Министерство образования Российской Федерации
Томский политехнический университет

Ю. Я. Кацман

МОДЕЛИРОВАНИЕ

Учебное пособие

Томск 2003

УДК 681.3
К 31

Кацман Ю. Я. Моделирование: Учеб. пособие / Том. политехн. ун-т. – Томск, 2003. – 91 с.

В пособии в краткой форме излагаются математические методы компьютерного моделирования и исследования вычислительных систем. При описании функционирования вычислительных систем используется аппарат цепей Маркова и методов стохастического моделирования. В изучаемой дисциплине дается сравнительный анализ языков имитационного моделирования, их преимущества (недостатки) в сравнении с языками программирования высокого уровня; излагаются элементы теории массового обслуживания. Теоретические знания подкрепляются на лабораторных и практических занятиях, которые посвящены изучению методов проектирования и оценки характеристик систем массового обслуживания. В пособии представлены краткие сведения по *Q*-схемам и языку имитационного моделирования GPSS. Пособие подготовлено на кафедре вычислительной техники, соответствует программе дисциплины и предназначено для студентов специальности 220100 «Вычислительные машины, системы, комплексы и сети» Института дистанционного образования.

Печатается по постановлению Редакционно-издательского Совета Томского политехнического университета.

Рецензенты:

И. В. Карелина – доцент кафедры высшей математики Северского государственного технологического института, кандидат физико-математических наук;

М. С. Квасница – доцент кафедры электронных приборов Томского университета систем управления и радиоэлектроники, кандидат технических наук.

Темплан 2003

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Сегодня трудно назвать область человеческой деятельности, в которой бы в той или иной степени не использовались методы моделирования. Обусловлено это тем, что решение многих сложных научных и технических задач значительно упрощается при моделировании, т. е. замещение одних объектов другими, обеспечивающими фиксацию наиболее существенных свойств и особенностей замещаемых объектов.

Моделирование является мощным средством анализа и синтеза сложных процессов, явлений, объектов и систем. В настоящее время оно широко применяется в автоматике, вычислительной и измерительной технике, радиотехнике и связи, термодинамике и гидродинамике, физике, математике и других областях науки и техники. Основные методы теории моделирования оказываются незаменимыми при малой изученности рассматриваемых сложных объектов, при отсутствии полного математического описания объектов. Моделирование применяется и при существовании математического описания системы (объекта), если оно слишком сложно для аналитического исследования или исследования на ЭВМ.

Моделирование на ЭВМ (*компьютерный эксперимент*) по сравнению с реальным экспериментом имеет ряд преимуществ. К ним относятся:

1. Экономия средств.

Проведение исследований в современной физике, космонавтике, в самолето- и автомобилестроении требует дорогостоящих, а порой и уникальных установок. Стоимость исследовательского атомного реактора, синхрофазотрона, автополигона и т. п. может составить сотни миллионов (миллиарды) долларов.

2. Экономия времени.

Благодаря быстродействию современных компьютеров и их относительной дешевизне, по сравнению с вышеназванными установками, результат эксперимента достигается намного быстрее. Сокращение времени достигается и за счет того, что *компьютерный эксперимент* позволяет выявить наиболее перспективные направления экспериментирования. Объем реальных (физических) исследований при этом уменьшается кратно.

3. Невозможность проведения реальных экспериментов.

При исследовании объектов микро- и макромира физическое исследование либо невозможно, либо крайне затратно. Исследование критических режимов (разрушающих нагрузок) мостов, высотных зданий, тепловых, химических и атомных установок значительно безопаснее проводить в *компьютерном эксперименте*, чем на реальном объекте (авария на Чернобыльской АЭС). Успешным запускам космических ракет, поступлению на вооружение новых образцов техники предшествуют десятки и сотни тысяч исследований на моделях.

4. Повторяемость модели.

При реальных исследованиях невозможно, при необходимости, повторить все начальные условия (модель Бородинского сражения).

5. Модель – тренажер

В ряде случаев модель может заменять исходный объект при обучении. Например, компьютерные модели широко используются в качестве тренажера при подготовке персонала к последующей работе в реальных условиях (диспетчерская подготовка).

Наряду с вышеперечисленными преимуществами моделирование обладает и рядом недостатков:

1. Справедливость выводов, полученных при анализе результатов моделирования, подтверждается их реализацией в конкретных устройствах, образцах и т. п.

2. Результаты исследований не могут отразить те зависимости, которые не были первоначально заложены в исходных характеристиках модели:

а) при моделировании закона Ома ($R = \frac{U}{I}$) мы никогда не получим зависимости сопротивления от материала проводника, площади его сечения, температуры и т. п. Причина этого заключается в том, что в исходной модели (формуле) данные зависимости никак не отражены.

б) если при моделировании зависимости потока нейтронов от количества урана мы не предусмотрим эффекта «критическая масса», то атомного взрыва в исследуемой модели не произойдет, даже при неограниченном увеличении массы урана.

1. ИСТОРИЯ РАЗВИТИЯ МОДЕЛИРОВАНИЯ

Исторически первыми моделями как заместителями некоторых объектов были символические условные модели. Ими являлись языковые знаки, естественно возникшие в ходе развития человечества и составившие разговорный язык. Первое применение символически условных моделей другого типа связано, по-видимому, с возникновением обмена в первобытном обществе. Первоначально схема общего обмена имела вид

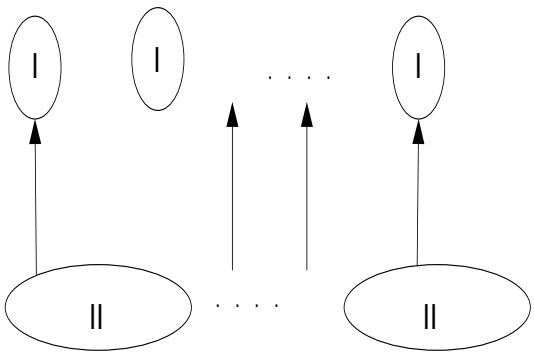

Рис. 1. Модель обмена товарами

Как видно на рис. 1, предметы раскладывались в два ряда, и этим достигалось взаимное однозначное соответствие. В процессе исторического развития было замечено, что соответствие между объектами первого и второго рода может быть установлено с помощью объектов третьего рода как посредников:

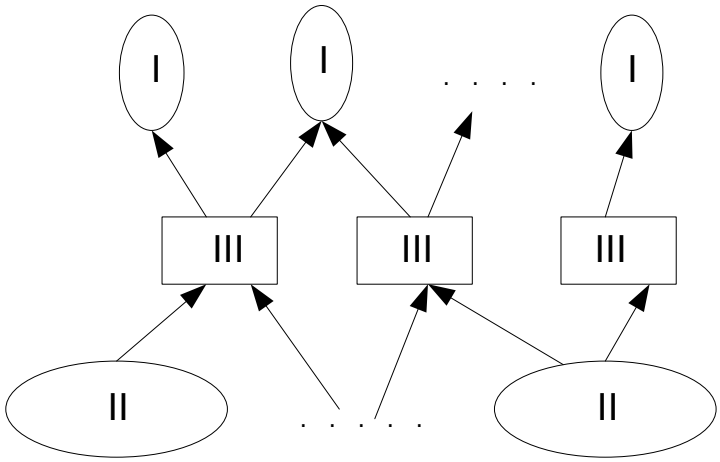

Рис. 2. Модель обмена товарами с помощью посредника

В этой схеме объекты (III) выступают в роли заместителей объектов (I) и (II). Очевидно, что этими заместителями первоначально были пальцы рук, ног, в дальнейшем палочки и … наконец числа. Можно предположить, что именно зарубки (засечки) были прототипами римских цифр I, V, X.

Дальнейшее развитие логических знаковых моделей связано с возникновением письменности и математической символики.

В древней Греции в V – III вв. до н.э. была создана геометрическая модель солнечной системы. Греческий врач Гиппократ для изучения человеческого глаза воспользовался его физической аналогичной моделью – глазом быка. Развитие физического (геометрического) моделирования, основанного на теории подобия, широко развивалось в средние века. Однако в XVI – XVIII вв. использование геометрического моделирования (теории подобия) столкнулось с определенными трудностями, выявившими его недостатки при решении ряда задач, – череда необъяснимых морских катастроф. Решение этой проблемы сформулировал Галилео Галилей (1564 – 1642 гг.): «Прочность подобных тел не сохраняет того же отношения, которое существует между величиной тел». В дальнейшем вопросами физического моделирования занимались такие великие ученые, как И. Ньютон (1643 – 1727), Л. Эйлер (1707 – 1783), И.П. Кулибин (1735 – 1818) и т. д.

В настоящее время выработалась технология исследования сложных проблем, основанная на построении и анализе с помощью ЭВМ математических моделей изучаемого объекта, системы. Такой метод исследования называют вычислительным (компьютерным) экспериментом (рис. 3).

