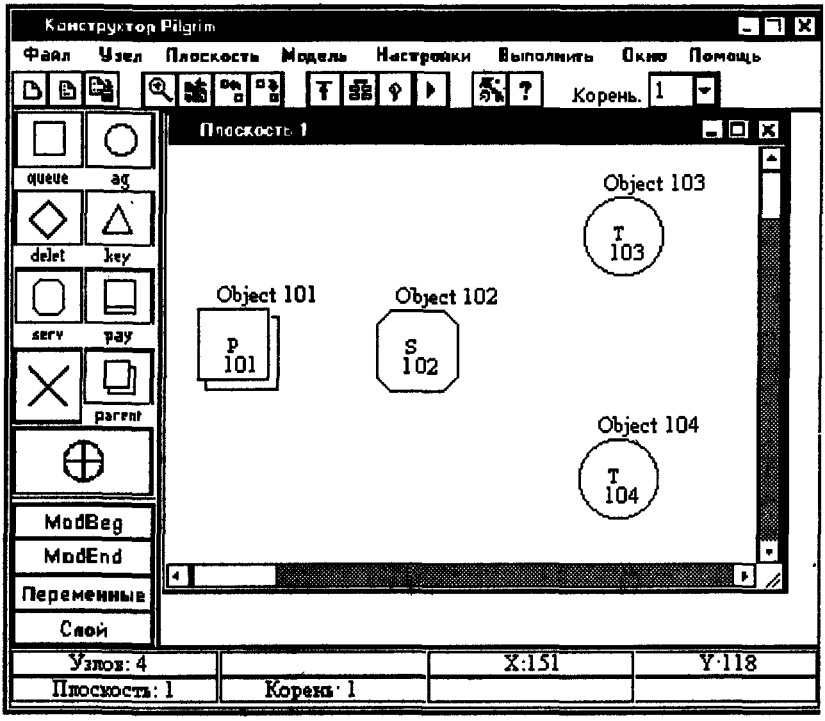


Рис. 5.15. Окно создания узлов в корневой плоскости

6. Зададим имена узлов. Для этого нужно дважды щелкнуть мышью по каждому из узлов и в появившемся диалоговом окне в поле «Имя» вместо находящегося там «Object...» ввести: «Производство» для узла 102; «Отчет А» для узла 103, «Отчет В» для узла 104. Узел 101 будет содержать подуровень формирования заказов; он будет рассмотрен позже.

7. Определим маршруты транзактов. Для этого захватим изображение круга с перекрестием на панели узлов и перетащим на узел

«Производство». После отпускания кнопки мыши за курсором потянется стрелка. Далее необходимо щелкнуть мышью по узлу «Отчет А», в результате чего появится стрелка, символизирующая маршрут прохождения транзактов из узла «Производство» в «Отчет А». Аналогичную операцию повторим, задав маршруты из узла «Производство» в «Отчет А» и из «Object 101» в «Производство».

Предостережение. Необходимо сначала указывать узел-источник и потом – узел-приемник. Последующая смена направления невозможна. При создании неверного маршрута необходимо его удалить через свойства узла и потом задать новый. По окончании этого этапа должен появиться экран, подобный приведенному на рис. 5.16.

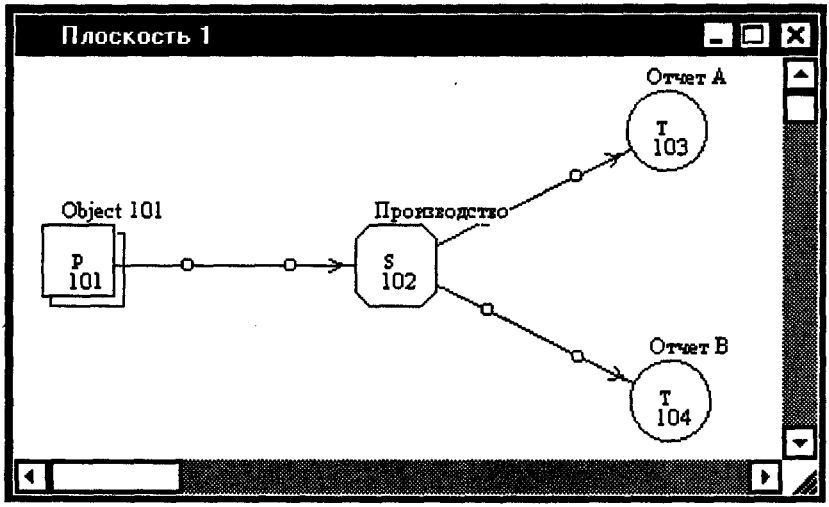


Рис. 5.16. Окно графа корневой плоскости

8. Создадим плоскость формирования заявок. Дважды щелкнем по узлу «Object 101», в результате чего получим новую пустую плоскость 10 с именем «Object 101». В этой плоскости теми же операциями, что и ранее, создадим граф, как показано на рис. 5.17. Останемся в этой плоскости.

9. Можно увидеть, что в информационной строке в нижней части экрана красным цветом отображены надписи «Вход не назначен» и «Выход не назначен». В нашем случае плоскость 10 не будет иметь

входа, так как сама предназначена для генерации заявок, однако выход необходимо назначить. В левой части экрана щелкнем по кнопке «Слой». В диалоговом окне введем имя слоя «Заказы» и выход 107. Вход установим на значение «none», т.е. отсутствующий. Результат приведен на рис. 5.18. После нажатия кнопки «OK» одна из красных надписей в информационной строке будет заменена на «Выход: 107».

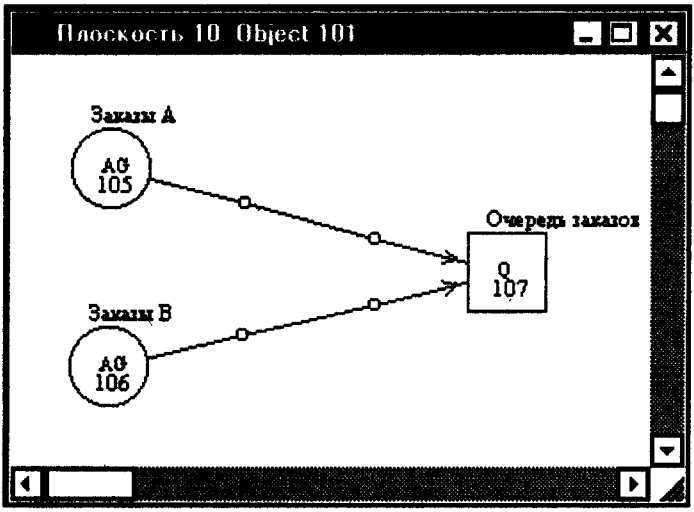


Рис. 5.17. Окно графа плоскости 10

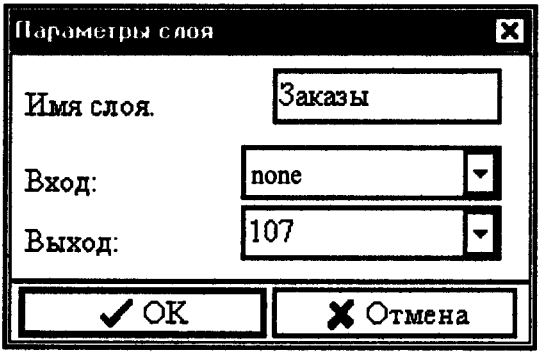


Рис. 5.18. Окно определения параметров слоя 10

Ш а г II. Определение глобальных переменных модели.

Определим переменные модели. Очевидно, для работы может понадобиться переменная времени обслуживания заявки. На левой панели щелкнем по кнопке «Переменные». Появится соответствующее диалоговое окно (рис. 5.19). В нем необходимо ввести: имя переменной proc_time, тип переменной float. Значение можно не указывать, так как оно будет определяться при выполнении модели в зависимости от типа продукции.

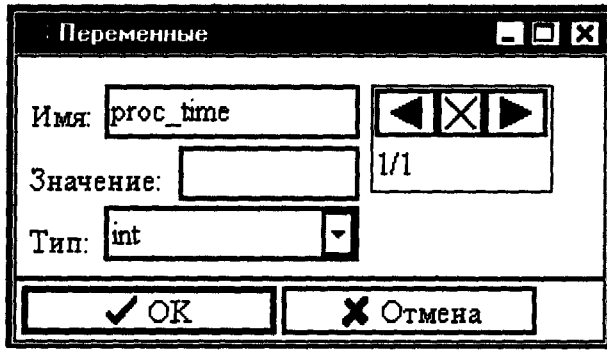


Рис. 5.19. Окно определения глобальных переменных модели

Ш а г III. Определение свойств узлов.

1. Определим свойства узлов плоскости 10. Узел 105 будет формировать заказы типа A, узел 106 – типа B. Необходимо дважды щелкнуть мышью по узлу 105, в результате чего появится диалоговое окно «Свойства узла». В нем щелкнуть по кнопке «Определить параметры». В результате появится окно определения параметров узлов типа ag (рис. 5.20). Очевидно, для заявок типа A значение «Приоритет» нужно указать равным 1. На выборе закона распределения не будем останавливаться, выберем его как norm – нормальное. В строке «Мат. ожидание» введем значение 30, а «Отклонение» примем равным 10. Повторим процедуру для узла 106. Но значение приоритета укажем 2, математическое ожидание примем равным 5, а отклонение – равным 2. Теперь генераторы подготовлены.

2. Аналогичным образом определим параметры для узла 107 «Очередь заявок». Здесь всего один параметр, его необходимо установить в значение «prty». Это будет означать, что заказы проходят

очередь в соответствии с приоритетами. Поэтому заявки на производство продукции типа А будут проходить прежде заявок на продукцию типа В.

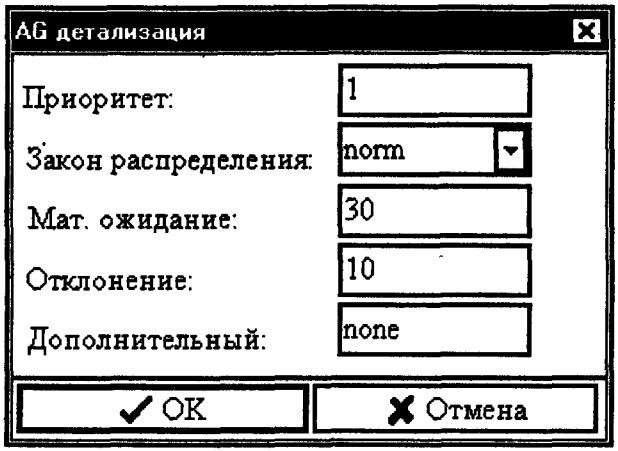


Рис. 5.20. Окно определения переменных узла типа ag Заказы А

3. Определим свойства узлов плоскости 1. Начнем с узла 102 типа serv. Дважды щелкнем по нему мышью – появится диалоговое окно «Свойства узла» (рис. 5.21). В поле «Выходы» на выберем строку «Из 102 в 103». В поле «Условие перехода» укажем «t→pr = = 1». Это значит, что по этой ссылке перейдут только те транзакты, которые в качестве параметра t→pr (приоритет) имеют значение 1. В поле «С++ текст», располагающееся под «Условием перехода», введем строку «proc_time = 10». Далее, выбрав в поле «Выходы» строку «Из 102 в 104», в поле «С++ текст» напишем «proc_time = 2». Щелкнем по кнопке «Определить параметры». В появившемся диалоговом окне «Serv детализация» зададим параметры: число каналов = 1; приоритетность = abs; закон распределения = norm; математическое ожидание = proc_time; отклонение = proc_time/2; дополнительно = none (рис. 5.22). Таким образом, определен обслуживающий прибор с одним каналом и временем обслуживания, зависящим от приоритета транзакта.

4. Для узлов типа term параметров не существует.

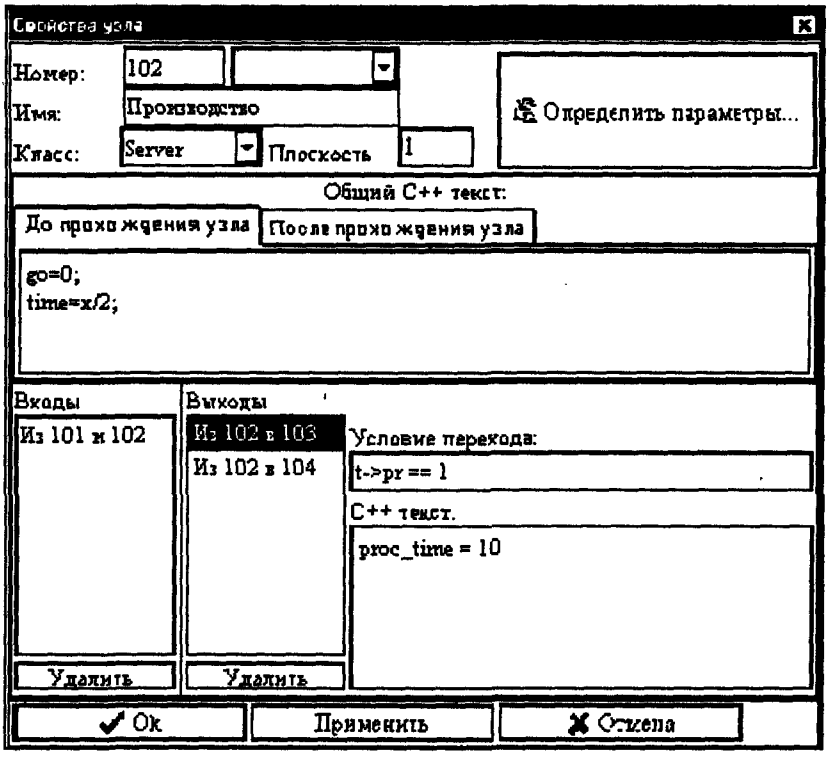


Рис. 5.21. Окно определения общих свойств узла 102 Производство

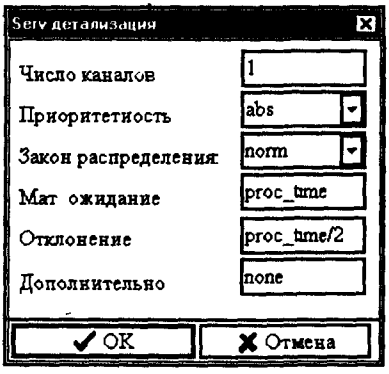


Рис. 5.22. Окно определения переменных функции serv для узла Производство

Ш а г IV. Определение параметров функций modbeg и modend.

1. Определим параметры функции modbeg. В левой панели инструментов щелкнем по кнопке modbeg либо выберем соответствующий пункт меню раздела «Модель». В появившемся диалоговом окне зададим имя модели «Очередь заказов». Параметр «Узлы» обозначает порядковый номер последнего узла модели и рассчитывается автоматически. Поле «Время» подразумевает время выполнения модели, в моделируемых единицах времени. Поскольку за единицу времени взят один день, то рассмотрим модель на отрезке времени 365 единиц, т.е. один год. Параметр «ПСЧ» служит для настройки датчика случайных чисел. Значение, отображенное по умолчанию, позволяет при каждом запуске получать различные результаты моделирования. Поле «Пространство» используется при пространственном моделировании и в нашем случае не понадобится. В поле «Задержка» необходимо ввести номер узла типа queue, информация о котором будет отображаться при выполнении модели, т.е. в нашем случае это номер 107. Поля «Поток» и «Точность» оставим без изменений. Результат определения параметров в диалоговом окне приведен на рис. 5.23.

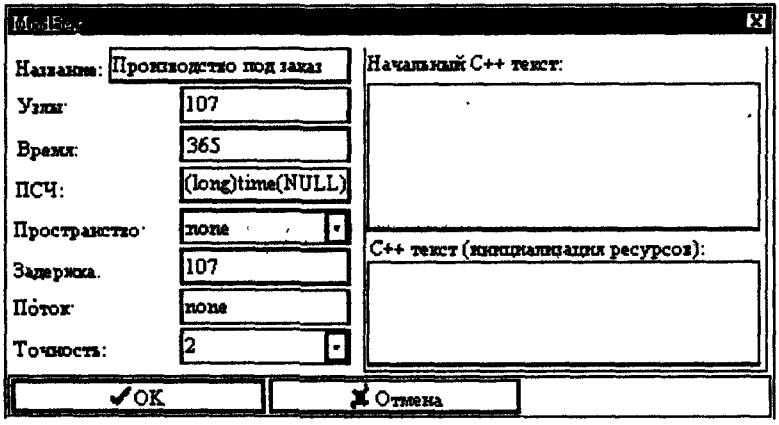


Рис. 5.23. Окно определения параметров функции modbeg

2. Определим параметры функции modend. Для нашей задачи оставим параметры modend, заданные по умолчанию. Вызов диалогового окна для ввода параметров осуществляется аналогично вызову окна modbeg.

Ш а г V. Генерация текста имитационной модели в операторах Pilgrim.

Итак, модель готова. Для нее сформирован граф, заданы переменные, определены параметры узлов и функций. Для генерации программного файла необходимо в основном меню выбрать «Выполнить», а затем – «Генерировать C++ файл». При этом конструктором сначала будет выполнена проверка модели, в нашем случае не обнаружившая никаких ошибок или подозрительных участков на графе модели (рис. 5.24). После нажатия кнопки OK будет выведено стандартное диалоговое окно, предлагающее сохранить файл с расширением «cpp». Сохраненный конструктором файл можно далее компилировать в среде Visual C++.

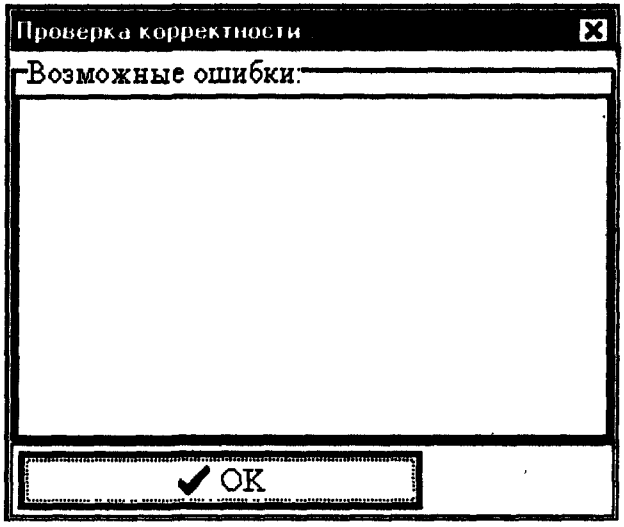


Рис. 5.24. Окно проверки корректности модели

Программная модель, автоматически сгенерированная конструктором и помещенная в cpp-файл, имеет следующий вид:

```
#include <Pilgrim.h>
forward
   {
    int fw;
    int proc_time;
```

```
 modbeg("Производство под заказ", 107, 365,
        (1ong)time(NULL), none, 107, none,none, 2);
 ag("Заказы А", 105, 1, norm, 30, 10, none, 107);
 ag("Заказы В",  106, 2, norm, 5, 2, none, 107);
 network(dummy, dummy)
    {
    top(102):
    if( t->pr == 1 )
       {
       proc_time = 10
       fw=103;
       }
    else
       {
       proc_time = 2
       fw=104;
       }
       serv("Производство", 1, abs, norm,
                proc_time,  proc_time/2, none, fw);
       place;
    top(103):
       term("Отчет А");
       place;
    top(104):
       term("Отчет В");
       place;
    top(107):
       queue("Очередь заказов", prty, 102);
       place;
    fault(123);
}
modend("pilgrim.rep", 1, 8, page);
return 0;
}
```

В тексте этой модели нет ни одной строчки, написанной вручную, без использования графического конструктора. Более того, работу с конструктором вполне может освоить специалист, не очень хорошо знающий программирование.

5.5 РАЗВИТИЕ ПРОБЛЕМНО-ОРИЕНТИРОВАННОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА С МОДЕЛЯМИ

В последние годы явно проявилась тенденция изменения технологий разработки приложений в направлениях, максимально ориентированных на проектирование систем и все более отдаляющихся от программирования как кодирования на языке программирования. Примером могут служить находящие все большее распространение CASE-средства, предусматривающие эффективную быструю разработку сложных моделей, баз данных, программ. В случае отсутствия в пакете графической оболочки, предоставляющей новые возможности создания программного обеспечения по сравнению с простым кодированием, его разработчикам (производителям компиляторов, имитаторов, интерпретаторов, словом, любых средств, использование которых подразумевает написание программы) все сложнее удержаться на рынке программных средств. Существуют многие причины такой тенденции. Приведем лишь некоторые из них.

1. Разработчики приложений стремятся создавать программное обеспечение (ПО), и в частности имитационные модели, в наиболее сжатые сроки по причине конкуренции со стороны альтернативных групп разработчиков.

2. Постоянно возрастающая сложность систем, а также увеличение объемов данных, используемых в рамках единой системы, порождают жесткое требование понятного графического отображения структуры информации.

3. Конечные пользователи стремятся применить новые технологии, возникающие вследствие развития аппаратного обеспечения вычислительных машин. Использование новых технологий – это следствие эволюции программных средств.

4. Расширяется круг разработчиков. В настоящее время (в отличие от 1970-х – 1980-х гг., когда большинство разработчиков ПО являлись профессиональными программистами) все больший круг программных продуктов разрабатывается людьми, имеющими опыт в той сфере, к которой относится разрабатываемый продукт, но не имеющими фундаментальных знаний в области программирования.

5. Расширяется круг пользователей. Грань между пользователями и разработчиками оказывается все более размытой. Практически

в каждой организации есть от одного до нескольких человек, умеющих не только пользоваться установленным ПО, но и разрабатывать дополнительное ПО, нужное для применения в конкретной области. Поэтому естественным плюсом любой инструментальной среды, предназначенной для разработки, является легкость ее использования.

Следовательно, чтобы поддерживать программный продукт, предназначенный для создания ПО, на привлекательном для клиента уровне, кроме хороших возможностей по выполнению основных функций нужно обеспечить в нем эффектную «внешность»: интерфейс с технологической оболочкой и с создаваемым программным изделием. Такой интерфейс должен согласовываться с методикой создания ПО.

Рассмотрим систему имитационного моделирования – пакет Pilgrim. Направленный на создание имитационных моделей, Pilgrim имеет круг потенциальных пользователей в лице системных аналитиков, экономистов-математиков, а также других специалистов, знакомых с программированием, но не являющихся программистами-профессионалами.

Изначально заложенные в Pilgrim функции моделирования процессов не теряют своей актуальности и применимы к широкому кругу задач. Однако развитие аппаратного обеспечения, а также эволюция операционных систем и конкурирующих продуктов заставляют изменять внешний вид системы, функциональность ее интерфейса для удобства пользователя. Пакет Pilgrim создавался как язык, лишенный оболочки, и позволял задавать модель в виде программного файла; результаты моделирования также сохранялись в файле. Далее Pilgrim приобрел интерфейс, содержащий меню и графики в текстовом режиме.

С появлением Windows 95/98/NT для Pilgrim были разработаны многочисленные модули, обеспечивающие графическое отображение результатов в окне Windows. Имеются возможности настройки окна отображения результатов в удобной для пользователя форме. Однако разработка самой модели оставалась задачей программиста. Новой задачей модификации Pilgrim стала разработка графического конструктора, позволяющего задавать модели в виде графа, отвлечься от кодирования и использовать иерархию процессов.

Конструктор позволяет создавать модели в соответствии с поставленными требованиями. Однако представляется целесообразным расширение функциональных возможностей, дополнение его новыми средствами.

Следует также отметить, что первые версии пакета Pilgrim отличались способностью переноса на различные платформы, единственным требованием к которым было наличие компилятора C++. Библиотеку пакета можно было одинаково эффективно использовать и в среде Unix, и в среде MS DOS. Созданная для Unix модель могла успешно компилироваться и в MS DOS, а ориентированный на конкретную платформу интерфейс только интерпретировал результаты выполнения.

При разработке интерфейса, естественно, возникает вопрос выбора операционной системы. Первый интерфейс Pilgrim в виде графического окна был реализован при использовании MS DOS; многие графические функции появились с переходом к Windows 95. Однако основная вычислительная библиотека языка Pilgrim практически не изменяется. Она наращивается по количеству новых расчетных программ, постепенно повышается точность в связи с переводом программ на более длинную разрядную сетку. Создание конструктора и оснащение его функциями, приведенными ниже, должны отразиться не только на библиотеке Pilgrim, но и на функциях, используемых в моделях. Дальнейшая эволюция Pilgrim должна привести к более тесной интеграции интерфейса системы и библиотеки, содержащей описание моделируемых функций.

Выделим перспективные диалоговые функции конструктора, добавление которых должно стать задачей для выпуска новой версии системы Pilgrim:

1) отображение процесса имитации на графе модели в реальном времени;

2) расширение графических средств отображения процесса имитации;

3) установление на графе модели контрольных точек, при попадании транзакта в которые возможно наступление различных событий, таких, как вывод диалогового окна или приостановка выполнения имитации;

4) предоставление интерфейса ввода начальных параметров;

5) хранение моделей в виде проектов, содержащих в качестве компонентов граф модели, описание интерфейса и дополнительных функций.

Многие сервисные функции уже есть в различных версиях системы. Рассмотрим их подробнее.

Конструктор работает с графом модели, описание которого хранится в собственном внутреннем формате. Граф можно сохранять,

загружать, модифицировать и на его основе в конечном итоге генерировать программный файл. Однако после генерации файл полностью теряет связь с конструктором моделей.

Модель при выполнении может выводить информацию на экран в виде, определяемом пользователем. Во время отладки модели (или при пошаговом просмотре процесса имитации) пользователю необходимо выполнять трассировку модели специально заложенной функцией Pilgrim. Результаты трассировки выводятся в окне выполнения модели в виде текстовых данных, содержащих номер активного транзакта, узел его нахождения и другие параметры. Естественно, не имея возможности помнить модель целиком с номерами узлов, пользователь вынужден постоянно сверять результаты с графом, построенным с использованием конструктора. Очевидным улучшением системы представляется отображение имитации непосредственно на графе модели, созданном в конструкторе. Такая функция позволит кроме удобной отладки модели также просматривать ход моделирования.

Помимо имитации на графе модели Pilgrim должен также предоставлять средства отображения выходных данных в виде графиков, таблиц, диаграмм, а также анимации. На данный момент в библиотеке Pilgrim такие средства заложены, однако набор их ограничен, и в случае желания заказчика получить дополнительные способы отображения результатов разработчику необходимо писать C++ код, использующий стандартные средства Visual C++.

Данный способ не очень эффективен. Поэтому предлагается разработать в рамках конструктора инструмент, позволяющий проектировать интерфейс для модели Pilgrim. В качестве входных параметров такой интерфейс должен использовать те же сигналы, что и блок имитации выполнения на графе модели в режиме трассировки.

Реализация дополнительных функций неизбежно приведет к изменению в составе и структуре конструктора (рис. 5.25). Основой для новых функций является механизм обмена данными между выполняющейся моделью и конструктором.

Для пакета Pilgrim такой механизм реализован в виде функции передачи данных во внешнюю программу через общую область памяти (или через временный файл). Данные, передаваемые моделью, обрабатываются конструктором через блок приема данных. В блоке приема данных и библиотеке Pilgrim должны быть механизмы синхронизации выполнения модели и отображения оперативных результатов на графе.

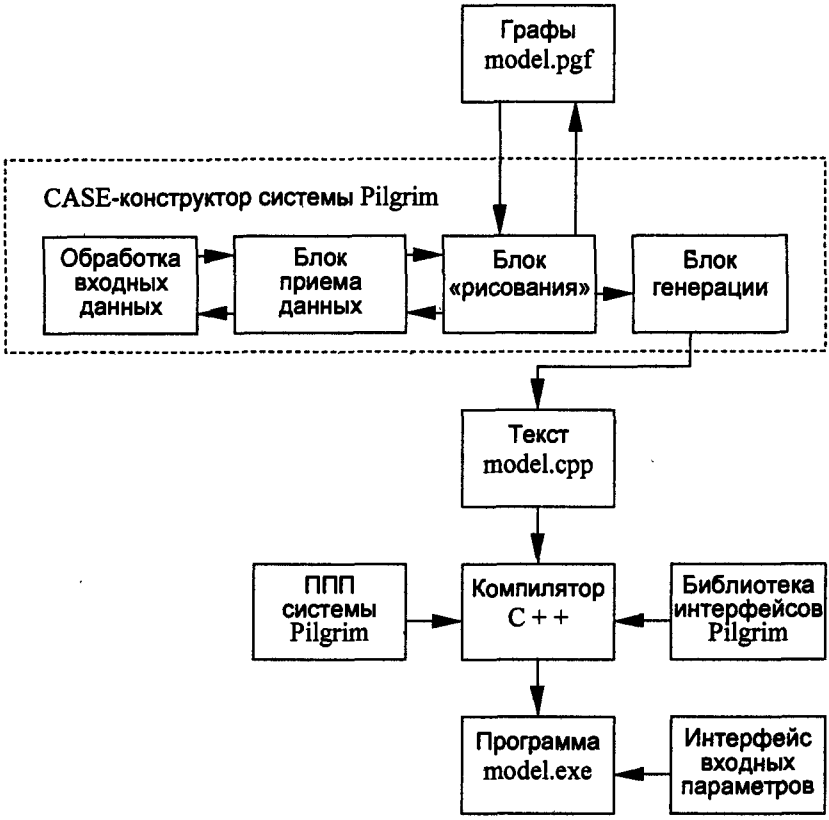


Рис. 5.25. Технологическая схема взаимодействия компонентов системы имитационного моделирования

Перспективный состав функций блока обработки может быть очень широк. Например, он мог бы включать в себя функции управления моделью через WEB или через рассылку электронных писем. Очевидно, такой блок должен иметь иерархическую структуру, на нижнем уровне которой будут располагаться способы представления и обработки информации. Набор способов можно последовательно обновлять.

Контрольные точки модели используются как средства отладки, а также для выполнения моделей, управляемых пользователями в реальном времени. Контрольные точки должны создаваться пользователем в виде установки меток на любом узле графа с описанием

действия, осуществляемого моделью, при соблюдении некоторого условия контрольной точки.

Система при выполнении контрольной точки может отрабатывать любые события. Что касается механизма реализации контрольных точек в системе Pilgrim, то наиболее удобным способом представляется пересылка информации о наступлении контролируемого события через общую систему обмена информацией между моделью и конструктором. В этом случае в тексте модели конструктор может размещать простые обращения к функциям, имитирующим наступление событий.

Зачастую пользователю модели для проведения эксперимента необходимо прогонять ее множество раз, изменяя входные параметры. При этом само изменение параметров модели может занимать гораздо меньше времени, чем перекомпиляция. Поэтому в системе Pilgrim реализована подсистема, позволяющая задавать входные параметры для уже скомпилированной модели. В таком случае набор входных данных, требующих настройки, определяется в самом конструкторе. Модуль, обеспечивающий их настройку, может быть выполнен как независимый программный файл или как часть конструктора.

Для реализации новых функций потребуется одновременное поддержание в проекте нескольких относительно независимых структур данных. В связи с этим предлагается хранить на диске проект, включающий в себя описание графа, интерфейса и общие настройки.

Добавление в конструктор всех перечисленных возможностей позволит разработчикам моделей создавать их быстрее и с большей эффективностью. Конечный пользователь, в свою очередь, получит совершенно новый уровень наглядности имитации и, следовательно, понимания процесса моделирования.

Все эти факторы в конечном итоге должны отразиться на увеличении привлекательности моделирующей системы Pilgrim как для разработчика моделей, так и для конечного пользователя.

Выводы

1. CASE-конструктор имитационных моделей позволяет проводить разработку многоуровневой Pilgrim-модели практически без применения языка Pilgrim.

2. В модели по невнимательности разработчика могут возникать ошибки, которые приводят к неадекватной работе и неправильным результатам. Многослойная конструкция модели позволяет повысить ее наглядность и резко уменьшить количество таких ошибок.

3. Видимое увеличение слоев модели – это следствие борьбы с ошибками (дефектами) имитационной модели. Умелое добавление нового слоя в модель приводит к уменьшению суммарного числа узлов на всех уровнях иерархической структуры.

Вопросы для самопроверки

1. Для чего нужен конструктор имитационных моделей?
2. Какие достоинства CASE-технологий учитываются при автоматизированном создании имитационных моделей?
3. Из каких компонентов состоит имитационная модель «с точки зрения» конструктора Pilgrim?
4. В чем состоит основное достоинство конструктора Pilgrim?
5. Какие файлы должны быть на входе и создаются на выходе конструктора?
6. Какие инструментальные приемы существуют для редактирования графа модели?
7. Как определяются и переопределяются параметры узла модели?
8. Каким образом можно определить параметры инициализации и завершения модели?
9. Возможно ли выполнять одновременно работу в разных плоскостях модели при ее проектировании?
10. Какие могут быть переменные в модели? Как они определяются?
11. Для чего нужны дополнительные (специальные) функции конструктора Pilgrim?
12. Какие возможности имеются для изменения настроек экрана конструктора?
13. Насколько подробно выполняется проверка корректности модели?
14. Какие особые приемы (средства) имеются для работы с планом бухгалтерских счетов?
15. Имеются ли средства однозначного поиска узла в сложной многослойной модели?
16. Какова технология копирования или вставки узла в модели?
17. Можно ли полностью очистить плоскость – слой модели?

18. Какие основные технологические шаги для создания модели необходимо выполнить с помощью конструктора Pilgrim?
19. Почему появляется многослойная конструкция модели и что она дает разработчику?
20. Что может привести к уменьшению суммарного числа узлов на всех уровнях иерархической структуры модели?
21. Почему узел parent виртуален? Как проводится детализация с его помощью?
22. Можно ли использовать узел типа pay в качестве средства реализации подпрограмм?
23. Какие основные технологические окна имеет конструктор Pilgrim?
24. В чем польза контрольных точек?
25. Какова технология генерации текста имитационной модели в операторах Pilgrim с помощью графического конструктора?

Глава 6

МОДЕЛИРОВАНИЕ В ГЕОПРОСТРАНСТВЕ И ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ

6.1 ГЕОГРАФИЧЕСКАЯ КАРТА, ПЛАН ТЕРРИТОРИИ И ГЕОИНФОРМАЦИОННАЯ СИСТЕМА

Карта – это математически определенное, уменьшенное, генерализованное изображение поверхности Земли, другого небесного тела или космического пространства, показывающее расположенные или спроецированные на них объекты в принятой системе условных знаков*.

Планом называется плоское графическое изображение контуров основных объектов хозяйственной территории, выполненное с соблюдением масштабов в предположении, что эта территория расположена на плоскости, а не на поверхности Земли. Для ориентации на плане вводятся только прямоугольные координаты X и Y. Географические координаты на плане не действуют по очень простой причине: для небольших участков они дают существенные погрешности. Рассмотрим земной меридиан; его протяженность составляет приблизительно 20 096 км. Расстояние по меридиану, соответствующее 1 градусу, – это приблизительно 111,6 км. Одной минуте соответствует 1,861 км, а 1 секунда – это 31 м. Причем таких карт, где масштаб позволял бы работать с точностью до секунды, практически нет в обращении.

* Берлянт А.М. Картография. – М.: Аспект Пресс, 2001. (Основные термины и определения картографии, не относящиеся к теории имитационного моделирования, использованы из данного учебника.)

Планы составляются для производственных комплексов (комбинатов, заводов, фабрик), фермерских хозяйств, небольших деревень (хуторов, усадеб), отдельных садовых участков и садовых обществ.

Элементы карты – это ее составные части, включающие само картографическое изображение, легенду и зарамочное оформление (рис. 6.1).

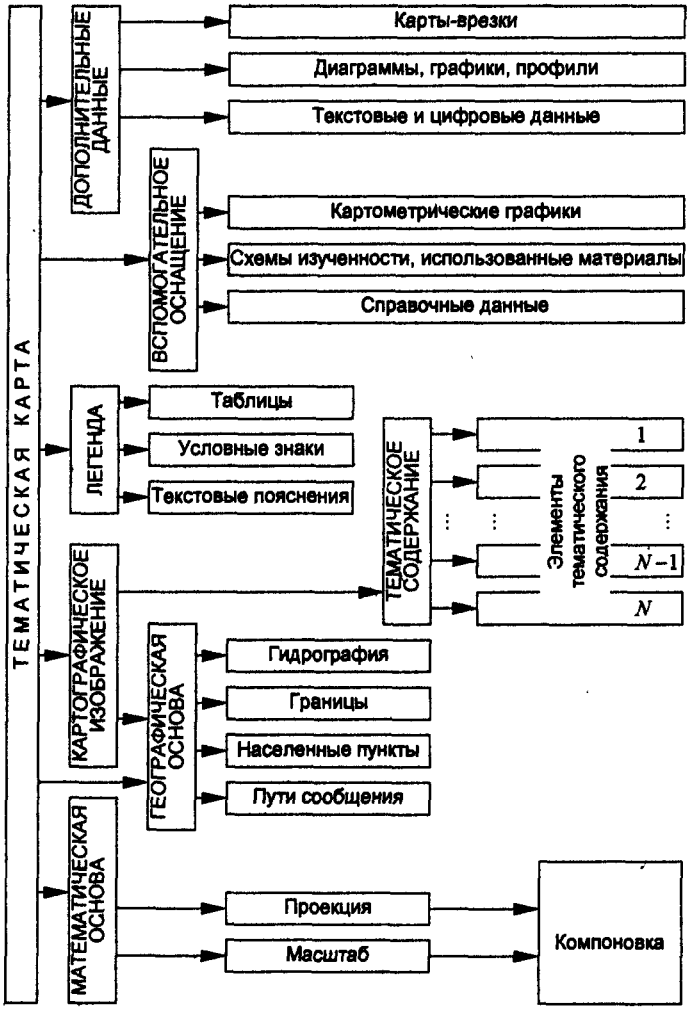


Рис. 6.1. Типы элементов тематической карты

Основной элемент – *картографическое изображение*, т.е. содержание карты, совокупность сведений об объектах и явлениях, их размещении, свойствах, взаимосвязи и динамике. Общегеографические карты имеют следующее содержание: населенные пункты, социально-экономические и культурные объекты, пути сообщения и линии связи, рельеф, гидрографию, растительность и грунты, политико-административные границы.

На тематических и специальных картах различают две составные части картографического изображения:

- географическую основу, т.е. общегеографическую часть содержания, которая служит для нанесения и привязки элементов тематического или специального содержания, а также для ориентировки по карте;
- тематическое, или специальное, содержание (насыщенность территории объектами экономической деятельности, спрос на конкретные товары, спрос-предложение товаров, демографическая обстановка, геологическое строение территории, навигационная обстановка, инфраструктура города).

Важнейший элемент всякой карты – *легенда,* т.е. система использованных на ней условных обозначений и текстовых пояснений к ним. Для топографических карт составлены специальные таблицы условных знаков. Они стандартизированы и обязательны к применению на всех картах соответствующего масштаба. На большинстве тематических карт обозначения не унифицированы, поэтому легенду размещают на самом листе карты. Она содержит разъяснения, истолкование знаков, отражает логическую основу и иерархическую соподчиненность картографируемых явлений.

Последовательность обозначений, их взаимное соподчинение в легенде, подбор цветовой гаммы, штриховых элементов и шрифтов – все это подчинено логике классификации изображаемого объекта или процесса. На сложных картах для повышения информативности легенды ее иногда представляют в табличной (матричной) форме. Тогда по строкам легенды дается один показатель (например, генетическая характеристика объекта), а по столбцам – другой (например, морфологические особенности этого объекта).

Картографическое изображение строится на *математической основе*, элементами которой на карте являются координатные сетки, масштаб и геодезическая основа. На мелкомасштабных картах элементы геодезической основы не показываются. С математической основой тесно связана и *компоновка* карты, т.е. взаимное размеще-

ние в пределах рамки самой изображаемой территории, названия карты, легенды, дополнительных карт и других данных.

Вспомогательное оснащение карты облегчает ее чтение и использование. Оно включает различные картометрические графики (например, на топографической карте помещают шкалу крутизны для определения углов наклона склонов), схемы изученности картографируемой территории и использованные материалы, разнообразные справочные сведения. К *дополнительным данным* относятся карты-врезки, фотографии, диаграммы, графики, профили, текстовые и цифровые данные. Они не принадлежат непосредственно картографическому изображению или легенде, но тематически связаны с содержанием карты, дополняют и поясняют его.

В самом определении карты обозначены основные ее свойства:

- математический закон построения – применение специальных картографических проекций, позволяющих перейти от сферической поверхности Земли к плоскости карты;
- знаковость изображения – использование особого условного языка картографических символов;
- генерализованность карты – отбор и обобщение изображаемых объектов;
- системность отображения действительности – передача элементов и связей между ними, отображение иерархии геосистем.

Свойства карты хорошо понятны при сопоставлении с аэро- и космическими снимками. Снимки дают подробный «портрет», копию местности, но без всяких условных знаков. На снимках территория предстает такой, какова она есть. Картографические условные знаки во многом обогащают изображение. Они позволяют передать количественные и качественные характеристики объектов (например, указать породы леса, ширину и покрытие автодорог, проходимость болот), отразить объекты, недоступные взору человека при рассмотрении снимка (рельеф, плотность населения), наглядно показать даже то, что не воспринимается органами чувств (магнитные склонения, аномалии силы тяжести и др.), передать динамику процессов, их ход во времени и перемещение в пространстве (атмосферные вихри, грузопотоки, миграция населения). Наконец, с помощью условных знаков на карте можно представить расчетные показатели и научные абстракции, скажем, градиент поля температур или степень устойчивости природных ландшафтов к химическому загрязнению, запасы полезных ископаемых или промышленный потенциал (по отраслям).

Картограф сам выбирает знаки и способы изображения, решает, что и как будет показано на карте. Одновременно он проводит отбор и обобщение объектов, т.е. определяет, что важно для данной карты и обязательно должно быть на ней показано, а что не очень существенно и может быть частично или полностью исключено. При этом составитель карты исходит не только из определенных научных принципов, правил и инструкций, но и руководствуется собственным пониманием сути отображаемого явления, его генезиса и значимости в картографируемой геосистеме.

Многие решения, которые принимает картограф, индивидуальны в каждой конкретной ситуации и поэтому трудно формализуемы. Карта в отличие от снимка не является копией местности; это изображение реальности, образно говоря, пропущенное через голову и руки картографа. На снимке представлены только факты, а на карте – еще и научные понятия, обобщения, логические абстракции.

Повышение эффективности государственного и муниципального управления практически невозможно без внедрения современных технологий с использованием вычислительной техники. Одна из важнейших задач, осуществляемых муниципальными и региональными органами власти, – управление недвижимостью. Основными инструментами, позволяющими решать задачи управления недвижимостью с помощью компьютера, являются специальные программные комплексы – *геоинформационные системы* (ГИС).

ГИС имеют следующие характеристики:

- в состав ГИС входит база данных (и не одна), причем полная технология обработки информации в ГИС значительно шире, чем просто работа с базой данных;
- ГИС рассчитана не просто на обработку данных, а на проведение экспертных оценок во многих ситуациях. Другими словами, ГИС должна включать в свой состав экспертную систему (одну или несколько), а этого только на уровне базы данных достигнуть невозможно, так как экспертная система является более общей по отношению к базе данных;
- данные, которые обрабатывает и хранит ГИС, имеют не только пространственную, но и временную характеристику, что важно в первую очередь для географических данных. Процент чисто географических данных в таких системах незначителен, технологии обработки данных имеют мало общего с традиционной обработкой географических данных и, наконец, географические данные служат лишь базой для решения большого числа прикладных задач, цели

которых далеки от географии. Поэтому современные геоинформационные системы не являются чисто географическими.

В рамках ГИС создаются электронные кадастры – аналоги Государственного земельного кадастра учета, оценки земель и регистрации прав на землю, регулирования земельных отношений и сведений о правовом, хозяйственно-экономическом, экологическом и природном состоянии городских земель и недвижимости (рис. 6.2). Как видно из приведенной схемы, электронный земельный кадастр обеспечивает решение следующих задач:

- в сфере управления инвестиционными проектами – планирование земельного участка под конкретный инвестиционный проект и одновременная оценка вариантов проекта с помощью разработанного экономико-математического инструментария;
- в области управления земельными ресурсами – предоставление полной и достоверной информации для планирования и управления земельными ресурсами города, межведомственное взаимодействие при формировании объектов недвижимости, контроль состояния и использования городских земель и земельных участков;
- в области правовых отношений – регистрация прав на землю, сделок с землей и, как следствие, помощь в защите интересов и инвесторов-землепользователей, и города в целом;
- в сфере экономики – поступление земельных платежей в городской бюджет, проведение кадастровой (экономической) оценки городских земель, информационная и правовая поддержка рыночного оборота земли, а в перспективе – создание фондового рынка земельных ценных бумаг;
- в сфере общеинформационных услуг – предоставление правительству города, его структурам, судам, банкам, другим юридическим и физическим лицам достоверной кадастровой информации, информационная поддержка других городских реестров и кадастров.

Геоинформационная система городского хозяйства относится к низшему (муниципальному) уровню относительно используемых масштабов территориальных данных (1 : 10 000 и более крупных). Вместе с тем это самые популярные ГИС, поскольку для их реализации требуются минимальные информационные и технические ресурсы.