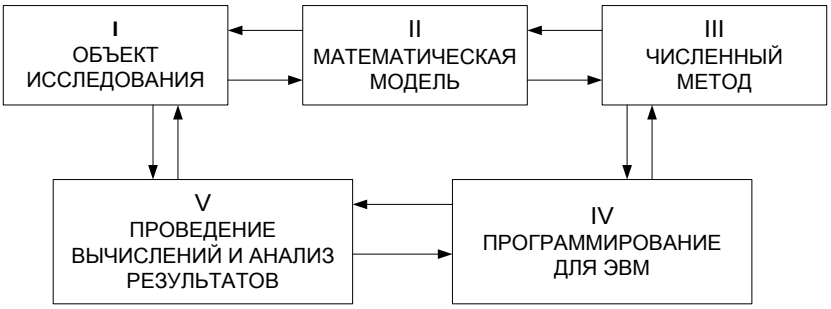

Рис. 3. Принципиальная схема компьютерного эксперимента

I – формулируются основные законы функционирования исследуемой системы (объекта).

II – строится соответствующая математическая модель, обычно представляющая запись этих законов в форме системы уравнений (алгебраических, дифференциальных, интегральных и т. п.).

III – именно на этом этапе необходимо привлечение компьютера (ЭВМ) и, как следствие, развитие численных методов. После того, как задача сформулирована в математической форме, необходимо найти ее решение. Ясно, что если даже удается доказать существование и единственность решения, то этого недостаточно. Необходимо еще изучить качественное поведение решения и найти те или иные количественные характеристики объекта.

IV – для реализации численного метода на ЭВМ необходимо разработать программу (пакет программ) или воспользоваться библиотекой стандартных подпрограмм.

V – после отладки программы в комплексе наступает этап проведения вычислительных экспериментов и анализа полученных результатов.

Полученные результаты анализируются с точки зрения их соответствия исследуемому явлению. При необходимости вносятся исправления в численный метод, уточняется математическая модель, и т. п. Таким образом, разработанная модель не является жесткой схемой, а может изменяться, т. е. носит итерационный характер.

2. СИСТЕМЫ И МОДЕЛИ

2.1. Основные понятия теории моделирования

Теория замещения одних объектов (оригиналов) другими объектами (моделями) и исследование свойств объектов на их моделях называется теорией моделирования.

В научных исследованиях большую роль играют *гипотезы,* т. е. определенные предсказания, основывающиеся на опытных данных, наблюдениях, интуиции. При формировании и проверке правильности гипотез большое значение в качестве метода суждения имеет *аналогия.*

Аналогией называется суждение о каком-либо частном сходстве двух объектов.

Современная научная гипотеза создается, как правило, по аналогии с проверенными на практике научными положениями. Таким образом, аналогия связывает *гипотезу с экспериментом.*

Модель – это объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.

Замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели называется *моделированием.*

Математическое моделирование – это замещение оригинала математической моделью, обеспечивающей фиксацию и исследование свойств и отношений оригинала, а также переход к оригиналу с помощью математических методов.

Особое значение среди математических моделей имеют *подобные,* обеспечивающие перенос данных на оригинал на основании *подобия.*

Подобие – это полная математическая аналогия при наличии пропорциональности между сходственными переменными, неизменно сохраняющаяся при всех возможных значениях этих переменных, удовлетворяющих сходственным уравнениям.

Геометрическое подобие – это подобие геометрических образов: точек, линий, поверхностей, фигур, тел. Широко распространен этот вид моделирования в архитектуре, дизайнерской работе.

Физическое подобие означает подобие физически однородных объектов. Все масштабы при этом являются безразмерными величинами. В качестве примера использования физического подобия можно привести исследование радиационной стойкости материалов и элементов радиоэлектронной техники под воздействием радиационного излучения реакторов и ускорителей. Проблема радиационной стойкости очень важна для космических объектов, работающих длительное время на орбите, и подвергающихся при этом воздействию интенсивного потока космического излучения.

Временное подобие – подобие функций времени. Данный подход часто используется в растениеводстве, биологии при выведении новых перспективных сортов растений либо при исследовании отдаленных генетических последствий при применении новых лекарств.

В теории и практике моделирования *подобие* имеет большее значение, чем *аналогия*. При *аналогии* двух объектов распространение свойств одного из них на другой носит характер предположения и нуждается в проверке. При *подобии* двух объектов знание поведения одного из них означает знание поведения другого.

В основе моделирования лежит теория подобия, которая утверждает, что абсолютное подобие может иметь место лишь при замене одного объекта другим, точно таким же. Поэтому стремятся к тому, чтобы модель достаточно хорошо отображала исследуемую сторону функционирования объекта. В соответствии с вышесказанным классификацию видов моделирования можно проводить по степени полноты моделей и разделить их в соответствии с этим признаком: *полные, неполные, приближенные*.

Система – это совокупность элементов, которые принадлежат ограниченной части реального мира, являющейся объектом исследования.

Модель – это описание системы.

2.2. Классический подход к разработке модели

Простой подход к изучению связей между отдельными частями модели предусматривает рассмотрение их как отражение связей между отдельными подсистемами объекта. Такой традиционный, классический подход может быть использован при создании достаточно простых моделей. Рассмотрим процесс синтеза модели **М** на основе классического (индуктивного) подхода (см. рис. 4).

Реальный объект, подлежащий моделированию, разбивается на отдельные подсистемы, т. е. выбираются исходные данные **Д** и формулируются цели **Ц**, отображающие отдельные стороны процесса моделирования. По отдельной совокупности исходных данных **Д** ставится цель моделирования отдельной стороны функционирования системы. На базе этой цели формируется некоторая компонента **К** будущей модели. Совокупность компонент объединяется в модель **М**. Таким образом, разработка модели **М** на базе классического подхода означает суммирование отдельных компонент в единую модель. Следует учесть, что каждая из компонент решает свои собственные задачи и изолирована от других частей модели.

Можно отметить две отличительные стороны классического подхода:

- При разработке модели наблюдается движение от частного к общему.
- Создаваемая модель (система) образуется путем суммирования отдельных, независимых компонент. При этом в разработанной модели не учитывается возникновение нового системного эффекта.

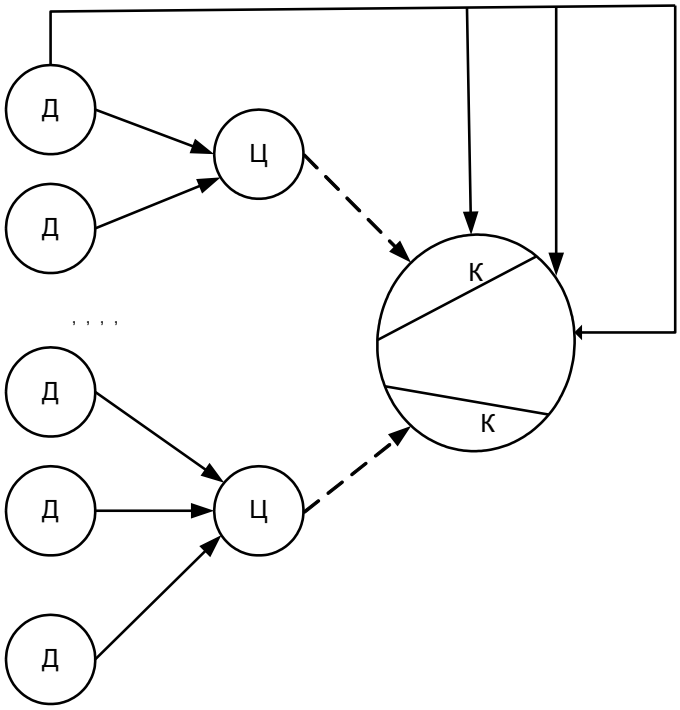

Рис. 4. Классический подход синтеза модели

2.3. Системный подход синтеза модели

Системный подход получил распространение в системотехнике в связи с необходимостью исследования больших реальных систем, когда сказалась недостаточность, а иногда ошибочность принятия каких-либо частных решений. На возникновение системного подхода повлияли увеличивающееся количество исходных данных при разработке, необходимость учета сложных стохастических связей в системе. В основе системного подхода лежит рассмотрение системы как интегрированного целого, причем это рассмотрение при разработке модели начинается с главного – формулировки цели функционирования (системы).

Рассмотрим процесс синтеза модели **М** на базе системного подхода (см. рис. 5). На основе исходных данных **Д**, которые известны из анализа внешней системы и ограничений, накладывающихся на систему сверху, либо исходя из возможностей ее реализации и на основе цели **Ц,** формулируются исходные требования **Т** к модели системы. На базе этих требований формируются ориентировочно некоторые подсистемы **П**, элементы **Э** и осуществляется наиболее сложный этап синтеза – выбор **В** составляющих системы, для чего используются специальные критерии выбора **КВ**. Здесь следует отметить, что при оценке эффективности функционирования системы или при оценке альтернативных решений требуется внесение изменений в описание системы, следовательно, перестройка модели. Поэтому на практике процесс построения модели является итеративным (см. рис. 5).

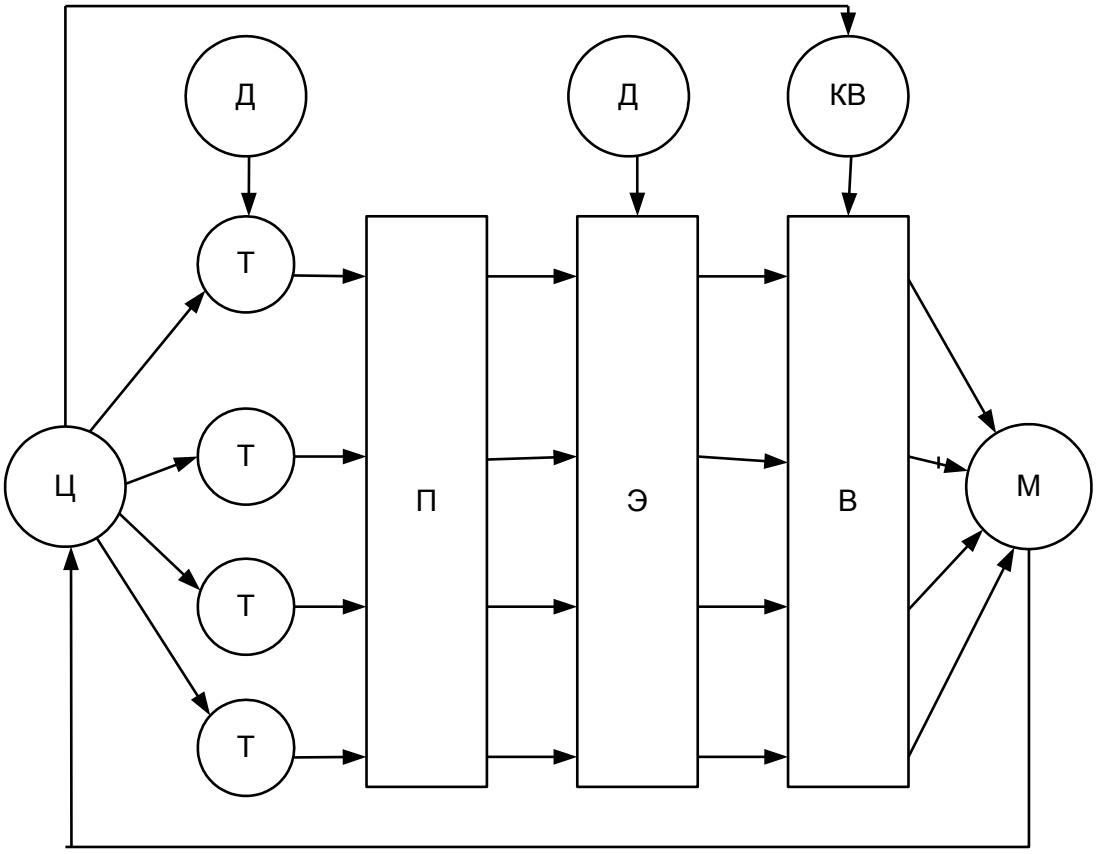

Рис. 5. Синтез модели – системный подход

Независимо от типа используемой модели **М** при ее построении необходимо руководствоваться рядом принципов системного подхода:

- пропорционально-последовательным продвижением по этапам и направлениям разработки модели;
- согласованием информационных, ресурсных, надежностных и других характеристик;
- правильным соотношением отдельных уровней иерархии в системе моделирования;
- целостностью отдельных обособленных стадий построения модели.

Использование системного подхода позволяет не только построить модель реального объекта, но и выбрать на базе этой модели необходимое количество управляющей информации в реальной системе, оценить показатели ее функционирования и тем самым на основе модели найти наиболее эффективный вариант структуры и оптимальный режим функционирования реального объекта (системы).

2.4. Классификация видов моделирования систем

Как уже было отмечено выше, первичная классификация моделей может быть проведена по степени полноты модели. Другой критерий классификации может основываться на характеристиках процессов, протекающих в исследуемом объекте.

Рис. 6. Классификация видов моделирования систем

Детерминированное моделирование отображает процессы, в которых предполагается отсутствие всяких случайных воздействий.

Стохастическое моделирование отображает вероятные процессы и события, обусловленные случайными воздействиями.

Статическое моделирование служит для описания поведения объектов в какой-либо момент времени.

Динамическое моделирование отражает поведение объекта во времени.

Дискретное моделирование служит для описания процессов, которые предполагаются дискретными, соответственно *непрерывное моделирование* позволяет отразить непрерывные процессы.

Дискретно-непрерывное моделирование используется для случаев, когда хотят выделить в системе функционирование как дискретных, так и непрерывных процессов.

В зависимости от формы представления объекта (системы) можно выделить мысленное и реальное моделирование.

Мысленное моделирование часто является единственным способом моделирования ситуаций микро- и макромира, кроме того, оно, как правило, используется на первых этапах других видов моделирования.

Мысленное моделирование может быть реализовано в виде *наглядного*, *символического* и *математического*.

При наглядном моделировании на базе представлений человека о реальных объектах создаются различные наглядные модели.

В основу *гипотетического* моделирования исследователями закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте. Обычно оно используется в тех случаях, когда знаний о реальном объекте недостаточно для построения формальных моделей.

Аналоговое моделирование основывается на применении аналогий различных уровней.

Макетирование мысленное – применяется в случаях, когда моделируемый процесс не поддается физическому моделированию, либо в случаях, когда это предшествует другим видам моделирования.

Символическое моделирование – искусственный процесс создания логического объекта, который замещает реальный и выражает основные свойства его отношений с помощью определенной системы знаков и символов.

Если ввести условное обозначение отдельных понятий, т. е. знаки, а также определенные операции между этими знаками, то можно реализовать *знаковое моделирование* и с помощью знаков отображать набор понятий – составлять отдельные цепочки из слов и предложений.

В основе *языкового моделирования* лежит некоторый *тезаурус*.

Тезаурус – словарь, в котором каждому слову может соответствовать лишь единственное понятие.

Математическое моделирование – процесс установления соответствия данному реальному объекту некоторого математического объекта, называемого математической моделью, и исследование этой модели с целью получения характеристик реального объекта.

Аналитическое моделирование – для него характерно то, что процессы функционирования элементов системы записываются в виде функциональных соотношений (алгебраических, разностных, интегро-дифференциальных и т. п.) или логических условий.

Аналитическая модель может быть исследована:

- аналитическим методом, т. е. получением решения в общем виде;
- численным методам (когда не можем решить уравнения в общем виде);
- качественным, когда, не имея решения в явном виде, можно найти некоторые свойства решения (например, оценить устойчивость решений).

Комбинированное моделирование – это разумное сочетание аналитического (на первом этапе) и имитационного (на последующих этапах) моделирования, позволяющее создавать модели более сложных систем.

Имитационное моделирование – реализует модель алгоритма, воспроизводящего процесс функционирования системы во времени, причем имитируются элементарные явления с сохранением их логической структуры и последовательности протекания во времени.

Основные преимущества имитационного моделирования по сравнению с аналитическим – возможность моделирования сложных систем.

Реальное моделирование – использует возможность исследования различных характеристик системы либо на реальном объекте, либо на его части.

Натурное моделирование – проведение исследований на реальном объекте с последующей обработкой результатов на основе теории подобия.

Производственный эксперимент и *комплексные испытания* обладают высокой степенью достоверности, однако по ряду ограничений не всегда возможны: невозможно исследовать критические режимы, тонкие эффекты, нереально исследовать все функциональные зависимости, и т. п.

Научный эксперимент – в настоящее время завоевал широкое распространение благодаря использованию средств автоматизации и управления, а также благодаря применению разнообразных вычислительных средств обработки и визуализации информации.

В научном эксперименте имеется возможность вмешательства человека в процесс проведения эксперимента и, благодаря этому, появилось новое направление – *автоматизация научных экспериментов*.

Другим видом реального моделирования является *физическое моделирование*.

Физическое моделирование отличается от натурного тем, что исследования проводятся на установках, которые сохраняют природу явлений и обладают физическим подобием.

Физическое моделирование может протекать в реальном и нереальном (псевдореальном) масштабах времени, а также может рассматриваться без учета времени: т. е. изучаются так называемые «замороженные процессы».

Представленная иллюстрация (см. рис. 6) и описание различных видов моделей дают представление о широком использовании моделирования в современной жизни.

3. ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

3.1. Цель и задачи имитационного моделирования

Имитационное моделирование – наиболее мощный и универсальный метод исследования и оценки эффективности сложных систем, поведение которых зависит от воздействия случайных факторов. К таким системам можно отнести и летательный аппарат, и популяцию животных, и предприятие, работающее в условиях рыночной экономики, и моделирование вычислительных сетей и систем.