Все координатные данные в ГИС должны входить в *единую систему координат*. Для позиционирования объектов в этой координатной системе должны быть определены *идентификаторы местоположения*, которые задают характеристики карты по всей юрисдикции, например коды объектов, акты переписи населения, номера домов и т.д.

Рис. 6.2. Функциональное содержание электронного кадастра

Тематические карты городского хозяйства представляют собой составную модель. В свою очередь, они подразделяются на *слои* (рис. 6.3). Таким образом, интегрированную графическую основу городской ГИС образует совокупность тематических карт-слоев и

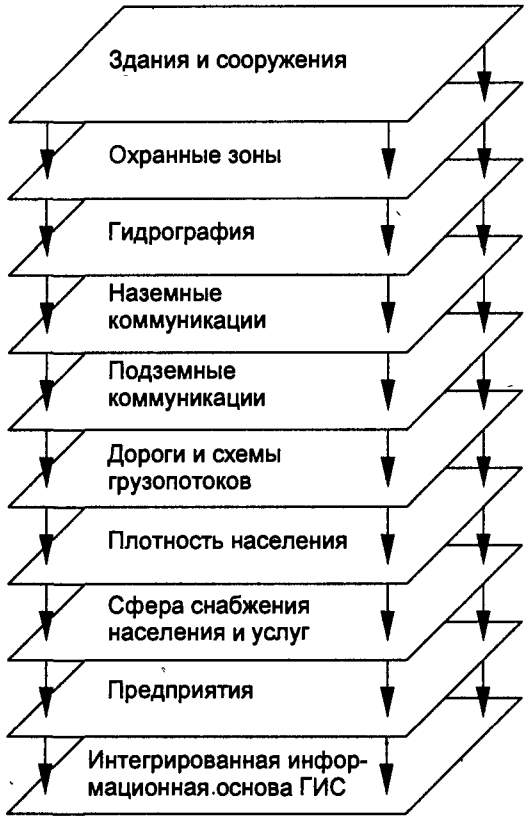


Рис. 6.3. Пример наложения слоев интегрированной ГИС

связанных с ними атрибутивных данных в табличной и текстовой форме.

Моделирующие системы типа Pilgrim и имитационные модели позволяют работать с различными базами данных, используя стандартный ODBC-интерфейс и другие средства, в том числе и с ГИС, позволяющими обращения из других программ (например, с геоинформационной системой ArcView). Кроме того, система Pilgrim имеет возможности создавать геопространства без применения ГИС. Обсуждение этих возможностей следует ниже. Рассмотрение работы с различными ГИС выходит за рамки данной книги.

6.2
МАТЕМАТИЧЕСКИЕ ОСНОВЫ КАРТОГРАФИЧЕСКИХ ИЗМЕРЕНИЙ

Известно, что Земля шарообразна, но не обладает формой идеального шара. Фигура ее неправильна и, как всякое вращающееся тело, она немного сплюснута у полюсов. Кроме того, из-за неравномерного распределения массы земного вещества и глобальных тектонических деформаций Земля имеет обширные, хотя и довольно пологие, выпуклости и вогнутости. Сложную фигуру нашей планеты называют *геоидом*. Точно определить его форму практически невозможно, но современные высокоточные измерения со спутников позволяют иметь о нем достаточно хорошее представление и даже описать уравнением.

Наилучшее геометрическое приближение к реальной фигуре Земли – *эллипсоид вращения* – геометрическое тело, которое образуется при вращении эллипса вокруг его малой оси. Сжатие эллипсоида моделирует сжатие планеты у полюсов.

Исторически сложилось так, что в разные времена и в разных странах были приняты и законодательно закреплены различные эллипсоиды, и их параметры не совпадают между собой. В России принят *эллипсоид Ф.Н. Красовского*, вычисленный в 1940 г. Его параметры таковы:

- большая полуось, $z_1 = 6\,378\,245{,}000$ м;
- малая полуось, $z_2 = 6\,356\,863{,}019$ м;
- сжатие, $\alpha = (z_1 - z_2) / z_1 - 1 : 298{,}3$.

В США и Канаде до недавнего времени использовали эллипсоид Кларка, рассчитанный еще в 1866 г., его большая полуось на 39 м короче, чем в российском эллипсоиде, а сжатие определено как 1 : 295,0. Во многих странах Западной Европы и некоторых государствах Азии принят эллипсоид Хайфорда, вычисленный в 1909 г., а в бывших английских колониях (в Индии и странах Южной Азии) используют рассчитанный англичанами в 1830 г. эллипсоид Эвереста.

В 1984 г. на основе спутниковых измерений определен международный эллипсоид WGS-84 (World Geodetic System). Всего в мире насчитывается около полутора десятков эллипсоидов. Их примеры приведены в табл. 6.1.

Таблица 6.1

Значения элементов земных эллипсоидов

Название эллипсоида	Полуоси		Сжатие, α	Страна использования эллипсоида
	Большая z_1 (м)	Малая z_2 (м)		
Красовского (1940 г.)	6 378 245	6 356 863	1 : 298,3	Россия, страны СНГ, страны Восточной Европы, Антарктида
Бесселя (1841 г.)	6 377 397,2	6 356 079	1 : 299,15	Европа и Азия
Хайфорда (1909 г.)	6 378 388	6 356 912	1 : 297,0	Европа, Азия, Южная Америка, Антарктида
Кларка I (1866 г.)	6 378 206	6 356 584	1 : 294,98	Северная и Центральная Америка
Кларка II (1880 г.)	6 378 249	6 356 515	1 : 293,46	Африка, Барбадос, Израиль, Иордания, Иран, Ямайка
Эйри (№ 1)	6 377 563,4	6 356 257	1 : 299,32	Великобритания
Эйри (№ 2)	6 377 340,2	6 356 034	1 : 299,32	Ирландия
Эвереста (1830 г.)	6 377 276,3	6 356 075	1 : 300,8	Индия, Пакистан, Непал, Шри-Ланка
Эвереста (1956 г.)	6 377 301,24	6 356 100	1 : 300,80	Индия, Непал
Австралийский (1965 г.)	6 378 160	6 356 775	1 : 298,25	Австралия, Папуа (Новая Гвинея)
GKS (1980 г.)	6 378 137	6 356 752	1 : 298,26	Аляска, Центральная Америка, Мексика, США, Канада
Южно-Американский (1969 г.)	6 378 160	6 356 775	1 : 298,25	Южная Америка
WGS-84	6 378 137	6 356 752	1 : 298,257	Международный

Карты, составленные на основе разных эллипсоидов, получаются в несколько различающихся координатных системах, что создает неудобства. Однако для принятия единого международного эллипсоида требуется перевычислить координаты и пересоставить все карты, а это долгое, сложное и, главное, дорогостоящее занятие. Несовпадения бывают заметны главным образом на крупномасштабных картах при определении по ним точных координат объектов. Но на широко используемых географических средне- и мелкомасштабных картах такие различия не очень чувствительны. Более того, иногда вместо эллипсоида берут шар и тогда в качестве среднего радиуса Земли принимают величину $R = 6\,367{,}6$ км.

Погрешности разовых измерений при замене эллипсоида шаром или при использовании прямоугольных координат с карт невелики. Однако если прокладывать маршрут транспортного средства по местности с многочисленными поворотами (галсами), пусть даже небольшими в угловом измерении, то погрешность накапливается. Например, если при прокладке сложного маршрута вертолета в рамках квадратного района 100×100 км пренебречь тем, что Земля – эллипсоид, и использовать прямоугольные координаты с карты, то погрешность может оказаться такой, что этот вертолет выйдет к конечной точке, расположенной в стороне от истинной точки финиша на несколько километров (до 10 и более) – за счет многочисленных галсов во время полета.

Чтобы добиться наименьших искажений, применяют также способ двойного проектирования: сначала эллипсоид проектируют на шар, а затем шар – на плоскость. При равновеликом отображении, когда площадь поверхности эллипсоида Красовского должна быть равна площади поверхности шара, радиус его оказывается равным $R = 6\,371\,116$ м.

Для упрощения проектирования применяют и иные способы отображения эллипсоида на шар.

Масштаб карты – степень уменьшения объектов на карте относительно их размеров на земной поверхности (точнее, на поверхности эллипсоида). Строго говоря, масштаб постоянен только на планах, охватывающих небольшие участки территории. На географических картах он меняется от места к месту и даже в одной точке – по разным направлениям, что связано с переходом от сферической поверхности планеты к плоскому изображению. Поэтому различают главный и частный масштабы карт. *Главный масштаб* показывает, во сколько раз линейные размеры на карте уменьшены по отношению к эллипсоиду или шару. Этот масштаб подписывают на карте, но необходимо иметь в виду, что он справедлив лишь для отдельных линий и точек, где искажения отсутствуют. *Частный масштаб* отражает соотношения размеров объектов на карте и эллипсоиде (шаре) в данной точке. Он может быть больше или меньше главного. Частный масштаб длин μ показывает отношение длины бесконечно малого отрезка на карте ds^* к длине бесконечно малого отрезка ds на поверхности эллипсоида или шара, а частный масштаб площадей ρ передает аналогичные соотношения бесконечно малых площадей на карте dp^* и эллипсоиде или шаре dp:

$$\mu = \frac{ds^*}{ds} \quad \text{и} \quad \rho = \frac{dp^*}{dp}.$$

В общем случае, чем мельче масштаб картографического изображения и чем обширнее территория, тем сильнее сказываются различия между главным и частным масштабами.

Масштаб указывается на картах в различных вариантах. *Численный масштаб* представляет собой дробь с единицей в числителе. Он показывает, во сколько раз линейные размеры на карте меньше соответствующих длин на местности (например, 1 : 1 000 000). *Линейный (графический) масштаб* дается на полях карты в виде линейки, разделенной на равные части (обычно сантиметры), с подписями, означающими соответствующие расстояния на местности. Он удобен для измерений по карте. *Именованный масштаб* указывает в виде подписи, какое расстояние на местности соответствует одному сантиметру на карте (например, в 1 см 1 км).

Картографическая проекция – математически определенное отображение поверхности эллипсоида или шара (глобуса) на плоскость карты. Проекция устанавливает однозначное соответствие между геодезическими координатами точек. Каждая точка имеет две координаты (рис. 6.4):

- широту φ, которая на иностранных картах иногда обозначается как latitude (или lat);
- долготу λ, иногда обозначаемую как longitude (или lon).

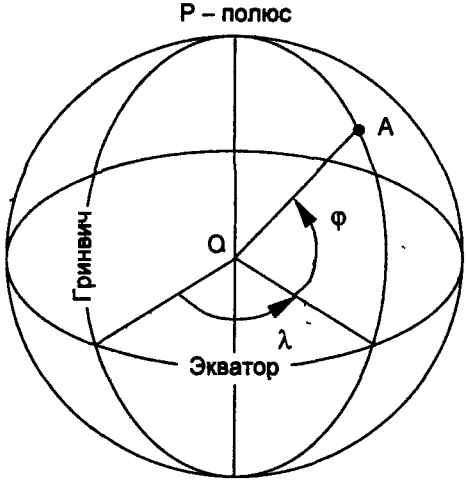


Рис. 6.4. Географические координаты точки A:
φ – северная широта; λ – восточная долгота

На карте этой же точке соответствуют прямоугольные координаты X и Y. Уравнения проекций в общей форме выглядят так:

$$X=f_1(\varphi,\lambda);\ Y=f_2(\varphi,\lambda).$$

Конкретные реализации функций f_1 и f_2 часто выражены довольно сложными математическими зависимостями, их число бесконечно, а, следовательно, разнообразие картографических проекций практически неограничено.

Теория картографических проекций составляет главное содержание математической картографии. В этой области разрабатывают методы изыскания новых проекций для разных территорий и разных задач, создают приемы и алгоритмы анализа проекций, оценки распределения и величин искажений. Особый круг задач связан с учетом этих искажений при измерениях по картам, с переходом из одной проекции в другую и т.п. Компьютерные технологии позволяют рассчитывать проекции с заданными свойствами. Существуют различные виды проекций шара или эллипсоида на плоскую карту, которые различаются по типу развертки и вносимых при этом искажений.

Цилиндрические проекции – проектирование шара (эллипсоида) ведется на поверхность касательного или секущего цилиндра, а затем его боковая поверхность разворачивается в плоскость. Если ось цилиндра совпадает с осью вращения Земли, а его поверхность касается шара по экватору (или сечет его по параллелям), то проекция называется *нормальной (прямой) цилиндрической.* Тогда меридианы нормальной сетки представляются в виде равноотстоящих параллельных прямых, а параллели – в виде прямых, перпендикулярных к ним. На таких проекциях меньше всего искажений в тропических и приэкваториальных областях.

Если ось цилиндра расположена в плоскости экватора, то это *поперечная цилиндрическая* проекция. Цилиндр касается шара по меридиану, искажения вдоль него отсутствуют, и следовательно, в такой проекции наиболее выгодно изображать территории, вытянутые с севера на юг. В тех случаях когда ось вспомогательного цилиндра расположена под углом к плоскости экватора, проекция называется *косой цилиндрической.* Она удобна для вытянутых территорий, ориентированных на северо-запад или северо-восток.

Конические проекции – поверхность шара (эллипсоида) проектируется на поверхность касательного или секущего конуса, после чего она как бы разрешается по образующей и разворачи-

вается в плоскость. Как и в предыдущем случае, различают *нормальную (прямую) коническую* проекцию, когда ось конуса совпадает с осью вращения Земли, *поперечную коническую* – ось конуса лежит в плоскости экватора и *косую коническую* — ось конуса наклонена к плоскости экватора.

В нормальной конической проекции меридианы представляют собой прямые, расходящиеся из точки полюса, а параллели – дуги концентрических окружностей. Воображаемый конус касается земного шара или сечет его в районе средних широт, поэтому в такой проекции удобнее всего картографировать территории России, Канады, США, вытянутые с запада на восток в средних широтах.

Азимутальные проекции – поверхность земного шара (эллипсоида) переносится на касательную или секущую плоскость. Если плоскость перпендикулярна к оси вращения Земли, то получается *нормальная (полярная) азимутальная* проекция. Параллели в ней являются концентрическими окружностями, а меридианы – радиусами этих окружностей. В этой проекции всегда картографируют полярные области Земли и других планет.

6.3 ОТОБРАЖЕНИЕ ГЕОИНФОРМАЦИИ В ФУНКЦИОНАЛЬНОМ ОКНЕ ИМИТАЦИОННОЙ МОДЕЛИ

При оценке крупномасштабных геопроцессов имитационные модели должны учитывать, что поверхность Земли – это эллипсоид. Географические координаты точек на поверхности Земли – это широта и долгота, которые измеряются в градусах °, минутах ' и секундах ". Причем 1 минута в 60 раз меньше градуса (1° = 60'), а 1 секунда в 60 раз меньше минуты (1' = 60").

Расстояние между двумя пунктами на поверхности Земли определяется по формулам сферической тригонометрии. Рассмотрим сферический треугольник PAB на сфере с центром в O. Вершина P – полюс Земли. Введем следующие обозначения:

φ_1 – географическая широта точки A (угол);

φ_2 – географическая широта точки B;

λ_1 – географическая долгота точки A;

λ_1 – географическая долгота точки B.

Необходимо найти l_{AB} – длину дуги AB.

Угол AOB обозначим γ. Одна из основных формул сферической тригонометрии Красовского определяет косинус этого угла:

$$\cos\gamma = \sin\varphi_1 \sin\varphi_2 + \cos\varphi_1 \cos\varphi_2 \cos(\alpha_2 - \alpha_1). \qquad (6.1)$$

Радиус Земли R_x для широты φ_x вычисляют по эллипсоиду Красовского:

$$R_x = \frac{z_1 z_2}{\sqrt{z_1^2 \cos^2\varphi_x + z_2^2 \sin^2\varphi_x}}, \qquad (6.2)$$

где $z_1 = 6\,378\,245{,}000$ м – большая полуось эллипсоида Земли;
$z_2 = 6\,356\,863{,}019$ м – малая полуось.

Функция geoway (lat *X*, lon *X*, lat *Y*, lon *Y*), входящая в состав комплекса Pilgrim, служит для определения расстояния между точками *X* и *Y* по их географическим координатам. Она использует формулы Красовского (6.1) и (6.2).

Взаимное расположение точек на поверхности Земли в средних широтах, характерных для России, стран Европы и США, рекомендуется изображать на плоскости (в частности, на экране монитора) в виде *нормальной конической проекции*. При отображении земного эллипсоида используется промежуточная поверхность земного шара с радиусом *R*.

Пример 6.1. Допустим, что необходимо подготовить экран монитора для отображения точек, находящихся на территории России (включая Калининградскую область), Белоруссии, Украины, Литвы и Латвии. Этот участок ограничен координатами:

- от 48°04' до 59°57' Северной широты (обозначим φ_1 и φ_2);
- от 19°55' до 38°10' Восточной долготы (обозначим λ_1 и λ_2).

Перейдем к радианной системе измерения углов. Пересчет географических координат, измеряемых в градусах, минутах и секундах, в радианы не представляется сложной задачей. Отображаемую поверхность необходимо так расположить на плоскости, чтобы середина региона соответствовала центру той части экрана, на которой строится поверхность. Меридианы изображаются прямыми, исходящими из одной точки – проекции Северного полюса, которая находится за пределами экрана. Параллели – это дуги окружностей с радиусами, исходящими из полюса. На рис. 6.5 показано расположение региона, максимально вписанное в отведенные прямоугольные

границы экрана с размерами x_{max} – по горизонтали и y_{max} – по вертикали. Все точки, имеющие координаты φ и λ, должны попасть на экран с прямоугольными координатами x и y. Обычно при планировании места для отображения на экране выдерживают соотношение

$$\frac{x_{max}}{y_{max}} = \frac{4}{3},$$

но могут быть и другие.

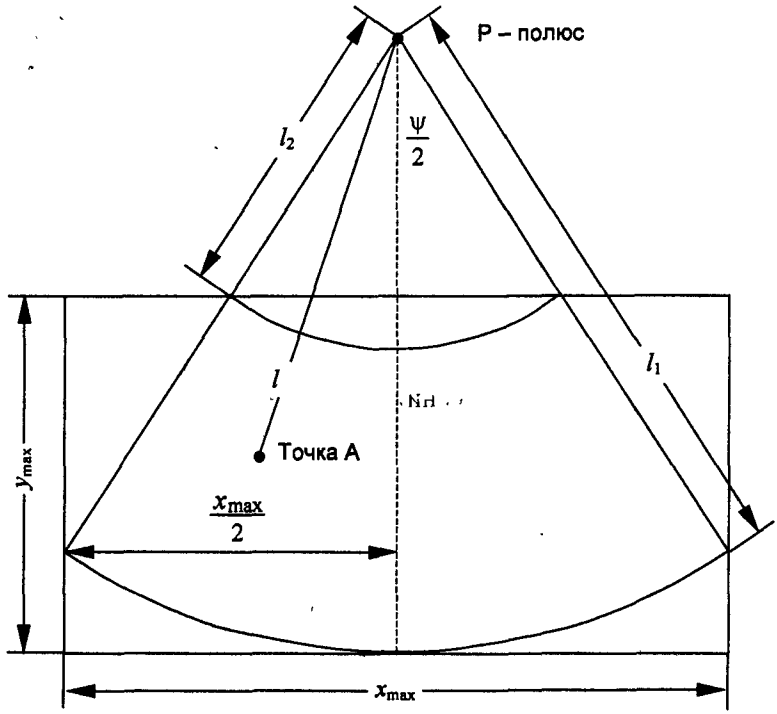


Рис. 6.5. Нормальная коническая проекция на экране монитора

Рассмотрим угол конической развертки на экране $\psi = |\lambda_2 - \lambda_1|$. Нетрудно заметить, что радиус окружности, касающейся нижней границы прямоугольника и соответствующей южной граничной параллели региона, определяется как

$$l_1 = \frac{0{,}5\,x_{max}}{\sin(0{,}5\psi)}.$$

Радиус окружности, соответствующей северной граничной параллели региона, определяется по формуле

$$l_2 = \frac{l_1 - y_{max}}{\cos(0{,}5\psi)}.$$

Работа с прямоугольными координатами на экране монитора должна следовать правилам компьютерной графики:

- левый верхний угол прямоугольника имеет координаты $x = 0$ и $y = 0$;
- единица измерения – 1 пиксель.

Поэтому, если имеется точка А с координатами φ (широта) и λ (долгота), причем $\varphi_1 \le \varphi \le \varphi_2$ и $\lambda_1 \le \lambda \le \lambda_2$, а угол конической развертки на экране монитора изменяется на отрезке от $-0{,}5\Psi$ до $+0{,}5\Psi$, то справедливы следующие соотношения для координат этой точки x и y.

Радиус окружности l, соответствующей параллели на широте φ, определяется из выражения

$$l = l_1 - (l_1 - l_2)\frac{\varphi - \varphi_1}{\varphi_2 - \varphi_1}.$$

Координата x на экране монитора вычисляется по формуле

$$x = l\sin(-0{,}5\psi + (\lambda - \lambda_1)) + 0{,}5x_{max}.$$

Координата y на экране монитора определяется как

$$y = y_{max} - \{l_1 - l\cos(-0{,}5\psi + (\lambda - \lambda_1))\}.$$

Для дополнительных расчетов в рассматриваемом регионе необходимо знать радиус Земли. Он усредняется по диапазону широт $\varphi_1 - \varphi_2$:

$$R = \frac{R_1 + R_2}{2},$$

где R_1 – радиус Земли на широте φ_1;
R_2 – радиус на широте φ_2.

При изображении взаимного расположения точек на экране монитора возникает необходимость в *корректировке масштабов*, т.е. требуется сделать масштабы равновеликими. Дело в том, что, если последовать расчетным формулам, приведенным выше, масштаб по

параллели, проходящей через центр региона, может существенно отличаться от масштаба в направлении меридиана. Поэтому необходимо определить масштабы:

- m_E – по средней параллели в восточном направлении;
- m_N – по меридиану в северном направлении.

Сначала найдем l_M – расстояние между восточной и западной точками, имеющими координаты

$$\left(\frac{\varphi_1+\varphi_2}{2},\lambda_1\right) \text{ и } \left(\frac{\varphi_1+\varphi_2}{2},\lambda_1\right),$$

например, с помощью функции Pilgrim geoway.

Соответствующий масштаб определяется из отношения

$$m_E = \frac{R|\lambda_2-\lambda_1|}{l_M}, \text{ км / пиксель.}$$

Далее определим расстояние по меридиану между северной и южной границами региона: $l_N = l_1 - l_2$. Масштаб в северном направлении равен отношению

$$m_N = \frac{R|\varphi_2-\varphi_1|}{l_N}, \text{ км / пиксель.}$$

Введем в рассмотрение отношение масштабов

$$k_m = \frac{m_E}{m_N}.$$

Корректировка линейных масштабов осуществляется по следующим правилам:

1) если $k_m > 1$, то угол $\psi = |\lambda_1 - \lambda_2|$ нужно увеличить, умножив на k_m;

2) если $k_m < 1$, то угол $\theta = |\varphi_1 - \varphi_2|$ следует увеличить, умножив на k_m;

3) если $k_m = 1$, то масштабы установлены правильно и корректировка не требуется (но такой случай нереален).

После корректировки границы региона несколько расширяются, поэтому нужно заново рассчитать l_1 и l_2, так как от них зависят x и y – прямоугольные координаты точек на экране.

Вернемся к примеру 6.1. После программирования всех манипуляций получается небольшая программа на Visual C++. Функцио-

нальное окно на экране (рис. 6.6) действительно выводит коническую проекцию региона. Некоторая ступенчатость параллелей и меридианов является следствием специфики конкретной функциональной задачи, при решении которой потребовалась кусочно-линейная аппроксимация элементарных площадок (прямые линии и дуги, если их провести простыми средствами Visual C++, таких изъянов не имеют). В данном примере при программировании окна выбран нестандартный формат области экрана:

$$\frac{x_{\max}}{y_{\max}} \neq \frac{4}{3}.$$

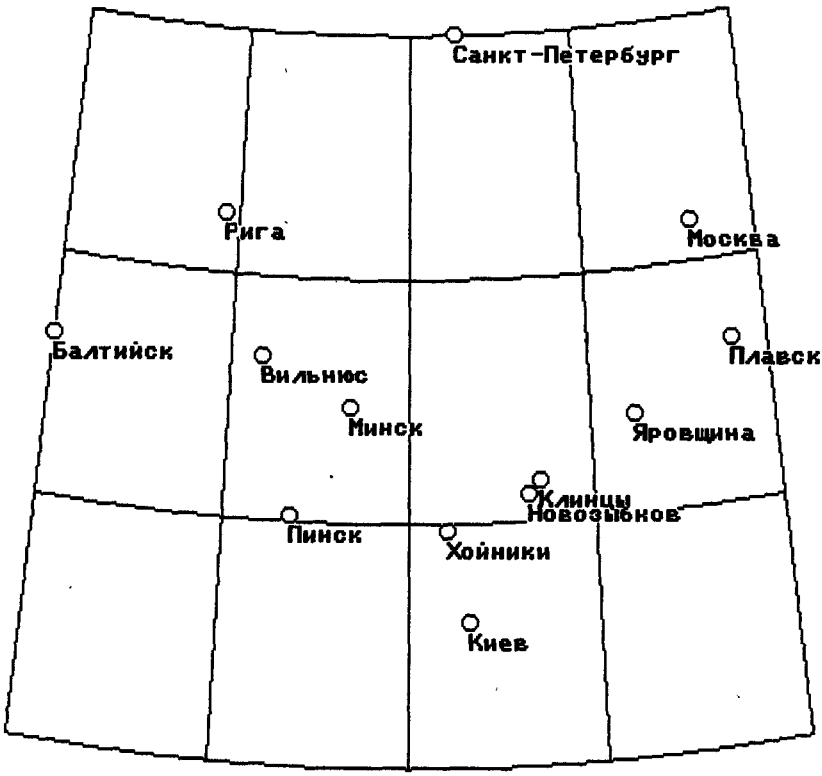


Рис. 6.6. Развертка выбранного региона на плоскости (учебный пример)

6.4 ПРИЕМЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И АНАЛИЗА ГЕОПОВЕРХНОСТЕЙ

Формализованное картографическое изображение хорошо приспособлено для математическою анализа. Как упоминалось выше, каждой точке карты с координатами x и y поставлено в соответствие лишь одно значение картографируемого параметра z, что позволяет представить изображение рассматриваемого явления как функцию $z = F(x, y)$. В других случаях картографическое изображение удобно представить как поле случайных величин и использовать для его анализа вероятностно-статистические методы.

В принципе почти все разделы математики применимы для обработки и анализа картографического изображения. Проблема лишь в том, чтобы точно подобрать математическую модель и, главное, дать надежное содержательное истолкование результатам моделирования. Достаточно прочно в картографический анализ вошли некоторые разделы численного анализа, многомерной статистики, теории вероятностей и теории информации.

Аппроксимации. Под *аппроксимациями* в математике понимают замену (приближение) сложных или неизвестных функций другими, более простыми функциями, свойства которых известны. Любую сложную поверхность (поле), изображенную на изолинейной карте, можно аппроксимировать, т.е. приблизительно представить в виде

$$z = f(x, y) + \varepsilon,$$

где $f(x, y)$ – некая аппроксимирующая функция;
ε – остаток, не поддающийся аппроксимации.

Функцию $f(x, y)$ можно далее разложить в ряд, представив уравнение поверхности в виде

$$z = f_1(x, y) + f_2(x, y) + \ldots + f_n(x, y) + \varepsilon,$$

где $f_i(x, y)$ – компоненты разложения, которые предстоит определить.

В общем случае для этого с аппроксимируемой карты снимают ряд значений, после чего составляют систему уравнений, решаемых совместно по способу наименьших квадратов так, чтобы

$$\sum_{i=1}^{n} \varepsilon_i = \sum_{i=1}^{n} [F(x_i, y_i) - f(x_i, y_i)]^2 = \min.$$

Существуют разные способы аппроксимации. Это обычные алгебраические многочлены, ортогональные многочлены Чебышева и Лежандра, которые определенным образом упрощают вычисления, спалйн-функции и др. Не останавливаясь на особенностях математического аппарата, отметим, что во всех случаях задача сводится к тому, чтобы аппроксимирующее уравнение наилучшим образом описывало исходную поверхность, а сумма квадратов отклонений была бы минимальна.

Аппроксимация 1-го порядка (линейное уравнение) дает плоскость, отражающую только общий уклон поверхности, это очень грубое, слишком общее приближение. Поверхность 2-го порядка уже больше похожа на исходную модель, а аппроксимация 3-го порядка (кубическое уравнение) дает достаточно хорошее приближение к исходной поверхности.

Тригонометрические функции позволяют описывать сложные, сильно расчлененные поверхности. Сферические функции применяют, если при вычислениях нельзя пренебречь кривизной земной поверхности. Аппроксимация с помощью двойных рядов Фурье позволяет вводить постепенное усложнение поверхности за счет добавления двухмерных синусоид с разными фазами и амплитудами. Компьютерное моделирование обеспечивает выполнение подобных аппроксимаций для поверхностей любой сложности с помощью уравнений высокого порядка, содержащих порой несколько десятков членов разложения.

В исследовательской практике аппроксимации используют для аналитического описания поверхностей, изображенных на картах, и выполнения с ними различных действий: суммирования, вычитания, интегрирования и дифференцирования; для подсчета объемов тел, ограниченных этими поверхностями; для решения множества других задач. Одно из направлений использования аппроксимаций – это разложение поверхностей на составляющие, что позволяет выделять и анализировать нормальные и аномальные факторы развития и пространственного размещения явлений.

Статистический анализ картографического изображения преследует главным образом три цели:

- изучение характеристик и функций распределения явления;
- определение формы и тесноты связей между явлениями;
- оценку степени влияния отдельных факторов на изучаемое явление и выделение ведущих факторов.

В основу всякого статистического исследования кладется *выборка*, т.е. некоторое подмножество однородных величин a_i, снятых с карты по регулярной сетке точек (систематическая выборка), в случайно расположенных точках (случайная выборка), на ключевых участках (ключевая выборка) или по районам (районированная выборка).

Выборочные данные группируют по интервалам, составляют гистограммы распределений и затем вычисляют различные *статистики* – количественные показатели, характеризующие пространственное распределение изучаемого явления. Наиболее употребительные показатели – среднеарифметическое, среднее взвешенное арифметическое, среднеквадратичное, дисперсия, вариация и др. Кроме того, с помощью специальных показателей (критериев согласия) можно оценить соответствие данного конкретного распределения тому или иному теоретическому закону распределения. Например, установить, согласуется ли эмпирическое распределение высот рельефа с кривой нормального распределения или подчиняется какой-либо иной функции.

Другая типичная исследовательская задача – оценка взаимосвязи между явлениями – решается с помощью хорошо разработанного в математической статистике аппарата *теории корреляции*. Для этого необходимо иметь выборки по сравниваемым явлениям, показанным на картах разной тематики (например, А и В). Значения a_i и b_i берут в одних и тех же i-х точках, т.е. строго скоординированно, и затем строят график поля корреляции.

Если поле корреляции может быть аппроксимировано прямой, которая называется линией регрессии, то приступают к вычислению *коэффициента парной корреляции* r. Его числовые значения заключены в интервале [–1, 1]. Если r равно 1 или –1, то существует функциональная прямая или обратная связь. Когда r близок к нулю, связь между явлениями отсутствует, а при $r \geq |0{,}7|$ связь считается существенной. Коэффициент корреляции рассчитывают по формуле

$$r = \frac{\sum_{i=1}^{n}(a_i - M_a)(b_i - M_b)}{n\sigma_a\sigma_b},$$

где a_i и b_i – выборочные данные, полученные по картам А и В;
n – объем выборки (число пар данных);
M_a и M_b – значения средних соответственно для a_i и b_i;

$$M_a = \frac{\sum_{i=1}^{n} a_i}{n} \quad \text{и} \quad M_b = \frac{\sum_{i=1}^{n} b_i}{n};$$

σ_a и σ_b – значения среднеквадратичных соответственно для a_i и b_i:

$$\sigma_a = \sqrt{\frac{\sum_{i=1}^{n} a_i^2}{n} - M_a^2} \quad \text{и} \quad \sigma_b = \sqrt{\frac{\sum_{i=1}^{n} b_i^2}{n} - M_b^2}.$$

Оценку точности вычисления коэффициента корреляции r получают по формуле

$$m_r = \frac{1 - r^2}{\sqrt{n}},$$

из которой видно, что при прочих равных условиях погрешность вычисления коэффициента корреляции всегда уменьшается с увеличением объема выборки. Отсюда следует, что определение объема выборки – важная проблема при расчете коэффициента корреляции, да и вообще при вычислении всех статистических показателей. Достаточно представительной обычно считается выборка объемом 30 – 50 значений.

В практике исследования взаимосвязей часто необходимо получить предварительно приблизительную оценку коэффициента корреляции. В простых случаях это можно сделать, используя представление о статистических поверхностях. Доказано, что коэффициент корреляции примерно равен косинусу угла α между направлениями наибольших скатов (градиентов) двух сравниваемых статистических поверхностей: $r = \cos \alpha$.

Значения r заключены в интервале [0°, 180°]. Если $\alpha = 0°$, что свидетельствует о полном совпадении направлений скатов поверхностей, то $r = \cos 0° = 1$, т.е. между явлениями существует прямая связь. При $\alpha = 180°$ скаты поверхностей направлены в противоположные стороны, и $r = \cos 180° = -1$, следовательно, связь сильная, но отрицательная, а при $\alpha = 90°$ связь между явлениями отсутствует, поскольку $r = \cos 90° = 0$.

На рис. 6.7 представлены две статистические поверхности и показаны направления их наибольших скатов. Угол между ними оказался равен 36°, тогда $r = \cos 36° = 0{,}81$. Такие приближенные вычисления особенно удобны при сравнении изолинейных карт.

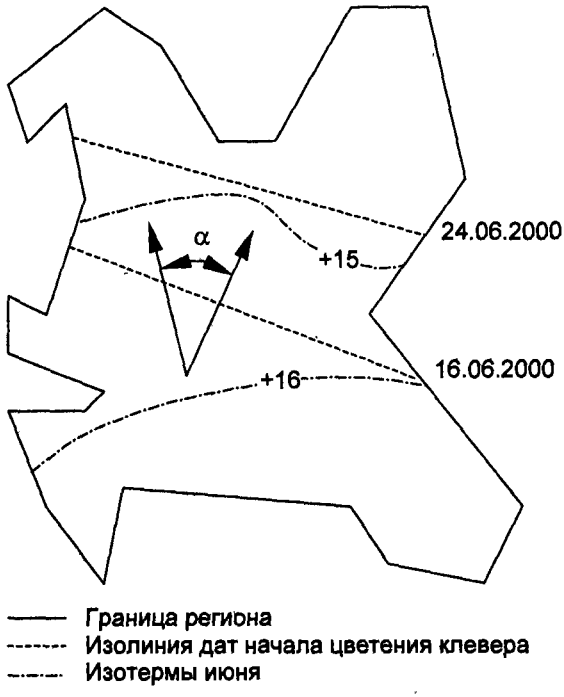


Рис. 6.7. Приближенное определение коэффициента корреляции по карте региона

Для оценки взаимосвязи явлений в случаях, когда трудно или невозможно получить большие выборки, используют показатель *ранговый коэффициент корреляции* γ, который вычисляют по формуле

$$\gamma = 1 - \frac{6\sum_{i=1}^{n}\left(p_{a_i} - p_{b_i}\right)^2}{n^3 - n},$$

где p_{a_i} и p_{b_i} – ранги значений, полученных соответственно по картам A и B, т.е. их порядковые номера в возрастающей последовательности (1, 2, 3 и т.д.);

n – объем выборки.

По смыслу коэффициент γ аналогичен коэффициенту парной корреляции r, он изменяется в интервале от –1 до +1. При этом не требуется больших объемов выборки, расчеты можно выполнять даже при $n = 3$. К тому же не нужны точные количественные значения a_i и b_i, достаточно знать их ранги. Все это удобно для работы с картограммами, где используются интервальные шкалы, а объем выборки ограничен числом административных районов.

Аппарат теории корреляции достаточно разнообразен. В нем есть показатели, удобные для анализа взаимосвязей по картам ареалов (где явления характеризуются только двумя состояниями: «есть» и «нет»), по картам качественного фона (где каждое явление имеет много состояний, но не охарактеризовано количественно). Существуют коэффициенты для расчета криволинейных зависимостей и связей между тремя явлениями (коэффициенты множественной корреляции) и т.п.

Расчет корреляций дает основу для более сложных видов анализа: регрессионного, дисперсионного, факторного и др. Часто при исследованиях ставится задача выделить основные факторы, определяющие развитие и размещение того или иного явления. Эту задачу решает многомерный *факторный анализ*. Он позволяет свести к минимуму (к трем-четырем главным факторам) большие совокупности исходных показателей, характеризующих сложное явление. Уравнение факторного анализа имеет вид

$$a_k = \sum_{i=1}^{n} l_{k_i} f_k + e_k,$$

где a_k – исходные показатели;

f_k – выделенные главные факторы, дающие синтетическую оценку изучаемого явления;

l_{k_i} – вес каждого фактора в синтетической оценке (факторная нагрузка);

e_k – остаток, характеризующий неучтенные отклонения.

Приемы теории информации. Эти приемы используют для оценки степени однородности и взаимного соответствия явлений, изучаемых по картам. Речь идет об основной функции теории информации – *энтропии*. В теории информации энтропия характеризует степень неопределенности каких-либо исходных данных или передаваемых сообщений, а в картографическом анализе эта функция оказалась довольно удобной для оценки степени однородности / неоднородности (разнообразия) картографического изображения.

Энтропией $E(A)$ некоторой системы A называется сумма произведений вероятностей ω_i различных состояний этой системы на логарифмы вероятностей, взятая с обратным знаком:

$$E(A) = E(\omega_1, \omega_2, ..., \omega_n) = -\sum_{i=1}^{n} \omega_i \log_2 \omega_i .$$

В теории информации принято брать логарифмы вероятностей по основанию 2, что связано с двоичной системой счисления. Смысл функции не изменится, если пользоваться десятичными или натуральными логарифмами. Функция $E(A)$ остается неотрицательной. Она обращается в нуль, когда на карте изображен только один контур или выдел (т.е. изображение совершенно однородно), и монотонно возрастает с увеличением числа контуров n. Это свойство функции энтропии позволяет косвенно характеризовать неоднородность картографического изображения, понимаемую как разнообразие контуров и неравномерность их распространения по площади (контурам с разными площадями соответствуют различные значения ω_i).

Кроме того, информационные функции используют для оценки степени взаимного совпадения контуров на разных картах. В этом случае они выполняют роль своеобразных показателей взаимосвязи явлений наподобие коэффициентов корреляции.

6.5 РЕШЕНИЕ ЗАДАЧИ КОММИВОЯЖЕРА С ПРИВЯЗКОЙ К ГЕОГРАФИЧЕСКИМ КООРДИНАТАМ

Существует классическая задача исследования операций, которая в различных модификациях решается для оптимизации процессов в логистике, при сплошном обследовании населенных пунктов в каком-либо регионе в связи с экономическими, экологическими или медицинскими проблемами, а также в зонах чрезвычайных ситуаций. Она получила название «задачи коммивояжера». Рассмотрим три постановки этой задачи.

Простейшая постановка задачи коммивояжера. Допустим, имеется множество M населенных пунктов. Необходимо составить такой маршрут посещения этих пунктов, который удовлетворял бы двум требованиям:

• каждый пункт необходимо посетить только один раз;

• длина маршрута должна быть минимальной.

Данная постановка предполагает два взаимосвязанных, но практически невыполнимых предположения:

• коммивояжер (или транспортное средство) всегда может передвигаться в данном регионе по прямой линии;

• вся поверхность региона – это идеально гладкая и твердая плоскость, пригодная для движения транспортных средств.

При такой постановке задачи можно использовать следующие методы составления маршрута: линейное программирование, метод «ветвей и границ», динамическое программирование и алгоритм «ближайшего непосещенного города». Первые три метода при большом числе населенных пунктов требуют применения ЭВМ большой мощности и создания довольно сложных расчетных программ, а четвертый метод дает решения, которые на 20–50% хуже оптимального (в зависимости от густоты и неравномерности расстояний между пунктами).

Существуют другие модификации этой задачи, учитывающие реальные возможности движения коммивояжера по региону.

Постановка задачи коммивояжера с привязкой к дорожной сети. В регионе с развитой сетью дорог имеется:

• множество M населенных пунктов;

• множество N перекрестков (развилок) вне населенных пунктов;

• множество K участков дорог.

Участком дороги назовем дорогу от пункта А до пункта Б, причем пункт – это населенный пункт или перекресток.

Необходимо составить такой маршрут посещения населенных пунктов, чтобы длина маршрута, включая суммарные повторные пробеги по участкам дорог, была минимальной.

В данной постановке учитывается то обстоятельство, что из-за конечных возможностей дорожной сети все-таки возможны повторные посещения населенных пунктов и возвраты к перекресткам (развилкам) дорог. Однако не существует метода, который сразу мог бы привести к оптимальному маршруту при такой постановке: любой метод, включая алгоритмы динамического программирования, дает тупиковые псевдооптимальные решения, из которых методом перебора нужно отыскать наилучший. Причем нет гарантии, что найден весь набор решений, среди которых есть и оптимальное.

Поэтому, если есть ЭВМ большой мощности и время для ожидания результата (оно может быть довольно большим), то на компьютере реализуется алгоритм полного «тупого» перебора вариантов. Но «тупой перебор» не имеет универсального алгоритма. Поэтому необходимо написать довольно сложную расчетную программу, учитывающую правила движения по данному региону.

Постановка задачи коммивояжера для выполнения вертолетных работ. В регионе имеется множество M населенных пунктов. Необходимо составить такой алгоритм посещения этих пунктов, который удовлетворял бы следующим требованиям:

- каждый пункт необходимо посетить только один раз;
- длина маршрута должна быть минимальной;
- при определении отрезков маршрута учитывается, что поверхность Земли – эллипсоид;
- на маршруте могут работать один или два вертолета (если два – то они движутся навстречу друг другу).

Задача в данной постановке может решаться теми же методами.

Существует «алгоритм двух вертолетов», который подходит для практического использования как для действий с привязкой к дорожной сети, так и вертолетных работ, а по качеству решений может уступить только «тупому» перебору, так как в этом случае нет гарантии того, что среди набора псевдооптимальных решений имеется расписание движения по оптимальному маршруту.

Прежде чем перейти к описанию этого алгоритма, введем два определения и сформулируем одну теорему (без доказательства). Рассмотрим рис. 6.8. Предположим, что нужно составить полетное расписание для самолета, который вылетает из Санкт-Петербурга в Москву, причем он должен пролететь по кратчайшему маршруту над одиннадцатью промежуточными населенными пунктами. Предварительное расписание было составлено вручную и выдано экипажу самолета (табл. 6.2). Траектория маршрута показана на рисунке.

Определение 1. Под корректировкой понимается такое исправление маршрута следования, которое приводит к его сокращению без исключения каких-либо населенных пунктов из маршрутного расписания. Корректировка выполняется на основании графической траектории движения, изображенной на карте, и опыта руководителя движения.

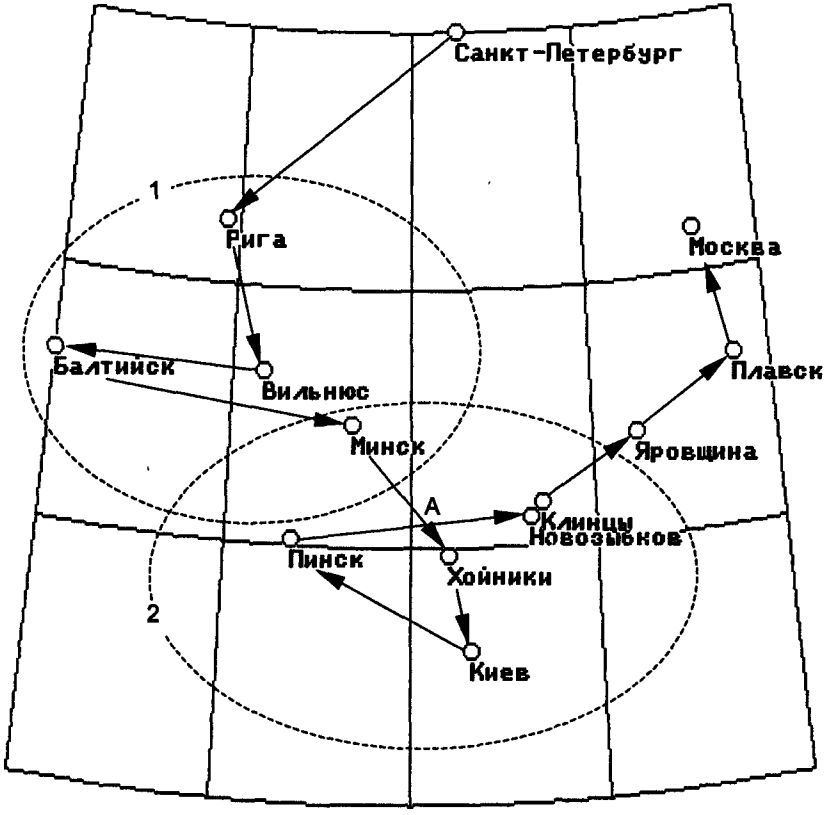


Рис. 6.8. Пример составления оптимального маршрута коммивояжера с двумя корректируемыми участками:
1 – неоптимального участка; 2 – петли

На рис. 6.8 пунктирным контуром 1 выделен неоптимальный участок, где установлен следующий порядок движения (полета):

Рига→Вильнюс→Балтийск→Минск.