Цель имитационного моделирования – воспроизведение взаимодействий, в которых участвуют различные компоненты, и изучение поведения и функциональных возможностей исследуемой системы. Для этого конкретизируются состояния системы и описываются действия, которые переводят ее из одного состояния в другое. Говорят, что система находится в определенном состоянии, когда все ее компоненты находятся в состояниях, совместимых с областью значений, описывающих это состояние характеристик. Таким образом, имитация – это динамический «портрет» состояний системы во времени, т. е. воспроизведение поведения системы во времени.

В имитационном моделировании предполагается, что система может быть описана ограниченным количеством понятий (терминов). Такими общими понятиями являются «*процесс*», «*событие*» и «*активность*» (действие). Взаимосвязь между этими понятиями можно проиллюстрировать на примере работы банка (рис. 7).

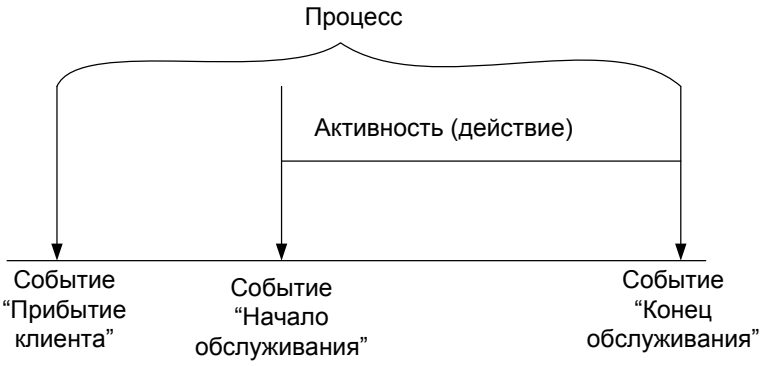

Рис. 7. Имитация работы банка

Процесс – это ориентированная во времени последовательность событий, каждое из которых может состоять из нескольких действий. Некоторый процесс может выступать в роли активности, или субпроцесса, в процессе более высокого уровня. Выполнение программы на ЭВМ можно представить процессом, составленным из активности вычислений и активности обмена с дисками (внешними носителями).

Событие – представляет собой мгновенное изменение состояния системы, которое может быть как активным, так и пассивным. Событие происходит в тот момент, когда принимается решение о начале или окончании действия. Активности инициируются в результате совершения событий.

Активность (*действие*) – характеристика производительности, прямо или косвенно связанная со скоростью, с которой система выполняет свою работу, и поэтому она содержат время в качестве независимой переменной. Работа совершается путем выполнения активностей. Сама активность с точки зрения моделирования систем является наименьшей единицей работы.

При дискретной имитации состояние системы может меняться только в момент свершения событий. Т. к. состояние системы не изменяется между этими моментами, полный динамический портрет состояния системы может быть получен путем продвижения имитационного времени от одного события к другому.

Функционирование дискретной имитационной модели (ИМ) можно задать следующим образом: определяя изменения системы, происходящее в момент свершения событий; описывая действия, в которых принимают участие элементы системы, или процесс, через который проходят элементы.

Очевидно, что разработка модели является достаточно сложным и специфическим делом. Рассмотрим подробнее различные этапы создания модели на примере создания имитационной модели. При этом необходимо учесть, что большие размеры ИМ, сложность поведения ее компонент и высокая стоимость разработки требуют применения математических методов на всех этапах разработки. Этапы разработки ИМ:

1. **Формулирование проблемы:** описание исследуемой проблемы и определение целей исследования.
2. **Разработка модели:** логико-математическое описание моделируемой системы в соответствии с формулировкой проблемы.
3. **Подготовка данных:** идентификация, спецификация и сбор данных.
4. **Трансляция модели:** перевод модели на язык, применимый для используемой ЭВМ.
5. **Верификация:** установление правильности машинной программы.
6. **Валидация:** оценка требуемой точности и соответствия ИМ реальной системе.
7. **Стратегическое и тактическое планирование:** определение условий проведения машинного эксперимента с имитационной моделью.
8. **Экспериментирование:** прогон имитационной модели на ЭВМ для получения требуемой информации.
9. **Анализ результатов** имитационного эксперимента для подготовки выводов и рекомендаций по решению проблемы.
10. **Реализация и документирование:** реализация рекомендаций, полученных на основе имитации, и составление документации по модели и ее использованию.

На примере ИМ мы рассмотрели основные этапы разработки модели системы. И, несмотря на то, что были рассмотрены только укрупненные этапы, они дают достаточно полное представление о сложности разработки модели.

3.2. Языки имитационного моделирования

Широкое применение ИМ в качестве средства анализа систем привело к разработке целого ряда языков, предназначенных специально для имитации. В общем случае языки ИМ включают средства обработки языковых конструкций (компилятор, транслятор или интерпретирующую программу), систему выполнения имитационного процесса во времени и, желательно, графический модуль, позволяющий визуализировать процесс функционирования модели во времени. Ряд основных конструкций языковых средств имеют общие черты во всех языках ИМ.

Имитационные языки используют специальное понятие и операторы, позволяющие отображать состояние системы в некоторый момент времени, а также воспроизводить переход системы из одного состояния в другое. В зависимости от подхода к описанию рассматриваемых объектов языки ИМ можно классифицировать следующим образом:

- языки, ориентированные на активности;
- языки, ориентированные на события;
- языки, ориентированные на процессы.

Наряду с предложенной классификацией языки имитационного моделирования (ЯИМ) можно классифицировать по эффективности их применения. При этом не следует забывать и о возможности использования для этих целей языков общего назначения (ЯОН), таких, как C, C++, Pascal, Fortran. При анализе эффективности использования ЯИМ для описания работы конкретных систем выделяют следующие моменты:

- возможность описания структуры и алгоритмов функционирования исследуемой системы в терминах ЯИМ;
- простоту применения языка для конкретной задачи;
- машинную реализацию и методы обработки и визуализации результатов прогона модели (сервисы);
- предпочтение пользователя, которое отдается либо знакомому языку, либо ЯИМ, имеющему максимальную универсальность, простоту и т. п.

Нетрудно заметить, что приведенные характеристики противоречивы. Действительно, универсальные ЯИМ должны иметь множество команд, операторов, позволяющих гибко моделировать сложные системы. Однако использование таких языков требует от разработчика модели высокой квалификации программиста. Исходя из того, что разработчик модели является специалистом высокого класса в своей предметной области, предпочтение в выборе отдается языкам, обладающим максимальной гибкостью при мини-

мальном количестве команд. В работе [1] приведена сводная таблица, в которой эффективность ЯИМ дана в порядке уменьшения их эффективности.

Возможность языка	Простота применения	Предпочтение пользователя
SIMULA	*GPSS*	*SIMSCRIPT*
SIMSCRIPT	*SIMSCRIPT*	*GPSS*
GPSS	*SIMULA*	*SIMULA*

Рассмотрим подробнее применение различных ЯИМ на конкретных примерах.

3.2.1. Подход сканирования активностей

Как было отмечено выше, действия активностей в модели совершаются только в начале и в конце выполнения активностей. Нетрудно прийти к выводу, что все изменения состояний в моделируемой системе происходят в моменты времени, соответствующие окончанию активностей. Инициирование активности совершается как результат окончания ранее выполнявшейся активности или совокупности активностей, которые могут существовать вне рассматриваемой системы. События, которые начинают или завершают действия, не планируются разработчиком модели, а имитируются по условиям, определенным для данного действия. Условия начала или окончания действия проверяются после очередного продвижения имитационного времени. Если заданные условия удовлетворяются, происходит соответствующее действие. Для того чтобы было выполнено каждое действие в модели, сканирование условий производится для всего множества действий при каждом продвижении имитационного времени.

Проиллюстрируем эти рассуждения конкретным примером (рис. 8).

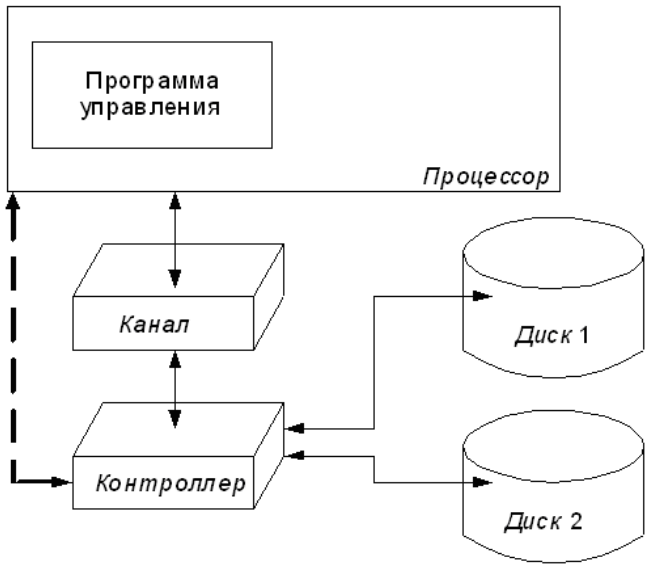

Рис. 8. Подсистема дисковой памяти

На рис. 8 схематически показана работа подсистемы дисковой памяти, в состав которой входят два дисковода с перемещаемыми головками записи-чтения и устройством управления вместе с каналом, через который происходит обмен данными между дисковой и оперативной памятью.

Описание работы системы

Запросы к дискам, инициируемые процессором, могут быть направлены к любому из двух дисководов:

a) когда инициируется запрос к диску №…, управляющая программа процессора определяет, свободен ли указанный дисковод, и если это так, то через устройства управления выдает команду установки на соответствующий дисковод. Если дисковод занят, запрос помещается в очередь;

b) вслед за завершением операции установки головки на соответствующую дорожку, активизируется управляющая программа, которая выдает команду передачи, если канал в это время свободен. Если канал занят, запрос помещается в очередь;

c) после выдачи команды передачи канал остается занятым до того момента, когда нужный сектор дорожки окажется под головкой записи-чтения и данные будут считаны с диска (записаны на диск);

d) по окончании передачи данных снова активизируется управляющая программа, дисковод и канал освобождаются. Если возможно, начинается обработка запросов, находящихся в очереди.

Ограничения модели

1. Полагаем, что время, затрачиваемое управляющей программой настолько мало, что им можно пренебречь.

2. Активности процессора, инициирующие запросы к диску для нас не представляют интереса, т. к. выходят за рамки рассматриваемой системы. Правда, следует заметить, что именно в результате завершения одной из таких активностей в подсистеме дисковой памяти и «появляется» запрос.

Запрос к диску можно рассматривать как процесс, включающий две активности: активность установки и активность канала. Время выполнения активности установки определяется временем, затрачиваемым на перемещение механизма доступа с одной дорожки на другую; время выполнения активности канала складывается из времени поворота диска в нужное положение и времени передачи данных. В рамках нашей модели дисководы и каналы рассматриваются в качестве элементов системы, которые имеют всего два состояния: «*свободно*» и «*занято*».

Проиллюстрируем выполнение одной конкретной последовательности запросов на временной диаграмме (см. рис. 9).

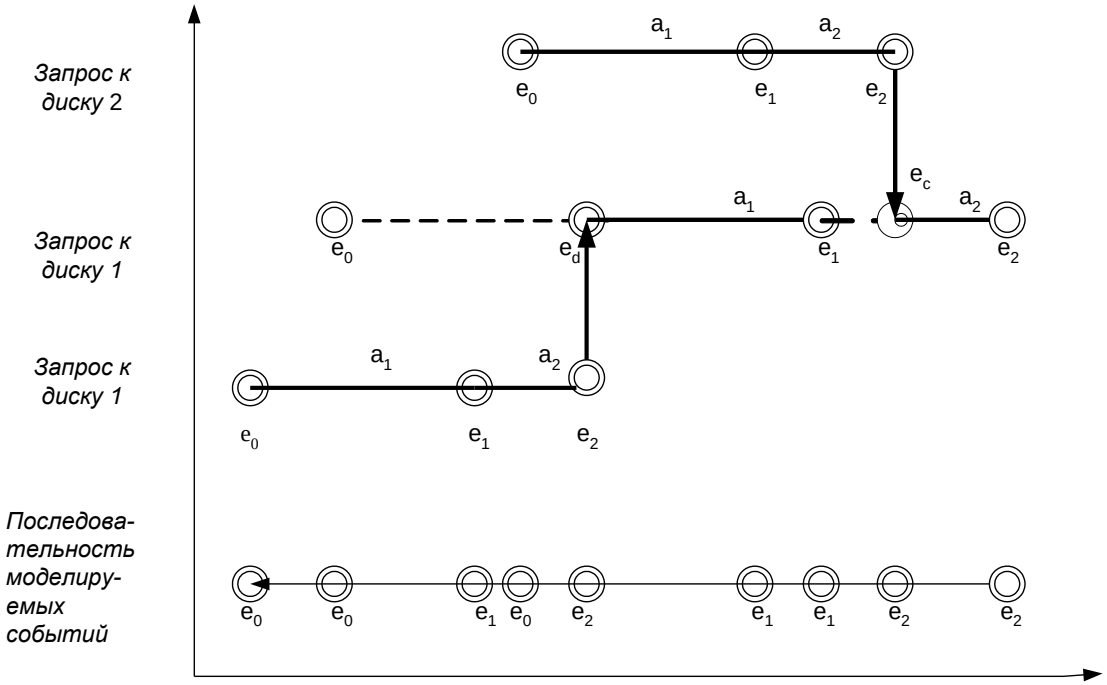

Рис. 9. Диаграмма активностей, выполняющихся при обращениях к диску:
a_1 и a_2 – активности установки и канала соответственно;
e_0 – события, соответствующие окончанию активности процесса;
e_1 и e_2 – события, соответствующие завершению активностей a_1 *и* a_2;
e_d и e_c – события, отвечающие освобождению диска и канала соответственно

При этом $e_{0,}\, e_1$, e_2 – события следования, а e_d и e_c – события изменения состояния системы.

Анализ диаграммы помогает понять, каким образом в модели осуществляется (отображается) совместное выполнение нескольких активностей. Глобальный порядок следования моделируемых событий устанавливается путем динамической сортировки величин времени, представляющих окончания активностей; эта последовательность событий управляет порядком выполнения различных частей моделирующей программы. В языках, использующих подход сканирования активностей, после завершения каждой активности последовательно просматриваются (сканируются) все имеющиеся в программе активности; и, если будет установлено, что набор условий выполнения какой-либо активности удовлетворен, выполняются действия, соответствующие этой активности.

В современных языках моделирования продвижение по оси времени осуществляется на длину очередного интервала между событиями, а не механическим увеличением счетчика времени на постоянную величину шага.

Подход сканирования активностей обеспечивает простую схему моделирования для решения целого ряда проблем. Тем не менее, т. к. необходимо сканировать условия для каждого действия, подход сканирования активностей менее эффективен, чем событийный подход. Примером языка, применяющего подход сканирования активностей, является язык CSL.

3.2.2. Языки, ориентированные на события

В языках, ориентированных на события, таких, как GASP, SIMSCRIPT или SMPL, моделирующая программа организована в виде совокупности секций событий или, другими словами, процедур (подпрограмм) событий. Процедура события состоит из последовательности операций, которые, как правило, выполняются после завершения какой-либо активности. Здесь система моделируется путем идентификации изменений, происходящих в ней в моменты свершения событий. Задача исследователя (разработка модели) заключается в описании событий, которые могут изменить состояние системы, и в определении логических взаимосвязей между ними. Имитация функционирования системы осуществляется путем выполнения упорядоченной во времени последовательности логически взаимосвязанных событий.

Обычно события подразделяются на две категории:

- ***события следования***, которые управляют инициализацией процессов (или отдельных работ внутри процесса);
- ***события изменения состояний*** (элементов системы или системы в целом).

Для того чтобы в модели обеспечивалось относительно независимое описание активностей, программа моделирования должна представлять собой совокупность независимых процедур, в которых разделены события следования и события изменения состояния. На следующем уровне описания программы активности объединяются в рамках описания процессов и т. п. Данный подход не имеет существенных преимуществ при создании простых моделей, однако при моделировании сложных систем, учитывающих взаимодействие процессов различных классов, этот подход имеет решающее значение.

Для иллюстрации событийного подхода рассмотрим пример работы банка с одним кассиром. Клиенты приходят в банк, ожидают обслуживания (если кассир занят), обслуживаются кассиром и затем покидают банк. Состояние системы в этом примере определяется состоянием кассира и числом ожидающих обслуживания клиентов. Оно остается неизменным, за исключением моментов, когда клиент прибывает в систему или покидает ее. Поэтому событийная модель в данном случае состоит из описания действий, происходящих в момент прибытия и окончания обслуживания очередного клиента. Т. к. изменения состояния системы могут происходить только в эти моменты времени, использование событий ***прибытие*** и ***конец обслуживания*** полностью обеспечивает воспроизведение логики и динамики функционирования системы. Рассмотрим сначала логику события ***прибытие*** и запишем операторную схему этого события:

- ПЛАНИРОВАНИЕ ПРИБЫТИЯ СЛЕДУЮЩЕГО КЛИЕНТА.
- ЕСЛИ КАССИР ЗАНЯТ: ЧИСЛО ОЖИДАЮЩИХ = ЧИСЛО ОЖИДАЮЩИХ + 1; ВЫХОД.
- ЕСЛИ КАССИР СВОБОДЕН: ПЕРЕВОД КАССИРА В СОСТОЯНИЕ «ЗАНЯТ». ПЛАНИРОВАНИЕ СОБЫТИЯ «ОКОНЧАНИЕ ОБСЛУЖИВАНИЯ» В МОМЕНТ ВРЕМЕНИ = ТЕКУЩЕЕ ВРЕМЯ + ВРЕМЯ ОБСЛУЖИВАНИЯ; ВЫХОД.
- КОНЕЦ.