Корректировка, проведенная экипажем, заключается в изменении порядка посещения (пролета). Оптимальным будет (это заметно на рисунке):

Рига→ Балтийск→Вильнюс→ Минск.

Таблица 6.2

Полетное расписание (исходный вариант)

Порядок (номер)	Географическое наименование	Географические координаты	
		Широта	Долгота
Старт	Санкт-Петербург	59°57'	30°19'
1	Рига	56°57'	24°06'
2	Вильнюс	54°41'	25°17'
3	Балтийск	54°39'	19°55'
4	Минск	53°54'	27°34'
5	Хойники	51°54'	29°58'
6	Киев	50°27'	30°30'
7	Пинск	52°07'	26°06'
8	Новозыбков	52°30'	32°00'
9	Клинцы	52°41'	32°17'
10	Яровщина	53°38'	34°42'
11	Плавск	54°41'	37°18'
Финиш	Москва	56°39'	36°31'

Определение 2. Петлей называется траектория движения по маршруту, имеющая замкнутый контур из ломаных линий, причем один угол такого контура находится вне перекрестка или населенного пункта.

На рис. 6.8 пунктирным контуром 2 выделен участок, имеющий характерную петлю. Точка А (угол Пинск-А-Хойники на пересечении траекторий Минск→Хойники и Пинск→Новозыбков) действительно находится за пределами населенных пунктов, через которые проходит маршрут.

Теорема. Независимо от метода, с помощью которого определяется минимальный маршрут коммивояжера, необходимым условием оптимизации пути является отсутствие петель в траектории.

Исходя из этой теоремы участок маршрута внутри контура 2 неоптимален. На нем установлен порядок:

Минск→Хойники→Киев→Пинск→Новозыбков.

После корректировки порядок изменится, а длина траектории уменьшится:

Минск→Пинск→Киев→Хойники→Новозыбков.

В результате двух выполненных корректировок из исходного (псевдооптимального) полетного расписания получено расписание полета по оптимальному маршруту (табл. 6.3).

Таблица 6.3

Оптимальное полетное расписание

Порядок (номер)	Географическое наименование	Географические координаты	
		Широта	Долгота
Старт	Санкт-Петербург	59°57'	30°19'
1	Рига	56°57'	24°06'
2	Балтийск	54°39'	19°55'
3	Вильнюс	54°41'	25°17'
4	Минск	53°54'	27°34'
5	Пинск	52°07'	26°06'
6	Киев	50°27'	30°30'
7	Хойники	51°54'	29°58'
8	Новозыбков	52°30'	32°00'
9	Клинцы	52°41'	32°17'
10	Яровщина	53°38'	34°42'
11	Плавск	54°41'	37°18'
Финиш	Москва	56°39'	36°31'

Алгоритм двух вертолетов. Допустим, имеется множество M населенных пунктов, которые нужно посетить. Сначала сделаем предварительные замечания о способах измерения расстояний между населенными пунктами. Для простоты будем считать, что их два:

- при планировании вертолетных работ расстояние по поверхности Земли вычисляется с помощью функции geoway;
- для действий с привязкой к дорожной сети расстояние определяется с помощью матрицы достижимости

$$D = \begin{vmatrix} d_{11} & d_{12} & \cdots & d_{1M} \\ d_{21} & d_{22} & \cdots & d_{2M} \\ \cdots & \cdots & \cdots & \cdots \\ d_{M1} & d_{M2} & \cdots & d_{MM} \end{vmatrix},$$

где d_{ij} – расстояние между пунктами A_i и A_j по дорожной сети, $i, j = 1, 2, \ldots, M$.

Такая матрица D составляется с помощью полной математической индукции при решении задачи коммивояжера для числа пунктов 2, 3, ..., M–1 при использовании карты региона. При этом расстояния отрезков дорожной сети определяются одним из двух способов:

- простейшего – с использованием обычных инструментов (циркуль, курвиметр, линейка);
- компьютерного – с помощью функции geoway.

Второй способ наиболее точный, если известны географические координаты всех пунктов с точностью до сотых долей секунды (электронные способы определения координат с такой точностью имеются).

Алгоритм индукции рассматривать не будем, так как он имеет довольно большое описание. Подготовленный читатель сможет самостоятельно убедиться в том, что предлагаемый ниже алгоритм работает и при постановке задачи коммивояжера с привязкой к дорожной сети.

Допустим, что необходимо построить маршрут из пункта $A_{\text{старт}}$ (с аэродрома базирования) до другого пункта $A_{\text{финиш}}$ с посещением заданного множества населенных пунктов, причем путь должен проходить через все пункты один раз и иметь минимальную длину. Только точки $A_{\text{старт}}$ и $A_{\text{финиш}}$ в частном случае могут принадлежать одному и тому же населенному пункту. Это самая общая постановка задачи.

При моделировании пространственных перемещений, связанных с поставкой товаров во многие пункты местности, используются узлы queue и proc (*см.* рис. 2.10). В аспекте имитации перемещений по маршруту необходимо установить такой порядок элементов в структурном массиве space, чтобы этот порядок и был оптимальным маршрутным расписанием. Для определенности считаем, что пункты $A_{\text{старт}}$ и $A_{\text{финиш}}$ имеют специальные признаки.

Сначала необходимо пункт $A_{\text{старт}}$ поставить на место 1, а $A_{\text{финиш}}$ – на место M в массиве space.

Далее предположим, что в пунктах $A_{\text{старт}}$ и $A_{\text{финиш}}$ находятся два вертолета (условно назовем их А и Б), которые начинают полет с оптимизацией единого маршрута одновременно. Зоной ответственности вертолета назовем замкнутый контур, населенные пункты внутри которого посещаются его экипажем (рис.6.9). В этой зоне необходимо оптимизировать маршрут по согласованию с экипажем другого вертолета, чтобы минимизировать суммарный маршрут.

Все пункты, которые включены в маршрут конкретного вертолета, относятся к его зоне ответственности. Но полностью эта зона становится известной только после построения всего маршрута. Поэтому в процессе построения можно говорить только о вероятности принадлежности пункта к такой зоне (или об экспертной оценке такой вероятности – о весовом коэффициенте, показывающем на большую или меньшую степень возможности включить пункт в зону ответственности). Экипаж вертолета, принимая очередное решение, выбирает для посещения тот населенный пункт, который с мини-

мальной вероятностью относится к зоне ответственности другого экипажа, с учетом оставшегося множества непосещенных пунктов и взаимного расположения.

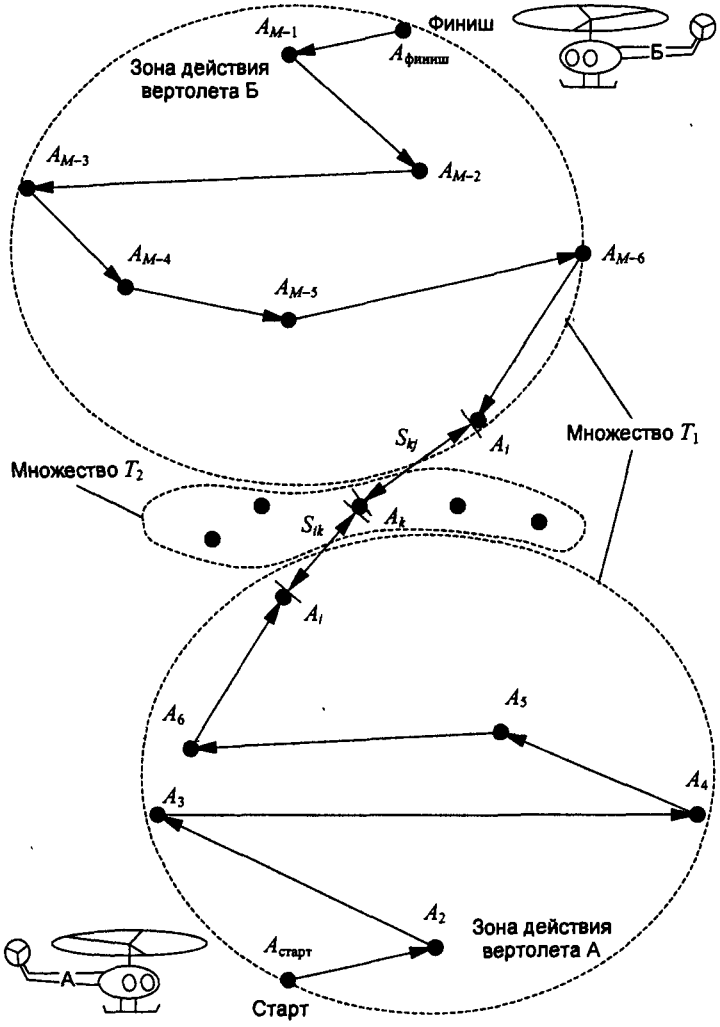


Рис. 6.9. Схема оперативного решения задачи коммивояжера (задача о двух вертолетах)

Если вертолеты при построении маршрута находятся в пунктах A_i и A_j (в процессе составления расписания) и можно считать, что они имеют одинаковую среднюю скорость, то определим вероятность того, что непосещенный пункт A_k принадлежит к зоне ответственности вертолета А, находящегося в пункте A_i, по формуле

$$p_{ik} = 1 - \frac{d_{ik}}{d_{ik} + d_{kj}},$$

где d_{ik} и d_{kj} – соответствующие расстояния между пунктами A_i, A_k и A_j.

При составлении маршрута для действий с привязкой к дорожной сети (автомобили, речные суда, железнодорожный транспорт) d_{ik} и d_{kj} – это элементы матрицы достижимости.

При таком составлении маршрута для одного или двух вертолетов задача коммивояжера может быть сведена к минимаксной игровой задаче двух партнеров, оптимизация решения которой достигается за счет выбора ближайшего пункта с максимальной вероятностью принадлежности к своей зоне ответственности.

Маршруты двух вертолетов заканчиваются в каких-то двух соседних населенных пунктах, а начинаются в общем случае с разных аэродромов. Поэтому после проигрывания маршрута для двух игроков получаем реальный маршрут, начинающийся на одном аэродроме и заканчивающийся на другом.

Рассмотрим алгоритм подробнее. Предположим, что имеется неупорядоченный список населенных пунктов, помещенный в массив space. Введем условные обозначения:

- i, j, k – номера пунктов в неупорядоченном структурном массиве space;
- A_i – название пункта;
- N_i – порядковый номер пункта i в массиве space после составления маршрута;
- L_i – расстояние от пункта N_i–1 до пункта N_i;
- G – общая длина пути после составления расписания;
- G_1 – длина пути вертолета А от пункта $A_{\text{старт}}$;
- G_2 – длина пути вертолета Б от пункта $A_{\text{финиш}}$ (навстречу первому);
- M – число упорядочиваемых элементов в массиве space (количество пунктов).

Перейдем к процедуре составления расписания. Присвоим начальные значения пунктам старта и финиша:

$$A_1 = A_{\text{старт}},\ N_1 = N_{\text{старт}};$$

$$A_M = A_{\text{финиш}},\ N_M = N_{\text{финиш}}.$$

Построение маршрута начнем от пункта старта (только для определенности) и предположим, что уже выполнено $x - 1$ шагов этой процедуры ($x = 1, 2, ..., M–2$).

Допустим, что вертолет А находится в пункте i, а вертолет Б – в пункте j. При этом множество T_1 всех упорядоченных пунктов, попавших в расписание после выполнения шага x, содержит $2 + x$ элементов, а множество T_2 всех неупорядоченных пунктов содержит $M–2–x$ элементов. Если T – множество всех M пунктов, то справедлива операция объединения непересекающихся множеств $T = T_1 \cup T_2$, где $\cup$ – операция объединения (сложения). Причем $T_1 \cap T_2 = 0$, где $\cap$ – операция пересечения (умножения).

Рассмотрим отдельно действия для каждого вертолета на шаге x.

1. *Вертолет А.* В множестве T_2 выбирается пункт A_k, имеющий минимальное значение выражения:

$$\min\left(\frac{d_{ik}}{d_{ik} + d_{kj}}\right),$$

где $A_i \in T_1$, $A_j \in T_1$, $A_k \in T_2$.

После определения такого пункта делаются следующие изменения:

$$G_1 = G_1 + d_{ik};$$
$$G = G + d_{ik};$$
$$N_k = N_i + 1;$$
$$L_k = d_{ik};$$
$$A_k \notin T_2;$$
$$A_k \in T_1;$$
$$i = k;$$
$$x = x + 1.$$

Если текущий путь вертолета А станет таким, что $G_1 > G_2$, то следующий шаг составления маршрута будет производиться для вертолета Б, так как считаем, что он долетел до пункта j своей зоны ответственности. В противном случае следующий шаг составления расписания опять делаем в отношении вертолета А.

2. *Вертолет Б.* В множестве T_2 выбирается пункт A_k, имеющий минимальное значение выражения:

$$\min\left(\frac{d_{kj}}{d_{ik}+d_{kj}}\right),$$

где $A_i \in T_1$, $A_j \in T_1$, $A_k \in T_2$.

После определения такого пункта делаются следующие изменения:

$$G_2 = G_2 + d_{kj};$$
$$G = G + d_{kj};$$
$$N_k = N_j - 1;$$
$$L_j = d_{kj};$$
$$A_k \notin T_2;$$
$$A_k \in T_1;$$
$$j = k;$$
$$x = x + 1.$$

Если текущий путь вертолета Б станет таким, что $G_2 \geq G_1$, то следующий шаг составления маршрута будет производиться для вертолета А, так как считаем, что он долетел до пункта i своей зоны ответственности. В противном случае следующий шаг составления расписания делаем в отношении вертолета Б.

Процедура составления маршрута делается всего за M–2 шагов, после чего множество T_2 будет пустым, а $T_1 = T$.

Рассмотренный метод был внедрен в практику работы оперативной группы при обследовании районов Брянской области, пострадавших в результате аварии на Чернобыльской АЭС. Проведенное моделирование позволило оценить максимальный эффект (в смысле дополнительного получения информации) по сравнению с расписанием, составленным вручную методом сканирования местности в сочетании с алгоритмом «ближайшего непосещенного города». Результаты представлены в табл. 6.3.

Таблица 6.3

Эффективность применения «алгоритма двух вертолетов»

№ п/п	Обследуемый регион	Выигрыш, %
1	Гордеевский район	48
2	Клинцовский район	53
3	Красногорский район	54
4	Новозыбковский и Злынковский районы (вместе)	61

Имеется закономерность: чем больше пунктов и чем больше разброс расстояний между ними, тем больше выигрыш. На практике этот выигрыш несколько меньше по двум причинам:

- пилоты эвристически применяют алгоритм «ближайшего непосещенного города»;
- вертолет затрачивает часть времени на посадки в населенных пунктах.

Реальный эффект в течение летней экспедиционной кампании 1989 г. составил 25 –35% экономии летного времени.

Следует отметить, что если алгоритм «ближайшего непосещенного города» никогда не даст возможности построить оптимальный маршрут, то алгоритм двух вертолетов, упрощенное описание которого мы рассмотрели, дает либо оптимальное решение, либо существенно лучшее (по сравнению с «алгоритмом ближайшего непосещенного города»). То, что данный алгоритм не всегда дает оптимальное решение, объясняется просто: в данном случае имеет место задача целочисленного программирования с тупиковыми локальными оптимумами. Однако петли маршрута легко обнаруживаются визуально на экране монитора, поэтому неоптимальные участки легко корректируются вручную.

Выводы

1. Изображение пространства (картографическое изображение) строится на математической основе, элементами которой на карте являются координатные сетки, масштаб и геодезическая основа. На мелкомасштабных картах элементы геодезической основы не показываются. С математической основой тесно связана и компоновка карты, т.е. взаимное размещение в пределах рамки самой изображаемой территории, названия карты, легенды, дополнительных карт и других данных.

2. Геоинформационные системы имеют следующие свойства:

- в состав ГИС входят базы данных, причем полная технология обработки информации в ГИС значительно шире, чем просто работа с базой данных;
- ГИС рассчитана не просто на обработку данных, а на проведение экспертных оценок во многих ситуациях;
- данные, которые обрабатывает и хранит ГИС, имеют не только пространственную, но и временную характеристику, что важно в первую очередь для географических данных.

Процент чисто географических данных в таких системах незначителен, технологии обработки данных имеют мало общего с традиционной обработкой географических данных и, наконец, географические данные служат лишь базой решения большого числа прикладных задач, цели которых далеки от географии. Поэтому современные геоинформационные системы не являются чисто географическими, а применяются вместе с другими компьютерными средствами, в том числе с *моделирующими системами*.

3. Наилучшее геометрическое приближение к реальной фигуре Земли – эллипсоид вращения, геометрическое тело, которое образуется при вращении эллипса вокруг его малой оси. Сжатие эллипсоида моделирует сжатие планеты у полюсов. В России принят эллипсоид Ф.Н. Красовского.

4. Взаимное расположение точек на поверхности Земли в средних широтах, характерных для России, стран Европы и США, рекомендуется изображать на экране монитора в виде нормальной конической проекции.

5. Большинство задач логистики, связанных с грузопотоками, необходимо решать с учетом информации о географических особенностях местности, дорожной сети и расстояниях, имеющейся в электронных картах (в ГИС) или получаемой из обычных карт.

Вопросы для самопроверки

1. Какая проекция рекомендуется к использованию при компьютерном изображении земной поверхности на территории России?
2. Как происходит отображение геоинформации в функциональном окне имитационной модели?
3. Какое средство в составе системы Pilgrim служит для определения расстояния между точками *X* и *Y* по их географическим координатам? Какие математические соотношения при этом используются?
4. Что такое карта? Какие основные свойства она имеет?
5. Что такое географические координаты?
6. Чем отличается план местности от карты?
7. Что такое картографическое изображение?
8. Из чего состоит легенда карты?
9. Из каких элементов состоит математическая основа карты?
10. Чем отличается карта от снимка, полученного с самолета или искусственного спутника Земли?

11. Для каких целей используются геоинформационные системы?
12. Каковы основные функции ГИС?
13. Что такое «электронный кадастр»?
14. Чем отличается геоид от эллипсоида вращения?
15. Когда необходимо учитывать, что Земля не является шаром?
16. Как определяется масштаб карты?
17. Какие картографические проекции используются в средних и приэкваториальных широтах?
18. Что означает аппроксимация (пространственная аппроксимация)?
19. Как решается задача оценки взаимосвязи между явлениями на поверхности Земли?
20. Как зависит коэффициент корреляции от угла между направлениями наибольших скатов (градиентов) двух сравниваемых статистических поверхностей?
21. Какие приемы теории информации используются для оценки степени однородности и взаимного соответствия явлений, изучаемых по картам?
22. Как формулируется задача коммивояжера в простейшей постановке? С привязкой к сети речных, шоссейных и железных дорог? Для выполнения вертолетных работ?
23. В чем заключается преимущество алгоритма «двух вертолетов» при моделировании логистических задач по сравнению с алгоритмом «ближайшего непосещенного города» и другими алгоритмами?

Глава 7

ПЛАНИРОВАНИЕ ИМИТАЦИОННОГО КОМПЬЮТЕРНОГО ЭКСПЕРИМЕНТА

7.1 КИБЕРНЕТИЧЕСКИЙ ПОДХОД К ОРГАНИЗАЦИИ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ СЛОЖНЫХ ОБЪЕКТОВ И ПРОЦЕССОВ

Имитационная модель независимо от выбранной системы моделирования (например, Pilgrim или GPSS) позволяет получить два первых момента и информацию о законе распределения любой величины, интересующей экспериментатора (экспериментатор – это субъект, которому нужны качественные и количественные выводы о характеристиках исследуемого процесса).

Если набор стандартных параметров, получаемых автоматически с помощью модели, не устраивает экспериментатора, то существуют следующие простейшие вспомогательные приемы.

Рассмотрим получение первого и второго момента произвольной величины, не являющейся параметром узла модели. Если неизвестная величина x является интервалом времени (или пропорциональна интервалу), то ее можно связать:

1) с интервалом пребывания клапана key, дополнительно введенного в модель, в запертом состоянии;

2) с временем жизни дополнительно сгенерированного транзакта в узле creat, помещенного в запертый клапан key, который в нужный момент открывается и направляется в дополнительный терминатор term.

В первом случае математическое ожидание длительности пребывания клапана в запертом состоянии m определяется автоматически в качестве параметра узла key. Дисперсия – это произведение

квадрата математического ожидания m^2 на коэффициент вариации в квадрате c^2, который также подсчитывается автоматически в этом узле.

Во втором случае математическое ожидание m времени жизни транзакта в узле term также определяется автоматически, а дисперсия получается в качестве произведения m^2c^2.

Для получения вида закона распределения, если не хватает стандартных выходных данных, предлагается простейший прием. Интересующий нас интервал возможных значений переменой x, которая имеет произвольный смысл (денежная сумма, объем партии товара и др.), делится на k равных интервалов: $(x_0, x_1]$, $(x_1, x_2]$, ..., $(x_{k-1}, x_k]$. В модели объявляется массив k переменных с фиксированной точкой: int p[k]. Во время прогона модели частоты появления значений x в этих интервалах значений подсчитываются в соответствующих элементах массива p. Вид закона определяется в виде ступенчатой функции. Поэтому получение доверительного интервала значений измеряемой величины x или проверка гипотезы о равенстве математического ожидания $M[x]$ заданному значению $d=const$ не вызывает затруднений.

Более сложной является задача планирования имитационного эксперимента для определения той области, в которой находится оптимальная (в каком-то смысле, с точки зрения экспериментатора) точка. Далее словом «опыт» будем называть один прогон модели, который дает возможность получить два первых момента интересующих нас величин. Серию целенаправленных опытов, позволяющих с некоторой достоверностью определить искомое экстремальное значение, назовем *экспериментом*.

Планирование эксперимента можно рассматривать как кибернетический подход к организации и проведению экспериментальных исследований сложных объектов и процессов. Основная идея метода состоит в возможности оптимального управления экспериментом в условиях неопределенности, что родственно тем предпосылкам, на которых базируется кибернетика. Целью большинства исследовательских работ является определение оптимальных параметров сложной системы или оптимальных условий протекания процесса:

- определение параметров инвестиционного проекта в условиях неопределенности и риска;
- выбор конструкционных и электрических параметров физической установки, обеспечивающих наиболее выгодный режим ее работы;

• получение максимально возможного выхода реакции путем варьирования температуры, давления и соотношения реагентов – в задачах химии;

• выбор легирующих компонентов для получения сплава с максимальным значением какой-либо характеристики (вязкость, сопротивление на разрыв и пр.) – в металлургии.

При решении задач такого рода приходится учитывать влияние большого количества факторов, часть из которых не поддается регулированию и контролю, что чрезвычайно затрудняет полное теоретическое исследование задачи. Поэтому идут по пути установления основных закономерностей с помощью проведения серии экспериментов. Методы эмпирического поиска оптимального решения долгое время оставались неформализованными. Исследователь выбирал ту или иную схему постановки эксперимента (стратегию), базируясь только на своем опыте и интуиции. Однако во второй половине XX в. начала усиленно развиваться математическая теория экстремальных экспериментов, которая помогает экспериментатору выбрать оптимальную стратегию. Основными показателями оптимальности при этом являются уменьшение числа экспериментов при обеспечении той же точности результатов исследования или сохранение числа экспериментов при увеличении точности. Существенные упрощения при этом достигнуты в методах обработки результатов эксперимента. Исследователь получил возможность путем несложных вычислений выражать результаты эксперимента в удобной для их анализа и использования форме.

7.2 РЕГРЕССИОННЫЙ АНАЛИЗ И УПРАВЛЕНИЕ МОДЕЛЬНЫМ ЭКСПЕРИМЕНТОМ

В общем случае объект исследования можно представить как некоторый «черный ящик» (рис. 7.1), на входе которого действуют управляющие параметры x_i ($i = 1, 2, \ldots, k$) и неконтролируемые возмущения z_j ($j = 1, 2, \ldots, m$). Выходом объекта исследования являются показатели качества или какие-либо другие характеристики объекта η_v ($v = 1, 2, \ldots, n$). Например, в задаче исследования движения самолета или ракеты управляющими входными параметрами будут углы отклонения рулей направления и рулей высоты, режим работы двигательной установки, качество топлива и т.д. Выходные параметры –

высота, направление, скорость полета. К возмущающим факторам или помехам, которые не оказывают преимущественного влияния на процесс движения, но все же заметно искажают выходные параметры, можно отнести изменения температуры, влажность, скорость и направление воздушных течений и т.п. В задачах металлургии под переменными x_i можно понимать процентный состав компонентов сплава, под помехами z_j – вредные примеси, количество которых колеблется от плавки к плавке, η_v – характеристики сплава.

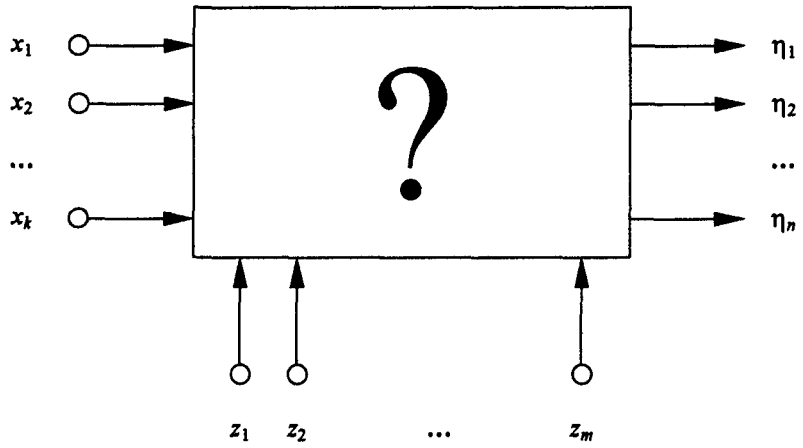


Рис. 7.1. Схема исследования системы или процесса

В электронных установках управляющими факторами являются параметры электронных деталей, величины напряжений и токов. Помехи возникают в результате работы других электронных установок (путем наводок и через общие цепи питания), а также в результате изменения характеристик параметров (температурный и временной дрейф).

Если рассмотреть зависимость одной из характеристик системы $\eta_v(x_i)$, как функцию только одной переменной x_i, (рис. 7.2), то при фиксированных значениях x_i будем получать различные значения $\eta_v(x_i)$. Разброс значений η_v в данном случае определяется не только ошибками измерения, а главным образом влиянием помех z_j. Сложность задачи оптимального управления характеризуется не только сложностью самой зависимости $\eta_v(x_1, x_2, \ldots, x_k)$, но и влиянием z_j,

что вносит элемент случайности в эксперимент. График зависимости $\eta_v(x_i)$ определяет корреляционную связь величин η_v и x_i, которая может быть получена по результатам эксперимента с помощью методов математической статистики. Вычисление таких зависимостей при большом числе входных параметров x_i и существенном влиянии помех z_j и является основной задачей исследователя-экспериментатора. При этом чем сложнее задача, тем эффективнее становится применение методов планирования эксперимента.

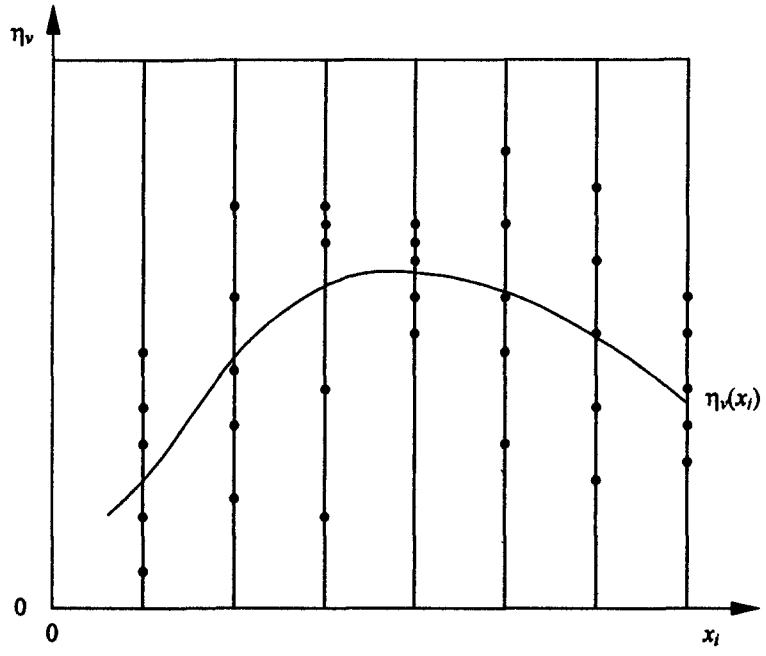


Рис. 7.2. Пример усреднения результатов эксперимента

Различают два вида эксперимента: пассивный и активный. При пассивном эксперименте исследователь только ведет наблюдение за процессом (за изменением его входных и выходных параметров). По результатам наблюдений затем делается вывод о влиянии входных параметров на выходные. Пассивный эксперимент обычно выполняется на базе действующего экономического (производственного) процесса, который не допускает активного вмешательства экспериментатора. Этот метод малозатратный, но требует большого времени.

Активный эксперимент проводится главным образом в лабораторных условиях, где экспериментатор имеет возможность изменять входные характеристики по заранее намеченному плану. Такой эксперимент быстрее приводит к цели, и именно к нему применимы идеи планирования экстремального эксперимента.

На математическом языке задача установления взаимосвязей оптимизируемого процесса формулируется следующим образом: нужно получить некоторое представление о функции отклика

$$\eta = \varphi(x_1, x_2, \ldots, x_k),$$

где η – параметр процесса, подлежащий оптимизации;

$x_1, x_2, \ldots, x_k$ – независимые переменные, которые можно варьировать при постановке экспериментов.

Координатное пространство с координатами $x_1, x_2, \ldots, x_k$ называют *факторным пространством*. Геометрический образ соответствующей функции отклика называется *поверхностью отклика*.

Будем рассматривать самый общий случай, когда исследование поверхности отклика ведется при неполном знании механизма изучаемых явлений. Естественно, что и в этом случае аналитическое выражение функции отклика неизвестно. Наиболее удобным оказалось представление функции отклика в виде полинома:

$$\eta = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i \neq j} \beta_{ij} x_i x_j + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \ldots,$$

где β_0, β_i, β_{ij}, β_{ii} – коэффициенты регрессии.

Пользуясь результатами эксперимента, можно определить выборочные коэффициенты регрессии b_0, b_i, b_{ij}, b_{ii}, которые являются лишь оценками (приближенными значениями) для теоретических коэффициентов регрессии. Уравнение регрессии, полученное на основе опыта, запишется так:

$$\bar{y} = b_0 + \sum_{i=1}^{k} b_i x_i + \sum_{i \neq j} b_{ij} x_i x_j + \sum_{i=1}^{k} b_{ii} x_i^2 + \ldots, \qquad (7.1)$$

где $\bar{y}$ – значение выхода, предсказанное уравнением (выборочная оценка для η).

Допустим, что у нас имеется N результатов наблюдения величины y, зависящей от $x_1, x_2, \ldots, x_k$. Положим, что результаты наблюдений нужно представить полиномами степени d. Тогда число коэффициентов регрессии будет равно C_{k+d}^{d} (число сочетаний из $k+d$ по d). Очевидно, необходимо, чтобы $N \geq C_{k+d}^{d}$.

Для определения численных значений выборочных коэффициентов регрессии используется так называемый регрессионный анализ (метод наименьших квадратов). В регрессионном анализе полагается, что выполняется ряд предпосылок.

1. Результаты наблюдений $y_1, y_2, \ldots, y_N$ – независимые, нормально распределенные случайные величины. Речь идет о распределении y относительно некоторой фиксированной точки $x_1, x_2, \ldots, x_k$, так как на значение y влияют и другие неконтролируемые параметры. Если эта предпосылка не удовлетворяется, то коэффициенты регрессии найти можно, однако ничего нельзя будет сказать об эффективности метода, т.е. нельзя оценить точность уравнения регрессии. Если y не подчиняется нормальному распределению, то стараются подобрать такую функцию преобразования, чтобы перейти от y к новой случайной величине $q = f(y)$, распределенной приближенно нормально. Например, для многих асимметричных распределений делается замена $q = \ln y$.

2. Дисперсии $\sigma_y^2 = \sigma^2\{y_u\}$, $u = 1, 2, \ldots, N$ равны друг другу. Это значит, что если производить многократные и повторные наблюдения над величиной y_u при некотором определенном наборе значений $x_{1u}, x_{2u}, \ldots, x_{ku}$, то получим дисперсию σ_y^2, которая не будет зависеть от математического ожидания $M\{y_u\}$, т.е. не будет отличаться от σ_y^2, полученной при повторных наблюдениях для любого другого набора независимых переменных. Это требование также не всегда выполняется для реального эксперимента.

3. Независимые переменные $x_1, x_2, \ldots, x_k$ измеряются с пренебрежимо малой ошибкой по сравнению с ошибкой в определении y.

При таких исходных предпосылках оказывается возможным вычислить коэффициенты b_0, b_i, b_{ij}, b_{ii}, а также оценить их точность и точность уравнения регрессии (7.1) в целом.

7.3 ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТОВ РЕГРЕССИИ

Упростим систему обозначений, т.е. введем фиктивную переменную $x_0 = 1$ и заменим члены второго порядка линейными, положив

$$\begin{cases} x_1^2 = x_{k+1}, \ x_2^2 = x_{k+2}, \ \ldots, \ x_k^2 = x_{2k}; \\ x_1 x_1 = x_{2k+1}, \ \ldots, \ x_{k-1} x_k. \end{cases}$$

Аналогичным образом можно заменить линейными членами члены любого порядка. В новой системе обозначений полином степени d будет записываться как однородное линейное уравнение

$$\overline{y} = b_0 x_0 + b_1 x_1 + \ldots + b_{k'} x_{k'}, \qquad (7.2)$$

где $k' = C_{k+d}^{d} - 1$.

Штрих при k в дальнейшем будем опускать.

Чтобы методом наименьших квадратов найти оценки коэффициентов регрессии, нужно минимизировать сумму квадратов отклонений:

$$S = \sum_{u=1}^{N} \left\{ y_u - \left(b_0 x_{0u} + b_1 x_{1u} + \ldots + b_k x_{ku} \right) \right\}^2, \qquad (7.3)$$

где y_u – результат измерения выходного параметра при значениях входных параметров x_{iu}; $i = 0, 1, \ldots, k$; $u = 1, 2, \ldots, N$.

Приравнивая нулю частные производные от квадратичной формы (7.3), по искомым коэффициентам $b_0, b_1, \ldots, b_k$ получаем

$$\frac{\partial S}{\partial b_i} = 2 \sum_{u=1}^{N} \left\{ (y_u - b_0 x_{0u} - b_1 x_{1u} - \ldots - b_k x_{ku}) x_{iu} \right\} = 0, \qquad (7.4)$$

где $i = 0, 1, 2, \ldots, k$.

Если ввести обозначения

$$(ij) = (ji) = \sum_{u=1}^{N} x_{iu} x_{ju};$$

$$(iy) = \sum_{u=1}^{N} x_{iu} y_u,$$

то выражение (7.4) можно представить следующей системой так называемых нормальных уравнений:

$$\begin{cases} b_0(00) + b_1(01) + \ldots + b_k(0k) = (0y); \\ b_0(10) + b_1(11) + \ldots + b_k(1k) = (1y); \\ \ldots \quad \ldots \quad \ldots \quad \ldots = \ldots \\ b_0(k0) + b_1(k1) + \ldots + b_k(1k) = (ky). \end{cases}$$

Чтобы найти интересующие нас коэффициенты регрессии, нужно решить систему нормальных уравнений относительно неизвестных $b_0, b_1, \ldots, b_k$. Для упрощения системы обозначений обратимся к

матричной алгебре. Матрицы, с помощью которых представляются результаты наблюдений, будут иметь следующий вид:

$$X = \begin{vmatrix} x_{01} & x_{11} & \dots & x_{k1} \\ x_{02} & x_{12} & \dots & x_{k2} \\ \dots & \dots & \dots & \dots \\ x_{0N} & x_{1N} & \dots & x_{kN} \end{vmatrix}, \quad Y = \begin{vmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{vmatrix}.$$

Если обозначить транспонированную матрицу X через X^*, то матрицы для записи нормальных уравнений могут быть представлены в виде:

$$X^*X = \{(ij)\} = \begin{vmatrix} (00) & (01) & \dots & (0k) \\ (10) & (11) & \dots & (1k) \\ \dots & \dots & \dots & \dots \\ (k0) & (k1) & \dots & (kk) \end{vmatrix}; \quad X^*Y = \{(iy)\} = \begin{vmatrix} (0y) \\ (1y) \\ \dots \\ (ky) \end{vmatrix};$$

$$B = \{b_i\} = \begin{vmatrix} b_0 \\ b_1 \\ \dots \\ b_k \end{vmatrix}.$$

В матричной форме система нормальных уравнений запишется следующим образом:

$$X^*XB = X^*Y.$$

Введем матрицу $(X^*X)^{-1}$, обратную матрице X^*X, ее элементы обозначим как c_{ij}:

$$(X^*X)^{-1} = \{c_{ij}\} = \begin{vmatrix} c_{00} & c_{01} & \dots & c_{0k} \\ c_{10} & c_{11} & \dots & c_{1k} \\ \dots & \dots & \dots & \dots \\ c_{k0} & c_{k1} & \dots & c_{kk} \end{vmatrix}. \qquad (7.5)$$

Умножив слева обе части матричного уравнения (7.5) на $(X^*Y)^{-1}$, получим

$$(X^*X)^{-1}(X^*X)B = B = (X^*X)^{-1}(X^*Y).$$

Отсюда следует, что интересующие нас коэффициенты регрессии (элементы матрицы B) определяются выражением

$$b_i = \sum_{j=0}^{k} c_{ij}(jy). \qquad (7.6)$$

Оказывается, что коэффициенты регрессии не могут быть определены независимо друг от друга. Сумма произведений, определяющая i-й коэффициент регрессии, состоит из k+1 членов, соответствующих k+1 коэффициенту регрессии. Если мы почему-либо изменим порядок полинома d, то все вычисления нужно будет производить заново. Уменьшить объем вычислений для получения b_i и обеспечить их независимость можно, если эксперименты планировать по некоторой схеме так, чтобы в матрице планирования X скалярные произведения для всех векторов-столбцов были равны нулю:

$$\sum_{u=1}^{N} x_{iu} x_{ju} = 0 .$$

При таком планировании, называемом ортогональным, матрица X^*X станет диагональной, т.е. система нормальных уравнений распадается на k+1 независимых уравнений:

$$b_0(00) = (0y);$$
$$b_0(11) = (1y);$$
$$\ldots \quad \ldots \quad \ldots$$
$$b_k(kk) = (ky).$$

Коэффициенты регрессии будут определяться независимо друг от друга:

$$b_i = c_{ii}(iy), \tag{7.7}$$

где $c_{ii} = \dfrac{1}{(ii)}$.

Увеличение или уменьшение членов в уравнении регрессии не будет изменять коэффициенты у остальных членов уравнения.

7.4 СТАТИСТИЧЕСКИЙ АНАЛИЗ УРАВНЕНИЯ РЕГРЕССИИ

После того как вычислены коэффициенты регрессии, нужно произвести статистический анализ уравнения регрессии, т.е. дать статистические оценки точности этого уравнения.

Прежде всего определим остаточную сумму квадратов

$$S_R = \sum_{u=1}^{N} (y_u - \overline{y})^2, \tag{7.8}$$

характеризующую рассеяние точек относительно найденного уравнения регрессии.

Рассмотрим

$$\overline{Y} = \begin{vmatrix} \overline{y}_1 \\ \overline{y}_2 \\ \dots \\ \overline{y}_N \end{vmatrix}$$

– вектор-столбец значений, предсказанных уравнением регрессии. Запишем матричное уравнение: $\overline{Y} = XB$.

Остаточная сумма квадратов в матричной форме

$$\begin{aligned} S_R &= (\overline{Y} - Y)^* (\overline{Y} - Y) = (\overline{Y}^* - Y^*)(\overline{Y} - Y) = \\ &= \overline{Y}^* \overline{Y} - \overline{Y}^* Y - Y^* \overline{Y} + Y^* Y = \overline{Y}^* \overline{Y} - 2\overline{Y}^* Y + Y^* Y. \end{aligned} \tag{7.9}$$

Заметим, что

$$\begin{aligned} \overline{Y}^* Y &= Y^* \overline{Y}; \\ \overline{Y}^* Y &= (XB)^* Y = B^* X^* Y, \end{aligned} \tag{7.10}$$

и следовательно,

$$\overline{Y}^* Y = B^* X^* XB = B^* X^* X [(X^* X)^{-1} X^* X] = B^* X^* Y. \tag{7.11}$$

С учетом равенств (7.10) и (7.11) из выражения (7.9) получаем

$$(\overline{Y} - Y)^* (\overline{Y} - Y) = B^* X^* Y - 2B^* X^* Y + Y^* Y = Y^* Y - B^* X^* Y.$$

Переходя от матричной формы к обычной алгебре, получаем остаточную сумму квадратов в виде выражения

$$S_R = (yy) - \sum_{i=0}^{k} b_i (iy), \tag{7.12}$$

которым значительно легче пользоваться, чем непосредственно соотношением (7.8). Значения b_i нам известны, суммы (iu) подсчитываются в процессе вычисления b_i, остается только вычислить сумму квадратов:

$$(yy) = \sum_{u=1}^{N} y_u^2.$$

Опыт показывает, что в целях проверки всей системы вычислений полезно остаточную сумму квадратов вычислять дважды, пользуясь формулами (7.9) и (7.12).

Сама величина S_R недостаточно удобна для определения степени разброса экспериментальных точек относительно уравнения регрессии, так как она зависит от N. Поэтому обычно пользуются остаточной дисперсией, которая характеризует разброс, отнесенный к одной точке измерения,

$$s_y^2 = \frac{S_R}{f},$$

где f – число степеней свободы; $f = N - k - 1$.

Так, если $\bar{y} = b_0$, то $f = N - 1$; если же $\bar{y} = b_0 + b_1 x_1$, то $f = N - 1$ и т.д. Для вычисления каждого коэффициента b_i требуется минимум одна точка y_u. Оставшиеся точки могут рассматриваться как свободные, и к их числу относят остаточную сумму квадратов S_R. Если число опытов равно числу коэффициентов, т.е. $N = k + 1$, то уравнение регрессии пройдет через все N точек и разброса вообще не будет ($S_R = 0$).

Если s_y^2 мало, то, следовательно, уравнение регрессии достаточно точно характеризует процесс; если s_y^2 велико, то или в уравнении регрессии не учтены какие-то существенные факторы x_i, или неправильно выбрана степень полинома. При этом если на основе проведения аналогичных экспериментов известна ошибка опыта σ_y^2, т.е. ошибки измерения и влияние неконтролируемых факторов z_j, то можно найти F-отношение:

$$F = \frac{s_y^2}{\sigma_y^2} \qquad (7.13)$$

и проверить гипотезу об адекватности представления результатов полиномом заданной степени d.

В математической статистике F-распределение используется для проверки равенства дисперсий в двух сериях опытов. Остановимся несколько подробнее на этом понятии. В выражении для дисперсии

$$s_y^2 = \frac{1}{f} \sum_{u=1}^{N} (y_u - \bar{y})^2$$

в числителе находится сумма квадратов случайных, нормально распределенных чисел (по предпосылке регрессионного анализа) с математическим ожиданием, равным нулю ($\bar{y}$ является математическим ожиданием для y_u). Эта сумма сама есть случайное число, так как для каждой новой серии опытов N можно получать другое значение s_y^2. Такое случайное число имеет свой закон распределения, который зависит от числа степеней свободы. Этот закон получил название χ^2 (хи-квадрат). F-отношение есть отношение двух случайных величин, каждая из которых подчинена закону χ^2:

$$F = \frac{s_{y_1}^2}{s_{y_2}^2},$$

поэтому F также является случайной величиной, она подчинена так называемому закону F-распределения. Плотность распределения этой величины определяется выражением

$$W_F(F, f_1, f_2) = C \frac{F^{0,5 f_1 - 1}}{(f_2 + f_1 F)^{0,5(f_1 + f_2)}}, \qquad (7.14)$$

где

$$C = \frac{\Gamma\left(\frac{f_1 + f_2}{2}\right)}{\Gamma\left(\frac{f_1}{2}\right)\Gamma\left(\frac{f_2}{2}\right)} \cdot f_1^{0,5 f_1} \cdot f_2^{0,5 f_2};$$

где $\Gamma(\alpha)$ – гамма-функция от аргумента α;

f_1, f_2 – степени свободы соответственно для $s_{y_1}^2$ и $s_{y_2}^2$.

Случайная величина F является отношением двух положительных величин, характер $W_R(F, f_1, f_2)$ представлен на рис. 7.3 при фиксированных значениях f_1 и f_2. На практике непосредственно выражение (7.14) не применяют, а используют таблицы, которые приводятся в пособиях по математической статистике (*см.* приложение 3).

Чтобы проверить гипотезу о равенстве дисперсий ($s_{y_1}^2 = s_{y_2}^2$, т.е. $F = 1$), нужно задаться областью недопустимых значений F, которую считаем неприемлемой. Только тогда можно судить о том, будет ли полученное числовое значение F слишком большим или малым.

За эту критическую область принимают два интервала: интервал больших значений $F > F_2$ и интервал малых значений $0 < F < F_1$ (*см.* рис. 7.3). Причем точки F_1 и F_2 подбираются так, чтобы выполнялись равенства для соответствующих вероятностей:

$$P[F > F_2] = q/2,$$

$$P[F < F_1] = q/2,$$

где q – уровень значимости (он часто задается в процентах).

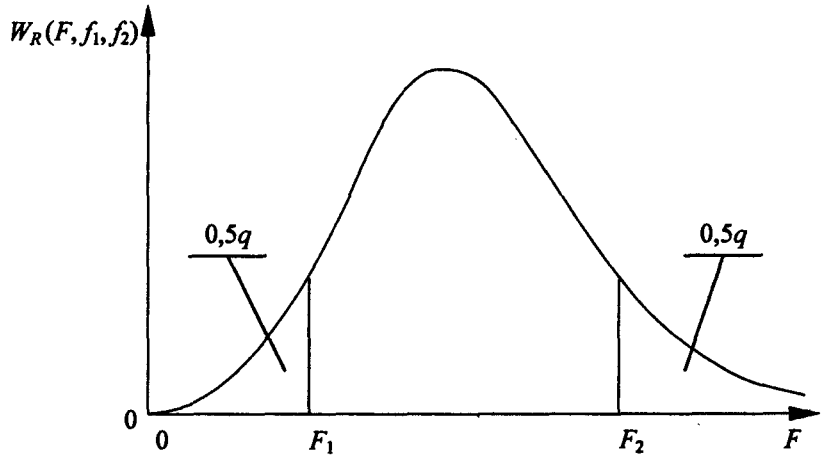


Рис. 7.3. Вид F-распределения

Если полученное значение F окажется вне интервала (F_1, F_2), то гипотеза о равенстве $s_1^2 = s_2^2$ отбрасывается. Причем правильность этого решения будет гарантирована с достоверностью $(1 - q)$. В $100q$ процентах случаев гипотеза будет отвергаться напрасно. С увеличением q как бы налагаются более жесткие условия на совпадение результатов, и естественно, что гипотеза будет отвергаться чаще. Уменьшение q означает меньшую требовательность. Поскольку для сравнения дисперсий можно брать и обратное отношение

$$\frac{s_{y1}^2}{s_{y2}^2} = F^1,$$

то получается, что F_1 соответствует F_2^1. В итоге оказывается достаточно вычислить одну границу интервала, а именно F_2, при этом в

числителе берут большее значение s_y^2. В таблицах дается одна величина $F_т$, с которой сравнивается полученное расчетное значение $F_р$.

Пример 7.1. Использование *F*-отношения. Рассмотрим результаты моделирования инвестиционного процесса по двум вариантам бизнес-планов инвестиционного проекта. Оба варианта в конечном итоге приводят к получению одного и того же показателя чистого приведенного дохода *NPV*. По варианту 1 было проведено $n_1 = 10$ опытов с имитационной моделью, а по варианту 2 проведено $n_2 = 15$ опытов. Обе серии опытов, естественно, имеют какое-то отклонение от требуемого значения *NPV*. Дисперсии были подсчитаны, и оказалось, что $s_{y_1}^2 = 9{,}6$ млн руб., $s_{y_2}^2 = 5{,}7$ млн руб. Откуда

$$F_р = \frac{s_{y1}^2}{s_{y2}^2} = \frac{9{,}6}{5{,}7} = 1{,}68.$$

Степень свободы для $s_{y_1}^2$ равна $f_1 = 10 - 1 = 9$, а для $s_{y_2}^2$ она равна $f_2 = 15 - 1 = 14$. Уравнение регрессии в этом примере имеет вид $\bar{y} = b_0$, где b_0 – искомое среднее значение «чистого» приведенного дохода, т.е. *NPV* не зависит от управляющих факторов x_i. На основе полученного значения $F = 1{,}68$ требуется сделать вывод о том, одинакова ли степень неопределенности (и риска) у вариантов 1 и 2. Разница в величинах $s_{y_1}^2$ и $s_{y_2}^2$ может быть как результатом разной неопределенности (различного риска), так и следствием случайности. Зададим уровень значимости $q = 10\%$. Для $f_1 = 9$ и $f_2 = 14$ из таблицы (*см.* приложение 3) найдем $F_т = 2{,}65$. Поскольку $F_р < F_т$, значение $F_р = 1{,}68$ при $q = 10\%$ является незначимым, т.е. предположение о равенстве дисперсий (а следовательно, и одинаковой неопределенности вариантов бизнес-планов) не противоречит результатам серий опытов. Если бы получилось $F_р > 2{,}65$, то гипотезу об одинаковой неопределенности нужно было бы отбросить, при этом в 10% случаев гипотеза была бы отброшена напрасно.

Аналогичным образом, используя отношение (7.13), можно убедиться в правильности выбора степени полинома *d*. Если априори есть достаточно оснований для выбора *d*, то остаточную дисперсию s_y^2 можно рассматривать как оценку дисперсии, характеризующей ошибку эксперимента.

Кроме точности уравнения регрессии в целом большое значение имеет точность в определении самих коэффициентов регрессии. Коэффициентами, значение которых соизмеримо с погрешностью их определения, очевидно, следует пренебречь.

Дисперсии $s^2_{b_i}$, характеризующие ошибки в определении коэффициентов регрессии (в каждой новой серии опытов для одного и того же уравнения регрессии будут получаться различные значения одних и тех же коэффициентов b_i из-за влияния помех z_j), также можно определить с помощью преобразования матриц. Определим вектор ошибок:

$$B-\beta=\begin{vmatrix}\Delta b_1\\ \Delta b_2\\ \dots\\ \Delta b_k\end{vmatrix},$$

где β – теоретическое значение коэффициентов.

Вычислим математическое ожидание:

$$M\left\{(B-\beta)(B-\beta)^*\right\}=M\left\{\begin{vmatrix}\Delta b_1\\ \Delta b_2\\ \dots\\ \Delta b_k\end{vmatrix}\times\left|\Delta b_1\ \Delta b_2\ \dots\ \Delta b_k\right|\right\}=$$

$$=\begin{vmatrix}\overline{\Delta b_1\Delta b_1} & \overline{\Delta b_1\Delta b_2} & \dots & \overline{\Delta b_1\Delta b_k}\\ \overline{\Delta b_2\Delta b_1} & \overline{\Delta b_2\Delta b_2} & \dots & \overline{\Delta b_2\Delta b_k}\\ \dots & \dots & \dots & \dots\\ \overline{\Delta b_k\Delta b_1} & \overline{\Delta b_k\Delta b_2} & \dots & \overline{\Delta b_k\Delta b_k}\end{vmatrix}.$$

Черта над элементами матрицы означает усреднение по всем сериям опытов. Введем следующие обозначения:

$$\beta=M\{B\},\ \Delta=Y-M\{Y\}.$$

Поскольку

$$B=(X^*X)^{-1}X^*Y,$$

то

$$M\{B\}=(X^*X)^{-1}X^*M\{E\}.$$

Случайными являются величины y и b, а x измеряется точно, поэтому $(X^*X)^{-1}$ и X^* – константы.

Сделаем следующие преобразования:

$$
\begin{aligned}
&M\{(B-\beta)(B-\beta)^*\} = \\
&= M\{[(X^*X)^{-1}X^*Y - M\{B\}]\times[(X^*X)^{-1}X^*Y - M\{B\}]^*\} = \\
&= M\{[(X^*X)^{-1}X^*(Y - M\{Y\})]\times[(X^*X)^{-1}X^*(Y-(M\{Y\})]^*\} = \\
&= M\{[(X^*X)^{-1}X^*\Delta]\times[(X^*X)^{-1}X^*\Delta]^*\} = \\
&= M\{(X^*X)^{-1}X^*\Delta\Delta^*X(X^*X)^{-1}\} = (X^*X)^{-1}\sigma_y^2 .
\end{aligned}
\qquad (7.15)
$$

При получении результата (7.15) использованы следующие свойства матрицы:

$$(ABC)^* = A^*B^*C^*,\ (A^{-1})^* = (A^*)^{-1},$$

поэтому

$$\left[\left(X^*X\right)^{-1}\right]^* = \left[\left(X^*X\right)^*\right]^{-1} = \left[X^*X\right]^{-1} \equiv \left(X^*X\right)^{-1}.$$

Поясним также преобразования

$$
M\left\{\Delta^*\Delta\right\} = M\left\{\begin{vmatrix}\Delta y_1\\ \Delta y_2\\ \dots\\ \Delta y_k\end{vmatrix}\times\left|\Delta y_1\ \Delta y_2\ \dots\ \Delta y_k\right|\right\} =
$$

$$
= \begin{vmatrix}\overline{\Delta y_1^2} & 0 & \dots & 0\\ 0 & \overline{\Delta y_1^2} & \dots & 0\\ \dots & \dots & \dots & \dots\\ 0 & 0 & \dots & \overline{\Delta y_1^2}\end{vmatrix} = \sigma_y^2 E,
$$

где E – единичная матрица.

В соответствии с предпосылками регрессионного анализа (*см.* разд. 7.2) все Δy_1^2 равны, а усредненные произведения $\overline{\Delta y_i^2 \Delta y_j^2} = 0$ при $i \neq j$ (*см.* первую предпосылку, независимость y_u).

Переходя к обычной форме записи, из выражения (7.15) получаем

$$\sigma^2_{b_i} = c_{ii}\sigma^2_y, \quad s^2_{b_i} = c_{ii}s^2_y, \tag{7.16}$$

где c_{ii} – коэффициенты (будут определены позднее).

Зная $s^2_{b_i}$, можно установить доверительные границы для коэффициентов b_i. Введем случайную величину

$$t = \frac{b_i - \beta_i}{s_{b_i}},$$

где β_i – теоретическое значение коэффициента.

Поскольку b_i получается из линейной комбинации y_u, а y_u имеет нормальное распределение по условиям регрессионного анализа, то разность $b_i - \beta_i$ также распределена по нормальному закону. Величина t, являющаяся отношением нормально распределенной случайной величины к случайной величине, распределенной по $\sqrt{\chi^2}$, имеет плотность так называемого t-распределения:

$$W_t(t,f) = \frac{1}{\sqrt{\pi f}} \cdot \frac{\Gamma(0{,}5(1+f))}{\Gamma(0{,}5f)} \left(1 + \frac{t^2}{f}\right)^{-0{,}5(f+1)}.$$

Данное распределение напоминает по характеру нормальное распределение при $f = 1$ и переходит в него при $f \to \infty$ (рис. 7.4). Вероятность того, что случайная величина t примет значение $|t| > t_q$, будет равна q. Величина q называется уровнем значимости величины t (рис. 7.5). Если зададимся значением $q = 10\%$ и найдем в таблице $t_q(f)$ (*см.* приложение 4), то сможем утверждать, что в 90% случаев истинное значение коэффициента β_i будет лежать в пределах

$$b_i - t_q(f)s_{b_i} \le \beta_i \le b_i + t_q(f)s_{b_i}. \tag{7.17}$$

Коэффициент b_i незначим, если

$$b_i < t_q(f)s_{b_i}.$$

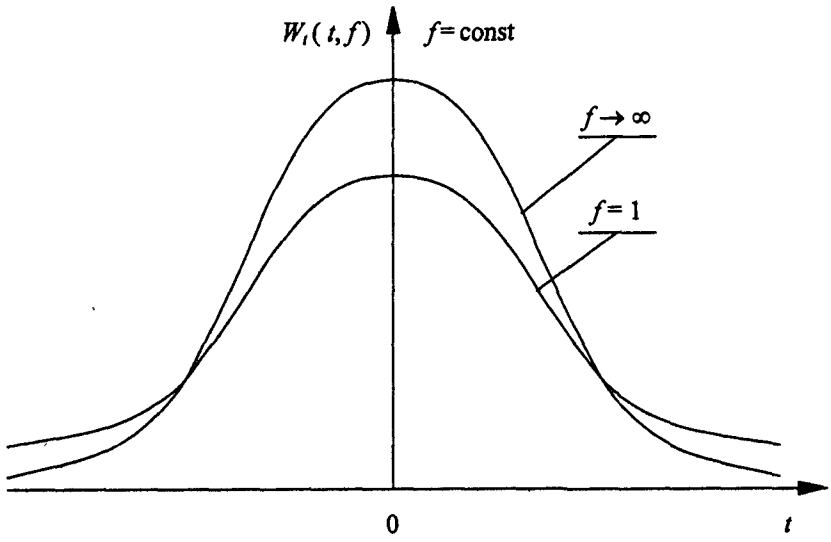


Рис. 7.4. Вид t-распределения

В математической статистике нельзя дать абсолютно утвердительного ответа ни по одному из параметров. Можно только с определенной степенью достоверности указывать пределы, в которых находится значение параметра. Так, в данном случае степень достоверности равна 0,9.

Величина $t_q(f)$ возрастает с уменьшением q и уменьшается с увеличением f. Это естественно: увеличение степени достоверности (уменьшение q) вызывает увеличение диапазона возможного значения параметра (*см.* рис. 7.5). Увеличение f означает увеличение числа испытаний, т.е. более точное определение параметров процесса, поэтому с ростом f диапазон разброса параметра $t_q(f)$ уменьшается.

Доверительные границы по соотношению (7.17) можно установить только для случая ортогонального планирования, когда диагональные элементы обратной матрицы c_{ii} определяются независимо:

$$c_{ii} = \frac{1}{(ii)}.$$

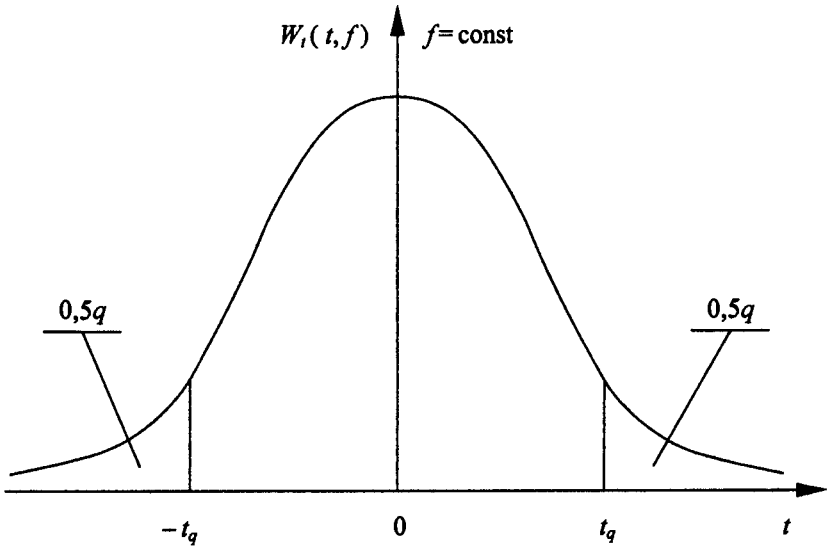


Рис. 7.5. Пример использования t-распределения для определения интервала достоверности

Если матрица не диагональная (для неортогонального планирования), то в вычислении c_{ii} участвуют все элементы матрицы (X^*X). При изменении числа переменных k, естественно, c_{ii} получают другое значение. Поэтому определение дисперсии для каждого коэффициента b_i становится возможным только при фиксировании значений остальных коэффициентов.

Рассмотрим пример на получение и анализ уравнения регрессии.

Пример 7.2. Неортогональное планирование эксперимента. Для того чтобы лучше представлять, как идет процесс решения, зададим сами функцию двух переменных x_1, x_2, и полученные значения функции в заданных точках «засорим» шумами. По этим «засоренным» точкам, которые будут имитировать экспериментальные точки, будем искать нами же заданный полином (в действительности же вид полинома и даже его степень бывают неизвестны):

$$\eta(x_1, x_2) = 10 + 4_{x_1} + 6_{x_2} + 3_{x_1 x_2} .$$

Значения функции $\eta(x_1, x_2)$ и измеренные значения y_u представлены в табл. 7.1.

Таблица 7.1

Значения функции $\eta(x_1, x_2)$

u	1	2	3	4	5	6	7	8	9
x_1	0	0	0	1	2	1	2	0	–1
x_2	0	1	2	0	0	1	2	–1	–1
$\eta(x_1, x_2)$	10	16	22	14	18	23	42	4	3
y_u	10	17	20	14	18	24	40	3	3
Δy_u	0	1	–2	0	0	1	–2	–1	0
y_u^2	0	1	4	0	0	1	4	1	0

В ряде случаев бывает известна ошибка опыта. В данном примере можно принять за ошибку опыта разность между y_u и $\eta(x_1, x_2)$. Тогда

$$\sigma_y^2 = \frac{1}{9}\sum_{u=1}^{9}\Delta y_u^2 = \frac{11}{9} = 1{,}22\,;\ \sigma_y = 1{,}11;\ f = N = 9.$$

Забудем теперь об уравнении для $\eta(x_1, x_2)$ и начнем искать уравнение регрессии в линейной форме [*см.* формулу (7.2)], так как полином нулевой степени явно не обеспечит адекватности:

$$\bar{y} = b_0 x_0 + b_1 x_1 + b_2 x_2,\ k = 2;$$

$$X = \begin{vmatrix} x_0 & x_1 & x_2 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 0 & -1 \\ 1 & -1 & -1 \end{vmatrix}.$$

Система точек неортогональна:

$$\sum_{u=1}^{9} x_0 x_1 = 5 \neq 0\,;$$

$$\sum_{u=1}^{9} x_0 x_2 = 4 \neq 0;$$

$$\sum_{u=1}^{9} x_1 x_2 = 6 \neq 0;$$

$$Y = \begin{vmatrix} 10 \\ 17 \\ 20 \\ 14 \\ 18 \\ 24 \\ 40 \\ 3 \\ 3 \end{vmatrix}; \quad X^* X = \{(ij)\} = \begin{vmatrix} 9 & 5 & 4 \\ 5 & 11 & 6 \\ 4 & 6 & 12 \end{vmatrix}; \quad X^* Y = \{(iy)\} = \begin{vmatrix} 149 \\ 151 \\ 155 \end{vmatrix}.$$

Определяем элементы обратной матрицы:

$$\left[X^* X\right]^{-1} = \left\{(c_{ij})\right\} = \frac{X^* X^{\Delta}}{\left|X^* X\right|},$$

где $X^* X^{\Delta}$ – присоединенная матрица для заданной квадратной матрицы $X^* X$.

Рассчитаем определитель

$$\left|X^* X\right| = 628.$$

Далее получим коэффициенты:

$$c_{11} = 11 \times 12 - 6 \times 6 = 96;$$

$$c_{12} = 36;$$

$$c_{13} = -14 \text{ и т.д.}$$

Определяем матричное выражение:

$$\left\{(c_{ij})\right\} = \left[X^* X\right]^{-1} = \frac{1}{628} \begin{vmatrix} 96 & -36 & -14 \\ -36 & 92 & -34 \\ -14 & -34 & 74 \end{vmatrix}.$$

С помощью равенства (7.6) вычисляем коэффициенты b_i:

$$\begin{cases} b_0 = (96 \cdot 149 - 36 \cdot 151 - 14 \cdot 155)\frac{1}{628} = 10{,}65; \\ b_1 = 5{,}2; \\ b_2 = 6{,}77. \end{cases}$$

Таким образом,

$$\bar{y} = 10{,}65 + 5{,}2x_1 + 6{,}77x_2 .$$

Промежуточные результаты вычисления дисперсии, которую дает уравнение регрессии, представлены в табл. 7.2.

Таблица 7.2

Промежуточные результаты вычисления дисперсии

u	1	2	3	4	5	6	7	8	9
$\bar{y}$	10,65	17,4	24,2	15,85	21	22,6	34,6	3,9	–2
y_u	10	17	20	14	18	24	40	3	3
Δy	0,65	–0,4	–4,2	–1,85	–3	1,4	5,4	–0,9	5
y_u^2	0,1	0,16	17,5	3,5	9	2	30	0,8	25

Получаем

$$S_R = \sum_{u=1}^{9} \Delta y_u^2 = 89 .$$

Степень свободы $f = 9 - 3 = 6$, а дисперсия и среднеквадратичное отклонение имеют следующие значения:

$$\sigma_y^2 = \frac{S_R}{f} = 14{,}8 ; \; \sigma_y \cong 3{,}8 .$$

Поскольку мы считаем, что нам известна ошибка опыта $\sigma_y^2 = 1{,}22$, можно определить на основе F-отношения адекватность полученного уравнения регрессии:

$$F = \frac{s_y^2}{\sigma_y^2} = \frac{14{,}8}{1{,}22} = 12{,}1.$$

Случайно ли, что s_y^2 в 12 раз превышает σ_y^2, или выбрана недостаточная степень полинома? Попытаемся ответить на этот вопрос: s_y^2 имеет $f_1 = 6$, а σ_y^2 имеет $f_2 = 9$ степеней свободы.

Для 10%-ного уровня значимости по таблице (*см.* приложение 3) найдем $F_т = 3{,}37$. Поскольку $12{,}1 > 3{,}37$, гипотезу о случайном отклонении нужно отбросить.

Найдем уравнение регрессии в виде неполного уравнения 2-й степени:

$$\overline{y} = b_0 x_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2 =$$
$$= b_0 x_0 + b_1 x_1 + b_2 x_2 + b_3 x_3,$$

где $x_3 = x_1 x_2$.

Матрица результатов наблюдений имеет вид:

$$X = \begin{vmatrix} x_0 & x_1 & x_2 & x_3 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 4 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & -1 & 1 \end{vmatrix}.$$

Выполнив промежуточные расчеты (объем выкладок значительно возрастет, особенно при определении обратной матрицы), получим

$$\overline{y} = 9{,}9 + 4{,}1 x_1 + 5{,}8 x_2 + 2{,}7 x_1 x_2; \; S_R = 9; \; f_1 = 9 - 4 = 5;$$

$$s_y^2 = \frac{S_R}{f} = \frac{9}{5} = 1{,}8; \; s_y = \sqrt{1{,}8} = 1{,}35; \; F_p = \frac{1{,}8}{1{,}22} = 1{,}47.$$

По таблице для q = 10% при $f_1 = 5$, $f_2 = 9$ найдем $F_т = 3{,}48$ (*см.* приложение 3).

Так как $F_р < F_т$, гипотеза об адекватности уравнения регрессии может быть принята.

Пример 7.3. Ортогональное планирование. Изменим систему точек, чтобы выполнялось условие ортогональности (табл. 7.3).

Таблица 7.3

Система точек с выполнением условия ортогональности (план 1-го порядка)

u	1	2	3	4	5	6	7	8	9
x_1	0	–2	2	0	0	1	–1	–1	–1
x_2	0	0	0	1	–1	1	–1	–1	1
$\eta(x_1, x_2)$	10	2	18	16	4	23	3	5	9
y_u	10	0	18	17	3	24	3	3	10
Δy_u	0	–2	0	1	–1	1	0	–2	1
y_u^2	0	4	0	1	1	1	0	4	1

Хотя число точек $N = 9$ сохранено, естественно, что условия задачи изменились: в четырех новых точках нужно задать какое-то новое значение шума.

Вычисляем:

$$\sigma_y^2 = \frac{1}{9}\sum_{u=1}^{9}\Delta y_u^2 = \frac{12}{9} = 1{,}34;\ \sigma_y = 1{,}16;$$

$$\sum_{u=1}^{9} x_{1u}x_{2u} = \sum_{u=1}^{9} x_{0u}x_{1u} = \sum_{u=1}^{9} x_{0u}x_{2u} = 0.$$

Зададим линейное уравнение регрессии:

$$\bar{y} = b_0 x_0 + b_1 x_1 + b_2 x_2;$$

$$X = \begin{vmatrix} x_0 & x_1 & x_2 \\ 1 & 0 & 0 \\ 1 & -2 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & -1 \\ 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \end{vmatrix};\quad Y = \begin{vmatrix} 10 \\ 0 \\ 18 \\ 17 \\ 3 \\ 24 \\ 3 \\ 3 \\ 10 \end{vmatrix};\quad X^*X = \begin{vmatrix} 9 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 6 \end{vmatrix};\quad X^*Y = \begin{vmatrix} 88 \\ 50 \\ 42 \end{vmatrix} = \{(iy)\}.$$

В данном случае не нужно искать обратную матрицу:

$$b_i = c_{ii}(iy),\ c_{ii} = \frac{1}{(ii)};$$

$$b_0 = c_{00}(0y) = \frac{1}{(00)}(0y) = \frac{1}{9} \cdot 88 = 9{,}8;$$

$$b_1 = c_{11}(1y) = \frac{1}{12} \cdot 50 = 4{,}17;$$

$$b_2 = c_{22}(2y) = \frac{1}{6} \cdot 42 = 7;$$

$$\overline{y} = 9{,}8 + 4{,}17x_1 + 7x_2.$$

Значение S_R (для разнообразия) подсчитаем по формуле (7.12):

$$S_R = (yy) - \sum_{i=0}^{2} b_i(iy);\ \sum_{u=1}^{9} y_u^2 = 1416;$$

$$\sum_{i=0}^{2} b_i(iy) = 9{,}8 \cdot 88 + 4{,}17 \cdot 50 + 7 \cdot 42 = 1363;\ S_{R_2} = 53\ (k = 3);$$

$$f = 9 - 3 = 6;\ s_y^2 = \frac{53}{6} = 8{,}85;\ s_y \approx 2{,}6.$$

В данном случае адекватность линейной гипотезы будет отброшена по критерию F.

Поскольку матрица X^*X в ортогональной системе диагональна, здесь можно применить t-распределение при определении доверительных интервалов для b_i. По формуле (7.16) получим:

$$s_{b_0}^2 = c_{00}s_y^2 = \frac{1}{9}8{,}85 = 0{,}98;\ s_{b_0} = 0{,}99 \approx 1;$$

$$s_{b_1}^2 = c_{11}s_y^2 = \frac{1}{12}8{,}85 = 0{,}74;\ s_{b_1} = 0{,}86 \approx 0{,}9;$$

$$s_{b_2}^2 = c_{22}s_y^2 = \frac{1}{6}8{,}85 = 1{,}47;\ s_{b_2} = 1{,}22 \approx 1{,}2.$$

Для 10%-ного уровня значимости и рассчитанного $f = 6$ по таблице (*см.* приложение 4) найдем q-процентный предел: $t_{10}(6) = 1{,}943 \approx 1{,}9$.

Таким образом, с достоверностью 0,9 можно утверждать, что истинные коэффициенты регрессии лежат в пределах (*см.* формулу (7.17)):

$$\begin{cases} 7{,}9 \le b_0 \le 11{,}7; \\ 2{,}5 \le b_1 \le 5{,}9; \\ 4{,}7 \le b_2 \le 9{,}3. \end{cases}$$

Все коэффициенты b_i значимы, т.е. ими нельзя пренебречь (приравнять нулю), так как $b_i > t_{10} s_{b_i}$. Найдем уравнение регрессии в виде

$$\overline{y} = b_0 x_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2 = b_0 x_0 + b_1 x_1 + b_2 x_2 + b_3 x_3,$$

где $x_3 = x_1 x_2$.

Матрица результатов наблюдений примет вид:

$$X = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{vmatrix}.$$

Остальные матрицы записываются следующим образом:

$$Y = \begin{vmatrix} 10 \\ 0 \\ 18 \\ 17 \\ 3 \\ 24 \\ 3 \\ 3 \\ 10 \end{vmatrix}; \quad X^* Y = \begin{vmatrix} 88 \\ 50 \\ 42 \\ [14] \end{vmatrix}; \quad X^* X = \begin{vmatrix} 9 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & [4] \end{vmatrix}.$$

Коэффициенты b_0, b_1, b_2 останутся теми же; требуется определить только b_3:

$$b_3 = \frac{14}{4} = 3{,}5;$$

Тогда уравнение регрессии имеет вид

$$\overline{y} = 9{,}8 + 4{,}17x_1 + 7x_2 + 3{,}5x_1x_2.$$

Значение S_R определяется с использованием выполненных выше расчетов:

$$S_{R_3} = S_{R_2} - b_3(3y) = 53 - 3{,}5 \cdot 14 = 4.$$

Дисперсия, среднеквадратичное отклонение и F-отношение определяются выражениями:

$$s_y^2 = \frac{S_R}{f} = \frac{4}{9-4} = 0{,}8; \quad s_y = 0{,}9; \quad F = \frac{\sigma_y^2}{s_y^2} = \frac{1{,}34}{0{,}8} \approx 1{,}7.$$

Для $q = 10\%$ и $f = 5$ найдем $t_{10}(5) = 2{,}015 \approx 2$.

Для коэффициентов уравнения регрессии справедливы следующие соотношения:

$$s_{b_0}^2 = \frac{0{,}9}{9} = 0{,}1; \quad s_{b_0} \approx 0{,}34; \quad 9{,}1 \le b_0 \le 10{,}5;$$

$$s_{b_1}^2 = \frac{0{,}9}{12} = 0{,}08; \quad s_{b_1} \approx 0{,}28; \quad 3{,}6 \le b_1 \le 4{,}7;$$

$$s_{b_2}^2 = \frac{0{,}9}{6} = 0{,}15; \quad s_{b_2} \approx 0{,}38; \quad 6{,}2 \le b_2 \le 7{,}8;$$

$$s_{b_3}^2 = \frac{0{,}9}{4} = 0{,}22; \quad s_{b_3} \approx 0{,}47; \quad 2{,}5 \le b_3 \le 4{,}5.$$

Итак, рассмотрены примеры применения регрессионного анализа для неортогонального и ортогонального планирования. Хочется подчеркнуть одну важную особенность ортогонального планирования, кроме очевидного значительного упрощения расчетов.

В неортогональном планировании при изменении степени полинома все вычисления проводятся заново, а значения всех коэффициентов регрессии изменяются. В ортогональном же планировании ранее вычисленные коэффициенты остаются без изменения.

Неопределенность в оценке коэффициентов регрессии при неортогональном планировании затрудняет их физическую интерпретацию; уравнение регрессии можно рассматривать только как интерполяционную формулу. В ортогональном же планировании можно придавать определенный физический смысл индивидуальным коэффициентам регрессии. В данном случае уравнение регрессии – не просто интерполяционная формула, а некоторая математическая модель процесса.

7.5 ФАКТОРНЫЙ ЭКСПЕРИМЕНТ И МЕТОД КРУТОГО ВОСХОЖДЕНИЯ

Одной из основных идей планирования эксперимента является выбор экспериментальных точек. Факторный эксперимент обеспечивает наиболее удобный для описания процесса выбор точек факторного пространства, при этом обеспечивается и свойство ортогональности. Факторный эксперимент применяется в тех случаях, когда неизвестная поверхность достаточно гладкая и не имеет многочисленных локальных экстремумов, например при определении зависимостей от различных факторов свойств химических и физических процессов. Факторный анализ используется и при обработке большого числа экономических данных, собранных органами государственной статистики, если исследуемые свойства экономического процесса достаточно гладко меняются при варьировании отдельных факторов.

При построении *полного факторного эксперимента* управляющие переменные x_i принимают только два возможных значения: +1 или –1. К такой схеме планирования можно свести любой эксперимент. Например, управляющими переменными процесса в химическом реакторе являются давление и температура. Несмотря на очень простое построение плана, полный факторный эксперимент имеет существенный недостаток: с ростом числа факторов k число опытов растет по показательной функции $N = 2k$.

Число опытов факторного эксперимента можно сократить, применяя так называемый *дробный факторный эксперимент* (дробные реплики от полного факторного эксперимента). Однако уменьшение числа опытов полного факторного эксперимента при сохранении всех его расчетных преимуществ может сопровождаться неприятным явлением взаимного влияния различных эффектов при необоснованном пренебрежении некоторыми взаимодействиями.

Основные *преимущества* и *возможности* факторного эксперимента:

1) очень просто производятся все вычисления;

2) можно получать математическое ожидание процесса как в форме линейного уравнения, так и с учетом взаимодействий;

3) все коэффициенты регрессии определяются независимо друг от друга, что дает некоторую возможность рассматривать уравнение регрессии как физическую модель процесса;

4) все коэффициенты уравнения регрессии определяются с одинаковой и минимальной дисперсией;

5) применение дробного факторного эксперимента и насыщенного планирования позволяет уменьшать число опытов полного факторного эксперимента;

6) имеется возможность исключать временной дрейф.

Рассмотрим метод крутого восхождения с применением факторного эксперимента. Определение оптимальных условий протекания экономических, химических, физических и металлургических процессов, или задача определения оптимального состава компонентов системы, всегда решалась чисто интуитивно. При попытке дать строго обоснованные методы решения этой задачи приходится сталкиваться с большими трудностями. Чтобы найти оптимум, нужно дать описание поверхности отклика в широком интервале варьирования независимых переменных. Адекватное описание таких больших участков поверхности требует очень большого числа опытов.

Для решения этой задачи используется последовательный, пошаговый метод изучения поверхности отклика. Исследователь вначале ставит серию опытов для описания небольшого участка поверхности отклика полиномом 1-го порядка. Далее он двигается по поверхности отклика в направлении градиента линейного приближения. Если одного линейного приближения оказывается недостаточно, то ставится новая небольшая серия опытов и находится новое направление для движения по поверхности отклика. Такой процесс продолжается до тех пор, пока исследователь не попадет в окрестность экстремума. Если требуется более точно определить положение оптимума, то ставится большая серия опытов, и поверхность отклика описывается полиномом 2-го, а иногда даже 3-го порядка. При таком подходе к задаче достигается весьма высокая концентрация опытов в той части поверхности отклика, которая преимущественно интересует исследователя.

Градиент функции отклика может быть задан выражением

$$\nabla y = \frac{\partial y}{\partial x_1}\bar{x}_1 + \frac{\partial y}{\partial x_2}\bar{x}_2 + \ldots + \frac{\partial y}{\partial x_k}\bar{x}_k,$$

где $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_k$ – единичные векторы в направлении осей $x_1, x_2, \ldots, x_k$ факторного пространства.

Движение в направлении градиента – это движение по кратчайшему, наиболее крутому пути; отсюда название «крутое восхождение» (если отыскивается максимум функции) или «наискорейший спуск» (минимум функции).

Здесь следует отметить несколько разновидностей движения по поверхности отклика. Если бывает затруднительно определить градиент, используют метод Гаусса–Зейделя. По этому методу производится поочередное изменение каждого параметра в направлении оптимума с помощью пробных шагов. Это относительно длинный путь к оптимуму. Сам метод градиента тоже имеет несколько разновидностей. Градиент может вычисляться на основе выполнения одного пробного шага по каждой переменной (для двух переменных будет использоваться одна центральная точка и две пробных).

Более точно градиент может быть вычислен, если известно линейное приближение поверхности отклика, полученное по числу точек, превышающему число переменных. Боксом и Уилсоном предложено определять градиент по линейному приближению поверхности отклика на основе дробного факторного эксперимента. Если градиент рассчитывается заново после каждого шага решения, то это метод градиента. Если же в направлении градиента выполняется несколько шагов до тех пор, пока не перестанем приближаться к оптимуму, то это метод крутого восхождения или наискорейшего спуска.

Рассмотрим метод крутого восхождения при определении градиента по линейному приближению поверхности отклика, полученному на основе факторного эксперимента. На рис. 7.6 нанесены кривые равного уровня поверхности отклика для двух независимых переменных. Если построить нормали к кривым равного уровня, то получим направления градиента. Движение из точки O в направлении OP – это наиболее крутой путь подъема по поверхности отклика. В направлении OP исследователь будет двигаться до тех пор, пока не перейдет точку Q. В окрестности точки Q надо будет поставить новую серию опытов и заново найти направление градиента (QM).

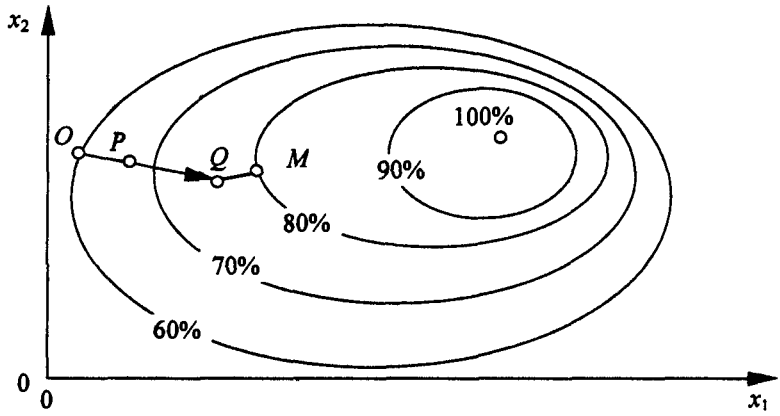


Рис. 7.6. Схема движения по градиенту

Если поверхность отклика локально может быть описана линейным уравнением, то частные производные, очевидно, будут равны коэффициентам уравнения регрессии

$$\frac{\partial y}{\partial x_i} = b_i.$$

В этом случае при движении по поверхности отклика в направлении крутого восхождения нужно будет независимые переменные изменять пропорционально величине соответствующих коэффициентов регрессии с учетом их знака. При постановке экспериментов всегда приходится переходить к натуральным переменным. В натуральных переменных величина шага должна быть пропорциональна произведению b_i на единицу варьирования.

Пример 7.4. Определение состава компонентов сплава. Рассмотрим задачу выбора оптимального состава сплава*. Требуется найти такое соотношение легирующих компонентов, чтобы сплав на основе железа имел максимальное сопротивление на разрыв при температуре 800°С. Железо легировали следующими элементами: Cr, Ni, Mo, V, Nb, Mn, C. Легирующие элементы и уровни варьирования задавались исходя из металловедческих и экономических соображений. Опустим для краткости процесс планирования движения по градиенту, и сразу отметим следующее.

* Кузин Л.Т., Плужников Л.Н., Белов Б.Н. Математические методы в экономике и организации производства. – М.: Изд-во МИФИ, 1968. (Пример взят из этой книги и перепроверен авторами с помощью модели.)

Лучший результат получен в 11-м опыте. Дальнейшее движение по направлению градиента линейного приближения приводит к понижению сопротивления на разрыв. Таким образом, был найден следующий состав стали: Cr 10,4%; Ni 1,2%; Mo 0,66%; V 0,18%; Nb 0,58%; Mn 0,24%; C 0,8% с сопротивлением на разрыв $y = 11{,}500$ т/см^2.

Эксперимент с семью факторами после всего 13 опытов позволил увеличить сопротивление на разрыв более чем в 2,5 раза (11,5/4,5). При обычных традиционных методах исследования решение такой задачи потребовало бы длительных и дорогостоящих усилий. Результаты работы были бы представлены не двумя таблицами, а громоздким отчетом, содержащим десятки таблиц и графиков. При желании полученный результат может быть уточнен, если повторить эксперимент, выбрав полученную точку оптимума в качестве исходной.

В рассмотренном примере имеется достаточно априорных сведений о процессе (состав переменных, величина шага, исходная точка). Если априорных сведений мало, то исследование методом крутого восхождения также эффективно, однако его приходится повторять многократно. Чем меньше данных априори, тем длительнее процесс поиска оптимума.

Основными *достоинствами* факторного эксперимента являются простота и возможность отыскания экстремальной точки (с какой-то погрешностью), если неизвестная поверхность достаточно гладкая и нет локальных экстремумов.

Основные *недостатки* факторного эксперимента:

1) невозможность поиска экстремума при наличии ступенчатых разрывов неизвестной поверхности или локальных эктремумов;

2) отсутствие средств описания характера поверхности вблизи экстремальной точки, так как используются простейшие линейные уравнения регрессии.

7.6 ОРТОГОНАЛЬНОЕ ПЛАНИРОВАНИЕ ВТОРОГО ПОРЯДКА: ПОИСК ЭКСТРЕМАЛЬНЫХ ТОЧЕК С ПОМОЩЬЮ МОДЕЛИ

В окрестности оптимума линейного приближения уже недостаточно, доминирующими становятся коэффициенты регрессии, характеризующие эффекты взаимодействия. Уменьшение шага при сохранении линейного приближения также неэффективно из-за

большого влияния помех. Обычно окрестность экстремума, которую называют почти стационарной областью, удается описать полиномами 2-го порядка. Для этого нужно иметь такую систему планирования, в которой каждая переменная будет принимать хотя бы три разных значения. Такое планирование может быть получено путем добавления некоторого количества специально расположенных точек к ядру, образованному планированием для линейного приближения. Такие планы называют композиционными, а само планирование – центральным композиционным планированием (ЦКП).

Рассмотрим случай, когда число независимых переменных $k = 2$ при полном факторном эксперименте, а количество результатов наблюдений $N = 4$. Нужно построить уравнение регрессии

$$\overline{y} = b_0 x_0 + b_1 x_1 + b_2 x_2 + b_{12} x_1 x_2 + b_{11} x_1^2 + b_{22} x_2^2.$$

Как минимум, необходимы еще две точки для определения всех шести коэффициентов регрессии. Для увеличения общего числа точек вводят так называемые звездные точки (рис. 7.7), по две для каждой переменной $(\pm\alpha, 0)$, $(0, \pm\alpha)$ и используют центральную точку (0,0). Общее число точек будет равно 9. Такой метод построения точек распространяется и на общий случай k точек. Общее число точек будет равно $N_{цкп} = 2^k + 2k + 1$ ($k = 3$, $N_{цкп} = 15$, вершины куба – 8 и 6 звездных точек). Если просто для каждой переменной задавать три уровня, то потребовалось бы 3^k точек. ЦКП требует меньшего числа опытов по сравнению с 3^k, особенно преимущество ЦКП проявляется с ростом k:

$$k = 3;\ 3^k = 27;\ N_{цкп} = 15;$$

$$k = 4;\ 3^k = 81;\ N_{цкп} = 25.$$

Возникает вопрос, как оптимальным образом выбрать величину плеча звездных точек. При линейном приближении, когда использовался факторный эксперимент, получалось ортогональное планирование, и дисперсии всех b_i были минимальны и равны друг другу. В этом состояла оптимальность планирования. Попытаемся добиться этих же условий для описания почти стационарной области.

Обеспечение ортогональности планирования. В общем случае матрица точек ЦКП не обеспечивает ортогональности всех векторов-столбцов. Так,

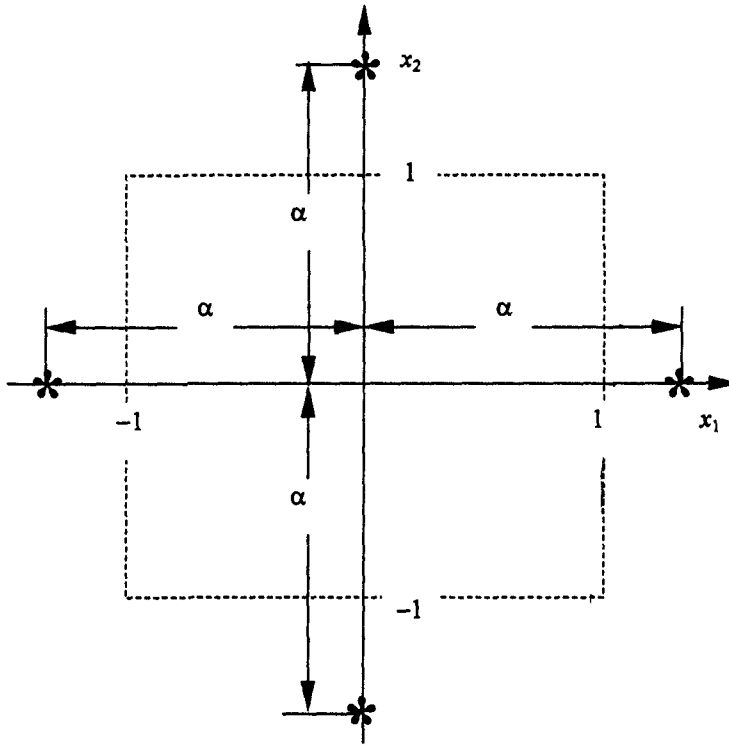


Рис. 7.7. Схема расположения «звездных» точек

$$\sum_{u=1}^{N} x_{0u} x_{iu}^2 \neq 0,$$

поскольку $x_{0u} \equiv 1$, и неотрицательные величины x_{iu}^2 не могут быть все равны нулю. Очевидно также, что

$$\sum_{u=1}^{N} x_{iu}^2 x_{ju}^2 \neq 0.$$

Чтобы получить ортогональность, нужно произвести некоторые преобразования квадратичных переменных и специальным образом выбрать величину звездного плеча α. Введем преобразование

$$x_i' = x_i^2 - \frac{1}{N} \sum_{u=1}^{N} x_{iu}^2 = x_i^2 - \overline{x}_i^2,$$

где $\overline{x}_i^2$ – среднее значение x_i^2.