На первом шаге планируется прибытие следующего клиента, что позволяет при неоднократном обращении к подпрограмме организовать непрерывный поток прибывающих клиентов.

Дальнейшее поведение клиента зависит от состояния системы, и если кассир занят, то длина очереди к кассиру увеличивается на единицу и на этом работа подпрограммы заканчивается.

Если кассир свободен, то состояние системы меняется путем перевода кассира в состояние «занят». В тот же момент начинается обслуживание клиента и планируется время окончания обслуживания. На этом работа подпрограммы заканчивается.

Рассмотрим теперь логику обработки события ***конец обслуживания***. Операторная схема этого события имеет следующий вид:

- ЕСЛИ ЧИСЛО ОЖИДАЮЩИХ В ОЧЕРЕДИ БОЛЬШЕ НУЛЯ: ЧИСЛО ОЖИДАЮЩИХ = ЧИСЛО ОЖИДАЮЩИХ – 1; ПЛАНИРОВАНИЕ ОКОНЧАНИЯ ОБСЛУЖИВАНИЯ В МОМЕНТ ВРЕМЕНИ, РАВНЫЙ ТЕКУЩЕМУ ВРЕМЕНИ + ВРЕМЯ ОБСЛУЖИВАНИЯ; ВЫХОД.
- ЕСЛИ ЧИСЛО ОЖИДАЮЩИХ В ОЧЕРЕДИ РАВНО НУЛЮ: ПЕРЕВОД КАССИРА В СОСТОЯНИЕ «СВОБОДЕН»; ВЫХОД.
- КОНЕЦ.

По окончании обслуживания очередного клиента сначала проверяем, не пуста ли очередь. Если длина очереди не равна нулю, то длина очереди уменьшается на единицу и начинается обслуживание клиента из очереди, т. е. планируется время окончания обслуживания очередного клиента. После этого происходит выход из подпрограммы. Если длина очереди равна нулю, то кассир переводится в состояние «свободен» и на этом работа подпрограммы заканчивается.

При имитации функционирования банка с одним кассиром на основе событийного подхода необходимо воспроизвести хронологию (*календарь*) событий и причины, вызывающие их появление в соответствующие моменты имитационного времени. Календарь событий первоначально содержит отметку только о первом событии – «прибытие первого клиента». В ходе имитации возникновения других событий ***прибытие*** и ***конец обслуживания***

должны быть запланированы в календаре в соответствии с логикой функционирования системы. Все моделируемые события выполняются в упорядоченной по времени последовательности, при этом имитационное время продвигается от одного события к другому.

Двумя распространенными событийно-дискрентыми языками являются GASP и SIMSCRIPT.

Если при построении событийно-дискретной модели разработчик использует какой-либо универсальный язык, например Фортран, то программирование календаря событий и механизма продвижения имитационного времени, обеспечивающих обработку событий в хронологической последовательности, требуют значительных затрат и мастерства программиста.

3.2.3. Процессно-ориентированный подход

Многие имитационные модели содержат последовательности компонентов, которые возникают в них по определенной схеме, например очередь, в которой клиенты ожидают обслуживания. Логика возникновения компонентов по требуемой схеме может быть обобщена и задана в одном операторе. Имитационный язык затем транслирует такие операторы в соответствующую последовательность событий, происходящих с компонентами модели. Динамические (временные) элементы (объекты) этого языка называются ***транзактами***.

Транзакт – это некоторое сообщение (заявка на обслуживание), которое поступает извне на вход системы и подлежит обработке. В ряде случаев, например при моделировании автоматизированных систем управления, более удобно проследить функционирование системы именно относительно алгоритма обработки транзакта. В рамках одной ИМ могут рассматриваться транзакты нескольких типов. Каждый транзакт характеризуется соответствующим алгоритмом обработки и необходимыми для его реализации ресурсами системы. Учитывая это, продвижение транзакта по системе можно в некоторых случаях рассматривать как последовательную активизацию *процессов*, реализующих его обработку («обслуживание заявки»). Термин «обслуживание заявки» тесно связан с теорией массового обслуживания, которая будет рассмотрена несколько позже. Поэтому при разработке и исследовании имитационных моделей на основе транзактов целесообразно использовать методику и показатели, применяемые при анализе систем массового обслуживания.

Проиллюстрируем вышесказанное следующим набором операторов, используемых для описания процесса в модели банка с одним кассиром:

- *создать прибывающих клиентов через каждые T единиц времени*;
- *ожидать кассира*;
- *продвинуть время на продолжительность обслуживания*;
- *освободить кассира*;
- *удалить клиента.*

Здесь первый оператор генерирует прибывающих в систему клиентов через каждые *T* единиц имитационного времени. Величина *T* может быть либо константой, либо принимать случайные значения в соответствии с заданным законом распределения.

Клиент ожидает до тех пор, пока кассир освободится. Этот тип оператора аналогичен понятию действия по условию в подходе сканирования активностей.

Оператор моделирует период времени, необходимый для обслуживания клиента кассиром. (Аналогичен оператору планирования конца события в событийном подходе.) В календарь событий помещается метка о том, что обслуживание клиента будет окончено в момент времени, равный текущему моменту плюс время обслуживания.

По завершении обслуживания кассир освобождается, и клиент покидает систему. Освобождение кассира позволяет сразу же приступить к обработке какого-либо ожидающего клиента из оператора «ожидать кассира».

Как видно из примера, процессно-ориентированный подход сочетает в себе черты событийного подхода и подхода сканирования активностей. Он обеспечивает описание прохождения компонентов через процесс, содержащий ресурсы. Простота этого подхода состоит в том, что определяемая операторами логика событий заложена в самом имитационном языке. Однако набор стандартных операторов языка обычно ограничен, поэтому данный подход является менее гибким, чем событийный. Кроме того, требуется постоянный анализ состояния ресурсов после их использования. Наиболее распространенными процессно-ориентированными языками являются GPSS, SIMULA, ASPOL и SOL.

Наряду с рассмотренными выше ЯИМ существуют комплексные языки, используемые для описания функционирования сложных систем. В них могут использоваться понятия процесса, события и частично действия и соответствующие им программные средства языка. Управление последовательностью выполнения процессов и событий в этих языках осуществляется с помощью общего календаря событий. При этом выполняемое событие может изменить характеристики процессов (например, среднее время обслуживания запроса (заявки)), а процессы могут вызывать возникновения событий. Порядок работы программы, написанной на таком языке, аналогичен тому, как работают программы, ориентированные на процессы и события. Только в этом случае будет выполняться либо подпрограмма, соответствующая событию, либо начинает выполняться какой-то процесс, в зависимости от того, время начала какого события, занесенного в календарь, будет меньшим. Примерами таких языков являются SIMSKRIPT-2, SLAM1, SLAM2. Как правило, они включают и моделирование непрерывных процессов.

3.3. Последовательность разработки программной (машинной) реализации модели системы

Ранее были кратко рассмотрены основные этапы создания имитационной модели (см. п. 3.1). Сейчас имеет смысл вернуться к этой задаче на новом уровне, используя полученную информацию. Для этого рассмотрим задачу «*Моделирование управления памятью в многопроцессорной системе с общей шиной*».

1. Описание исследуемой проблемы и определение цели исследования.

Изучаемый объект представляет собой управление памятью в схеме с общей шиной, которая используется в малых ЭВМ. Основные компоненты рассматриваемой системы представлены на рис. 10. Система состоит из n модулей (блоков) памяти и m модулей процессоров, соединенных двумя общими шинами. По одной из шин передаются адреса, а по другой – данные. Все модули работают асинхронно; обмен между блоками памяти и процессорами координируется устройством управления. Время цикла памяти равно t_m (нс).

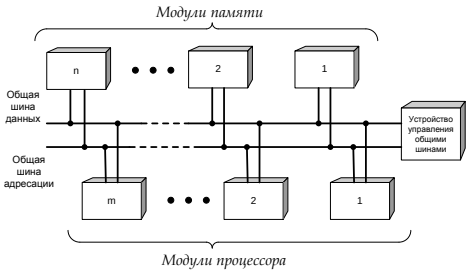

Рис. 10. Организация модулей процессоров и памяти по схеме с общей шиной

Цель исследований (моделирования) – оценить среднее время доступа в зависимости от позиции процессорного модуля на общей шине при условии, что все модули обращаются к памяти с одинаковой средней частотой.

2. Разработка модели.