Тогда будут равны нулю скалярные произведения

$$\sum_{u=1}^{N} x_{0u} x'_{iu} = \sum_{u=1}^{N} x_{iu}^2 - N\overline{x}_i^2 = 0.$$

Неортогональными останутся векторы-столбцы для квадратичных членов

$$\sum_{u=1}^{N} x'_{iu} x'_{ju} \neq 0. \qquad (7.18)$$

Если теперь составить матрицу $[X'X]$, то в ней ряд недиагональных элементов, а именно элементов для квадратичных переменных, не будет равен нулю (для ортогонального планирования только диагональные элементы будут иметь значение, не равное нулю). При этом, так как все звездные точки выражаются через один и тот же параметр α, все недиагональные элементы матрицы, не равные нулю, оказываются равны между собой. Поэтому, приравнивая нулю выражение для недиагонального элемента, однозначно получим значение α, которое обеспечит ортогональность планирования. Значения α в функции числа независимых переменных k представлены в табл. 7.4.

Таблица 7.4

Величина звездного плеча α

Число независимых переменных k	2	3	4	5
Ядро планирования	2^2	2^3	2^4	$x_5 = x_1 x_2 x_3 x_4$
Величина звездного плеча α	1,000	1,215	1,414	1,547

Рассмотрим пример матрицы ортогонального планирования 2-го порядка для случая $k = 3$:

$$\overline{x}_1^2 = \overline{x}_2^2 = \overline{x}_3^2 = \frac{1}{15}\sum_{u=1}^{15} x_{1u}^2 = \frac{1}{15}\left[8 + 2 \cdot 1{,}215^2\right] = 0{,}73.$$

Определим для звездных точек

$$x'_i = x_i^2 - \overline{x}_i^2 = 1{,}475 - 0{,}73 = 0{,}745;$$

$$x'_i = x_i^2 - \overline{x}_i^2 = 0 - 0{,}73 = -0{,}73.$$

Нетрудно убедиться, что все векторы-столбцы матрицы (табл. 7.5) ортогональны. Существующая до преобразований ортогональность не нарушилась, например

$$\sum_{u=1}^{15} x_{1u} x_{3u} = 0; \quad \sum_{u=1}^{15} x_{2u} x_{3u} = 0.$$

Таблица 7.5

Ортогональный план второго порядка

M	0	1	2	3	4	5	6	7	8	9	
	x_0	x_1	x_2	x_3	$x_1^2-0{,}73$	$x_2^2-0{,}73$	$x_3^2-0{,}73$	x_1x_2	x_1x_3	x_2x_3	y_u
Ядро 2^3	+	–	–	–	0,27	0,27	0,27	+	+	+	y_1
	+	+	–	–	0,27	0,27	0,27	–	–	+	y_2
	+	–	+	–	0,27	0,27	0,27	–	+	–	y_3
	+	–	–	+	0,27	0,27	0,27	+	–	–	y_4
	+	–	+	+	0,27	0,27	0,27	–	–	+	y_5
	+	+	–	+	0,27	0,27	0,27	–	+	–	y_6
	+	+	+	–	0,27	0,27	0,27	+	–	–	y_7
	+	+	+	+	0,27	0,27	0,27	+	+	+	y_8
Звездные точки	+	–1,215	0	0	0,745	–0,73	–0,73	0	0	0	y_9
	+	1,215	0	0	0,745	–0,73	–0,73	0	0	0	y_{10}
	+	0	–1,215	0	–0,73	0,745	–0,73	0	0	0	y_{11}
	+	0	1,215	0	–0,73	0,745	–0,73	0	0	0	y_{12}
	+	0	0	–1,215	–0,73	–0,73	0,745	0	0	0	y_{13}
	+	0	0	1,215	–0,73	–0,73	0,745	0	0	0	y_{14}
Нулевая точка	+	0	0	0	–0,73	–0,73	–0,73	0	0	0	y_{15}

Видно, что удовлетворяется условие (7.3):

$$\sum_{u=1}^{15} x_{0u}(x_{1u}^2 - 0{,}73) = 8 \cdot 0{,}27 + 2 \cdot 0{,}745 - 5 \cdot 0{,}73 = 2{,}16 + 1{,}49 - 3{,}65 = 0.$$

Превратились в равенства выражения (7.18):

$$\sum_{u=1}^{15} \left(x_{1u}^2 - 0{,}73\right) \cdot \left(x_{2u}^2 - 0{,}73\right) = 8 \cdot 0{,}27^2 - 4 \cdot 0{,}745 \cdot 0{,}73 + 3 \cdot 0{,}73^2 = 0.$$

Регрессионный анализ ортогонального планирования второго порядка. После проведения эксперимента в силу ортогональности планирования все коэффициенты регрессии определяются независимо друг от друга.

Так как

$$c(mm) = \frac{1}{(mm)} = \frac{1}{\sum_{u=1}^{N} x_{mu}^2},$$

то из формулы (7.7) получаем

$$b_m = \frac{\sum\limits_{u=1}^{N} x_{mu} y_u}{\sum\limits_{u=1}^{N} x_{mu}^2},$$

где m – порядковый номер столбца;
x_{mu} – элементы соответствующего столбца.

Определяем дисперсию

$$s_{b_m}^2 = \frac{s_y^2}{\sum\limits_{u=1}^{n} x_{mu}^2}. \qquad (7.19)$$

Для факторного эксперимента мы бы имели

$$\sum_{u=1}^{n} x_{mu}^2 = N$$

для любого m. В данном случае

$$\sum_{u=1}^{N} x_{mu}^2 \neq N.$$

Например, для $k = 3$ основные соотношения примут следующий вид:

$$\sum_{u=1}^{N} x_{0u}^2 = N; \qquad (7.20)$$

$$\sum_{u=1}^{N} x_{mu}^2 = 2^k + 2\alpha^2 = 10{,}95 \text{ при } m = 1, 2, 3; \qquad (7.21)$$

$$\sum_{u=1}^{N} x_{mu}^2 = \sum_{u=1}^{N} \left(x_{iu}^2 - \overline{x}_i^2\right)^2 = 2^k\left(1 - \overline{x}_i^2\right) + 2\left(\alpha^2 - \overline{x}_i^2\right) + (2k-1)\overline{x}_i^2 = 4{,}36$$

при $m = 4, 5, 6$; (7.22)

$$\sum_{u=1}^{N} x_{mu}^2 = \sum_{u=1}^{N} \left(x_{iu} x_{ju}\right)^2 = 2^k = 8 \qquad (7.23)$$

при $m = 7, 8, 9$.

По выражениям (7.20) – (7.23) может быть подсчитана сумма $\sum_{u=1}^{N} x_{mu}^2$ для любого k, если известно значение α. Таким образом, вычисление b_m и s_{b_m} для ортогонального планирования незначительно сложнее, чем при факторном эксперименте.

Уравнение регрессии после преобразования квадратичной переменной запишется в форме

$$\bar{y} = b_0' + \sum_{i=1}^{k} b_i x_i + \sum_{i \neq j} b_{ij} x_i x_j + \sum_{i=1}^{k} b_{ii}\left(x_i^2 - \bar{x}_i^2\right),$$

для обычной формы записи –

$$\bar{y} = b_0 + \sum_{i=1}^{k} b_i x_i + \sum_{i \neq j} b_{ij} x_i x_j + \sum_{i=1}^{k} b_{ii} x_i^2,$$

Очевидно,

$$b_0 = b_0' - \sum_{i=1}^{k} b_{ii} \bar{x}_i^2.$$

При этом коэффициент b_0 оценивается с дисперсией

$$s_{b_0}^2 = s_{b_0'}^2 + \sum_{i=1}^{k} \left(\bar{x}_i^2\right) s_{b_{ii}}^2. \qquad (7.24)$$

Выражение (7.24) следует из известной теоремы теории вероятностей (закон накопления ошибок):

$$D\left[\sum_{i=1}^{k} \alpha_i x_i\right] = \sum_{i=1}^{k} \alpha_i^2 D[x_i].$$

Необходимо отметить, что ортогональное планирование не обеспечивает равенства дисперсий b_i (*см.* формулу (7.19)).

Выводы

1. Ортогональное планирование эксперимента (по сравнению с неортогональным) уменьшает число опытов и существенно упрощает расчеты при получении уравнения регрессии. Однако такое планирование осуществимо только при возможности проведения активного эксперимента.

2. Практичным средством отыскания экстремума является факторный эксперимент. Основные *достоинства* факторного эксперимента – простота и возможность отыскания экстремальной точки (с какой-то погрешностью), если неизвестная поверхность достаточно гладкая и нет локальных экстремумов. Следует отметить два основных *недостатка* факторного эксперимента. Первый заключается в невозможности поиска экстремума при наличии ступенчатых разрывов неизвестной поверхности и локальных экстремумов. Второй – в отсутствии средств описания характера поверхности вблизи экстремальной точки из-за использования простейших линейных уравнений регрессии, что сказывается на инертности системы управления, так как в процессе управления необходимо проводить факторные эксперименты для выбора управляющих воздействий.

3. Для целей управления наиболее подходит ортогональное планирование второго порядка. Обычно эксперимент состоит из двух этапов. Сначала с помощью факторного эксперимента отыскивается область, где существует экстремальная точка. Затем в районе существования экстремальной точки проводится эксперимент для получения уравнения регрессии 2-го порядка.

Уравнение регрессии 2-го порядка позволяет сразу определять управляющие воздействия, без проведения дополнительных опытов или экспериментов. Дополнительный эксперимент потребуется только в случаях, когда поверхность отклика существенно изменится под воздействием неконтролируемых внешних факторов (например, существенное изменение налоговой политики в стране серьезным образом повлияет на поверхность отклика, отображающую производственные затраты предприятия).

Вопросы для самопроверки

1. Какие статистические результаты позволяет получить имитационная модель, реализованная в любой системе моделирования (например, Pilgrim, ReThink или GPSS)?
2. В чем заключается кибернетический подход к организации экспериментальных исследований сложных объектов и процессов?
3. Какие характерные признаки имеет пассивный эксперимент?
4. В каких случаях проводится активный эксперимент?
5. Что такое функция (поверхность) отклика? Как она связана с факторным пространством?

6. Как представляется общий вид уравнения регрессии, полученного на основе опыта?
7. Каким образом вычисляются коэффициенты регрессии?
8. Как проводится статистический анализ уравнения регрессии?
9. Для чего применяется *F*-распределение?
10. Где применяется *t*-распределение?
11. Как осуществляется ортогональное планирование эксперимента?
12. Для чего определяются доверительные границы и при каких условиях их можно установить?
13. В чем заключается основное положительное свойство ортогонального плана эксперимента?
14. В чем состоит основной недостаток неортогонального планирования?
15. Для каких целей применяется факторный эксперимент (указать его достоинства и недостатки)?
16. Какие достоинства и недостатки имеет метод крутого восхождения?
17. В чем состоят отличия между полным и дробным факторными экспериментами?
18. Для чего необходимо ортогональное планирование второго порядка при проведении экстремального эксперимента?
19. Как осуществляется центральное композиционное планирование?
20. Каким образом можно обеспечить ортогональное планирование второго порядка?
21. Что собой представляет ядро планирования?
22. Как определяется величина «звездного плеча»?
23. Как выполняется регрессионный анализ при ортогональном планировании второго порядка?
24. Что такое «звездные точки» в ортогональном плане второго порядка?
25. Чем характеризуется нулевая точка ортогонального плана второго порядка?

Глава 8

ПРОБЛЕМНО-ОРИЕНТИРОВАННЫЕ ИМИТАЦИОННЫЕ МОДЕЛИ

8.1 МОДЕЛЬ «ПОСЕЩЕНИЕ ПУНКТОВ МЕСТНОСТИ КОММИВОЯЖЕРОМ»

Особенности решения задачи коммивояжера рассмотрены в главе 6. Предположим, что поставлена следующая задача: необходимо посетить все города, показанные на рис. 6.6, по кратчайшему маршруту, причем каждый город можно посетить только один раз. Коммивояжер находится в Санкт-Петербурге и имеет в своем распоряжении вертолет. Рассмотрим три варианта решения такой задачи.

Вариант 1. Коммивояжер, пользуясь только картой, строит маршрут по алгоритму «ближайшего непосещенного города». Чтобы не запутаться в большом числе частных маршрутов, он применяет сканирование на определенной широте. В данном случае он выполняет такие действия:

1) из Санкт-Петербурга проводит маршрут в западном направлении (ближайший город – Рига);

2) поскольку на данной широте не нашлось ближайшего города западнее, то далее маршрут строится южнее в восточном направлении (ближайший город – Москва);

3) так как на данной широте не нашлось ближайшего города восточнее, далее маршрут строится южнее в западном направлении (ближайшая оптимальная последовательность городов – Плавск, Вильнюс, Балтийск);

4) поскольку на данной широте не нашлось ближайшего города западнее, то далее маршрут строится южнее в восточном направлении (ближайшая оптимальная последовательность – Минск, Яровщина, Клинцы);

5) ввиду того что на данной широте не нашлось ближайшего города восточнее, далее маршрут строится южнее в западном направлении (ближайшая оптимальная последовательность – Новозыбков, Хойники, Пинск);

6) остался последний непосещенный город южнее – Киев; туда проводится последний отрезок маршрута.

Построенный маршрут (рис. 8.1) выглядит практически без изъянов: он действительно короткий, на нем нет петель (т.е. выполнено необходимое условие построения оптимального маршрута). Далее измеряем протяженность.

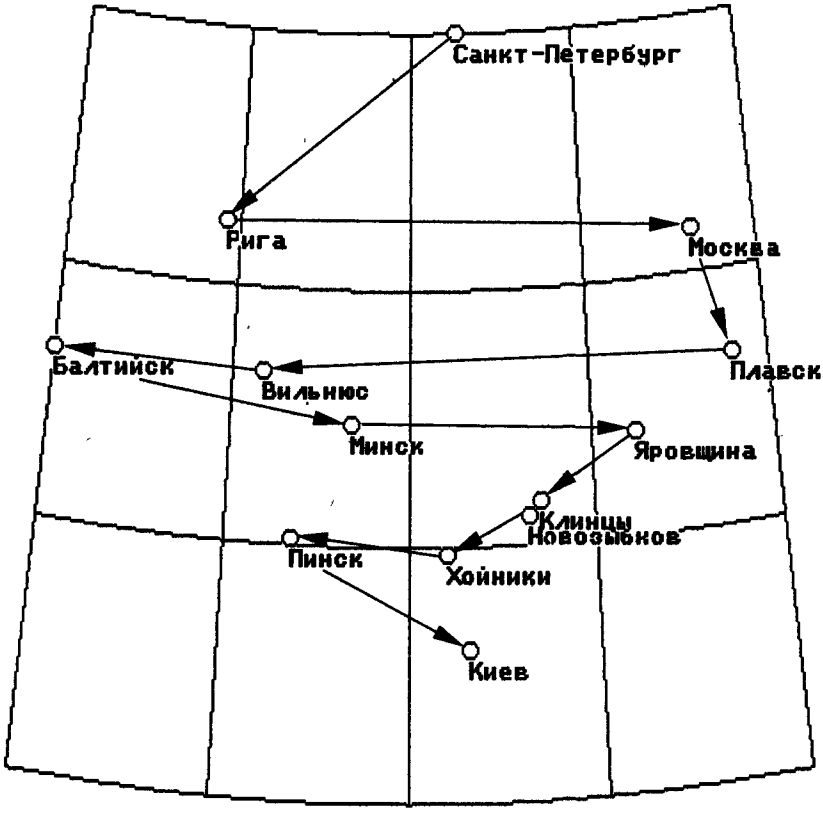


Рис. 8.1. Пример использования алгоритма «ближайшего непосещенного города»

Решение задачи: длина маршрута равна 4526,2 км.

Вариант 2. Введем более жесткие условия:

- пункт старта – Санкт-Петербург;
- пункт финиша – Москва.

В данном случае алгоритм «ближайшего непосещенного города» применить не удается: маршрут получится длиннее по сравнению с вариантом 1. Методы линейного и динамического программирования, метод «ветвей и границ» дают не лучшие решения при таком на первый взгляд несущественном усложнении задачи. Поэтому используем алгоритм «двух вертолетов». В результате создается маршрутное расписание, которое было показано в табл. 6.2. Сам маршрут (*см.* рис. 6.8) имеет два характерных участка, которые не позволяют его считать оптимальным, причем имеется петля (не выполнено необходимое условие оптимальности маршрута). Далее измеряем протяженность.

Решение задачи: длина маршрута равна 3338,2 км. На первый взгляд, это решение парадоксально: маршрут получился короче на 1188,0 км по сравнению с первым вариантом.

Но объяснение такого эффекта простое: при большом числе населенных пунктов алгоритм «двух вертолетов» всегда дает решение, лучшее по сравнению с алгоритмом «ближайшего непосещенного города» по следующей причине: оптимизация осуществляется двунаправленным просмотром. Алгоритм «ближайшего непосещенного города» начинает запутывать маршрут и при большом числе пунктов «уводить» его от оптимального, так как в конец маршрута заглянуть невозможно. Если иметь средства автоматического отображения маршрута (их предоставляет Pilgrim), то неоптимальные участки сразу видны (*см.* рис. 6.8). Поэтому возможна корректировка маршрута вручную.

Вариант 3. Используются условия варианта 2. Применяется алгоритм «двух вертолетов». Разрешена ручная корректировка неоптимальных участков. В результате получаем оптимальное маршрутное расписание (*см.* табл. 6.3). Далее измеряем протяженность.

Решение задачи: длина оптимального маршрута равна 2807,0 км. Это на 531,2 км короче по сравнению с вариантом 2 (но без варианта 2 этот маршрут нельзя было бы построить).

Далее необходимо разработать имитационную модель, которая должна обладать следующими свойствами:

- маршрутное расписание представляется в виде массива space, количество элементов которого – число населенных пунктов, а номер элемента массива – порядок посещения этого пункта;

• модель должна автоматически измерять длину маршрута (км) и время в пути (час).

Такая модель позволяет сравнивать эффективность различных алгоритмов.

Структура модели показана на рис. 8.2. Узел ag 1 «Генератор вызовов» поставляет транзакты в модель. Очередь 2 «Пункты вызовов» привязывает каждый транзакт к номеру пункта: если номер транзакта i, то он получает координаты из элемента i массива space, $i = 1, 2, \ldots, M$. Функцию накопления эта очередь не выполняет.

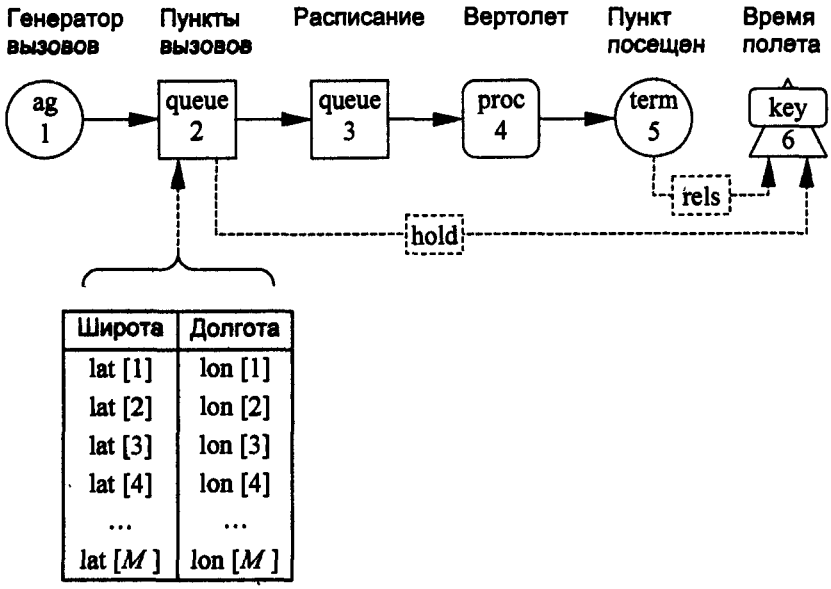


Рис. 8.2. Схема моделирования маршрутного расписания коммивояжера

Маршрутное расписание в виде упорядоченных транзактов образуется в узле 3 «Расписание»; это очередь на входе узла типа proc. Транзакты поступают на вход узла proc 4 «Вертолет», который имитирует полет с очередному населенному пункту. По завершении полета к каждому пункту соответствующий транзакт удаляется из узла proc, а далее и из модели (терминатор 5 «Пункт посещен»). Узел proc автоматически измеряет пройденный путь.

Время полета фиксируется с помощью клапана key (узел 6 «Время полета»). Этот клапан закрывается в начале имитации полета (операция hold из узла 2 queue при прохождении первого транзакта), а открывается в конце полета (операция rels из узла 5 term при удалении транзакта с номером *M*).

Рассмотрим текст модели в терминах Pilgrim:

```
#include <Pilgrim.h>
forward
   {
   int M=npoints; // Максимальное число пунктов
   int X=0;       // Текущая точка X - номер пункта
   modbeg("Воздушные рейсы",6,
          50.00,(long)time(NULL),earth,2,4,5,2);
   ag("Вызовы",1,none,expo,0.02,zero,zero,2);
   network(dummy, dummy)
      {
      top(2): t→iu0 = 3;
              X = 1 + addr[2]→na;
              f(X <= M)
                 sewt(X);
              else
                 t→iu0 = 5;
              if(addr[2]→na == 0)
                 hold(6);
              queue("Пункты вызовов", none, t→iu0);
              place;
      top(3): queue("Расписание",none,4);
              place;
      top(4): proc ("Вертолет",dummy,1,earth,
                    zero,20.0,zero,5);
              place;
      top(5): if(t→tx >= M)
                 rels(6);
              term ("Пункт посещен");
              place;
      top(6): key("Время полета", 5);
              place;
              fault(123);
      }
   modend ("Helicopter.doc",1,12,page);
   return 0;
   }
```

Сделаем следующие пояснения к этому тексту:

1) общее число населенных пунктов содержится в глобальной переменной Pilgrim – npoints;

2) параметр транзакта t→iu0 используется для изменения маршрутов транзактов, если их номера превысят номер *M*;

3) в узле 2 выполняется привязка транзакта к населенному пункту, номер которого равен 1+addr[2]→na, с помощью операции sewt;

4) в узле № 6 по завершении полета к последнему пункту измеряется время полета вертолета;

5) параметр транзакта t→tx содержит номер пункта, который посещает вертолет.

8.2 МОДЕЛЬ «СТОЯНКА МАРШРУТНОГО ТАКСИ»

Задача взаимного исключения возникает в следующих случаях:

- при создании сложных программных мультипроцессных средств;
- для моделирования логики экономического (логистического) процесса.

Ниже рассмотрено практическое решение такой задачи средствами Pilgrim на примере имитационной модели «Стоянка маршрутного такси».

Имеется остановка маршрутного такси. С определенными интервалами времени на остановку приходят пассажиры и подъезжают машины такси. Такси уезжает, когда в машину сядут 10 пассажиров. Если пришедший пассажир не обнаруживает стоящей машины такси, то он встает в очередь на посадку. Когда такси подъезжает на пустую остановку (нет ни одного человека), машина ждет пассажиров. Если такси подъехало, а на остановке уже стоит другая машина, которая пришла раньше, то вновь пришедшая машина встает в очередь.

Принцип работы узла delet: вошедший в него транзакт семейства «такси» становится транзактом семейства «пассажиры», входящие следом. Как только в узел войдет заданное количество транзактов-пассажиров, они поглотятся, а транзакт «тележка» переходит в следующий узел; транзакты, собранные в нее, перестают существовать как заявки на обслуживание в системе. Параметры модели представлены в табл. 8.1.

Опишем работу модели. Генератор 7 (рис. 8.3) создает транзакты, имитирующие пассажиров. Интервал прихода пассажиров в соответствии с теоремой о суперпозиции потоков событий имеет экс-

Таблица 8.1

Параметры имитационной модели

№ п/п	Параметр	Среднее значение (минуты)
1	Время моделирования: 10 ч	600,0
2	Интервал прихода пассажиров	1,0
3	Интервал приезда такси	10,0

поненциальное распределение. Интервал генерации транзактов, имитирующих такси, имеет нормальное распределение (обоснование: маршрут такси состоит из множества отрезков, поэтому имеют место условия центральной предельной теоремы, даже если на маршруте несколько машин). Использование в описании генератора 8 нормального закона распределения интервала генерации означает, что время между приездами такси на остановку чаще оказывается ближе к своему среднему значению и реже – дальше от него (чем больше отклонение интервала от среднего, тем реже это бывает).

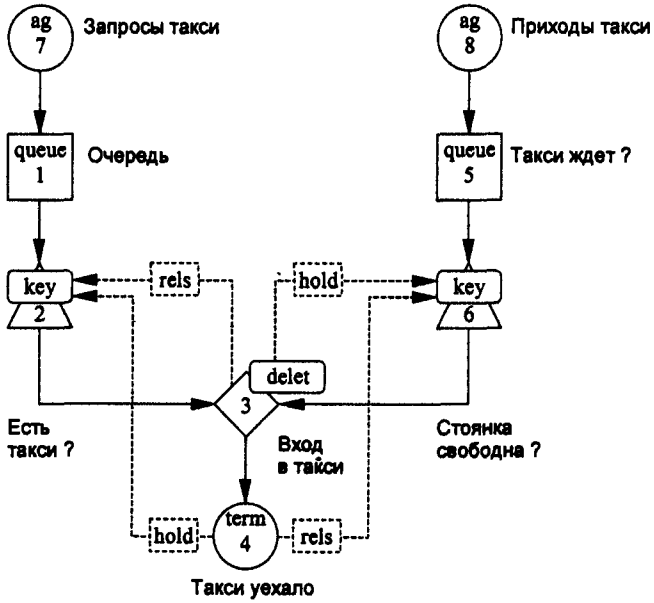


Рис. 8.3. Схема модели взаимного исключения и координации процессов типа «производитель-потребитель»

Узлы 1 и 5 имитируют соответственно очереди пассажиров и такси. Ключ 6 в начале работы модели находится в открытом состоянии (по умолчанию), а ключ 2 закрывается при приходе в очередь 1 первого транзакта-пассажира. Это делается для того, чтобы в узел delet первым вошел транзакт-такси, а не пассажир (иначе пассажир станет «тележкой»).

Как только в delet войдет первый транзакт-такси, ключ 6 закрывается, а ключ 2 одновременно с ним открывается. Теперь выходящие из генератора 6 транзакты-такси стоят в очереди 5, а транзакты-пассажиры заходят в узел delet (идет заполнение такси). Когда в узле delet накопятся 10 транзактов-пассажиров, транзакт-такси перейдет в терминатор (заполненное такси уедет). После этого ключ 6 открывается, чтобы в delet мог зайти следующий транзакт-такси, а ключ 2 в это же время закрывается, чтобы транзакты-пассажиры стояли в очереди 1 до прихода в delet транзакта-такси.

Таким образом, ключи 2 и 6 в модели всегда находятся в противоположных состояниях: если один открыт, то другой закрыт. Тем самым чередуется доступ к узлу delet разных типов транзактов – такси и пассажиров.

Рассмотренный принцип «тележки» позволил эффективно решить задачу взаимного исключения и координации процессов «производитель-потребитель».

Текст модели в комментариях не нуждается:

```
#include <Pilgrim.h>              // Модель
char commtext[81]="АВТОизвозчик и Ко";   // Фирма
char lefttext[81]="Минуты";    //Время в минутах
float win01      =   600.0;    // Моделируем 10 час
float win02      =     1.0;    // Приходы пассажиров
float win03      =     8.0;    // Поступление такси
char rightext[81]="Рабочий день"; // Интервал
float win11      =     0.0;    // Резерв (окно 11)
forward
   {
   Parametr;                   // Диалог настройки
   modbeg("Стоянка такси",8,
          win01,(long)time(NULL),none,1,none,4,2);
   ag("Запросы такси",7,none,expo,win02,zero,
      zero,1);
   ag("Приходы такси",8,none,norm,win03,win03/3,
      zero, 5 );
   network(dummy,dummy)
      {
      top(1): t→ft=1;
              queue("Очередь",none,2);
              place;
```

```
    top(2): if(addr[3]→tn == 1) // Такси подано?
               rels(2);          // Да: посадка
            else
               hold(2);          // Нет: ожидаем
            key("Есть такси?", 3);
            place;
    top(3): delet("Вход в такси",1,1,0,10,4);
            rels(2);
            hold(6);
            place;
    top(4): term("Такси уехало");
            hold(2);
            rels(6);
            place;
    top(5): queue("Такси ждет",none,6);
            place;
    top(6): key("Стоянка своб.?", 3);
            place;
    fault(123);
    }
modend("Result_taxi.doc",1,12,page);
return 0;
}
```

Нетрудно заметить, что в модели нет ни одного узла обслуживания (узлов типа serv или proc с ненулевым временем обслуживания), но в очередях могут возникать существенные задержки.

8.3 МОДЕЛЬ «ЭФФЕКТИВНОСТЬ КОМПЬЮТЕРОВ В АВТОМАТИЗИРОВАННОЙ БУХГАЛТЕРИИ»

Рассмотрим в несколько упрощенном изложении процесс обработки бухгалтерской информации в фирме с помощью экономической информационной системы, работающей в бухгалтерии. Такие системы часто называются автоматизированным рабочим местом (АРМ). Далее попытаемся создать структурную схему АРМ бухгалтерии и имитационную модель, соответствующую этой схеме.

В бухгалтерии выделим три подразделения (рис. 8.4):

- группа бухгалтеров в рабочей комнате бухгалтерии;
- архив, где документы, обработанные на компьютере, подшиваются в архивные папки;
- компьютерное подразделение, работающее в компьютерном зале бухгалтерии.

В рабочей комнате N бухгалтеров ($N \geq 1$) готовят независимо друг от друга различные документы, которые направляются в компьютерный зал для дальнейшей подготовки или обработки. В этом зале имеется один столик, на котором каждый документ находится до того, как оператор возьмет его на обработку. На таком столике может возникнуть очередь документов, ожидающих обработки.

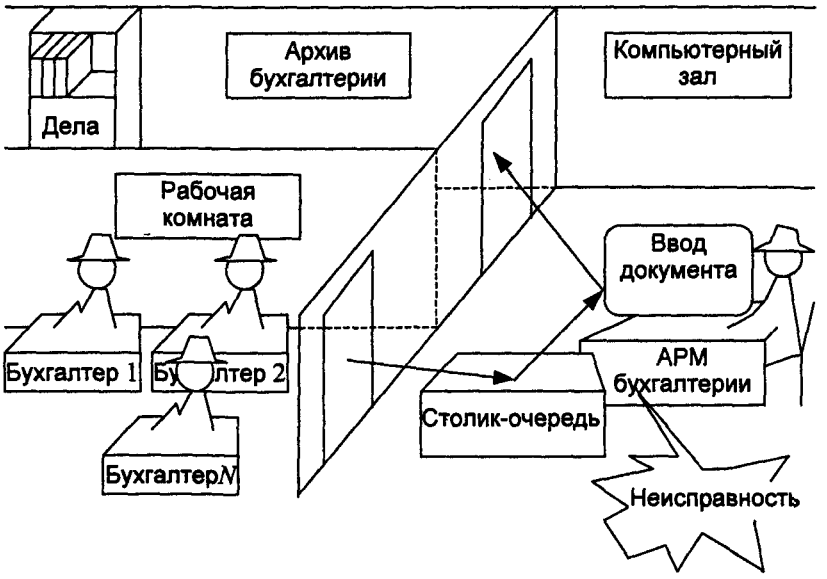


Рис. 8.4. Иллюстрация потока бухгалтерских документов

Обработка выполняется оператором, находящимся за компьютером, в течение какого-то интервала времени. Если в компьютерном зале несколько компьютеров, объединенных в сеть, то за каждым компьютером должен быть свой оператор. Общее число компьютеров M ($M \geq 1$). Обработанные (или распечатанные) на компьютере документы поступают в архив. Состав сотрудников архива нас не интересует.

Обработка документов затруднена по причине возникновения неисправностей в компьютере. Поэтому в компьютерном зале следующий состав сотрудников:

- один оператор за каждым компьютером;

• один наладчик, который приводит компьютеры в рабочее состояние после возникновения неисправностей и делает ежедневную профилактику каждого компьютера в конце каждого рабочего дня.

В качестве неисправности будем рассматривать любое событие, приводящее компьютер в нерабочее состояние в течение рабочего дня: выход из строя микросхемы, необходимость замены картриджа в печатающем устройстве, неисправность монитора, клавиатуры и ряд других. В любом случае после возникновения неисправности обработка документа с номером k прерывается, после чего вместо оператора за компьютером находится наладчик, который в течение какого-то интервала времени приводит компьютер в рабочее состояние.

После ремонта компьютера оператор продолжает обработку документа с номером k (полагаем, что программное обеспечение позволяет не начинать обработку документа заново).

При организации работы такой экономической информационной системы возникают два вопроса:

1) сколько компьютеров необходимо для того, чтобы справиться с потоком документов в бухгалтерии ?

2) сколько времени будет находиться документ, поступивший из рабочей комнаты, на столике в компьютерном зале, прежде чем его возьмет оператор на обработку?

Рассмотрим три основные причины, по которым компьютеры не смогут справиться с обработкой документов. Введем обозначения:

t_a – средний интервал времени между двумя последовательными поступлениями документов из двери рабочей комнаты на столик в компьютерном зале;

t_s – средний интервал времени обработки документа оператором на компьютере.

Первая причина, по которой компьютеры не справятся с обработкой потока документов, это высокая интенсивность потока:

$$t_a \le t_s / M.$$

Будем предполагать, что владелец фирмы достаточно хорошо владеет арифметическими расчетами, чтобы выбрать столько компьютеров, сколько необходимо для обработки потока.

Вторая причина заключается в том, что даже при выполнении соотношения

$$t_a > t_s / M$$

интервал поступления и интервал обработки – это случайные величины, которые имеют значительные отклонения от средних значений.

Третья причина – это ремонт компьютера в связи с неисправностями, возникающими в случайные моменты времени. Необходимый ремонт увеличивает разброс времени обработки документов и является дополнительной причиной снижения пропускной способности компьютера.

Если допустить, что в зале один компьютер и количество бухгалтеров велико (т.е. $N >> 1$), то можно считать поток документов простейшим, интервал поступления – случайной величиной, распределенной по экспоненциальному закону, а интервал обслуживания – случайной величиной, распределенной по любому закону со среднеквадратичным отклонением, обозначенным через σ.

При таких допущениях можно применить точную формулу Полячека–Хинчина и оценить среднее время задержки документа на столике t_q. Следует отметить, что при произвольном числе компьютеров $N > 1$ также можно найти аналитическое решение для определения времени t_q.

В нашем случае применение методов исследования операций становится невозможными по двум причинам: во-первых, число бухгалтеров не может быть большим и, во-вторых, присутствует фактор неисправности. Если попытаться создать математическую модель с помощью, например, аппарата вложенных цепей Маркова (метод Кендалла) или аппарата полумарковских процессов, то придется ввести большое число допущений, которые сделают погрешность метода при определении t_q крайне большой (буквально «плюс-минус в несколько раз»). Обе отмеченные причины приводят к тому, что от построения математической модели приходится отказаться. Поэтому будем применять метод имитационного моделирования.

В аспекте системного анализа можно представить модель «АРМ бухгалтерии» в виде «черного ящика» и выделить параметры четырех потоков (два входных и два выходных):

- входного потока документов;
- входного потока неисправностей;
- обработанного потока документов;
- потока устраненных неисправностей.

Структурный анализ данной системы довольно просто закончить уже на втором слое. Рассмотрим возможную схему модели (рис. 8.5).

Каждый бухгалтерский документ, исходящий из рабочей комнаты, будем считать неприоритетным транзактом. Тогда все бухгалтеры – это один генератор транзактов бесконечной емкости (узел 1 модели ag).

Представим неисправности в виде особых транзактов, которые могут занять компьютер, прервав обработку документа – неприоритетного транзакта. Для этого необходимо ввести в рассмотрение второй генератор транзактов бесконечной емкости (узел 2 модели ag).

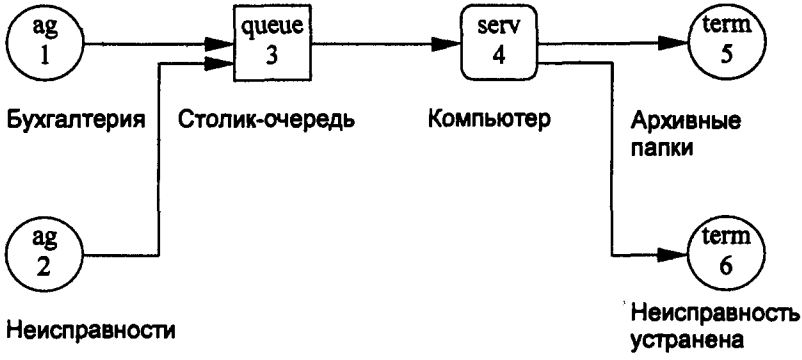


Рис. 8.5. Структурная схема модели «АРМ бухгалтерии»

Столик-очередь представим в виде узла типа очередь (узел 3 модели queue). Будем считать, что документы забираются со столика в хронологическом порядке по принципу «первым пришел – первым обслужен», т.е. оператор берет нижний документ, если вновь поступающие документы помещаются сверху на пачку необработанных документов.

Документы обрабатываются на одном (или нескольких) компьютере. Процесс обработки представим в виде узла – обслуживающего прибора (узел 4 модели serv). Обслуживающий прибор имеет M независимых каналов обслуживания ($M \geq 1$). Логично допустить, что каждый канал – это компьютер со своим оператором, обрабатывающим документы. Канал занят, если соответствующий оператор взял документ и обрабатывает его.

Архив представим в виде особого узла – терминатора, в который транзакты входят и в нем остаются (узел 5 модели term).

Рассмотрим путь транзакта-неисправности. Такой нематериальный транзакт выходит из второго генератора. Этот транзакт должен обладать абсолютным приоритетом по сравнению с транзактом-документом. Если на столике находятся несколько документов, то неисправность их «отодвигает», проникает в компьютер, и если на АРМ проходила обработка документа, то транзакт-неисправность

прерывает эту обработку. Прерванный транзакт-документ после этого попадает в специальный накопитель (стек), где находится до тех пор, пока не будет отремонтирован компьютер.

По завершении ремонта транзакт-неисправность будет направлен в особый узел модели – терминатор (узел 6 модели term), в котором он уничтожается, а прерванный транзакт-документ возвращается из стека, и обработка его будет продолжена.

Средний интервал времени между двумя последовательными неисправностями – это случайная величина со средним значением t_e. Транзакт-неисправность должен попасть на столик-очередь, отодвинуть все транзакты-документы, если они есть в очереди, войти в канал обслуживающего прибора и вытеснить из него транзакт-документ в особое хранилище. Наладчик должен заняться ремонтом соответствующего канала-компьютера. Через некоторое время, которое является случайной величиной со средним значением t_r, наладчик починит компьютер. Это означает, что транзакт-неисправность должен покинуть канал обслуживающего прибора и уйти в особый узел-терминатор (узел 6 модели term), так как в архив такой транзакт, естественно, поступать не может. В этот же момент времени транзакт-документ, обработка которого была прервана из-за неисправности, извлекается из хранилища, и обработка его продолжается.

В соответствии с предельной теоремой о суперпозиции потоков событий, а также учитывая, что в компьютере может быть несколько сотен независимых потоков элементарных неисправностей, будем считать, что интервал между двумя последовательными неисправностями распределен по экспоненциальному закону (значение соответствующего параметра – expo).

Если допустить, что в бухгалтерии работает не менее трех сотрудников, которые могут подготавливать информацию для обработки, и характер их работы приблизительно одинаков, то в качестве закона распределения интервала поступления документов выбираем экспоненциальный закон (значение соответствующего параметра – также expo). Следует отметить, что если это допущение не выполняется, то можно выбрать и любой другой закон.

Длительность ремонта компьютера всегда можно представить в виде суммы длительностей многочисленных элементарных операций – случайных величин, распределенных по произвольному закону. Поэтому в соответствии с центральной предельной теоремой бу-

дем считать, что эта длительность распределена по нормальному закону (значение соответствующего параметра – norm). Если допустить, что процесс обработки оператором одного документа также можно представить в виде последовательности многочисленных элементарных операций и длительность каждой элементарной операции распределена по произвольному закону, то можно считать, что интервал обработки одного документа – это величина, распределенная по нормальному закону (значение соответствующего параметра – также norm).

После создания схемы модели можно перейти к написанию ее текста в операторах Pilgrim. При составлении модели сразу встретимся с двумя сложными моментами.

Первая сложность, с которой мы столкнемся, – это описание условия перехода транзакта-документа в узел 5, а транзакта-неисправности – в узел 6. Вторая сложность заключается в том, что среднее время обработки транзакта и среднеквадратичное отклонение этого времени существенно зависят от природы транзакта, т.е. будут различными для транзакта-неисправности и транзакта-документа.

Введем в рассмотрение три переменные пользователя:

1) int Dist – для изменения закона распределения длительности обслуживания в узле serv в зависимости от вида транзакта;

2) float Tobs – для изменения средней длительности обслуживания в узле типа serv;

3) float Pogr – для изменения среднеквадратичного отклонения длительности обслуживания в этом же узле serv.

Транзакты можно различать по значению приоритета. Например, если транзакт-документ неприоритетный, то значение его приоритета обозначается none; а если транзакт-неисправность имеет приоритет 1, то это число можно сразу определить. Приоритет транзакта хранится в нем самом – в параметре t→pr. Будем использовать параметр пользователя с номером 0, находящийся внутри транзакта, для временного помещения номера узла, в который должен перейти транзакт из узла 4 (в узел 5 или 6). Этот параметр символически обозначается как t→iu0.

Следует учесть, что при прохождении транзакта через операторы узла адрес этого транзакта всегда находится в указателе t . Поэтому всегда можно определить приоритет транзакта как t→pr, а параметр пользователя с номером 0 можно получить как t→iu0.

После таких предварительных действий можно оформить следующее условие, позволяющее справиться с обоими вышеуказанными моментами (конструктор автоматически поможет написать его):

```
if (t→pr == 1)    // Приоритет равен 1?
   {
   t→iu0=6;       // Номер следующего транзакта
   Dist=norm;      // Закон распределения
   Tobs=1.0;       // Время обслуживания
   Pogr=Tobs/3.0; // Среднеквадратичное отклонение
   }
else
   {
   t→iu0=5;       // Номер следующего транзакта
   Dist=norm;      // Закон распределения
   Tobs=0.08;      // Время обслуживания
   Pogr=Tobs/3.0; // Среднеквадратичное отклонение
   }
```

Далее приступаем к созданию модели, формализованный текст которой приведен ниже:

```
#include <Pilgrim.h>
forward              // Начало моделирования
   {
   int Dist;         // Закон обслуживания
   float Tobs;       // Среднее время обслуживания
   float Pogr;       // Среднеквадратичное отклонение
   modbeg ("АРМ бухгалтерии",6,
           1200,(long)time(NULL),none,3,none,5,2);
   ag ("Бухгалтерия",1,none,expo,0.1,zero,zero,3);
                                  //  ta =  0,10 часа
   ag ("Неисправности",2,1,expo,24.0,zero,zero,3);
                                  //  te =  24,0 часа
   network (dummy,dummy)
      {
      top(3): queue("Столик-очередь",prty,4);
              place;
      top(4): if (t→pr == 1)
                 {
                 t→iu0=6;
                 Dist=norm;
                 Tobs=1.0;      // tr = 1,0 часа
                 Pogr=Tobs/3.0;
                 }
              else
                 {
                 t→iu0=5;
                 Dist=norm;
```

```
                Tobs=0.08;     // ts = 0,08 часа
                Pogr=Tobs/3.0;
                }
            serv("Компьютер",1,abs,Dist,Tobs,
                 Pogr, zero, t→iu0);
            place;
    top(5): term("Архивные папки");
            place;
    top(6): term("Неиспр. устр-на");
            place;
    fault(123);
    }
modend("Bookkeep.doc",1,24,page);
return 0;
}
```

В процессе моделирования в стандартном окне на экране монитора можно наблюдать динамику задержек, пространственных процессов, потоков транзактов, а также информацию о любом узле, входящем в состав модели. Результаты моделирования показаны на рис. 8.6. (график задержек) и в табл. 8.2, получаемой с помощью компьютера.

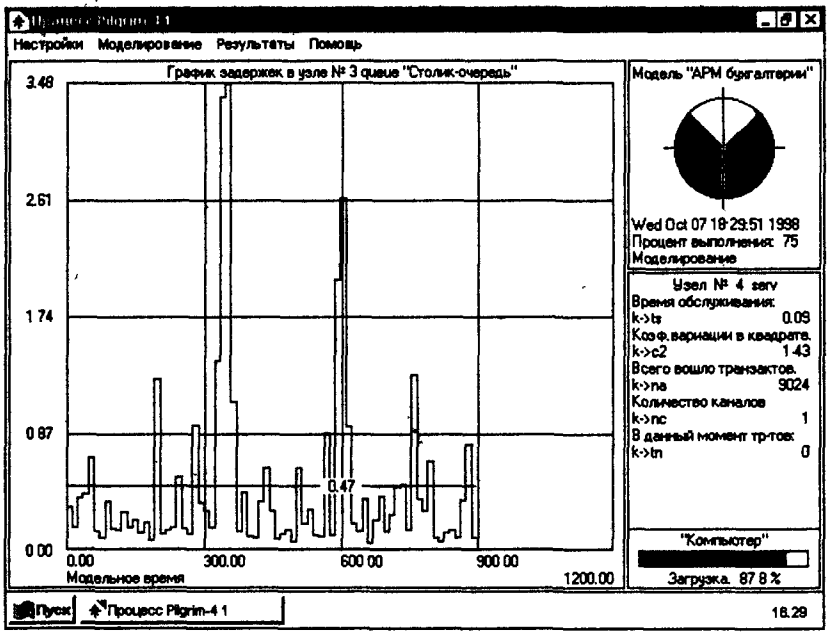


Рис. 8.6. Экран с результатами прогона сконструированной модели

Таблица 8.2

Результаты моделирования

НАЗВАНИЕ МОДЕЛИ: АРМ бухгалтерии							Лист: 1
ВРЕМЯ МОДЕЛИРОВАНИЯ: 1200,02							
№ узла	Наименование узла	Тип узла	Точка	Загрузка, % Путь, км	M[t] среднее	C^2 [t] квадрат коэфф. вариации	Счетчик входов
1	Бухгалтерия	ag	–	–	0,10	1,00	8974
2	Неисправности	ag	–	–	17,49	1,18	50
3	Столик-очередь	queue	–	–	0,46	2,15	9024
4	Компьютер	serv	–	%=87,8	0,09	1,43	9024
5	Архивные папки	term	–	–	0,54	1,54	8974
6	Неисправность устранена	term	–	–	1,08	0,16	50

Рассмотренная выше модель относится к разновидности имитационных моделей, которые называются *разомкнутыми*. В таких моделях не просматриваются какие-либо обратные связи или взаимные влияния компонентов экономической системы.

8.4 МОДЕЛЬ «МИНИМИЗАЦИЯ ПРОИЗВОДСТВЕННЫХ ЗАТРАТ»

Рассмотрим модель, которая относится к *замкнутым* имитационным моделям. Предположим, что мы открыли собственную швейную фабрику «Мануфактура Ко». На этой фабрике есть швейный цех, где имеются 50 рабочих мест со швейными машинами. Машины выходят из строя и требуют ремонта. Поэтому на фабрике есть ремонтный цех, в котором работают специалисты – наладчики машин*.

Среднее время безотказной работы одной машины 157,0 ч. Естественно допустить, что это время – случайная величина, распределенная по экспоненциальному закону в соответствии с предельной теоремой о суперпозиции потоков. Допущение основано на том, что машина – сложное устройство, состоящее из сотен деталей и узлов, которые могут выйти из строя.

* Впервые эту задачу в упрощенной постановке Т. Дж. Шрайбер решил с помощью имитационного моделирования на GPSS.

Среднее время ремонта машины равно 7,0 ч; среднеквадратичное отклонение времени ремонта – 3,0 ч. В соответствии с центральной предельной теоремой считаем, что время ремонта, состоящего из многих последовательных элементарных операций, распределено по нормальному закону.

Сформулируем следующую задачу: сколько нужно арендовать резервных машин j (дополнительно к 50 собственным) и сколько необходимо нанять наладчиков i, чтобы сделать минимальными затраты на производство, связанные с ремонтом и наладкой машин? Все исходные данные для решения поставленной задачи сведены в табл. 8.3. В качестве денежной единицы выберем американский доллар (только для определенности).

Таблица 8.3

Параметры швейной фабрики

№ п/п	Параметр швейного производства	Значение параметра
1	Название швейной фабрики	«Мануфактура Ко»
2	Заработная плата одного наладчика (в час)	3,75 долл.
3	Оплата за аренду одной швейной машины (в день)	30,0 долл.
4	Убыток из-за простоя одного рабочего места по причине неисправности швейной машины (в час)	20,0 долл.
5	Среднее время безотказной работы одной швейной машины	157,0 ч
6	Среднее время ремонта (наладки) одной машины	7,0 ч
7	Среднеквадратичное отклонение времени ремонта (наладки) одной машины	3,0 ч
8	Продолжительность одного рабочего дня	8,0 ч
9	Число рабочих часов в одну неделю	40,0 ч
10	Число рабочих недель в одном году	52 недели

Сначала попытаемся решить эту задачу без применения имитационной модели. Допустим, что необходимо j дополнительных машин, которые будем арендовать на каком-либо складе.

Нетрудно представить, что каждая машина находится в двух состояниях:

1) находится в швейном цехе и работает;

2) находится в ремонтном цехе, неисправна и ремонтируется наладчиками.

Средняя длительность такого цикла равна $157 + 7 = 164$. Поэтому введем в рассмотрение полезную загрузку одной машины $\rho = 157/164 = 0{,}957$.

Составим простое уравнение, показывающее, сколько нужно иметь дополнительных машин, чтобы все 50 мест не простаивали: $50 = \rho\,(50 + j)$.

Решение этого уравнения элементарно: $j = 50(1-\rho) = 2{,}15$ машин. Однако решение должно быть целочисленным. Поэтому округляем в бóльшую сторону и получаем целое число $j = 3$. Время ремонта составляет 7 ч. Учитывая, что постоянно будут неисправны 3 машины, а рабочий день составляет 8 ч, наймем трех наладчиков, которые будут хорошо загружены в течение дня (два наладчика могут не справиться с потоком машин, поступающих в ремонт). Получили искомую величину: $i = 3$.

Если поступим в соответствии с полученным решением и будем проводить натурное моделирование, собирая в течение длительного периода времени (не менее года) хронометрическую и финансовую информацию, то увидим, что решение, полученное выше, неверное.

Основная ошибка заключалась в гипотезе о том, что машины находятся в двух состояниях. На самом деле таких состояний четыре (рис. 8.7).

Выбранные выше состояния (исправная работа и ремонт) осуществляются в двух цехах: швейном и ремонтном. В начальный момент, при организации производства, все машины были исправны. Введем в рассмотрение следующие переменные, которые далее будут использоваться в модели:

- Nowon – количество рабочих мест в швейном цехе и соответственно количество собственных швейных машин;
- Arend – число арендуемых дополнительных машин для замены вышедших из строя ;
- Men – количество наладчиков, производящих ремонт (наладку) швейных машин.

Отметим все значимые для задержек состояния в табл. 8.4.

Таблица 8.4

Состояния машин

Номер фазы	Практический смысл фазы	Тип узла в модели	Число каналов
1	Помещение, где находятся исправные швейные машины (их начальное число Arend=*j*)	queue	–
2	Швейный цех, в котором имеется Nowon рабочих мест для Nowon=50 швейных машин	serv	Nowon
3	Помещение, где находятся неисправные швейные машины, ожидающие ремонта	queue	–
4	Ремонтный цех, в котором работают специалисты-наладчики, число которых Men=*i*	serv	Men

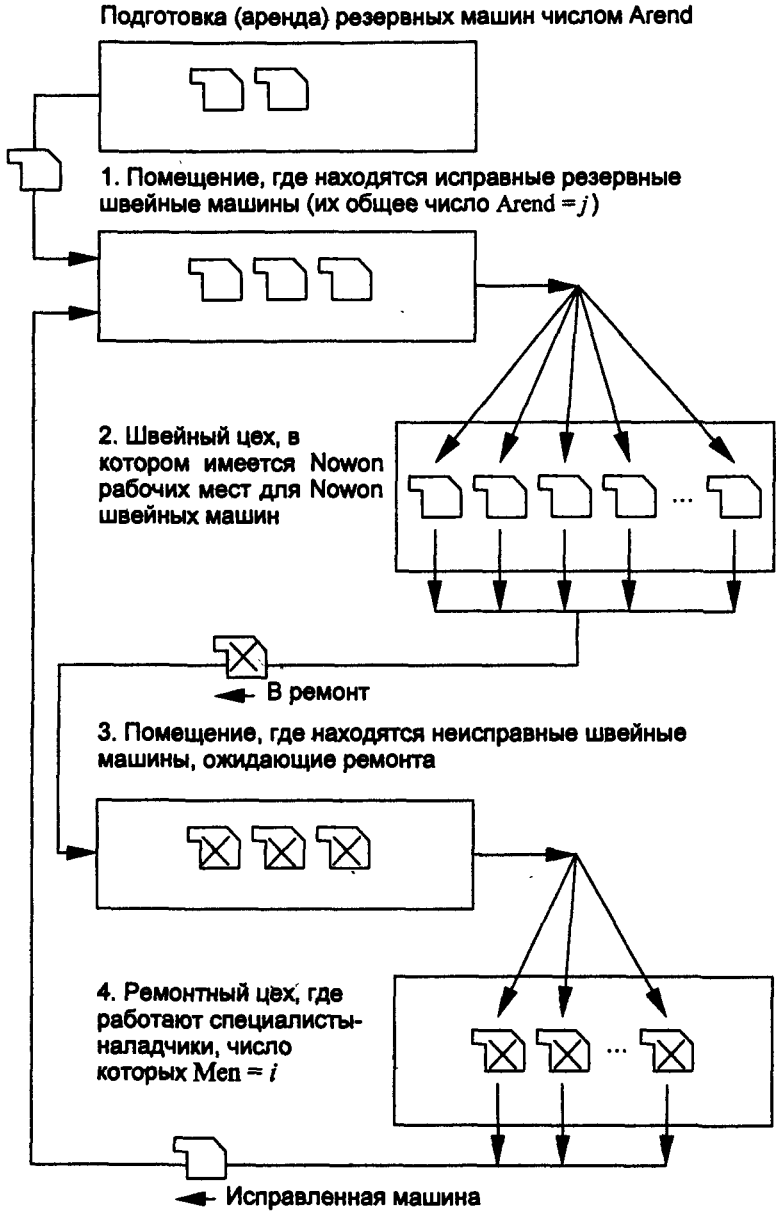


Рис. 8.7. Иллюстрация к задаче оптимизации производственных затрат

В начальный момент решено арендовать Arend машин. Эти машины должны некоторое время находиться в каком-то помещении и «ждать», пока не выйдет из строя машина в цехе. Далее в процессе выхода из строя и восстановления машин в этом помещении будут находиться машины, число которых – случайная величина. Пребывание машин в этом помещении является состоянием 1. Пребывание машин в швейном цехе в исправности – состояние 2.

Если какая-либо машина выходит из строя, то она поступит в ремонт только в том случае, если хотя бы один наладчик свободен (не занят ремонтом другой машины). Вероятность такого события, естественно, не равна единице. Поэтому неисправные машины в ожидании ремонта направляются в другое помещение. Пребывание в этом помещении является состоянием 3.

И, наконец, если подходит очередь ремонта неисправной машины, она поступает в ремонтный цех. Ремонт – состояние 4 каждой машины.

Далее начнем строить имитационную модель. Наиболее рациональное решение заключается в том, чтобы выбрать машины в качестве транзактов. В табл. 8.4 приведены типы узлов, имитирующих соответствующие состояния. Количество каналов узла serv в состоянии 2 равно Nowon, а число каналов узла serv в состоянии 4 – это значение Men.

Единственная проблема, с которой мы немедленно столкнемся, заключается в том, что в данной замкнутой схеме нет генераторов транзактов. В связи с этим непонятно, откуда транзакты попадут в модель. Решение этой проблемы довольно простое: необходимо ввести в имитационную модель специальную часть, которая называется *схемой зарядки*. Рассмотрим эту схему и всю модель на рис. 8.8.

Узлы с номерами 1 (queue), 2 (serv), 3 (queue) и 4 (serv) имитируют пребывание машин в состояниях с соответствующими номерами. Эта замкнутая схема не требует дальнейших комментариев.

Далее рассмотрим *схему зарядки*, которая состоит из трех узлов: узел 5 (генератор ag), узел 6 (транзактно-управляемый генератор creat) и узел 7 (терминатор term со вспомогательным оператором cheg). Модельное время, в течение которого будем осуществлять эксперимент, задается переменной Protime. Если мы хотим, чтобы это время составляло 3 года, а единицей измерения установим 1 ч, то это время равно 6240,0 ч.

Узел ag генерирует первый транзакт через какой-либо заданный или случайный интервал времени t_{beg}. Этот интервал можно назвать временем подготовки производства. Сгенерированный первый тран-

закт (назовем его прародителем) войдет в узел creat. В результате из этого узла выйдут Nowon + Arend транзактов, которые поступят в узел 1.

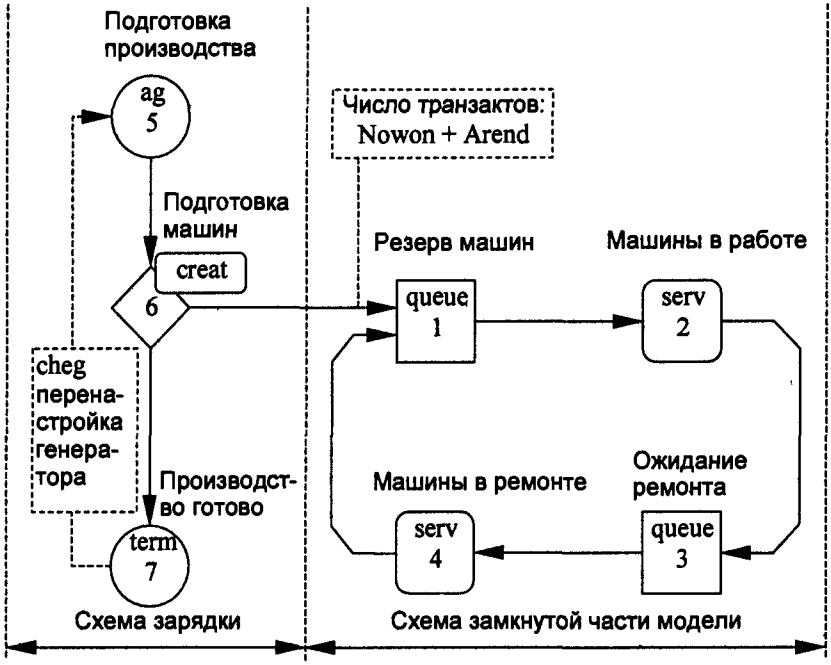


Рис. 8.8. Структурная схема модели для решения задачи минимизации производственных затрат

Первые Nowon транзактов без задержки пройдут этот узел и поступят в узел 2; причем каждый транзакт-машина займет свой канал, т.е. рабочее место. Остальные Arend транзактов останутся в узле 1.

Что касается транзакта-прародителя, то он выйдет из узла creat и войдет в узел term. В этом узле перед уничтожением транзакт-прародитель выполнит операцию cheg для перенастройки генератора ag. Эта операция установит время генерации второго транзакта, равное Protime. Учитывая, что с начала запуска модели уже прошло время t_{beg}, момент генерации второго транзакта будет равен Protime + + t_{beg} (т.е. после выполнения модели). Это означает, что второй транзакт не выйдет из генератора во время моделирования.

Далее перейдем к планированию имитационного эксперимента для определения рационального количества резервных машин и числа специалистов-наладчиков. Возможность существования оптимального решения вытекает из следующих предпосылок.

Нетрудно представить себе, что если число наладчиков мало или равно нулю, то машины в основном будут неисправны, а затраты на производство – велики из-за простоя рабочих мест. Если же число наладчиков очень велико, то будут большими затраты на заработную плату (рис. 8.9,*а*). Но при отсутствии наладчиков машины постепенно выйдут из строя, а затраты из-за простоя рабочих мест будут велики.

Что касается резервных машин, то, если их число мало или равно нулю, затраты на производство будут велики из-за простоя рабочих мест. Если же число арендуемых машин очень велико, то будут большими затраты на их аренду (рис. 8.9,*б*).

Поэтому можно допустить гипотезу о наличии какой-то поверхности оптимизации затрат (8.9,*в*). Однако, учитывая, что число наладчиков и количество машин – целые величины, эта поверхность включает в себя только точки с целочисленными координатами Men и Arend.

Необходимо найти минимальное значение на поверхности затрат E_{ij} и соответственно значения координат i_{opt} (переменная Men) и j_{opt} (переменная Arend). Не будем применять сложные методы поиска экстремальных значений на поверхности оптимизации затрат. Задача решается *методом перебора*, так как число возможных вариантов, которые необходимо сравнить, невелико. Однако в более сложных случаях требуется применять *регрессионный анализ* и строить поверхность отклика 2-го порядка.

Первое, что нужно сделать при проведении имитационного эксперимента, – это привести все времена и темпы затрат к одним единицам измерения. С помощью табл. 8.3 получим табл. 8.5 со статьями ежедневных расходов на производство. В ней три строки. Обозначим значение, указанное в каждой строке, как A_k, $k = 1,2,3$.

Таблица 8.5

Статьи ежедневных расходов A_k ($k = 1, 2, 3$)

Номер строки	Вид ежедневных расходов	Стоимость расходов, долл.
1	Дневная заработная плата одного наладчика	30
2	Дневная аренда одной машины	30
3	Убыток из-за простоя одного рабочего места	160

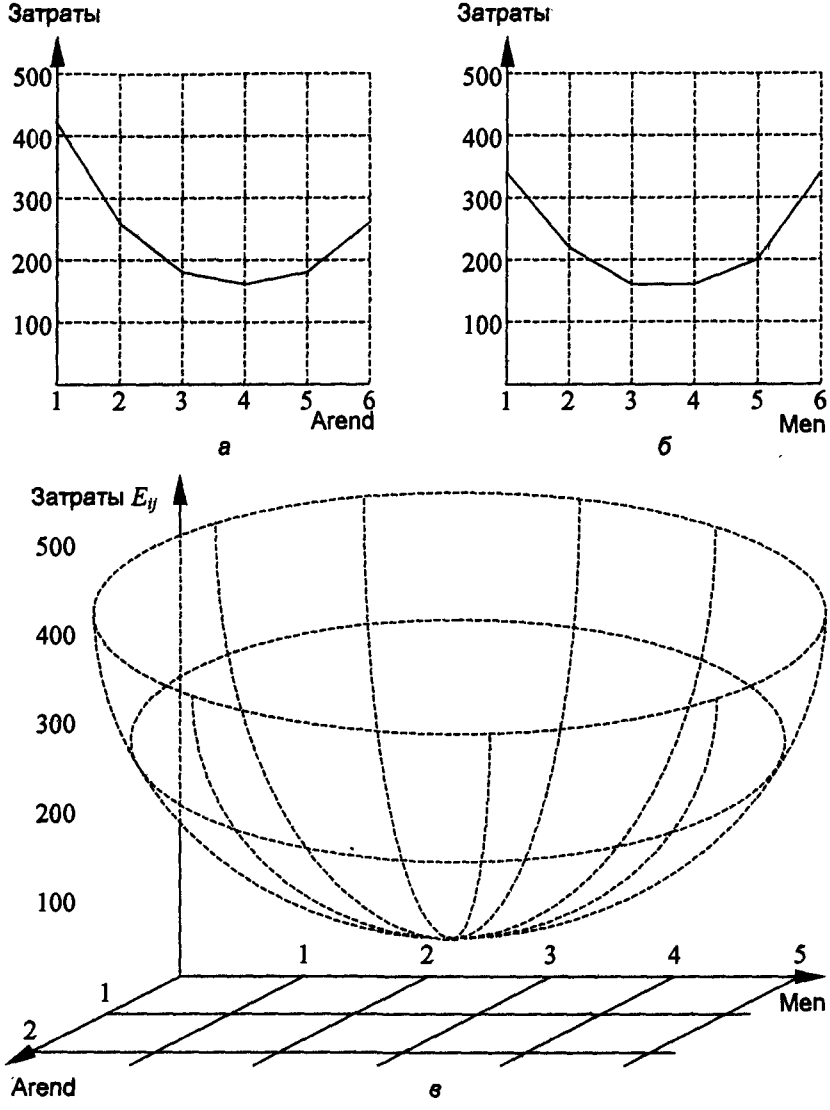


Рис. 8.9. Интерпретация возможности минимизации затрат: *а* – зависимость затрат от числа арендуемых машин Arend при фиксированном числе наладчиков; *б* – зависимость затрат от числа наладчиков Men при фиксированном числе арендуемых машин; *в* – предполагаемая поверхность оптимизации

Далее потребуется имитационная модель, схема которой была приведена на рис. 8.8. С ее помощью необходимо получить значения загрузки рабочих мест, например, с экрана (рис. 8.10).

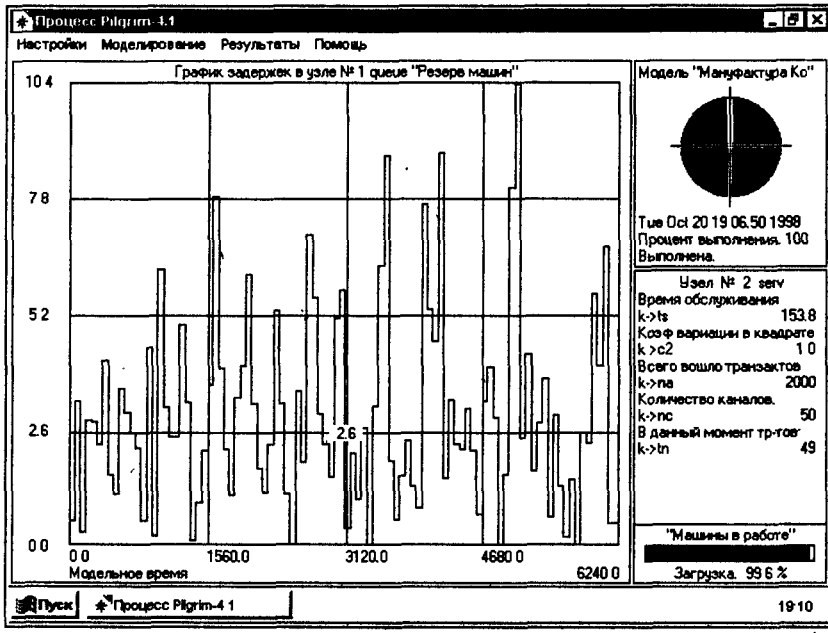


Рис. 8.10. Окно с результатами моделирования

Текст модели приведен ниже:

```
#include <Pilgrim.h>
int Nowon = 50;           // Число собственных машин
int Men = 3;              // Число наладчиков
int Arend = 3;            // Число арендуемых машин
float Protime = 6240.0; // Время прогноза 3 года
forward
  {
  modbeg("Мануфактура Ко",7,Protime,
         (long)1234567890,none,1,none,none,1);
  ag("Подг. произв.",5,none,none,1.0,zero,zero,6);
  network (dummy, dummy)
    {
    top(1): queue("Резерв машин", none, 2);
            place;
    top(2): serv("Машины в работе",Nowon,none,
                 expo,157.0,zero,zero,3);
            place;
```

```
    top(3): queue("Ожид. ремонта", none, 4);
            place;
    top(4): serv("Машины в рем.",Men,none,norm,
                 7.0,3.0,zero,1 );
            place;
    top(6): creat("Подгот. машин",none,
                  Nowon+Arend,copy,1,7);
            place;
    top(7): term("Пр-во готово");
            cheg(5,none,none,Protime,zero,zero,7);
            place;
    fault (123);
    }
modend ("Factory.doc",1,24,page);
return 0;
}
```

Построим матрицу загрузок для различных значений числа наладчиков i и количества швейных машин j. Элементы такой матрицы обозначим B_{ij}. Пример матрицы представлен в виде табл. 8.6. Как было показано выше, в оценочном и не совсем верном расчете, поиск оптимальных значений i и j можно начать со значений $i = 3$ и $j = 3$, пропустив меньшие значения (i – число наладчиков, $i = 3, 4, 5$; j – число арендуемых швейных машин, $j = 3, 4, 5$).

Таблица 8.6

Загрузка рабочих мест B_{ij} (получается с помощью имитационной модели)

Число наладчиков	Число арендуемых резервных машин		
	3	4	5
3	0,951	0,974	0,964
4	0,986	0,979	0,978
5	0,978	0,984	0,979

Далее определим ежедневные затраты на оплату труда наладчиков и аренду резервных машин. Для этого создадим еще одну матрицу (табл. 8.7) элементов C_{ij} с той же размерностью, какая существует в табл. 8.6. Элементы этой матрицы определяются с помощью табл. 8.5 из соотношения

$$C_{ij} = A_1 i + A_2 j, \quad i = 3, 4, 5, \quad j = 3, 4, 5.$$

Таблица 8.7

Затраты на оплату труда наладчиков и аренду резервных машин C_{ij}

Число наладчиков	Число арендуемых резервных машин		
	3	4	5
3	180	210	240
4	210	240	270
5	240	270	300

Перейдем к определению потерь из-за снижения объемов производства по причине простоя рабочих мест. Для этого построим матрицу (табл. 8.8) элементов D_{ij} с той же размерностью. Элементы этой матрицы определяются с помощью табл. 8.5 и 8.6. Элементы B_{ij} табл. 8.6 – это загрузка оборудования ρ при конкретных значениях i и j. Поэтому справедливо следующее соотношение:

$$D_{ij} = \text{Nowon}\,(1 - B_{ij})\,A_3,\ i = 3, 4, 5,\ j = 3, 4, 5.$$

Таблица 8.8

Потери из-за снижения объемов производства D_{ij}

Число наладчиков	Число арендуемых резервных машин		
	3	4	5
3	392	208	288
4	112	168	176
5	176	128	168

Последнее, что нужно сделать, – это определить суммарные ежедневные затраты на производство. Для этого построим матрицу с той же размерностью (табл. 8.9). Элементы матрицы обозначаются E_{ij} и определяются поэлементным суммированием данных табл. 8.7 и 8.8:

$$E_{ij} = C_{ij} + D_{ij},\ i = 3, 4, 5,\ j = 3, 4, 5.$$

Из табл. 8.9 следуют оптимальные значения:

1) количество нанимаемых наладчиков $i_{opt} = 4$;

2) число арендуемых резервных швейных машин $j_{opt} = 3$.

Таблица 8.9

Суммарные ежедневные затраты на производство E_{ij}

Число наладчиков	Число арендуемых резервных машин		
	3	4	5
3	571	418	528
4	322	408	446
5	416	398	468

Если бы мы выбрали ранее ошибочно рассчитанные значения $i_{opt} = 3$ и $j_{opt} = 3$, то затраты ежедневно составляли бы не 322, а 571 долл. Различие в результатах в данном случае весьма значительно.

В табл. 8.10 представлен стандартный набор выходных данных, автоматически полученных во время прогона варианта имитационной модели с конкретными значениями исходных данных.

Таблица 8.10

Результаты моделирования

НАЗВАНИЕ МОДЕЛИ: Мануфактура Ко							Лист: 1
ВРЕМЯ МОДЕЛИРОВАНИЯ: 6241.0							
№ узла	Наименование узла	Тип узла	Точка	Загрузка, % Путь, км	M[t] среднее	C^2 [t] квадрат коэфф. вариации	Счетчик входов
1	Резерв машин	queue	–	–	2,6	2,63	1973
2	Машины в работе	serv	–	%=98,6	154,9	1,03	1971
3	Ожидание ремонта	queue	–	–	2,9	2,9	1921
4	Машины в ремонте	serv		%=69,7	7,1	0,17	1921
5	Подготовка производства	ag	–	–	1,0	0,00	2
6	Подготовка машин	creat	–	–	0,0	1,00	53
7	Производство подготовлено	term	–	–	0,0	1,00	2

8.5 МОДЕЛЬ «ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ РЕСУРСОВ В СЕТИ ПОД УПРАВЛЕНИЕМ UNIX»

Локальная вычислительная сеть (ЛВС) имеет один процессор, один винчестер и несколько пользовательских терминалов. Пользователи посылают операционной системе OS UNIX запросы на выполнение своих прикладных задач, которые содержат некоторое количество операций ввода-вывода (I/O). Посланный запрос поступает на обработку процессором (CPU, число процессоров может быть 1, 2, 4 и др.). По каждому запросу создается процесс, который начинает работать в ЛВС.

Созданному процессу OS UNIX выделяет участок памяти (RAM) и затем несколько буферов I/O (ресурсы ЛВС). Процессы, требующие меньшего объема ресурса, имеют больший приоритет при его выделении. Если ресурсов ЛВС недостаточно для выполнения очередного процесса, он ожидает освобождения ресурсов другими процессами. После этого процесс получает доступ к винчестеру (HDD) для выполнения одной операции I/O. Процесс, завершивший свою работу, освобождает занимаемые им ресурсы, возвращая их операционной системе. Параметры модели приведены ниже, схема модели показана на рис. 8.11.

Параметр модели	Значение параметра
Количество пользователей	50
Время обдумывания запроса пользователем (среднее), с	10,00
Время обработки процесса в CPU (среднее), с	0,10
Время доступа HDD-винчестера (операция I/O):	
минимальное, с	0,05
максимальное, с	0,15
наиболее вероятное, с	0,10
Объем (мощность) ресурса «RAM», Мбайт	64
Объем (мощность) ресурса «Буферы I/O»	10
Вероятность завершения процесса I/O	0,20
Время моделирования, ч	8

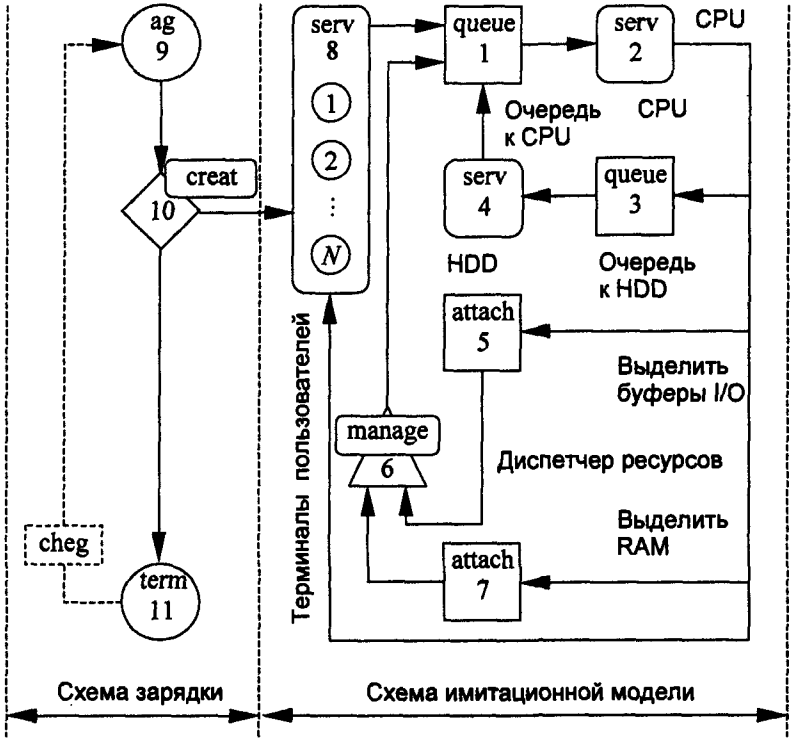


Рис. 8.11. Модель работы ЛВС под управлением OS UNIX

После выделения ресурса и выполнения каждой операции ввода-вывода CPU определяет очередной этап обработки процесса. Если при очередной проверке состояния процесса необходимое количество операций I/O окажется выполненным, пользователю направляется сигнал о выполнении его запроса. Получив сигнал, пользователь обдумывает и посылает следующий запрос.

Текст модели приведен ниже:

```
// Модель управления ресурсами в ЛВС под OS UNIX
// Время измеряется в секундах
#include <Pilgrim.h>
#define USERS 50          // Количество пользователей
#define RAM 0             // Флаг создания процесса
#define BUFF 1            // Флаг захвата буферов I/O
#define HDD 2             // Флаг операции I/O
#define MAX_RAM 64        // Максимальный объем RAM
```

```
#define MAX_BUFFERS 10  // Общее число буферов
#define PROCESS_STATUS t→iu0 // Состояние транзакта
#define RAM_ACCOUNT t→iu1    // Выделяемая RAM
#define BUF_AMOUNT t→iu2     // Выделяемые буферы
#define HDD_ACCESS t→iu3     // Флаг доступа к HDD
#define MEMORY 5             // Узел выделения RAM
#define IO_BUFFERS 7    // Узел управления буферами
#define MOD_TIME 28800.0     // Моделируется 8 час
int Next_top;           // Номер следующего узла
forward
  {
  modbeg("Сеть UNIX-V",1,MOD_TIME,
         (long)time(NULL),none,5,none,none,2);
  ag("Вспомогательный транзакт",9,none,none,
     0.0,zero,zero,10);
   supply(MEMORY, add, MAX_RAM);
                        // Объем памяти на сервере
   supply(IO_BUFFERS, add, MAX_BUFFERS);
                        // Объем буферного пула
   network(dummy, dummy)
     {
     top(1): queue("Очередь к CPU",none,2);
             place;
      top(2): switch(PROCESS_STAtUS)
               {
               case HDD:        // Процесс создан
                 if (rundum() > 0.8 && HDD_ACCESS)
                   Next_top=8; // Ответ на запрос
                 else
                   Next_top=3; // Операция I/O
                 break;
               case BUFF:       // Выделить буферы
                 Next_top=7;
                 break;
               case RAM:        // Создать процесс
                 Next_top=5;
                 break;
               }
             serv ("CPU", 1, none, unif ,0.1,0.1,
                   zero ,Next_top);
             clcode
               {
               if (Next_top ==8 )
                 {
                 HDD_ACCESS=0;
                          // Доступ к HDD прекращен
                 detach(MEMORY,RAM_AMOUNT);
                          // Освободить память
                 detach(IO_BUFFERS,BUF_AMOUNT);
                          // Вернуть буферы
                 }
             place;
```

```
    top(3): HDD_ACCESS=1;
                        // Получение доступа к HDD
            queue ("Очередь к HDD", none, 4);
            place;
    top(4): serv("HDD",1,none,beta,0.05,0.1,
                 0.15,1);
            place;
    top(5): RAM_AMOUNT+rundum()*(MAX_RAM-1);
            PROCESS_STATUS=BUFF;
                        // Запросить буферы I/O
            attach("Выделить RAM", RAM_AMOUNT,
                   prty, 6);
            place;
    top(6): manage("Диспетчер ресурсов",1);
            place;
    top(7): BUF_AMOUNT=1+rundum()*(MAX_BUFFERS-1);
            PROCESS_STATUS=HDD;
                        // Начать операцию I/O
            attach("Выделить буферы I/O",
                   BUF_AMOUNT,prty,6);
            place;
    top(8): PROCESS_STATUS=RAM;
                        // Создание процесса
            serv("Терминалы",USERS,none,norm,10.0,
                 2.5,zero,1);
            place;
//-----------------------------------------------
//  Схема зарядки модели:
//-----------------------------------------------
    top(10):creat("Пользователи",none,USERS,
                  copy,8,11);
            place;
    top(11):term("Выключение ag");
            cheg(9,none,none,MOD_TIME,zero,
                 zero,11);
            place;
    fault(123);
    }
  modend("Unix5.doc",1,13,page);
  return 0;
  }
```

Описание работы модели. В модели транзакт имеет несколько состояний:

- обдумываемый пользователем запрос;
- досланный пользователем запрос на организацию процесса ввода-вывода;
- созданный OS UNIX-V процесс I/O;
- процесс, получивший RAM;

- процесс, получивший буферы I/O;
- процесс, выполнивший хотя бы одну операцию I/O;
- сигнал пользователю о выполнении запроса.

Текущее состояние транзакта заносится в его параметр и является критерием перехода из сервера, имитирующего работу процессора. Изменение состояния транзакта осуществляется в узлах, имитирующих терминалы пользователей, работу CPU, обращение к HDD, выделение RAM и буферов I/O.

Имена используемых параметров транзактов переопределены операторами #define.

Модель является замкнутой. С помощью схемы зарядки (узлы 9, 10 и 11) генерируется семейство транзактов количеством USERS (число терминалов пользователей). Сгенерированные транзакты поступают в сервер 8, имитирующий пользовательские терминалы. Число каналов сервера равно константе USERS. Время обслуживания – это время обдумывания пользователем запроса к OS UNIX. В соответствии с центральной предельной теоремой оно распределяется по нормальному закону.

Перед входом в узел состояние транзакта (параметр PROCESS_STATUS) получает значение RAM. Это значит, что транзакт, выйдя из сервера, будет играть роль запроса на создание процесса I/O и ему нужно будет выделить память. Из сервера транзакт переходит в очередь 1, имитирующую очередь к процессору, и далее в сервер 2, имитирующий процессор. Перед входом в узел на основании значения параметра PROCESS_STATUS принимается решение о дальнейшей обработке транзакта. Если состояние транзакта имеет значение HDD, это значит, что данный транзакт имитирует уже созданный процесс I/O. Если при этом установлен флажок доступа к винчестеру (параметр транзакта HDD_ACCESS равен 1), значит, процесс выполнил хотя бы одну операцию ввода-вывода и с вероятностью 0,2 является завершенным (значение генератора rundum больше 0,8). В этом случае транзакт отправляется в сервер 8 – OS UNIX посылает пользователю сигнал о выполнении его запроса. Транзакт отправляется в очередь 3, имитирующую очередь запросов к винчестеру на выполнение операций I/O (процесс должен выполнить очередную операцию ввода-вывода), когда выполняется хотя бы одно из двух условий:

- значение HDD_ACCESS равно нулю (процесс не выполнил еще ни одной операции ввода-вывода);
- значение генератора rundum не превосходит 0,8 (процесс не завершен).

Если состояние транзакта имеет значение BUFF, это значит, что процесс получил память и теперь должен получить буферы I/O. В этом случае транзакт отправляется на «склад ресурсов» – узел attach 7, имитирующий выделение процессу буферов I/O. Если состояние транзакта имеет значение RAM, то транзакт имитирует только что пришедший запрос на создание процесса I/O. В этом случае транзакт отправляется на «склад ресурсов» 5, имитирующий выделение процессу памяти.

Число каналов сервера 2, имитирующего процессор, равно единице, поэтому процессор одновременно обрабатывает один процесс (запрос). Время обслуживания – это время обработки ветви программы процесса между двумя прерываниями; оно имеет равномерный закон распределения.

Если транзакт входит в узел CPU, а далее будет отправлен в сервер 8 (это ответ на терминал пользователя), то сбрасывается флажок доступа к винчестеру HDD_ACCESS (он получает значение нуль), а транзакт возвращает полученные им ресурсы (выполняются операции detach).

В очереди 3 устанавливается флажок доступа к винчестеру HDD_ACCESS, и транзакт переходит в сервер 4, имитирующий обращение к винчестеру – операцию I/O. Число каналов сервера равно единице, поэтому одновременно выполняется одна операция I/O. Время обслуживания в сервере есть время доступа к винчестеру; оно задано с помощью треугольного beta-распределения с минимальным временем доступа 0,05 с, максимальным – 0,15 с и наиболее вероятным – 0,1 с. Из этого сервера-винчестера транзакт переходит в очередь 1.

На «складе ресурсов» 5 транзакту выделяется ресурс типа RAM. Перед входом в узел определяется объем запрашиваемой памяти (параметр транзакта RAM_AMOUNT) как случайная величина, составляющая какую-то долю от мощности – максимального объема памяти. Состояние транзакта PROCESS_STATUS получает значение BUFF, в результате процесс помечается как ожидающий выделения буферов I/O. В описании узла параметр RAM_AMOUNT указывает количество запрашиваемого ресурса. Флажок учета приоритета запроса установлен. Далее транзакт переходит к диспетчеру ресурсов 6, а оттуда в очередь 1.

На «складе ресурсов» 7 транзакту выделяется требуемое количество буферов I/O (параметр транзакта BUF_AMOUNT). Состояние транзакта получает значение HDD (процесс готов к выполнению операций I/O, доступ к винчестеру разрешен). В остальном узел 7 аналогичен узлу 5.

Анализ результатов моделирования. За период моделирования к OS UNIX поступило 25 806 запросов (табл. 8.11); столько же было образовано вычислительных процессов (счетчик входов узла 8). Каждый пользователь послал в среднем 500 запросов, и 7 пользователей на момент окончания моделирования обдумывают очередной запрос. В среднем пользователи потратили на обдумывание запросов 18 % времени работы с ЛВС.

Было 25 799 запросов от процессов на получение RAM (счетчик входов узла 5). Однако 41 запрос остался неудовлетворенным: из-за дефицита 41 транзакт находится в очереди attach на момент завершения моделирования. Кроме того, 25 758 запросов было на получение буферов I/O, причем все они были удовлетворены. Диспетчер ресурсов произвел 51 516 операций по управлению ресурсами. Видно, что выполняется основное балансное соотношение:

$$(25\,799 - 41) + (25\,758 - 0) = 51\,516\,.$$

Всего процессор находился в состоянии работы 71,3% времени, операции ввода-вывода заняли 44,6% времени. Среднее время ожидания процессора составило 0,05 с, а среднее время пребывания в очереди ввода-вывода – 0,01 с.

Таблица 8.11

Результаты моделирования

НАЗВАНИЕ МОДЕЛИ: Сеть UNIX-V							Лист: 1
ВРЕМЯ МОДЕЛИРОВАНИЯ: 28800/00							
№ узла	Наименование узла	Тип узла	Загрузка, % Путь, км	M[t] среднее	Счетчик входов	Осталось транзактов	Состояние узла
1	Очередь к CPU	queue	–	0,05	205831	0	Открыт
2	CPU	serv	%=71,3	0,10	205831	1	Открыт
3	Очередь к HDD	queue	–	0,01	128517	0	Открыт
4	HDD	serv	%=44,6	0,10	128517	1	Открыт
5	Выделить RAM	attach	%=73,8	42,33	25799	41	9 S 2495 D
6	Диспетчер ресурсов	manage	–	0,00	51516	0	Открыт
7	Выделить буферы I/O	attach	%=55,5	0,38	25758	0	1 S 0 D
8	Терминалы	serv	%=17,9	10,1	25806	7	Открыт
9	Вспомогательный транзакт	ag	–	0,00	2	1	Открыт
10	Пользователи	creat	–	0,00	20	0	Открыт
11	Выключение ag	term	–	0,00	1	0	Открыт

Средняя загрузка ресурса RAM составляет 73,8%. На момент окончания моделирования остаток ресурса RAM составляет 9 Мбайт, а стоящие в очереди процессы требуют в общей сложности 2495 Мбайт памяти. Это объясняет большую задержку – 42,33 с при каждом обращении; данный ресурс – самое узкое место ЛВС. Средняя загруженность ресурса буферов I/O составляет 55,5%. Задержка в связи с возможным дефицитом сравнительно невелика – 0,38 с.

8.6 МОДЕЛЬ БИЗНЕС-ПРОЦЕССА «ЭФФЕКТИВНОСТЬ ПРЕДПРИЯТИЯ»

Рассмотрим модель бизнес-процесса предприятия (небольшой компании). Предприятие занимается выпуском товара небольшими партиями. Размер партии известен. Имеется рынок, где эта продукция продается. Размер партии покупаемого товара в общем случае – случайная величина.

Структурная схема бизнес-процесса содержит три слоя. На двух слоях расположены автономные процессы «Производство» (рис. 8.12) и «Сбыт» (рис.8.13), схемы которых независимы друг от друга (нет путей для передачи транзактов). Опосредованное взаимодействие этих процессов осуществляется только через ресурсы: материальные ресурсы в виде готовой продукции и денежные ресурсы (в основном через расчетный счет).

Управление денежными ресурсами происходит на отдельном слое – в процессе «Денежные операции» (рис. 8.14).

Данная модель позволяет решать следующую оптимизационную задачу, используя, например, методы регрессионного анализа. Введем целевую функцию: время задержки платежей с расчетного счета $T_{рс}$. Основные управляющие параметры – это цена единицы продукции, объем выпускаемой партии, сумма кредита, запрашиваемого в банке. Зафиксировав все остальные параметры (время выпуска партии, число производственных линий, интервал поступления заказа от покупателей, разброс размеров продаваемой партии, стоимость комплектующих изделий и материалов для выпуска партии, стартовый капитал на расчетном счете), можно минимизировать $T_{рс}$ для конкретной рыночной ситуации. Минимум $T_{рс}$ достигается при одном из максимумов среднего размера денежной суммы на расчетном счете. Причем вероятность рискового события – неуплаты долгов по кредитам – близка к минимуму (это можно доказать во время статистического эксперимента с моделью).