На данном этапе необходимо, в соответствии с формулируемой проблемой, дать логико-математическое описание элементов моделируемой системы. Опишем работу отдельных элементов системы на логическом уровне:

- Чтобы произвести обмен данными между блоком памяти и процессором, модуль, инициирующий обмен, должен, прежде всего, зарезервировать общую шину. Получив в распоряжение общую шину, процессор определяет, свободен или занят нужный ему блок памяти. Если блок свободен, он резервируется за процессором, после чего происходит передача адреса данных и самих данных в память (когда доступ связан с записью данных). Если блок памяти занят, то процессор освобождает общую шину, задерживается на время t_b и затем вновь пытается осуществить доступ. Если блок памяти уже отведен процессору, последующие действия определяются типом доступа. Обращение с записью в память заканчивается передачей сигнала «завершено» в процессор, инициировавший запрос, после чего он может продолжать работу. В этом случае общая шина освобождается и инициируется цикл записи в память, после которого блок памяти освобождается. Если доступ к памяти связан с чтением данных, то общая шина освобождается и начинается цикл выборки из памяти. После того, как слово данных, считанное из памяти, окажется в выходном регистре памяти, выполняется запрос к общей шине. Как только общая шина захвачена, слово данных вместе с сигналом «завершено» передается в процессор, инициировавший запрос. Теперь освобождается блок памяти и общая шина.
- Блок сканирования в составе устройства управления обслуживает запросы к общей шине с учетом приоритетов:
  - Запросы на чтение из памяти имеют наивысший приоритет.
  - Процессоры получают приоритеты в зависимости от позиции внешних контактов на общей шине (позиция 1 дает приоритет больший, по сравнению с позицией 2, и т. д.).
- Запросы к уже занятой общей шине помещаются в очередь и обслуживаются в порядке следования приоритетов.
- Если одновременно существуют несколько запросов на чтение из памяти, они обслуживаются в порядке их поступления.
- Любая операция с общей шиной занимает постоянное время.

3. Подготовка данных.

- Единица измерения времени в модели – наносекунда.
- Моделируемая конфигурация системы включает 4 блока памяти и 8 процессоров.
- Время цикла памяти – 150 нс, время операции с общей шиной – 8 нс.
- Параметр p – вероятность того, что очередной запрос определяет запись данных в память: $0 \le p \le 1$.
- Величина задержки, имитирующей вычислительную работу процессора, равномерно распределена на интервале $[t_1, t_2]$.
- При обращении к памяти номера блоков выбираются с равной вероятностью среди первых n чисел.

4. Трансляция модели.

В связи с тем, что представление алгоритма (блок-схемы) программы более наглядно и удобно для анализа, на этом этапе приведен не текст программы на алгоритмическом языке высокого уровня, а алгоритм (блок-схема) подсистемы доступа к памяти (см. рис. 11).

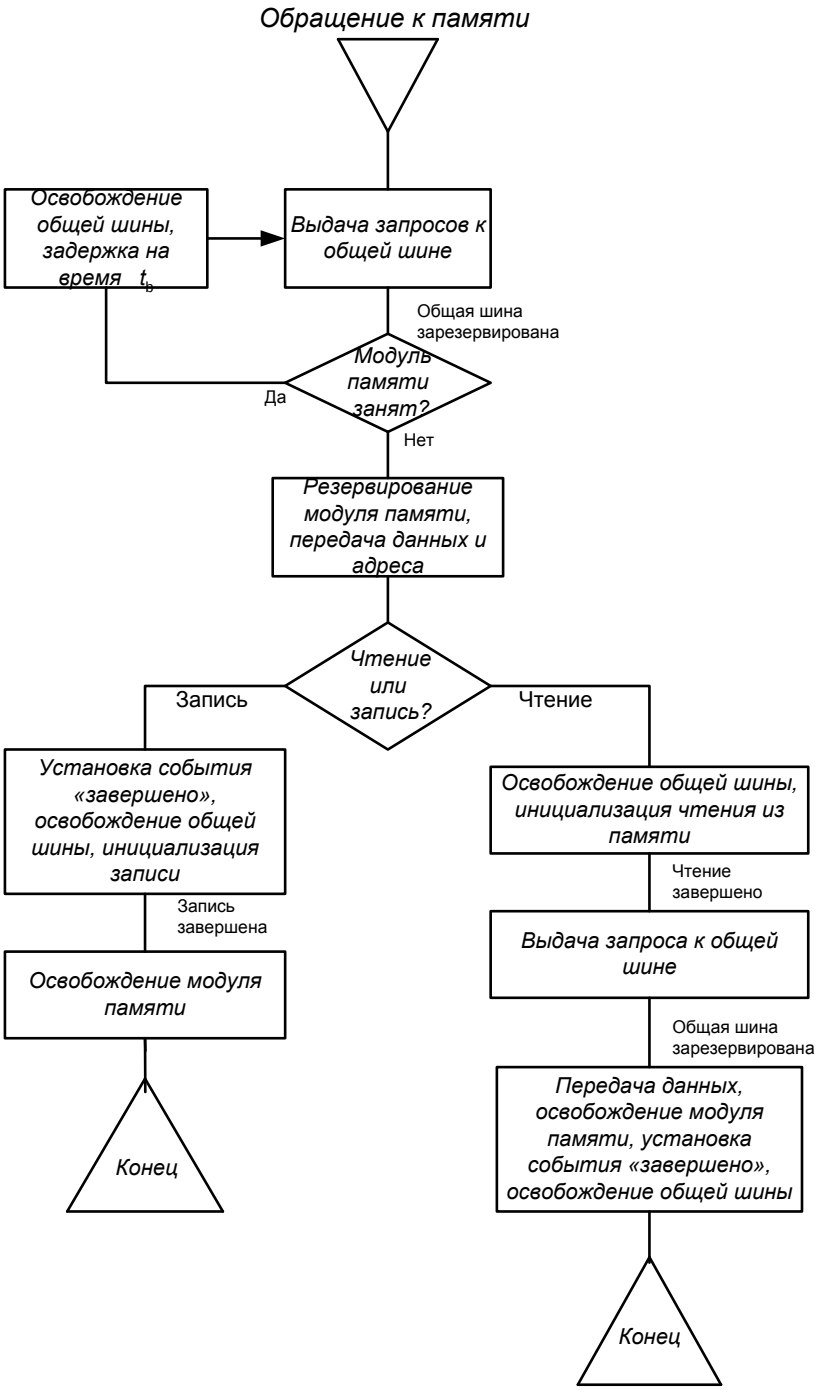

Рис. 11. Обработка обращения к памяти

5. Верификация.

На данном этапе проводится тестирование разработанной программы. Для установления правильности работы программы, обычно производят следующую последовательность операций:

- Производим отладку каждого модуля системы, для чего, естественно, придется написать отладочные программы для каждого модуля и, возможно, не одну.
- При отладке многофункционального модуля (подпрограммы) разумнее написать несколько отладочных программ, каждая из которых проверяет различные функции.
- Комплексную проверку разработанной программы проводим после того, как убеждаемся в работоспособности отдельных модулей. Для этого приходится писать отладочную программу порой более объемную и сложную, чем проверяемая.
- Особое внимание следует уделить проверке условных переходов (ветвлений). В отладочной программе необходимо предусмотреть все возможные переходы во всех возможных последовательностях. Возможно, что для проверки выполнения мало вероятных событий в отладочной программе следует расставлять «маяки» для выявления отдельных участков программы, которые никогда не работают.

6. Валидация.

Необходимо оценить требуемую точность и соответствие ИМ реальной системе. Рассмотрим сначала первое требование – оценка требуемой /полученной точности. В результате прогона ИМ мы должны получить среднее время доступа к процессорному модулю в зависимости от его номера. Нас интересует только один параметр для каждого процессора, который определяется выражением

$$M_i[t] = \frac{1}{n_i} \sum_{1}^{n_i} t_j ,$$

где i – номер процессора, n_i – количество обращений к i-процессору. Как правило, этого одного параметра не достаточно, поэтому обычно оценивают дисперсию; для выборочного среднего и выборочной дисперсии находят доверительные интервалы.

Проверка достоверности ИМ подразумевает сравнение поведения (характеристик) модели с поведением (характеристиками) ***реальной системы***. В большинстве случаев сравнение осуществить не удается, потому что, как правило, моделируется поведение системы, не имеющейся у исследователя. Часто создается модель системы, которая реально (в природе) не существует. Поэтому разработанная ИМ является мощным инструментом, достоверность работы которого для исследователя должна быть очевидна. Как можно убедиться в достоверности работы новой ИМ?

- Вновь созданная ИМ обычно не очень сложна, поэтому в ряде случаев полученные результаты можно сравнить с аналитическими.

• Задав конкретные начальные условия, можно свести свою задачу к известной (похожей) задаче, подробно описанной в литературе. Результаты, полученные при прогона модели, необходимо сравнить с эталонными (литературными).

• Работу новой ИМ можно проверить косвенным путем. Задав определенные начальные условия, можно сравнить полученные результаты с аналогичными, полученными на предыдущей модели. Такой подход правомочен при условии, что в достоверности результатов (предыдущая модель) вы уверены.

7. Стратегическое и тактическое планирование.

Тактика:

• Задание конкретных начальных условий и количества прогонов для исследуемой задачи.