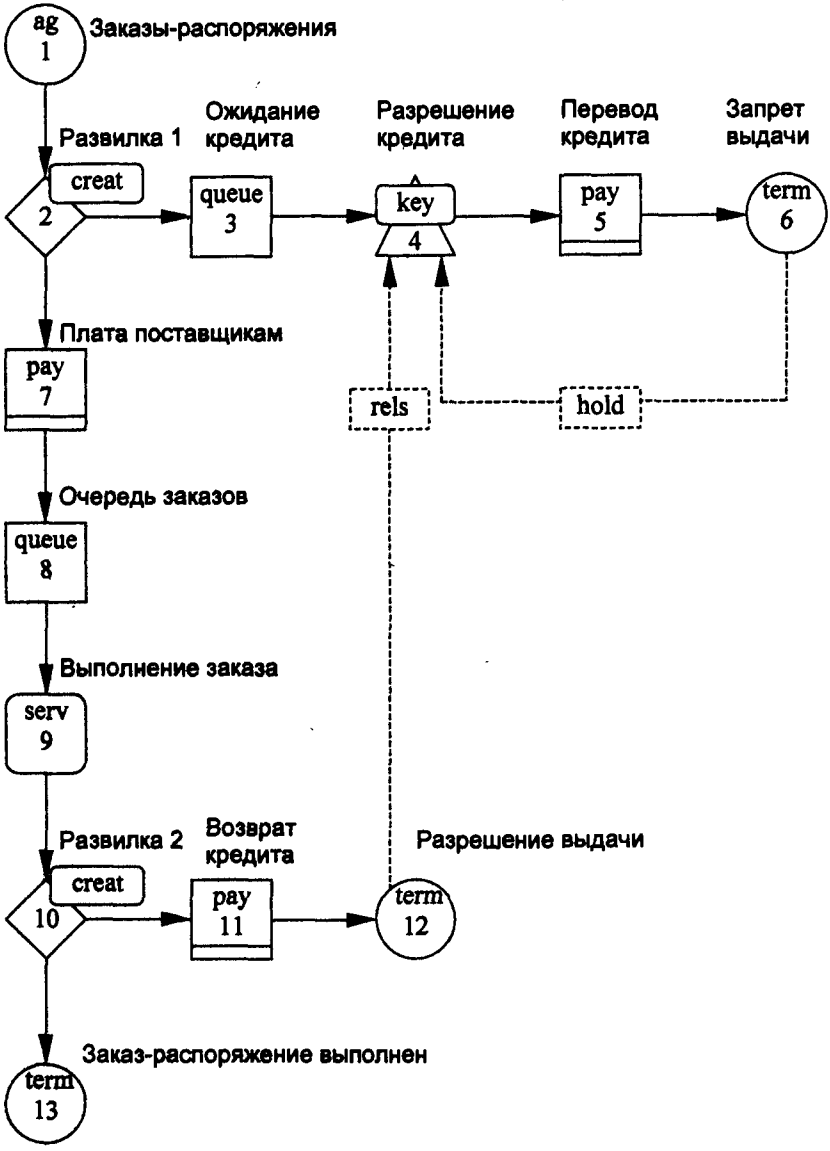


Рис. 8.12. Схема бизнес-процесса Производство: hold, rels – управление запретами /разрешениями

Рис. 8.13. Схема процесса реализации продукции Сбыт

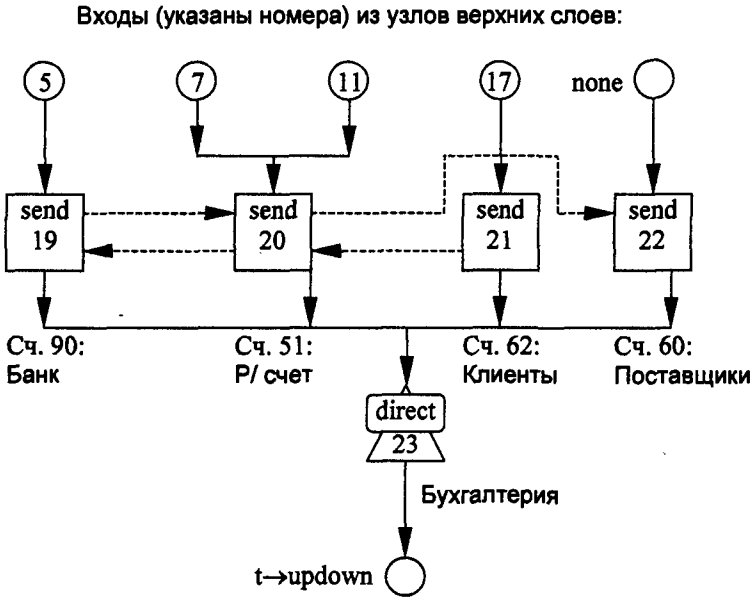


Рис. 8.14. Схема управления потоками Денежные операции

Первый процесс «Производство» реализует основные элементарные процессы. Узел 1 имитирует поступления распоряжений на изготовление партий продукции от руководства компании. Узел 2 – попытка получить кредит. В этом узле появляется вспомогательный транзакт – запрос в банк. Узел 3 – ожидание кредита этим запросом. Узел 4 – это администрация банка: если предыдущий кредит возвращен, то предоставляется новый (в противном случае запрос ждет

в очереди). Узел 5 осуществляет перечисление кредита на расчетный счет компании. В узле 6 вспомогательный запрос уничтожается, но информация о том, что кредит предоставлен, – это «шлагбаум» на пути следующего запроса на получение другого кредита (операция hold).

Основной транзакт-распоряжение проходит через узел 2 без задержки. В узле 7 производится оплата комплектующих, если на расчетном счете есть достаточная сумма (даже если кредит не получен). В противном случае происходит ожидание либо кредита, либо оплаты продаваемой продукции. В узле 8 транзакт становится в очередь, если все производственные линии заняты. В узле 9 осуществляется изготовление партии продукции. В узле 10 возникает дополнительная заявка на возврат кредита, если ссуда ранее была выделена. Эта заявка поступает в узел 11, где происходит перечисление денег с расчетного счета компании в банк; если денег нет, то заявка ожидает. После возврата кредита эта заявка уничтожается (в узле 12); в банке появилась информация о том, что кредит возвращен и компании можно выдать следующий кредит (операция rels).

Транзакт-распоряжение проходит узел 10 без задержки, а в узле 13 он уничтожается. Далее считается, что партия изготовлена и поступила на склад готовой продукции.

Второй процесс «Сбыт» имитирует основные функции по реализации продукции. Узел 14 – это генератор транзактов-покупателей продукции. Эти транзакты обращаются на склад, и если там есть запрашиваемое количество товара, то товар отпускается покупателю; в противном случае покупатель ждет. Узел 16 имитирует отпуск товара и контроль очереди. После получения товара покупатель перечисляет деньги на расчетный счет компании (узел 17). В узле 18 покупатель считается обслуженным; соответствующий ему транзакт больше не нужен и уничтожается.

Третий процесс «Денежные операции» имитирует проводки в бухгалтерии. Запросы на проводки поступают с первого слоя из узлов 5, 7, 11 (процесс «Производство») и из узла 17 (процесс «Сбыт»). Пунктирными линиями показано движение денежных сумм по счету 51 («Расчетный счет», узел 20), счету 60 («Поставщики, подрядчики», узел 22), счету 62 («Покупатели, заказчики», узел 21) и по счету 90 («Банк», узел 19). Условные номера примерно соответствуют плану счетов бухгалтерского учета.

Узел 23 имитирует работу финансового директора. Обслуженные транзакты после бухгалтерских проводок попадают обратно в те

узлы, откуда они поступили; номера этих узлов находятся в параметре транзакта t→updown.

Ниже приведен текст модели:

```
#include <Pilgrim.h>
forward
  {
  float T_cust=7;    // Интервал поступления заказов
  float T_work=14;        // Время выполнения заказа
  float S_bank=10000.00; // Сумма кредита в банке
  float S_supp=10000.00; // Сумма платы поставщику
  float The_price=90.00; // Цена единицы продукции
  float Mod_time=730;     // Моделируем 2 года
  int   N_work=2;         // Мощность фирмы
  int   Max=1200;         // Объем партии продукции
  modbeg("Бизнес-процесс",23,Mod_time,
         (long)1234567890,none,20,none,18,none);
  ag("Заказы",1,none,norm,T_work,0,zero,2);
  ag("Клиенты",14,none,norm,T_cust,T_cust/3,
     zero,15);
  assign(19,add,10000000.00); // Фонд банка
  assign(20,add,0.00);        // Фонд фирмы
  assign(21,add,10000000.00); // Фонды клиентов
  network(dummy,dummy)
    {
    //=============================================
    // Слой: процесс «Производство»
    //=============================================
    top( 2):creat("Развилка 1",0,1,none,3,7);
            place;
    top( 3):queue("Ждём кредит !",none,4);
            place;
    top( 4):key("Разреш. кредита",5);
            place;
    top( 5):pay("Перевод кредита",20,S_bank,19,
                none,19,6);
            place;       // На слой № 2: банк
    top( 6):term("Запрет выдачи");
            hold(4);
            place;
    top( 7):pay("Плата поставщ.",22,S_supp,20,
                none,20,8);
            place;       // На слой № 2:  Р/счет
    top( 8):queue("Очередь заказов",none,9);
            place;
    top( 9):serv("Выполн. заказа",N_work,none,norm,
                 T_work,zero,zero,10);
            place;
    top(10):creat("Развилка 2",0,1,none,11,13);
            place;
    top(11):pay("Возврат кредита",19,S_bank,20,
                none,20,12);
            place;          /* На слой № 2: Р/счет
```

```
    top(12):term("Разреш. выдачи");
            rels(4);
            place;
    top(13):term("Заказ выполнен");
            clcode
              supply(15,none,Max);
            place;
    //=============================================
    // Слой: процесс «Сбыт»
    //=============================================
    top(15):t->powr=rundum()*100;
                        // Объем закупаемой партии
            t->summ=t->powr*The_price;
                        // Стоимость этой партии
            attach("Склад гот.прод.",
                    t->powr,prty,16);
            place;
    top(16):manage("Отпускаем товар",17);
            place;
    top(17):pay("Оплата покупки",20,
                  t->summ,21,none,21,18);
            place; // На слой № 2: счет покупателя
    top(18):term("Товар оплачен");
            place;
    //=============================================
    // Слой: процесс «Денежные операции»
    //=============================================
    top(19):send("Банк:   90",   t->k1,t->summ,
                  t->dpr,23);
            place;
    top(20):send("Расч.счет: 51",t->k1,t->summ,
                  t->dpr,23);
            place;
    top(21):send("Клиент: 62",   t->k1,t->summ,
                  t->dpr,23);
            place;
    top(22):send("Поставщик: 60",t->k1,t->summ,
                  t->dpr,23);
            place;
    top(23):direct("Бухгалтерия",t->updown);
            place;      // Возврат на верхний слой
    //=============================================
    // "Дно" структурной модели
    //=============================================
    fault(123);
    }
  modend("Results.doc",1,30,page);
  return 0;
  }
```

Результаты моделирования, полученные автоматически после одного прогона модели, показаны в табл. 8.12.

Таблица 8.12

Результаты моделирования

НАЗВАНИЕ МОДЕЛИ: Бизнес-процесс							Лист: 1
ВРЕМЯ МОДЕЛИРОВАНИЯ: 734.23							
№ узла	Наименование узла	Тип узла	Загрузка, % Путь, км	M[t] среднее	Счетчик входов	Осталось транзактов	Состояние узла
1	Заказы	ag	–	14,00	0,00	1	Открыт
2	Развилка 1	creat	–	0,00	1,00	0	Открыт
3	Ожидание кредита	queue	–	3,47	4,13	0	Открыт
4	Разрешение выдачи	key	%=48,9	7,32	1,85	0	Открыт
5	Перевод кредита	pay	–	0,00	1,00	0	520 000,00 р
6	Запрет выдачи	term	–	3,47	4,13	0	Открыт
7	Плата поставщикам	pay	–	0,00	1,00	0	520 000,00 р
8	Очередь заказов	queue	–	0,06	25,04	0	Открыт
9	Выполнение заказа	serv	%=48,6	14,00	0,00	1	Открыт
10	Развилка 2	creat	–	0,00	1,00	0	Открыт
11	Возврат кредита	pay	–	0,00	1,00	0	510 000,00 р
12	Разрешение выдачи	term	–	8,90	0,97	0	Открыт
13	Заказ выполнен	term	–	18,58	0,10	0	Открыт
14	Клиенты	ag	–	7,12	0,11	1	Открыт
15	Склад	attach	%=2,8	0,38	35,84	1	1200 s 68 d
16	Отпуск товара	manage	–	0,00	1,00	1	Открыт
17	Оплата покупки	pay	–	0,00	1,00	1	515 235,00 р
18	Товар оплачен	term	–	0,38	35,84	0	Открыт
19	Сч.90: Банк	send	–	0,00	0,00	1	9990000,00 s 0,00 d
20	Сч.51: Р / счет	send	–	6,64	1,38	1	5235,00 s 0,00 d
21	Сч.62: Клиенты	send	–	0,00	0,00	1	9484765,00 s 0,00 d
22	Сч.60: Поставщики	send	–	0,00	1,00	0	520000,00 s 0,00 d
23	Бухгалтерия	direct	–	0,00	1,00	1	Открыт

8.7
МОДЕЛЬ «МУНИЦИПАЛЬНЫЕ ПРОЕКТЫ ИНВЕСТОРОВ-ЗЕМЛЕПОЛЬЗОВАТЕЛЕЙ»

Одним из важнейших условий привлекательности для инвестора той или иной территории Москвы являются:

- стоимость аренды земельного участка (включая оплату права аренды и арендные платежи);
- ТЭЗ – принадлежность к конкретной территориально-экономической зоне Москвы.

Рассмотрим пример. В Москве район «Очаково-Матвеевское» состоит из селитебной зоны – микрорайоны Матвеевское, Очаково и Аминьево, которые, обладая схожими градостроительными характеристиками, относятся к разным ТЭЗ (табл. 8.13): Матвеевское – к ТЭЗ 18, Очаково и Аминьево – к ТЭЗ 17. Промышленная зона относится к ТЭЗ 34. По базовым ставкам, используемым при расчете стоимости прав аренды земельного участка, соотношение между зонами следующее:

(ТЭЗ 17) : (ТЭЗ 18) = 1,2 : 1,0;
(ТЭЗ 18) : (ТЭЗ 34) = 1,8 : 1,0;
(ТЭЗ 17) : (ТЭЗ 34) = 2,0 : 1,0.

По описанию ТЭЗ района можно сделать заключение о том, что при эквивалентной привлекательности для инвестора земельных участков в микрорайонах Очаково и Матвеевское стоимость прав аренды в этих микрорайонах отличается на 20 %. Резкая разница между зоной микрорайона Очаково и Промышленной зоной приводит к тому, что в пределах одной транспортной развязки (Проектируемый проезд 1980) стоимость прав аренды на эквивалентные, с точки зрения инвестора, земельные участки может отличаться в 2 раза. При определенных условиях расчетная стоимость права аренды, определяемая по утвержденным методикам Москомзема, может быть увеличена или уменьшена в разрешенных пределах.

Увеличение стоимости права аренды земельного участка имеет положительные и отрицательные моменты. Положительным является увеличение разового денежного поступления в местный бюджет. Но, с другой стороны, увеличение стоимости прав аренды приводит к тому, что:

- увеличивается время поиска инвестора;
- количество потенциальных инвесторов уменьшается;
- увеличивается срок начала освоения земельного участка.

Таблица 8.13

Среднерыночная стонмость права долгосрочной аренды земельных участков

Номер оценочной зоны (ТЭЗ)	Среднерыночная стоимость права долгосрочной аренды 1 га, тыс. долл.	
	в зоне	в подзоне
1	8000	9000
2	4200	–
3	4400	–
4	5000	–
5	4300	–
6	4600	5500
7	5000	5000
8	5000	5500
9	5000	–
10,12,15,17	1000	1500
11	1200	1500
13	900	1500
19	1000	1500
14,20,36	700	–
16	860	–
18	750	–
21,22	500	–
23	380	–
24	400	800
25	350	690
26,27	320	690
28	500	750
29	290	690
30	350	690
31	400	690
32	380	–
33,34	500	–
35,37	400	–
38,39	240	–
46	300	–
40–45,47–69	260	–

Уменьшение стоимости права аренды приводит к уменьшению разового поступления в местный бюджет, однако количество потенциальных инвесторов на освоение земельного участка увеличивает-

ся, уменьшается время поиска инвестора и срок начала освоения участка. Кроме того, снижение размеров оплаты права аренды может позволить местным органам исполнительной власти более активно привлекать инвесторов к участию в районных, окружных социально-экономических и градостроительных программах.

Разовая оплата значительных сумм за право аренды экономически невыгодна инвестору, так как снижает эффективность оборотных средств. Уменьшение суммы за право аренды при наличии такой возможности или ее рассрочка позволяет органам исполнительной власти получить согласие арендатора на освоение земельного участка и регулярные выплаты в местный бюджет. При соответствующем определении обязательств можно добиться не только устранения негативных моментов, возникающих при уменьшении или рассрочке платежей за право аренды, но и получить достаточно хороший экономический эффект или повысить привлекательность земельного участка.

Опыт практической работы с инвесторами в районе «Очаково-Матвеевское» показывает следующее: добиваясь уменьшения стоимости права аренды земельного участка ориентировочно на 20%, можно получить денежных поступлений от арендатора в местный бюджет на 30 – 50% больше за время аренды по сравнению с суммой, на которую ранее была занижена стоимость права аренды земельного участка. Это достигается путем привлечения средств арендатора на реализацию социально-экономических и градостроительных программ района и округа.

Рассматривая возможности увеличения, уменьшения стоимости права аренды земельного участка и рассрочки по ее оплате, необходимо учитывать следующие факторы:

1) темпы развития инфраструктуры района, округа и улучшения их социально-экономических характеристик;

2) экономическую политику органов местной администрации;

3) финансовую надежность потенциальных инвесторов;

4) социально-экономическую ситуацию;

5) градостроительную перспективу района, округа.

По данным инвентаризации на 1997 г. только в районе «Очаково-Матвеевское» было зарегистрировано 715 земельных участков, имеющих владельца или землепользователя-арендатора. В настоящее время в этом районе имеется резерв на операции с землей на десятки лет. За 1998 г. было подписано более 100 договоров аренды: как с теми землепользователями, срок действия договора с которыми

истек, так и с новыми землепользователями. Причем были расторгнуты договоры с 15 землепользователями в основном по следующим причинам:

- землепользователь стал несостоятельным в смысле выполнения платежей по договорам аренды;
- землепользователь не реализует тот целевой проект, для которого арендовал участок (либо использует его не по назначению).

Перейдем к построению имитационной модели для целей управления таким сложным экономическим процессом. Структурный анализ начинается с рассмотрения классического «черного ящика» (слой 1, рис. 8.15), который для определенности изображается в виде узла parent. На этом этапе необходимо провести инвентаризацию источников всех входных потоков (материальных, информационных и денежных), влияющих на инвестиционный проект, и обозначить все генераторы транзактов. Такие потоки неявно формируют входное воздействие на систему $f(t)$. Соответствующие m генераторов имеют номера $f_1, f_2, \ldots, f_m$.

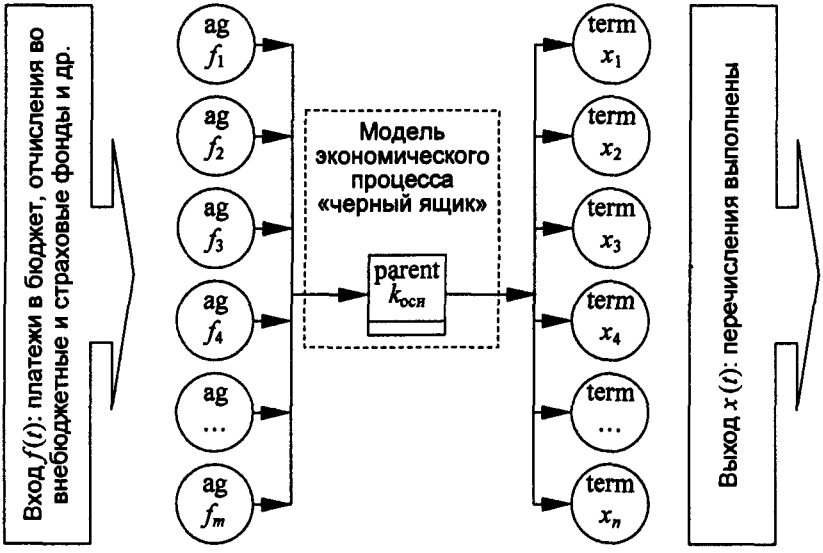


Рис. 8.15. Модель экономического процесса в виде «черного ящика» (слой 1)

Выходы модели – это совокупность терминаторов term. Терминаторы неявно влияют на выходные результаты $x(t)$. Соответствующие n генераторов имеют номера $x_1, x_2, \ldots, x_n$. Дальнейшая детализация структуры предприятия требует послойной декомпозиции «черного ящика». На рис. 8.16 (слой 2 структурного анализа) сделана декомпозиция основных процессов, связанных с материальными, информационными и денежными потоками на предприятии. На рис. 8.15, 8.16 видно, что процессы, связанные с бухгалтерскими проводками и перечислением денег, требуют дальнейшей, более глубокой декомпозиции.

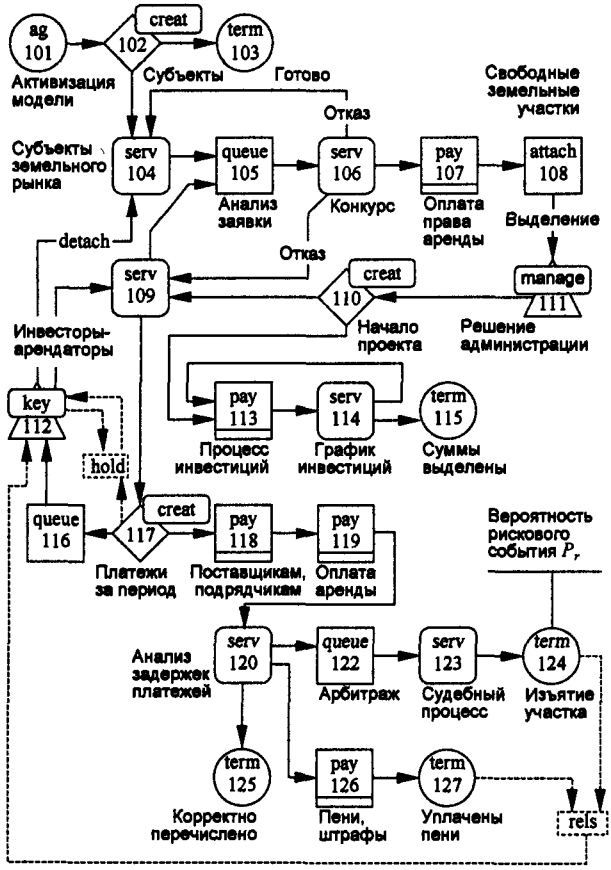


Рис. 8.16. Структурная модель взаимодействия администрации с инвесторами при реализации проекта (слой 2)

Слой 3 структурной функциональной схемы (рис. 8.17) показывает структуру операций со счетами бухгалтерского учета, реализацию требований налогов и договорных обязательств инвестора-землепользователя. Только на этом слое происходит взаимная увязка всех типов потоков.

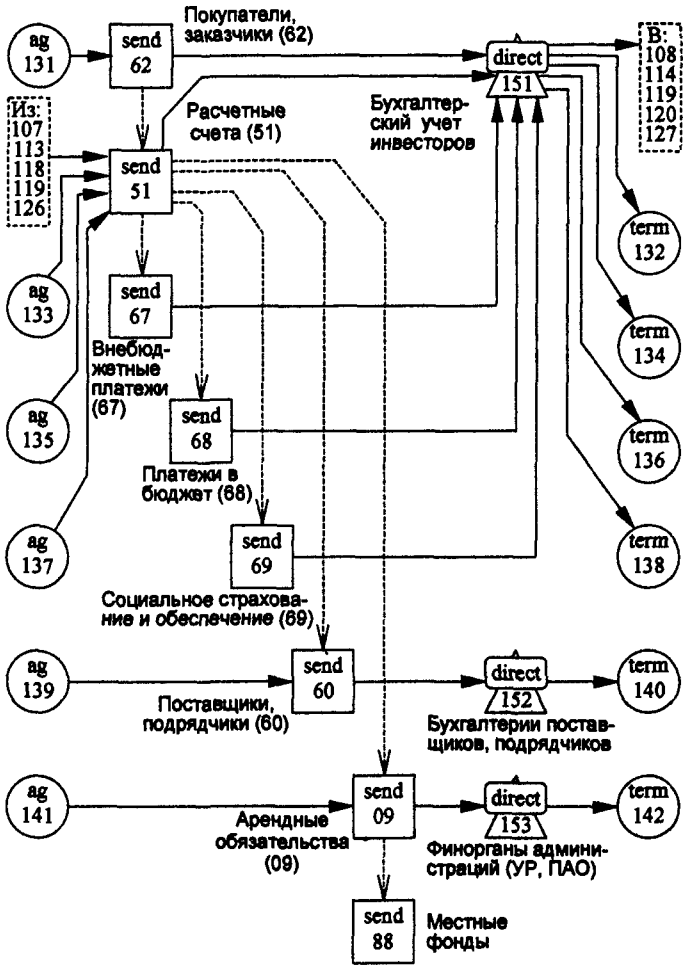


Рис. 8.17. Схема движения денежных сумм через расчетные счета инвесторов-землепользователей (слой 3)

В схеме экономического процесса, показанной на рис. 8.15 – 8.17, предполагается, что основной источник финансирования – это средства инвестора, а также привлеченные средства и, возможно, кредиты. Основным достоинством имитационной модели является возможность быстрого создания альтернативных решений:

- финансирования за счет 100%-ного единовременного авансирования работ;
- инвестирования работ по разработанному графику;
- привлечения компаньонов и кредитов.

Проработка альтернативного варианта означает лишь частичное изменение графической схемы модели, за которым последует частичное изменение ее структуры. Моделирование в данном случае является разновидностью макетирования на интеллектуальном уровне. Чем более высокоуровневыми средствами располагают система или пакет имитационного моделирования, тем более качественную модель можно создать: при тенденции сокращения времени создания или модернизации модели.

Следует отметить, что на практике «черный ящик» узлом модели не представляется; узел parent абстрактный, в тексте Pilgrim-модели он не генерируется. Основная польза слоя 1 состоит в формализации входов и выходов.

Вероятность неизвестного рискового события P_r – это вероятность события в имитационной модели. В данном случае (*см.* рис. 8.16) такое событие связано с изъятием участка несостоятельного инвестора-землепользователя и возвратом его в набор свободных земельных ресурсов (хотя можно рассматривать и более мягкие варианты). Изъятие означает выход транзакта из узла 123 для последующего уничтожения. Модель позволяет не только уточнить эту вероятность, но и высказать гипотезу о законе распределения времени наступления такого события.

При построении конкретных имитационных моделей использовались типовые схемы логистических потоков материальных и денежных средств на предприятии (рис. 8.18). В расчетах использовались предпосылки и исходные данные, показанные на примере реального инвестиционного проекта (табл.8.14).

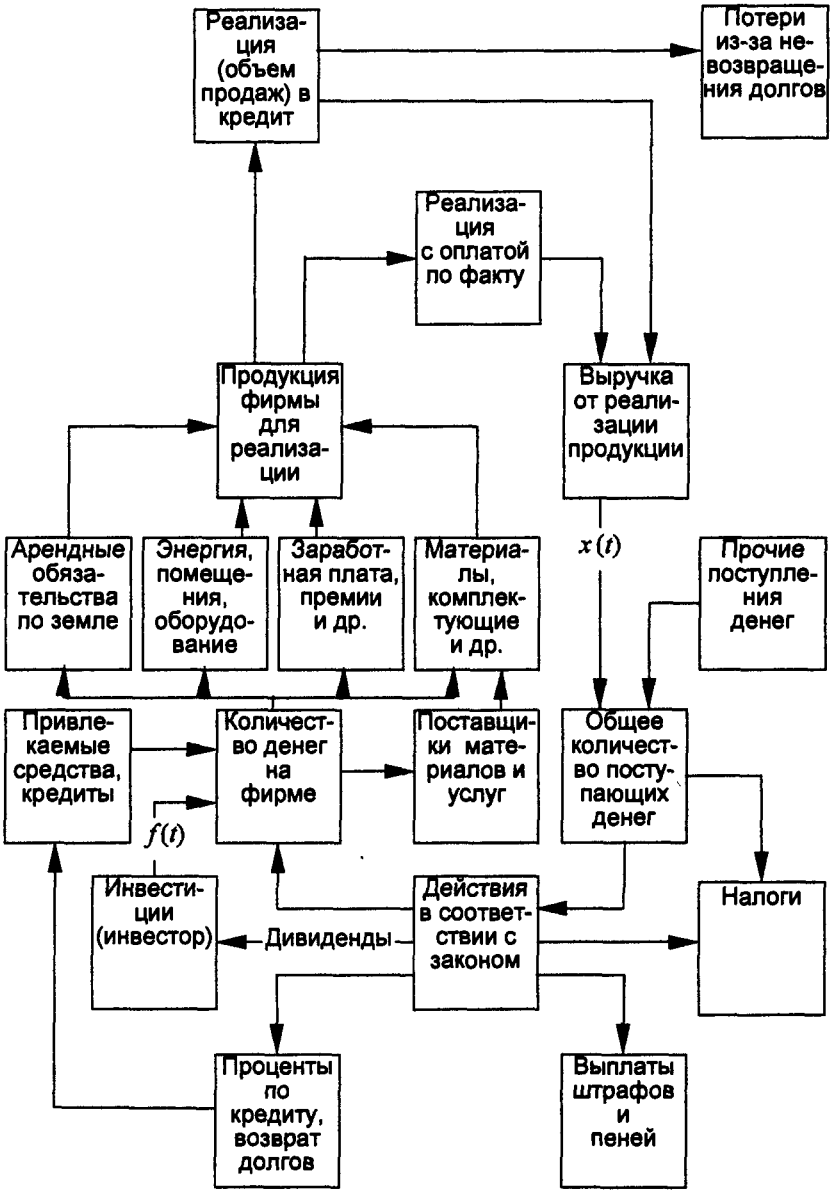


Рис. 8.18. Схема потоков материальных и денежных средств

Таблица 8.14

Структура затрат и доходов инвестора на земельном участке размером 1 га (ТЭЗ 18)

Номер статьи	Вид расходов или доходов по проекту	Сумма, долл.
1	Расходы на реализацию инвестиционного проекта ($t \geq t_р$)	37 500 000
1.1	Денежные ресурсы, изъятые из оборотных средств инвестора-землепользователя (объем инвестиций):	30 000 000
	заключение архитектурно-правового управления;	1000
	исходно-разрешительная документация в составе градостроительного заключения;	2000
	разработки и утверждение проекта объекта на земельном участке;	5000
	плата за право аренды земельного участка (за 49 лет);	2 500 000
	затраты на развитие района (расширение материально-технической базы школ, спортивных сооружений и др.) по трехстороннему контракту с ПАО и УР за $t_р = 3$ года;	600 000
	маркетинг в связи с возможной продажей объектов (зданий, сооружений), готовых к эксплуатации;	2000
	арендная плата за земельный участок, 2% стоимости за $t_р = 3$ года;	45 000
	затраты на строительство (включая стоимость проектирования)	19 345 000
1.2.	Неполученная прибыль в связи с отвлечением оборотных средств	7 500 000
2	Затраты на эксплуатацию объекта ($t > t_р$) в год	
2.1	Эксплуатация объекта (зданий, сооружений, объектов культуры), построенного или реконструированного в процессе реализации инвестиционного проекта	Параметр
2.2	Арендная плата за земельный участок, 2% стоимости в год	15 000
2.3	Платежи по трехстороннему контракту с ПАО и УР в год	600 000
3	Регулярные доходы инвестора после завершения инвестиционного проекта – доход в результате эксплуатации объекта (или его части) в год	Определяется в модели
4	Разовые доходы инвестора после завершения инвестиционного проекта ($t > t_р$) – выручка от продажи объекта (или его части)	Параметр

В табл. 8.14 применены следующие сокращения:

УР – управа района;

ПАО – префектура административного округа.

Инвестор-землепользователь получает землю по договору аренды для реализации инвестиционного проекта в течение времени $t_р$. Теоретически могут быть следующие результаты деятельности инвестора-землепользователя:

- по завершении времени $t_р$ он сам эксплуатирует частично или полностью объект (производственный объект), переоформив договор аренды;
- он частично или полностью продает объект (причем будущими землевладельцами прямо или косвенно станут покупатели этого объекта);
- он терпит неудачу и вынужден прекратить реализацию своего проекта.

Землепользователь начинает свой инвестиционный проект только в том случае, если после его реализации (при $t > t_р$) получаемые ежегодные доходы превысят ежегодные текущие затраты и обеспечат компенсацию всех расходов на реализацию инвестиционного проекта, понесенных за время $t_р$. Предположим, что для проведения инвестиционного проекта перечисляются инвестиционные суммы начиная с момента времени t =0.

Из табл. 8.14 видно, что стоимость проекта составляет 30 000 000 долл. Причем на строительство идет приблизительно две трети суммы, а одна треть – это платежи, связанные с земельным участком.

Ниже приведен текст одного из вариантов двухслойной структурной модели «Муниципальные проекты инвесторов-землепользователей», реализованной в терминах системы имитационного моделирования Pilgrim (рабочий вариант). Структурная схема и исходные данные этой модели были подробно рассмотрены выше. Текст модели снабжен комментариями к исходным данным:

```
#include <Pilgrim.h>     // ВРЕМЯ  измеряется в днях
forward          // ПЛОЩАДЬ земли измеряется в гектарах
  {              // ДЕНЬГИ измеряют в тыс. долл.
  float Mod_time=17885.0; // Моделируем 49 лет
  float T_subj=365.0;     // Инт. появления проекта
  float T_cust=7.0;       // Инт. Поступления заказов
  float T_fisk=365.0/4; // Инт. платежей в бюджет
  float T_juli=15.0;      // Инт. задержки платежей
  float T_fond=365.0/4; // Инт. пополнения фондов
  float T_aren=365.0/4; // Инт. арендных платежей
```

```
float T_conk=15.0;     // Инт. рассмотрения заявки
float T_admi=15.0;     // Инт. анализа задержек
float T_sude=15.0;     // Рассмотрение дела суде
float T_inve=150.0;    // Поступление инвестиций
float V_capi=1000000.00;// Капиталы инвесторов
float V_cust=1000000.00;// Средства заказчиков
float V_inve=1000.00;    // Стоимость проекта
float V_plat=1000.00; // Сумма от заказчика
float V_comp=1000.00; // Компаньонам
float V_prav=1000.00; // Стоимость права на 1 га
float V_aren=100.00;  // Арендная плата за 1 га
float V_vneb=1000.00; // Патеж во вн./бюдж. фонды
float V_budj=1000.00; // В среднем в бюджет
float V_socs=1000.00; // В среднем в соцстрах
float P_obor=10.00;    // «Поборы» с оборота (%)
float P_peni=2.00;     // Размер пеней (проценты)
float S_faze=0.10;// Средняя площадь участка (га)
int N_subj=150;  // Субъекты зем. рынка
int N_faze=150;  // Число земельных участков
int N_conk=2;     // Число параллельных заявок
int N_sude=2;     // Параллельное судопроизводство
int N_part=3;     // Число инвестиционных платежей
int A=1;  // Условие выигрыша земельного конкурса
int B=1;  // Условие перечисления сумм инвестиций
int C=1;  // Правильность арендных платежей
int D=1;  // Условие штрафов, пеней
modbeg("Земля-Инвестор",200,Mod_time,
       (long)time(NULL),none,51,none,42,none);
ag("Активизация мод.",101,none,none,1.0,
   zero,zero,102);
ag("Плат.Зак.-Пок.",131,132,expo,T_cust,
   zero,zero,62);
ag("Плат.Внебюджет",133,134,none,
   T_fisk+expont(T_juli),zero,zero,67);
ag("Плат.Бюджетные",135,136,none,
   T_fisk+expont(T_juli), zero, zero, 68);
ag("Плат.Соцстраху",137,138,none,
   T_fisk+expont(T_juli), zero, zero, 69);
ag("Бухг.Поставщик",139,none,expo,T_fond,
   zero, zero, 60);
ag("Бухг.Арендные",141,none,expo,T_fond,
   zero, zero, 9);
assign(51,add,V_capi);  // Капитал инвесторов
assign(62,add,V_cust);  // Фонды покупателей
supply(108,none,N_faze);// Число зем. участков
network(dummy,dummy)
  {
  //-----------------------------------------
  //  Слой 1. Схема основного бизнес-процесса
  //-----------------------------------------
  top(102): creat ("Субъекты рынка",0,N_subj,
                   none,104,103);
            place;
```

```
top(103): term ("Спрос готов");
          place;
top(104): serv ("Субъект-заявка",N_subj,none,
                 expo,T_subj,zero,zero,105);
          place;
top(105): queue ("Анализ заявок",none,106);
          place;
top(106): if (A) // Инвестор выиграл конкурс ?
            t→iu0=107;
          else   // Инвестор не выиграл конкурс
            t→iu0=104;
          serv ("Конкурс",N_conk,none,expo,
                 T_conk,zero,zero,t→>iu0);
          place;
top(107): pay("Право аренды",9,V_prav,51,
              none, 51, 108);
          place;
top(108): attach("Своб.участки",1,none,111);
          place;
top(109): serv("Инвест.-аренд.",N_subj,none,
                expo,T_aren,zero,zero,115);
          place;
top(110): creat("Начало проекта",0,1,
                 none,112,109);
          place;
top(111): manage("Отпускаем товар",110);
          place;
top(112): pay("Инвестиции",60,V_inve,51,
              none,51,113);
          place;
top(113): if(B)
            t→iu0=114; // Средства перечислены
          else
            t→iu0=112; // Продолжение
          serv ("График инвес.",1,none,norm,
                 T_inve,15.0,zero,t→iu0);
          place;
top(114): term("Суммы выделены");
          place;
top(115): creat ("Платеж периода",0,1,
                  none,116,109);
          place;
top(116): pay("Опл.участникам",62,V_comp,51,
              none, 51, 113);
          place;
top(117): pay("Арендная плата", 9,V_aren,51,
              none, 51, 113);
          place;
top(118): term ("Корректно пер.");
          place;
top(119): if(C)
            t→iu0=118; // Корректно перечислено
```

```
            else if(D)
              t→iu0=123; // Штрафы, пени
            else
              t→iu0=120; // Арбитраж
            serv("Анализ платеж.",1,none,norm,
                 T_admi,T_admi/3.0,zero,t→iu0);
            place;
top(120): queue("На арбитраж",none,121);
            place;
top(121): serv ("Суд. процесс",N_sude,none,
                 norm,T_sude,T_sude/3.0,
                 zero, 122);
            place;
top(122): term("Участок изъят");
            place;
top(123): pay("Пени,штрафы",9,
                0.01*P_peni*V_aren*S_faze,
                51,none,51,124);
            place;
top(124): term ("Уплачены пени");
            place;
//-----------------------------------------
//  Слой 2. Основные счета
//-----------------------------------------
top (62): send("Покупат.Заказ.",51,V_plat,
                 none,151);
            place;
top (51): send("Инвес.капиталы",t→k1,t→summ,
                 t→dpr,151);
            place;
top (67): send("Внебюдж. плат.",100,V_vneb,
                 none,151);
            place;
top (68): send("Плат. в бюджет",100,V_budj,
                 none,151);
            place;
top (69): send ("Соцстрах и др.",100,V_socs,
                 none, 151);
            place;
top (60): send ("Комп.-Подрядч.",88,
                 0.01*P_obor*addr[60]→saldo,
                 none, 152);
            place;
top ( 9): send ("Аренд.обязат. ",88,
                 V_aren*S_faze, none, 153);
            place;
top (88): send ("Местные фонды ",t→k1,t→summ,
                 t→dpr, 153);
            place;
//-----------------------------------------
//  Работа бухгалтерий
//-----------------------------------------
```

```
    top(151): if(t→pr > 0)
                t→updown = t→pr;
              direct("Бух.инвесторов",t→updown);
              place;
    top(152): direct("Бух.подрядчик. ",140);
              place;
    top(153): direct("Бух.адм.УР, ПАО",142);
              place;
    fault(123);
    }
  modend( "ModelRes.doc",1,30,page);
  return 0;
  }
```

Выводы

1. Задачу коммивояжера можно решать совместно с имитационным моделированием. Причем при большом числе населенных пунктов алгоритм «двух вертолетов» всегда дает решение, лучшее по сравнению с алгоритмом «ближайшего непосещенного города» и не уступающее по точности другим методам математического программирования. Это происходит потому, что оптимизация осуществляется двунаправленным одновременным просмотром. Алгоритм «ближайшего непосещенного города» начинает запутывать маршрут и при большом числе пунктов «уводить» его от оптимального, так как в конец маршрута заглянуть невозможно. Если иметь средства автоматического отображения маршрута (их предоставляет Pilgrim), то неоптимальные участки сразу видны в результатах.

2. Задача взаимного исключения возникает в следующих случаях:

- при создании сложных программных мультипроцессных средств;
- для моделирования логики экономического (часто логистического) процесса. Аналитические методы решения этой задачи отсутствуют.

3. При моделировании автоматизированной системы бухгалтерского учета применение методов исследования операций становится невозможным по двум причинам: во-первых, число бухгалтеров не может быть большим и, во-вторых, присутствует фактор неисправности. Если попытаться создать математическую модель с помощью, например, аппарата вложенных цепей Маркова (метод Кендалла) или аппарата полумарковских процессов, то придется ввести большое число допущений, которые сделают погрешность метода при определении t_q крайне большой (буквально «плюс-минус в несколь-

ко раз»). Обе отмеченные причины приводят к тому, что от построения математической модели приходится отказаться. Поэтому применяется метод имитационного моделирования.

4. Решение задачи минимизации производственных затрат, основанное на применении только двух первых моментов динамических параметров производственного процесса, как показывает практика, может дать решение, которое хуже оптимального в смысле потерь (из рассмотренного примера – потери в 1,5 раза больше). Правильные решения можно найти только с помощью имитационной модели.

5. Групповые потоки запросов, возникающие в локальных вычислительных сетях, не позволяют применять аналитические методы теории стохастических сетей для расчета трафика задержек в сети и эффективности всей системы, поскольку погрешность при этом такова, что реальные величины соответствующих показателей в несколько раз хуже расчетных. Имитационная модель позволяет получать реальные результаты с хорошей доверительной вероятностью.

6. Модель оценки эффективности предприятия, выпускающего продукцию, реализующего ее на конкурентном рынке, закупающего полуфабрикаты и пользующегося банковскими кредитами, состоит из трех независимых процессов: основного производственного процесса, процесса закупок полуфабрикатов и процесса сбыта.

Все процессы автономны. Они могут взаимодействовать только опосредованно: через банковские счета и счета бухгалтерского учета. Применение методов исследования операций в таких случаях зачастую затруднительно или невозможно. Но имитационное моделирование решает эту проблему.

7. Моделирование процесса реализации инвестиционного проекта инвесторов-землепользователей имеет следующие характерные особенности: инвесторы и муниципальные власти находятся во взаимной зависимости; существует заранее неизвестное оптимальное соотношение между платой за право аренды, арендными платежами и динамикой внесения этих средств инвесторами в местные бюджеты; существует неизвестная (для каждого проекта – уникальная) однозначная связь между объемами финансирования проекта, сроком реализации проекта, графиком перечисления инвестиционных сумм и параметрами риска, причем рискуют и инвесторы-землепользователи, и муниципальные власти.

Установить временные и финансовые параметры этих особенностей с помощью только аналитических методов и теории бухгалтерского учета невозможно. Но структурный анализ процесса реализации проекта позволяет построить имитационную модель.

Вопросы для самопроверки

1. Если маршрут, построенный с помощью алгоритма «ближайшего непосещенного города» выглядит практически без изъянов, он действительно короткий, на нем нет петель (т.е. выполнено необходимое условие построения оптимального маршрута), то чем он плох?
2. Какой узел имитационной модели может замерять пройденный путь?
3. Как вертолет коммивояжера в модели ориентируется на местности?
4. Можно ли в модели учитывать времена задержек коммивояжера в населенных пунктах?
5. В постановке задачи взаимного исключения нет ни одного узла обслуживания, требующего на обслуживание какое-то время. Тогда почему возникают значительные задержки транзактов?
6. Каков принцип работы узла типа delet в модели «Стоянка маршрутного такси» при решении задачи взаимного исключения?
7. Почему интервал прихода пассажиров в модели «Стоянка маршрутного такси» имеет экспоненциальное распределение?
8. Почему интервал генерации транзактов, имитирующих такси в этой же модели, имеет нормальное распределение?
9. Какая основная причина мешает в модели «Эффективность компьютеров в автоматизированной бухгалтерии» применить формулу Поллачека–Хинчина?
10. Какое решение обойдется дешевле для бухгалтерии: купить более надежный компьютер или поставить еще один аналогичный с той же надежностью?
11. Когда интервал времени между двумя последовательными поступлениями документов из двери рабочей комнаты на столик в компьютерном зале распределен не по экспоненциальному, а по нормальному закону?
12. По каким причинам компьютеры не смогут справиться с обработкой документов в модели «Эффективность компьютеров в автоматизированной бухгалтерии»?
13. Почему аналитическое решение в модели «Минимизация производственных затрат» неправильное?
14. Какие две основные тенденции показывают на возможность существования оптимального решения в этой модели?

15. Почему поиск оптимального решения методом перебора начинается со значений $i = 3$, $j = 3$, с последующим увеличением i и j в модели «Минимизация производственных затрат»?
16. Для чего нужна схема зарядки в этой модели?
17. Почему модель «Динамическое распределение ресурсов в сети под управлением Unix» нельзя исследовать как замкнутую стохастическую сеть?
18. Что показывает «состояние транзакта» в параметре t→iu0?
19. Какие ресурсы влияют на быстродействие работы в вычислительной сети не меньше, чем быстродействие и число процессоров на сервере?
20. Для чего нужна схема зарядки в модели «Динамическое распределение ресурсов в сети под управлением Unix»?
21. Сколько независимых (автономных) процессов содержит модель бизнес-процесса «Эффективность предприятия»? Является ли процесс получения кредита автономным?
22. Какими параметрами можно воздействовать на эту модель, чтобы получить закономерности спроса и предложения при условии рентабельности фирмы?
23. Через какие ресурсы автономные процессы оказывают влияние друг на друга?
24. Каким образом имитируется производственная мощность фирмы?
25. Каковы положительные и отрицательные стороны увеличения стоимости права аренды в модели «Муниципальные проекты инвесторов-землепользователей»?
26. К чему приводит уменьшение стоимости права аренды?
27. Почему невыгодна разовая оплата стоимости права аренды? Какие негативные последствия могут ожидать инвестора при разовой оплате?
28. Как сочетаются противоречивые интересы инвесторов-землепользователей и муниципальных властей в этой модели?
29. Можно ли найти рациональные решения по объемам платежей, срокам оплаты и стоимости арендных платежей, устраивающие инвесторов-землепользователей и муниципальные власти в этой модели? Возможно ли оптимальное решение?

1. ОПИСАНИЕ РЕСУРСОВ ДЛЯ СОЗДАНИЯ ДИАЛОГОВОГО ОКНА УПРАВЛЕНИЯ МОДЕЛЬЮ

Трактовка основных операторов языка описания ресурсов Visual C++ при создании диалогового окна, с помощью которого можно отображать и регулировать параметры модели в процессе ее выполнения, приведена ниже. Это описание содержится в файле UserRes.rc.

1. Оператор CAPTION. Определяет заголовок диалога.

Синтаксис:

CAPTION Text

Пример:

CAPTION «Диалоговая настройка модели»

Параметры:

Text представляет собой ASCII-строку, заключенную в двойные кавычки.

2. Оператор DIALOG. Определяет диалог.

Синтаксис:

Id_name DIALOG [Loading] [Memory] X, Y, Width, Height

(квадратные скобки указывают на необязательное использование параметра)

Пример:

PARMBOX DIALOG DISCARDABLE 1, 28, 315, 202

Параметры:

- Id_name – уникальное имя;
- Loading – способ загрузки ресурса. Значение RELOAD указывает, что ресурс загружается в память при старте программы, а значение LOADONCALL – что загрузка будет при первом обращении к нему;
- Memory – определяет, как система обращается с памятью, в которой расположен ресурс. Эта память может быть зафиксированной (FIXED), перемещаемой (MOVABLE) или выгружаемой (DISCARDABLE), когда в ней нет необходимости;
- X и Y – координаты левого верхнего угла диалогового окна относительно левого верхнего угла экрана;
- Width и Height – ширина и высота окна.

Сразу за оператором DIALOG может следовать оператор STYLE, определяющий стиль диалога. Стиль задается любой комбинацией эле-

ментарных стилей окна, сформированной с помощью логического оператора ИЛИ (вертикальная черта). Например, при определении диалогов иногда используется следующая комбинация стилей: DS_MODALFRAME | WS_POPUP | WS_VISABLE | WS_CAPTION | WS_SYSMENU.

3. Оператор EDITTEXT. Определяет окна ввода регулируемого параметра.

Синтаксис:

EDITTEXT Id, X, Y, Width, Height [, Style]

(квадратные скобки указывают на необязательное использование параметра)

Пример:

EDITTEXT DM_P9, 5, 171, 55, 12

Параметры:

- Id – целое число или уникальное имя, ассоциированное с элементом управления;
- X и Y – координаты левого верхнего угла окна ввода (относительно левого верхнего угла окна диалога);
- Width и Height – ширина и высота окна;
- Style – задается любой комбинацией элементарных стилей, сформированной с помощью логического оператора ИЛИ. Обычно включают элементарные стили WS_TABSTOP, WS_GROUP, WS_VSCROLL, WS_HSCROLL и WS_DISABLE. Стили по умолчанию – это ES_LEFT, WS_BORDER и WS_TABSTOP. Другие стили задаются при описании оператора STYLE.

4. Оператор GROUPBOX. Определяет рамку группы – элемент, представляющий собой прямоугольную рамку с текстовым заголовком, используемую для визуального группирования других элементов управления.

Синтаксис:

GROUPBOX Text, Id, X, Y, Width, Height [, Style]

(квадратные скобки указывают на необязательное использование параметра)

Пример:

GROUPBOX «Параметры бизнес-процесса», 100, 5, 3, 245, 30, WS_TABSTOP

Параметры:

- Text – выводимый текст, представляет собой ASCII-строку, заключенную в двойные кавычки;
- Id – целое число или уникальное имя, ассоциированное с элементом управления;

- X и Y – координаты левого верхнего угла рамки (относительно левого верхнего угла окна диалога);
- Width и Height – ширина и высота рамки;
- Style – задается любой комбинацией элементарных стилей BS_GROUPBOX, WS_TABSTOP и WS_DISABLE, сформированной с помощью логического оператора ИЛИ. Стиль по умолчанию определяется как BS_GROUPBOX; другие стили задаются при описании оператора STYLE.

5. Оператор LTEXT. Определяет статический элемент управления, отображающий текст, который выравнивается по левому краю ограничивающего прямоугольника. Он применяется исключительно в диалогах.

Синтаксис:

LTEXT Text, Id, X, Y, Width, Height [, Style]

(квадратные скобки указывают на необязательное использование параметра)

Пример:

LTEXT "Окошко 20", –1, 214, 188, 95, 8, NOT WS_GROUP

Параметры:

- Text – выводимый текст, представляет собой ASCII-строку, заключенную в двойные кавычки;
- Id – целое число или уникальное имя, ассоциированное со статическим текстом;
- X и Y – координаты левого верхнего угла элемента управления (относительно левого верхнего угла окна диалога);
- Width и Height – размеры элемента управления;
- Style – задается любой комбинацией элементарных стилей SS_LEFT, WS_GROUP и WS_TABSTOP, сформированной с помощью логических операторов ИЛИ (вертикальная черта), НЕ (NOT). Стиль по умолчанию определяется как SS_LEFT | WS_GROUP; другие стили задаются при описании оператора STYLE.

6. Оператор PUSHBUTTON. Определяет кнопку (push-button). Кнопка представляет собой прямоугольник со слегка закругленными краями, содержащий текст или графическое изображение, посылаемые родительскому окну при воздействии на эту кнопку щелчком мыши.

Синтаксис:

PUSHBUTTON Text, Id, X, Y, Width, Height [, Style]

(квадратные скобки указывают на необязательное использование параметра)

Пример:

PUSHBUTTON "Готово", IDOK, 265, 7, 34, 12

Параметры:

- Text – текст, отображаемый на «поверхности» кнопки (ASCII-строка, заключенная в двойные кавычки);
- Id – целое число или уникальное имя, ассоциированное с элементом управления;
- X и Y – координаты левого верхнего угла элемента управления (относительно левого верхнего угла окна диалога);
- Width и Height – размеры элемента управления;
- Style – задается любой комбинацией элементарного стиля BS_PUSHBUTTON со стилями WS_DISABLE, WS_GROUP и WS_TABSTOP, сформированной с помощью логических операторов ИЛИ (вертикальная черта), НЕ (NOT). Стиль по умолчанию определяется как комбинация BS_PUSHBUTTON | WS_GROUP; другие стили задаются при описании оператора STYLE.

7. Оператор STYLE. Определяет стиль создаваемого диалога. Используется вместе с оператором DIALOG.

Синтаксис:

STYLE Stiles

Пример:

STYLE WS_POPUP | WS_CAPTION

Параметры:

Stiles – стиль диалога по умолчанию определяется как комбинация WS_POPUP, WS BORDER и WS_SYSMENU. Ниже приводятся другие элементарные стили, которые могут использоваться для определения стиля окна, и вызываемые ими эффекты.

Стиль	Эффект
DS_LOCALEDIT	Сегмент данных программы будет использоваться окнами редактирования в диалоге. Стиль устаревший (используется для Windows 95)
DS_MODALFRAME	Окно диалога имеет широкую рамку, не позволяющую изменять размеры окна
DS_NOIDLEMSG	Уничтожение в очереди сообщений сообщения WM_ENTERIDLE, которое обычно посылается при активизации диалога его родительскому окну
DS_SYSMODAL	Отображение системного модального диалога (до его завершения продолжение работы с любой программой невозможно)
WS_BORDER	Окно имеет рамку
WS_CAPTION	Окно имеет титульную полосу, в которой отображается заголовок

Стиль	Эффект
WS_CHILD	Окно порождено другим окном
WS_CHILDWINDOW	То же, что и WS_CHILD
WS_CLIPCHILDREN	В процессе рисования родительского окна не разрешается рисование в порожденных окнах
WS_CLIPSIBLINGS	Перерисовка одного из порожденных окон не влечет за собой перерисовку других
WS_DISABLED	Неактивное окно, не получающее фокуса ввода
WS_DLGFRAME	Рисуется окно с рамкой, как у обычного диалога; заголовок его отсутствует
WS_GROUP	Определяет первый элемент в группе элементов управления
WS_HSCROLL	Окно имеет горизонтальную линейку прокрутки
WS_ICONIC	Отображение окна свернутым до иконки
WS_MAXIMIZED	Отображение окна в полноэкранном представлении
WS_MAXIMIZEBOX	Окно имеет кнопку полноэкранного представления (максимизации) в правом верхнем углу
WS_MINIMIZE	Отображение окна свернутым до иконки
WS_MINIMIZEBOX	Окно имеет кнопку минимизации в правом верхнем углу
WS_OVERLAPPED	Определение окна с рамкой и заголовком
WS_OVERLAPPEDWINDOW	Определяется как комбинация стилей: WS_OVERLAPPED | WS_SYSMENU | WS_CAPTION | WS_THICKFRAME | WS MINIMIZEBOX | WS_MAXIMIZEBOX
WS_POPUP	Определение всплывающего (popup) окна
WS_POPUPWINDOW	Создание всплывающего окна со следующей комбинацией стилей: WS_POPUP| WS_BORDER | WS_SYSMENU
WS_SIZEBOX	В правом верхнем углу окно содержит элемент управления, позволяющий изменять его размеры (sizebox)
WS_SYSMENU	Окно имеет кнопку системного меню в левом верхнем углу
WS_TABSTOP	Используется для элементов управления, которые можно выбрать клавишей табуляции
WS_THICKFRAME	Окно имеет толстую рамку, используемую для изменения его размеров
WS_VISIBLE	Окно изначально видимо
WS_VSCROLL	Окно с вертикальной линейкой прокрутки

2. НАБОР ФУНКЦИЙ VISUAL C++ ДЛЯ ПРОГРАММИРОВАНИЯ ПРОБЛЕМНО-ОРИЕНТИРОВАННОГО ОКНА

Большое число функций Visual C++ дают возможность создавать графические программы. Практика создания диалоговых окон для моделей Pilgrim позволила определить конкретный набор таких функций, с помощью которых можно создавать окна различного функционального назначения. Описание этих функций приведено ниже.

Названне функции или позицнонного параметра	Трактовка функции или значения параметра
BOOL Ellipse	Изобразить эллипс
1 HDC *hdc*	Дескриптор контекста устройства
2 int *nLeftRect*	Левая крайняя *x*-координата
3 int *nTopRect*	Верхняя крайняя *y*-координата
4 int *nRightRect*	Правая крайняя *x*-координата
5 int *nBottomRect*	Нижняя крайняя *y*-координата
BOOL WINAPI DllEntryPoint	Точка входа динамической библиотеки
1 HINSTANCE *hinstDLL*	Дескриптор dll-модуля
2 DWORD *dwReason*	Параметр вызывающей функции
3 LPVOID *lpvReserved*	Зарезервирован
BOOL GetTextExtentPoint32	Вычислить ширину и высоту строки
1 HDC *hdc*	Дескриптор контекста устройства
2 LPCTSTR *lpString*	Адрес текстовой строки
3 int *cbString*	Число символов в строке
4 LPSIZE *lpSize*	Адрес структуры для размеров строки
BOOL GetTextMetrics	Поместить в буфер метрики данного шрифта
1 HDC *hdc*	Дескриптор контекста устройства
2 LPTEXTMETRIC *lptm*	Адрес структуры для метрик текста
BOOL InvalidateRect	Управление прямоугольным полем окна
1 HWND *hWnd*	Оконный дескриптор с изменяемой областью
2 CONST RECT * *lpRect*	Адрес координат окна-прямоугольника
3 BOOL *bErase*	Флажок «стереть фон»
BOOL LineTo	Провести прямую до точки, не включая ее
1 HDC *hdc*	Дескриптор контекста устройства
2 int *nXEnd*	Точка: *x*-координата
3 int *nYEnd*	Точка: *y*-координата
struct tm *localtime	Корректировка вида значения времени
1 const time_t **timer*	Указатель на сохраненное значение времени

Название фуикции или позициоиного параметра	Трактовка функции или зиачения параметра
BOOL MessageBeep	Подать звуковой сигнал
1 *uType*	Тип сигнала
int MessageBox	Создать окно и вывести в него сообщение
1 HWND *hWnd*	Оконный дескриптор с изменяемой областью
2 LPCTSTR *lpText*	Адрес выводимого текста
3 LPCTSTR *lpCaption*	Адрес оконного заголовка
4 UINT *uType*	Стиль окна (способ и «кнопка» выхода)
BOOL MoveToEx	Указать на конкретную точку экрана
1 HDC *hdc*	Дескриптор контекста устройства
2 int *X*	Точка: *x*-координата
3 int *Y*	Точка: *y*-координата
4 LPPOINT *lpPoint*	Адрес для сохранения старых координат
BOOL Pie	Провести дугу (часть эллипса)
1 HDC *hdc*	Дескриптор контекста устройства
2 int *nLeftRect*	Левая крайняя *x*-координата
3 int *nTopRect*	Верхняя крайняя *y*-координата
4 int *nRightRect*	Правая крайняя *x*-координата
5 int *nBottomRect*	Нижняя крайняя *y*-координата
6 int *nXRadial1*	Начало дуги: *x*-координата
7 int *nYRadial1*	Начало дуги: *y*-координата
8 int *nXRadial2*	Конец дуги: *x*-координата
9 int *nYRadial2*	Конец дуги: *y*-координата
BOOL Rectangle	Изобразить прямоугольник
1 HDC *hdc*	Дескриптор контекста устройства
2 int *nLeftRect*	Левая крайняя *x*-координата
3 int *nTopRect*	Верхняя крайняя *y*-координата
4 int *nRightRect*	Правая крайняя *x*-координата
5 int *nBottomRect*	Нижняя крайняя *y*-координата
HGDIOBJ SelectObject	Выбрать объект (кисть, перо, шрифт или др.)
1 HDC *hdc*	Дескриптор контекста устройства
2 HGDIOBJ *hgdiobj*	Дескриптор объекта
char *strcat	Добавить строку в «хвост» к другой
1 char **string1*	Строка назначения с текстом и *null*-символом
2 const char **string2*	Исходная строка с хвостовым *null*-символом

Название функции или позиционного параметра	Трактовка функции или значения параметра
char *strcpy	Копировать одну строку в другую
1 char **string1*	Строка назначения
2 const char **string2*	Исходная строка с хвостовым *null*-символом
size_t strlen	Получить длину строки (исключая *null*)
1 const char **string*	Адрес строки
BOOL TextOut	Вывести строку текста
1 HDC *hdc*	Дескриптор контекста устройства
2 int *nXStart*	Левая крайняя *x*-координата
3 int *nYStart*	Верхняя крайняя *y*-координата
4 LPCTSTR *lpString*	Адрес строки текста
5 int *cbString*	Число символов в строке
time_t time	Получить системное время
1 time_t **timer*	Выделенная память для результата

3. ЗНАЧЕНИЯ ВЕЛИЧИНЫ *F* ПРИ 10%-ном УРОВНЕ ЗНАЧИМОСТИ

f_2	f_1											
	1	2	3	4	5	6	7	8	9	10	11	12
1	161	200	216	225	230	234	237	239	241	242	243	244
2	18,51	19,00	19,16	19,25	19,30	19,33	19,36	19,37	19,38	19,39	19,40	19,41
3	10,13	9,55	9,28	9,12	9,01	8,91	8,88	8,84	8,81	8,78	8,76	8,74
4	7,71	6,94	6,59	6,39	6,26	6,16	6,09	6,04	6,00	5,96	5,93	5,91
5	6,61	5,79	5,41	5,19	5,05	4,95	4,88	4,82	4,78	7,74	4,70	4,68
6	5,99	5,14	4,76	4,53	4,39	4,28	4,21	4,15	4,10	4,06	4,03	4,00
7	5,59	4,74	4,35	4,12	3,97	3,87	3,79	3,73	3,68	3,63	3,60	3,57
8	5,32	4,16	4,07	3,84	3,69	3,58	3,50	3,44	3,39	3,34	3,31	3,28
9	5,12	4,26	3,86	3,63	3,48	3,37	3,29	3,23	3,18	3,13	3,10	3,07
10	4,96	4,10	3,71	3,48	3,33	3,22	3,14	3,07	3,02	2,97	2,94	2,91
12	4,75	3,88	3,49	3,26	3,11	3,00	2,92	2,85	2,80	2,76	2,72	2,69
14	4,60	3,74	3,34	3,11	2,96	2,85	2,77	2,70	2,65	2,60	2,56	2,53
16	4,49	3,63	3,24	3,01	2,85	2,74	2,66	2,59	2,54	2,49	2,45	2,42
18	4,41	3,55	3,16	2,93	2,77	2,66	2,58	2,51	2,46	2,41	2,37	2,34
21	4,32	3,47	3,07	2,84	2,68	2,57	2,49	2,42	2,37	2,32	2,28	2,25
24	4,26	3,40	3,01	2,78	2,62	2,51	2,43	2,36	2,30	2,26	2,22	2,18
28	4,20	3,34	2,95	2,71	2,56	2,44	2,36	2,29	2,24	2,19	2,15	2,12
34	4,13	3,28	2,88	2,65	2,49	2,38	2,30	2,23	2,17	2,12	2,08	2,05
42	4,07	3,22	2,83	2,59	2,44	2,32	2,24	2,17	2,11	2,06	2,02	1,99
50	4,03	3,18	2,79	2,56	2,40	2,29	2,20	2,13	2,07	2,02	1,98	1,95
100	3,94	3,09	2,70	2,46	2,30	2,19	2,10	2,03	1,97	1,92	1,88	1,85
1000	3,85	3,00	2,61	2,38	2,22	2,10	2,02	1,95	1,89	1,84	1,80	1,76
∞	3,84	2,99	2,60	2,37	2,21	2,09	2,01	1,94	1,88	1,83	1,79	1,75

Продолжение

f_2	f_1											
	14	16	20	24	30	40	50	75	100	200	500	∞
1	245	246	248	249	250	251	252	253	253	254	254	254
2	19,42	19,43	19,44	19,45	19,46	19,47	19,47	19,48	19,49	19,49	19,50	19,50
3	8,71	8,69	8,66	8,64	8,62	8,60	8,58	8,57	8,56	8,54	8,54	8,53
4	5,87	5,84	5,80	5,77	5,74	5,71	5,70	5,68	5,66	5,65	5,64	5,63
5	4,64	4,60	4,56	4,53	4,50	4,46	4,44	4,42	4,40	4,38	4,37	4,36
6	3,96	3,92	3,87	3,84	3,81	3,77	3,75	3,72	3,71	3,69	3,68	3,67
7	3,52	3,49	3,44	3,41	3,38	3,34	3,32	3,29	3,28	3,25	3,24	3,23
8	3,23	3,20	3,15	3,12	3,08	3,05	3,03	3,00	2,98	2,96	2,94	2,93
9	3,02	2,98	2,93	2,90	2,86	2,82	2,80	2,77	2,76	2,73	2,72	2,71
10	2,86	2,82	2,77	2,74	2,70	2,67	2,64	2,61	2,59	2,56	2,55	2,54
12	2,64	2,60	2,54	2,50	2,46	2,42	2,40	2,36	2,35	3,32	2,31	2,30
14	2,48	2,44	2,39	2,35	2,31	2,27	2,24	2,21	2,19	2,16	2,14	2,13
16	2,37	2,33	2,28	2,24	2,20	2,16	2,13	2,09	2,07	2,04	2,02	2,01
18	2,29	2,25	2,19	2,15	2,11	2,07	2,04	2,00	1,98	1,95	1,93	1,92
21	2,20	2,15	2,09	2,05	2,00	1,96	1,93	1,89	1,87	1,84	1,82	1,81
24	2,13	2,09	2,02	1,98	1,94	1,89	1,86	1,82	1,80	1,76	1,74	1,73
28	2,06	2,02	1,96	1,91	1,87	1,81	1,78	1,75	1,72	1,69	1,67	1,65
34	2,00	1,95	1,89	1,84	1,80	1,74	1,71	1,67	1,64	1,61	1,59	1,57
42	1,94	1,89	1,82	1,78	1,73	1,68	1,64	1,60	1,57	1,54	1,51	1,49
50	1,90	1,85	1,78	1,74	1,69	1,63	1,60	1,55	1,52	1,48	1,46	1,44
100	1,79	1,75	1,68	1,63	1,57	1,51	1,48	1,42	1,39	1,34	1,30	1,28
1000	1,70	1,65	1,58	1,53	1,47	1,41	1,36	1,30	1,26	1,19	1,13	1,08
∞	1,69	1,64	1,57	1,52	1,46	1,40	1,35	1,28	1,24	1,17	1,11	1,00

4. ЗНАЧЕНИЯ Q-ПРОЦЕНТНЫХ ПРЕДЕЛОВ $T_Q(F)$

f	q								
	10,0	5,0	2,5	2,0	1,0	0,5	0,3	0,2	0,1
1	6,314	12,706	25,452	31,821	63,657	127,3	212,1	318,3	636,6
2	2,920	4,303	6,205	6,965	9,925	14,089	18,216	22,327	31,600
3	2,353	3,182	4,177	4,541	5,841	7,453	8,891	10,214	12,922
4	2,132	2,776	3,495	3,747	4,604	3,597	6,435	7,173	8,610
5	2,015	2,571	3,163	3,365	4,032	4,773	5,376	5,893	6,869
6	1,943	2,447	2,969	3,113	3,707	4,317	4,800	5,208	5,959
7	1,895	2,365	2,841	2,998	3,499	4,029	4,442	4,785	5,408
8	1,860	2,306	2,752	2,896	3,355	3,833	4,199	4,501	5,041
9	1,833	2,262	2,685	2,821	3,250	3,690	4,024	4,297	4,781
10	1,812	2,228	2,634	2,764	3,169	3,581	3,892	4,144	4,587
12	1,782	2,179	2,560	2,681	3,055	3,428	3,706	3,930	4,318
14	1,761	2,145	2,510	2,624	2,977	3,326	3,583	3,787	4,140
16	1,746	2,120	2,473	2,583	2,921	3,252	3,494	3,686	4,015
18	1,734	2,101	2,445	2,552	2,878	3,193	3,428	3,610	3,922
20	1,725	2,086	2,423	2,528	2,845	3, 153	3,376	3,552	3,849
22	1,717	2,074	2,405	2,508	2,819	3,119	3,335	3,505	3,792
24	1,711	2,064	2,391	2,492	2,797	3,092	3,302	3,467	3,745
26	1,706	2,056	2,379	2,479	2,779	3,067	3,274	3,435	3,704
28	1,701	2,048	2,369	2,467	2,763	3,047	3,250	3,408	3,674
30	1,697	2,042	2,360	2,457	2,750	3,030	3,230	3,386	3,646
∞	1,645	1,960	2,241	2,326	2,576	2,807	2,968	3,090	3,291

Алгоритмическое моделирование – разновидность имитационного моделирования с использованием универсального языка программирования (Паскаль, Бейсик или др.) и специальных алгоритмов. Более трудоемко по сравнению с применением моделирующих систем и технологий. В настоящее время применяется в некоторых вузах для преподавания основ компьютерного моделирования.

Виртуальный структурный узел – тип узла имитационной модели. Имеет наименование parent. В тексте модели он отсутствует. Позволяет объединить некоторое множество любых узлов модели и поместить их на более низкий слой, оставив на исходном слое только графический значок parent. Узел parent – мощное средство структурного анализа при создании модели. Работа с такими узлами возможна только в режиме CASE-технологии при использовании графического конструктора.

Временная динамика – основной вид динамики развития процесса, исследуемой в любых имитационных моделях.

Генератор транзактов (с бесконечной емкостью) – тип узла имитационной модели. Имеет наименование ag. Узлы-генераторы создают новые транзакты и передают их в другие узлы модели. Параметры генератора в случае необходимости можно изменить посредством информационного воздействия из другого узла – с помощью *сигнала* cheg (здесь и далее *сигнал* – это специальная функция, выполненная транзактом, находящимся в одном узле, в отношении другого узла).

Граф модели – объект имитационной модели, представляющий направленный граф, объединяющий все процессы имитационной модели независимо от количества уровней структурного анализа. Может иметь трехмерное «многослойное» изображение. Получается при структурном анализе процесса.

Датчик случайных величин – специальная программа, позволяющая получать псевдослучайные наборы чисел, распределенных по заданному закону. В современных компьютерах, если в качестве начальных кодов использовать коды таймера, реально получается последовательность случайных величин.

Замедленный масштаб времени – масштаб, задаваемый числом, выраженным в секундах. Это число меньше выбранной единицы модельного времени. Например, если в качестве единицы модельного вре-

мени выбран 1 ч, а в качестве масштаба задать число 7200, то модель будет выполняться медленнее реального процесса. Причем 1 ч реального процесса будет моделироваться в ЭВМ в течение 2 ч, т.е. примерно в 2 раза медленнее. Относительный масштаб в этом случае равен 2:1 (*см. масштаб времени*).

Имитационная модель (simulation model) – специальный программный комплекс, позволяющий имитировать деятельность какого-либо сложного объекта. Он запускает в компьютере параллельные взаимодействующие вычислительные процессы, которые являются по своим временным параметрам (с точностью до масштабов времени и пространства) аналогами исследуемых процессов. В странах, занимающих лидирующее положение в создании новых компьютерных систем и технологий, научное направление Computer Science ориентируется именно на такую трактовку имитационного моделирования, а в программах магистерской подготовки по данному направлению имеется соответствующая учебная дисциплина.

Имитационное моделирование (simulation) – распространенная разновидность аналогового моделирования, реализуемого с помощью набора математических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования, позволяющих посредством процессов-аналогов провести целенаправленное исследование структуры и функций реального сложного процесса в памяти компьютера в режиме «имитации», выполнить оптимизацию некоторых его параметров.

Имнтационное (компьютерное) моделирование экономическнх процессов – обычно применяется в двух случаях:

1) для управления сложным бизнес-процессом, когда имитационная модель управляемого экономического объекта используется в качестве инструментального средства в контуре адаптивной системы управления, создаваемой на основе информационных (компьютерных) технологий;

2) при проведении экспериментов с дискретно-непрерывными моделями сложных экономических объектов для получения и «наблюдения» их динамики в экстренных ситуациях, связанных с рисками, натурное моделирование которых нежелательно или невозможно.

Клапан, перекрывающий путь транзактам – тип узла имитационной модели. Имеет наименование key. Если на клапан воздействовать сигналом hold из какого-либо узла, то клапан перекрывается и транзакты не могут через него проходить. Сигнал rels из другого узла открывает клапан.

Коллектнвное управление процессом моделирования – особый вид эксперимента с имитационной моделью, применяемый в деловых играх и в учебно-тренировочных фирмах.

Компьютерное моделирование – иногда применяется как аналог термина *имитационное моделирование.*

Максимально ускоренный масштаб времени – масштаб, задаваемый числом «ноль». Время моделирования определяется чисто процессорным временем выполнения модели. Относительный масштаб в этом случае имеет очень малую величину; его практически невозможно определить (*см. масштаб времени*).

Масштаб времени – число, которое задает длительность моделирования одной единицы модельного времени, пересчитанной в секунды, в секундах астрономического реального времени при выполнении модели. Относительный масштаб времени – это дробь, показывающая, сколько единиц модельного времени помещается в одной единице процессорного времени при выполнении модели в компьютере.

Менеджер (или распорядитель) ресурсов – тип узла имитационной модели. Имеет наименование manage. Управляет работой узлов типа attach. Для правильной работы модели достаточно иметь один узел-менеджер: он обслужит все склады без нарушения логики модели. Чтобы различить статистику по разным складам перемещаемых ресурсов, можно использовать несколько узлов-менеджеров.

Метод Монте-Карло – метод статистических испытаний, проводимых с помощью ЭВМ и программ – датчиков псевдослучайных величин. Иногда название этого метода *ошибочно* применяется в качестве синонима *имитационного моделирования.*

Моделирующая система (система моделирования – simulation system) – специальное программное обеспечение, предназначенное для создания имитационных моделей и обладающее следующими свойствами:

- возможностью применения имитационных программ совместно со специальными экономико-математическими моделями и методами, основанными на теории управления;
- инструментальными методами проведения структурного анализа сложного экономического процесса;
- способностью моделирования материальных, денежных и информационных процессов и потоков в рамках единой модели, в общем модельном времени;
- возможностью введения режима постоянного уточнения при получении выходных данных (основных финансовых показателей, временных и пространственных характеристик, параметров рисков и др.) и проведении экстремального эксперимента.

Нормальный закон – закон распределения случайных величин, имеющий симметричный вид (функция Гаусса). В имитационных моделях экономических процессов используется для моделирования сложных многоэтапных работ.

Обобщенный закон Эрланга – закон распределения случайных величин, имеющий несимметричный вид. Занимает промежуточное положение между экспоненциальным и нормальным. В имитационных моделях экономических процессов используется для моделирования сложных групповых потоков заявок (требований, заказов).

Очередь (с относительными приоритетами или без приоритетов) – тип узла имитационной модели. Имеет наименование queue. Если приоритеты не учитываются, то транзакты упорядочиваются в очереди в порядке поступления. Когда приоритеты учитываются, транзакт попадает не в «хвост» очереди, а в конец своей приоритетной группы. Приоритетные группы упорядочиваются от «головы» очереди к «хвосту» в порядке уменьшения приоритета. Если транзакт попадает в очередь и не имеет своей приоритетной группы, то группа с таким приоритетом сразу возникнет: в ней будет один вновь поступивший транзакт.

Очередь с пространственно-зависимыми приоритетами – тип узла имитационной модели. Имеет наименование dynam. Транзакты, попадающие в такую очередь, привязаны к точкам пространства. Очередь обслуживается специальным узлом proc, работающим в режиме пространственных перемещений. Смысл обслуживания транзактов: необходимо посетить все точки пространства, с которыми связаны (или из которых поступили) транзакты. При поступлении каждого нового транзакта, если он не единственный в очереди, происходит переупорядочение очереди таким образом, чтобы суммарный путь посещения точек был минимальным (не следует считать, что при этом решается «задача коммивояжера»). Рассмотренное правило работы узла dynam в литературе называется «алгоритмом скорой помощи».

Произвольный структурный узел – тип узла имитационной модели. Имеет наименование down. Необходим для упрощения очень сложного слоя модели – для «развязывания» запутанной схемы, находящейся на одном слое, по двум разным уровням (или слоям).

Пропорционально ускоренный масштаб времени – масштаб, задаваемый числом, выраженным в секундах. Это число меньше выбранной единицы модельного времени. Например, если в качестве единицы модельного времени выбрать 1 ч, а в качестве масштаба задать число 0,1, то модель будет выполняться быстрее реального процесса. Причем 1 ч реального процесса будет моделироваться в ЭВМ в течение 0,1 с (с учетом погрешностей), т.е. примерно в 36 000 раз быстрее. Относительный масштаб равен 1:36 000 (*см. масштаб времени*).

Пространственная динамика – разновидность динамики развития процесса, позволяющей наблюдать во времени пространственные перемещения ресурсов. Изучается в имитационных моделях экономических (логистических) процессов, а также транспортных систем.

Пространство – объект модели, имитирующий географическое пространство (поверхность Земли), декартова плоскость (можно ввести и другие). Узлы, транзакты и ресурсы могут быть привязаны к точкам пространства или мигрировать в нем.

Равномерный закон – закон распределения случайных величин, имеющий симметричный вид (прямоугольник). В имитационных моделях экономических процессов иногда используется для моделирования простых (одноэтапных) работ, в военном деле – для моделирования сроков прохождения пути подразделениями, времени рытья окопов и строительства фортификационных сооружений.

Распорядитель финансов – тип узла имитационной модели «главный бухгалтер». Имеет наименование direct. Управляет работой узлов типа send. Для правильной работы модели достаточно одного узла direct: он обслужит все счета без нарушения логики модели. Чтобы различить статистику по разным участкам моделируемой бухгалтерии, можно использовать несколько узлов direct.

Реальный масштаб времени – масштаб, задаваемый числом, выраженным в секундах. Например, если в качестве единицы модельного времени выбрать 1 ч, а в качестве масштаба задать число 3600, то модель будет выполняться со скоростью реального процесса, а интервалы времени между событиями в модели будут равны интервалам времени между реальными событиями в моделируемом объекте (с точностью до поправок на погрешности при задании исходных данных). Относительный масштаб времени в этом случае равен 1:1 (*см. масштаб времени*).

Ресурс – типовой объект имитационной модели. Независимо от его природы в процессе моделирования может характеризоваться тремя общими параметрами: мощностью, остатком и дефицитом. Разновидности ресурсов: материальные (базируемые, перемещаемые), информационные и денежные.

Сигнал – специальная функция, выполненная транзактом, находящимся в одном узле в отношении другого узла для изменения режима работы последнего.

Система имитационного моделирования – иногда применяется как аналог термина *моделирующая система* (не вполне удачный перевод на русский язык термина simulation system).

Склад перемещаемых ресурсов – тип узла имитационной модели. Имеет наименование attach. Представляет хранилище какого-либо коли-

чества однотипного ресурса. Единицы ресурсов в нужном количестве выделяются транзактам, поступающим в узел attach, если остаток позволяет выполнить такое обслуживание. В противном случае возникает очередь. Транзакты, получившие единицы ресурса, вместе с ними мигрируют по графу и возвращают их по мере необходимости разными способами: либо все вместе, либо небольшими партиями, либо поштучно. Корректность работы склада обеспечивает специальный узел – менеджер.

Событие – динамический объект модели, представляющий факт выхода из узла одного транзакта. События всегда происходят в определенные моменты времени. Они могут быть связаны и с точкой пространства. Интервалы между двумя соседними событиями в модели – это, как правило, случайные величины. Разработчик модели практически не может управлять событиями вручную (например, из программы). Поэтому функция управления событиями отдана специальной управляющей программе – координатору, автоматически внедряемому в состав модели.

Структурный анализ процесса – формализация структуры сложного реального процесса путем разложения его на подпроцессы, выполняющие определенные функции и имеющие взаимные функциональные связи согласно легенде, разработанной рабочей экспертной группой. Выявленные подпроцессы, в свою очередь, могут разделяться на другие функциональные подпроцессы. Структура общего моделируемого процесса может быть представлена в виде графа, имеющего иерархическую многослойную структуру. В результате появляется формализованное изображение имитационной модели в графическом виде.

Структурный узел выделения ресурсов – тип узла имитационной модели. Имеет наименование rent. Предназначен для упрощения той части имитационной модели, которая связана с работой склада. Работа склада моделируется на отдельном структурном слое модели. Обращения на этот слой в нужные входы происходят с других слоев из узла rent без их объединения.

Структурный узел финансово-хозяйственных платежей – тип узла имитационной модели. Имеет наименование pay. Предназначен для упрощения той части имитационной модели, которая связана с работой бухгалтерии. Работа бухгалтерии моделируется на отдельном структурном слое модели. Обращения на этот слой в нужные входы происходят с других слоев из узла pay, без объединения этих слоев.

Счет бухгалтерского учета – тип узла имитационной модели. Имеет наименование send. Транзакт, который входит в такой узел, является запросом на перечисление денег со счета на счет или на бухгалтерскую проводку. Правильность работы со счетами регулируется специальным

узлом direct, который имитирует работу бухгалтерии. Если в узле send остаток денег достаточен, чтобы выполнить перечисление на другой счет, то перечисление выполняется. В противном случае в узле send образуется очередь необслуженных транзактов.

Терминатор – тип узла имитационной модели. Имеет наименование term. Транзакт, поступающий в терминатор, уничтожается. В терминаторе фиксируется время жизни транзакта.

Транзакт – динамический объект имитационной модели, представляющий формальный запрос на какое-либо обслуживание. В отличие от обычных заявок, которые рассматриваются при анализе моделей массового обслуживания, имеет набор динамически изменяющихся особых свойств и параметров. Пути миграции транзактов по графу модели определяются логикой функционирования компонентов модели в узлах сети.

Треугольный закон – закон распределения случайных величин, имеющего симметричный вид (равнобедренный треугольник) или несимметричный вид (треугольник общего вида). В имитационных моделях информационных процессов иногда используется для моделирования времени доступа к базам данных.

Узел обслуживания с многими параллельнымн каналами – тип узла имитационной модели. Имеет наименование serv. Обслуживание может быть в порядке поступления транзакта в освободившийся канал либо по правилу абсолютных приоритетов (с прерыванием обслуживания).

Узлы – объекты имитационной модели, представляющие центры обслуживания транзактов в графе имитационной модели (но необязательно массового обслуживания). В узлах транзакты могут задерживаться, обслуживаться, порождать семейства новых транзактов, уничтожать другие транзакты. В каждом узле порождается независимый процесс. Вычислительные процессы выполняются параллельно и координируют друг друга. Они выполняются в едином модельном времени, в одном пространстве, учитывают временную, пространственную и финансовую динамику.

Управляемый генератор транзактов (или размножитель) – тип узла имитационной модели. Имеет наименование creat. Позволяет создавать новые семейства транзактов.

Управляемый процесс (непрерывный или пространственный) – тип узла имитационной модели. Имеет наименование proc. Этот узел работает в трех взаимно исключающих режимах:

- моделирования управляемого непрерывного процесса (например, в реакторе);

- доступа к оперативным информационным ресурсам;
- пространственных перемещений (например, вертолета).

Управляемый терминатор транзактов – тип узла имитационной модели. Имеет наименование delet. В нем уничтожается (или поглощается) заданное число транзактов, принадлежащих конкретному семейству. Требование на такое действие содержится в уничтожающем транзакте, поступающем на вход узла delet. Он ждет поступления в узел транзактов указанного семейства и уничтожает их. После поглощения уничтожающий транзакт покидает узел.

Финансовая динамика – разновидность динамики развития процесса, позволяющей наблюдать во времени изменения ресурсов, денежных средств, основных результатов деятельности объекта экономики, причем параметры измеряются в денежных единицах. Изучается в имитационных моделях экономических процессов.

Экспоненциальный закон – закон распределения случайных величин, имеющего ярко выраженный несимметричный вид (затухающая экспонента). В имитационных моделях экономических процессов используется для моделирования интервалов поступления заказов (заявок), поступающих в фирму от многочисленных клиентов рынка. В теории надежности применяется для моделирования интервала времени между двумя последовательными неисправностями. В связи и компьютерных науках – для моделирования информационных потоков (пуассоновские потоки).

1. *Анфилатов В. С., Емельянов А. А., Кукушкин А. А.* Системный анализ в управлении / Под ред. А.А. Емельянова. – М.: Финансы и статистика, 2001. – 368 с.

2. *Берлянт А. М.* Картография. – М.: Аспект Пресс, 2001. – 336 с.

3. *Бусленко Н. П.* Моделирование сложных систем. – М.: Наука, 1978. – 399 с.

4. *Варфоломеев В. И.* Алгоритмическое моделирование элементов экономических систем. – М.: Финансы и статистика, 2000. – 208 с.

5. *Гаджинский А. М.* Практикум по логистике. – М.: Маркетинг, 2001. – 180 с.

6. *Дийкстра Э.* Взаимодействие последовательных процессов // Языки программирования / Под ред. Ф. Женюи. – М.: Мир, 1972. – С. 9–86.

7. *Дубров А. М., Мхитарян В. С., Трошин Л. И.* Многомерные статистические методы. – М.: Финансы и статистика, 2000. – 352 с.

8. *Емельянов А. А.* Имитационное моделирование в управлении рисками. – СПб.: Инжэкон, 2000. – 376 с.

9. *Емельянов А. А., Власова Е. А.* Имитационное моделирование в экономических информационных системах. – М.: Изд-во МЭСИ, 1998. – 108 с.

10. *Емельянов А. А., Мошкина Н. Л., Сныков В. П.* Автоматизированное составление оперативных расписаний при обследовании районов экстремально высокого загрязнения // Загрязнение почв и сопредельных сред. Вып. 7. – СПб: Гидрометеоиздат, 1991. – С. 46–57.

11. *Калянов Г. Н.* CASE структурный системный анализ (автоматизация и применение). – М.: Лори, 1996. – 241 с.

12. *Клейнрок Л.* Коммуникационные сети. Стохастические потоки и задержки сообщений. – М.: Наука, 1970. – 255 с.

13. *Круглински Д., Уингоу С., Шеферд Дж.* Программирование на Microsoft Visual C++ 6.0 для профессионалов. – СПб.: Питер, Русская редакция, 2001. – 864 с.

14. *Кузин Л. Т., Плужников Л. Н., Белов Б. Н.* Математические методы в экономике и организации производства. – М.: Изд-во МИФИ, 1968. – 220 с.

15. *Налимов В. В., Чернова И. А.* Статистические методы планирования экстремальных экспериментов. – М.: Наука, 1965. – 366 с.

16. *Нейлор Т.* Машинные имитационные эксперименты с моделями экономических систем. – М.: Мир, 1975. – 392 с.

17. *Ойхман Е. Г., Попов Э. В.* Реинжиниринг бизнеса. – М.: Финансы и статистика, 1997. – 336 с.

18. *Прицкер А.* Введение в имитационное моделирование и язык СЛАМ-II. – М.: Мир, 1987. – 544 с.

19. *Саати Т.* Элементы теории массового обслуживания и ее приложения. – М.: Сов. радио, 1970. – 377 с.

20. *Черемных С. В., Семенов И. О., Ручкин В. С.* Структурный анализ систем: IDEF-технология. – М.: Финансы и статистика, 2001. – 208 с.

21. *Чичерин И. Н.* Стоимость права аренды земельного участка и взаимодействие с инвесторами // Экономические информационные системы на пороге XXI века. – М.: Изд-во МЭСИ, 1999. – С. 229–232.

22. *Шеннон Р. Е.* Имитационное моделирование систем: наука и искусство. – М: Мир, 1978. – 420 с.

23. *Шрайбер Т. Дж.* Моделирование на GPSS. – М.: Машиностроение, 1979. – 592 с.

ОГЛАВЛЕНИЕ

Учебное издание

Емельянов Александр Анатольевич
Власова Екатерина Аркадьевна
Дума Роман Владимирович

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ ЭКОНОМИЧЕСКИХ ПРОЦЕССОВ

Заведующая редакцией *Л.А. Табакова*
Редактор *А.М. Маторина*
Младший редактор *Н.А. Федорова*
Художественный редактор *Ю.И. Артюхов*
Технический редактор *Т.С. Маринина*
Корректоры *Г.В. Хлопцева, Н.Н. Зубенко*
Компьютерная верстка *Е.В. Васильевской, И.В. Витте*
Оформление художника *Т.Л. Погорельцевой*

ИБ № 4469

Сдано в набор 21.02.2002. Подписано в печать 10.04.2002
Формат 60×88/16. Печать офсетная. Гарнитура «Таймс»
Усл. п. л. 22,54. Уч.-изд. л. 22,03
Тираж 4000 экз. Заказ 1398. «С» 106

Издательство «Финансы и статистика»
101000, Москва, ул. Покровка, 7
Телефон (095) 925-35-02, факс (095) 925-09-57
E-mail: mail@finstat.ru http://www.finstat.ru

Великолукская городская типография
Комитета по средствам массовой информации
Псковской области
182100, Великие Луки, ул. Полиграфистов, 78/12
Тел./факс: (811-53) 3-62-95
E-mail: VTL@MAPT.RU