• Вариация начальных условий (данных), влияние которых на исследуемые параметры мы хотим проанализировать. Здесь следует проявить разумную предусмотрительность. Желание выявить зависимость исследуемых параметров от всех (многих) переменных приведет к неограниченному росту материальных и временных ресурсов, при этом следует учесть и трудности обработки больших объемов информации. С другой стороны, неоправданное уменьшение объема исследований может не позволить выявить имеющиеся закономерности.

Стратегия. Разработка и создание ИМ – достаточно сложный и дорогостоящий процесс, чтобы ограничиться решением только одной конкретной задачи.

Что касается нашей модели, то на ее основе можно построить целый ряд более сложных систем, позволяющих расширить ее возможности. Например, при исследовании конфигурации подобной системы к модулю с наименьшим номером позиции на общей шине подключено внешнее устройство с высоким быстродействием или, наоборот, с низким быстродействием. Другой интересный пример усложненного эксперимента касается адресации к памяти и т. п.

8. Экспериментирование.

В соответствии с поставленной целью осуществляем прогон (серию прогонов) модели на компьютере для получения требуемой информации.

9. Анализ результатов.

Весь объем информации, полученной в результате прогонов модели, обрабатывают, визуализируют и анализируют с привлечением методов математической статистики. По результатам анализа делаются соответствующие выводы и формулируются рекомендации по решению проблемы.

10. Реализация и документирование.

На данном этапе реализуются рекомендации, полученные на основе имитационной модели. Параллельно создается подробная документация, отражающая все особенности модели, требуемые аппаратные и программные ресурсы, операционную систему и т. п.

4. СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ (СМО)

4.1. Основные понятия и задачи СМО

Функционирование многих систем представляет собой последовательность переходов системы из одного состояния в другое.

Система называется ***системой с дискретными состояниями***, если множество её состояний конечно (или счетно), а переходы из одного состояния в другое осуществляются скачками.

Процесс переходов из состояния в состояние называется ***цепью***. Переход системы в некоторое состояние S_i является ***событием***. Последовательность переходов системы в состояние S_i представляет поток таких событий $S_i^{'}, S_i^{''}$.

Потоком событий называется последовательность однородных событий S_i, следующих одно за другим, в некоторые моменты времени. Поток называется ***ординарным***, если события происходят поодиночке. Интервалы $T_1, T_2, \ldots, T_n$ одинарного потока могут быть одинаковыми или различными, дискретными или непрерывными, случайными или неслучайными.

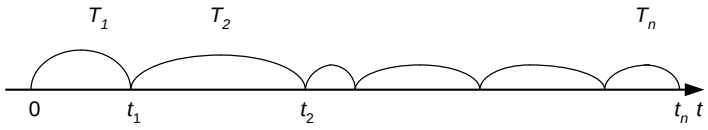

Рис. 12. Ординарный поток событий

Многие понятия теории массового обслуживания можно наглядно проиллюстрировать на примере *работы крупного аэропорта*, т. к. практически все в большей или меньшей степени сталкивались с его работой.

Мы предполагаем, что аэропорт имеет несколько взлетно-посадочных полос (***параллельных каналов***). Все эти полосы ведут к большему или меньшему числу дорожек, оканчивающихся на аэровокзале (***последовательные каналы***). После того как самолет, прибывший по расписанию (в соответствии с определенным ***распределением входящего потока***), приземляется, он присоединяется к ***очереди*** самолетов, ожидающих обслуживание (продвижение по дорожке к месту выгрузки). Таким образом, ***выходящий поток*** одной очереди становится входящим потоком для другой. Существуют очереди на земле (взлет самолетов) и в воздухе (посадка самолетов). Если для различных типов самолетов отведены различные взлетно-посадочные полосы, которые могут быть длиннее, например, для турбореактивных либо широкофюзеляжных самолетов, то ***распределение времени обслуживания*** (длительность операции обслуживания) может меняться от одной полосы к другой. При выборе самолетов из очереди на посадку важно определить соответствующий

показатель эффективности. Например, если желательно минимизировать общее время ожидания пассажиров, то вначале произведут посадку аэробусы и лишь потом самолеты местных линий (легкомоторные). Здесь же очень часто производится обслуживание по ***приоритетам***, когда разрешается посадка снижающемуся самолету раньше, чем взлет ожидающему. Система приоритетов очень эффективна при так называемых нештатных ситуациях.

Одной из основных проблем управления аэропортом является связь. Если входящий поток как на земле, так и в воздухе велик, то аэропорт должен быстро связаться с самолетами и получить ответ. При организации связи важной проблемой является определение числа операторов и ***каналов связи,*** необходимых для регулирования различных состояний перегруженности, которые могут возникнуть. В данном случае необходимо выбрать ***оптимальное число каналов*** для обслуживания требований, поступающих в соответствии с данным распределением (расписанием полетов). Можно произвести сравнение стоимости ***дополнительного канала*** со стоимостью возросшего объема обслуживания существующими каналами.

Обычно в системах массового обслуживания не существует запланированной регулярности событий. Например, заявки на посадку (взлет) в нашем примере могут поступать случайным образом, и поэтому нельзя быть заранее уверенным в точном времени их появления. Эти рассуждения справедливы и по отношению ко времени обслуживания. Очевидно, что с каждой заявкой (требованием) связан ряд случайных величин: τ – время поступления N-го требования, η – время ожидания, и ξ – время обслуживания и т. п.

Исследование задач теории массового обслуживания связано с нахождением распределения этих случайных величин (СВ) при наличии достаточной информации о них. Основная проблема состоит в том, чтобы:

1) правильно связать эти СВ с описанием задач массового обслуживания;
2) найти соответствующее распределение и определить его параметры на основе статических наблюдений;
3) использовать эти распределения для нахождения характеристик процесса обслуживания;
4) на основе полученных характеристик принять необходимые меры для улучшения (оптимизации) процесса обслуживания.

Очевидно, что для успешного решения этих проблем необходимо было разработать соответствующий математический аппарат либо привлечь известный, хорошо зарекомендовавший себя при решении подобных задач.

Оказалось, что задачи теории массового обслуживания успешно описываются с помощью марковских случайных процессов с дискретными или непрерывными состояниями. Что же представляют собой марковские процессы (цепи)?

4.2. Марковские случайные процессы

Рассмотрим марковский случайный процесс с дискретными состояниями и дискретным временем функционирования. Такой процесс описывает систему S с конечным числом состояний, причем переходы возможны только в фиксированные моменты времени $t_1, t_2, \ldots, t_n$. Процесс, происходящий в такой системе, можно представить в виде цепочки случайных событий $S_1(0) \to S_1(1) \to S_2(2) \to \cdots \to S_i(\nu) \to \cdots \to S_n(k)$.

Такая случайная последовательность называется *дискретной марковской цепью*, если для каждого шага ν ($\nu = 1,2,\ldots,k$) вероятность перехода из любого состояния S_i в любое состояние S_j ($i = 1,2,\ldots,n;\ \ j = 1,2,\ldots,n$) не зависит от того, как система пришла в состояние S_i.

Рассмотрим свойства марковских случайных процессов на примере известной игры «Тише едешь – дальше будешь».

Вспомним правила игры:

1. В этой игре фишка играющего должна пройти некоторое конечное число пунктов $1, 2, \ldots, m$.

2. Переход из одного пункта в другой каждый раз определяется исходом бросания игральной кости.

Таким образом, если на данном шаге фишка находится в пункте i, то правилами игры устанавливается пункт перехода ее на следующем шаге, в зависимости от числа очков, выпавших на игральной кости. Отметим очевидное свойство марковских случайных процессов.

Из любого пункта i фишка с некоторой вероятностью p_{ij} переходит в один из пунктов j независимо от характера ее движения до попадания в пункт i. Это свойство цепей маркова часто называют *системами без последствия*, или *системами с отсутствием памяти*.

Пусть имеется система, которая может находиться в одном из фазовых состояний $E_1, E_2, \ldots$ Состояние системы меняется в зависимости от некоторого параметра t, причем переход из состояния в состояние зависит от случайного фактора. Будем условно называть параметр t – временем и считать, что t пробегает либо целые, либо действительные числа. Пусть $\xi(t)$ – состояние системы в момент времени t, и пусть соблюдается закономерность: если в данный момент времени s система находится в фазовом состоянии i, то в последующий момент времени t будет находиться в состоянии j с некоторой вероятностью $p_{ij}(s,t)$ независимо от поведения системы до указанного момента s. Вероятности

$$p_{ij}(s,t) = P\{\xi(t) = j \,/\, \xi(s) = i\} \quad (i, j = 1,2,\ldots)$$

называются ***переходными вероятностями*** марковской цепи $\xi(t)$.

Марковская цепь $\xi(t)$ называется ***однородной***, если переходные вероятности $p_{ij}(s,t)$ зависят лишь от разности $t - s$:

$$p_{ij}(s,t) = p_{ij}(s-t) \quad (i, j = 1,2,\ldots).$$

Полным описанием однородной марковской цепи может служить квадратная матрица переходных вероятностей

$$\left[p_{ij}\right]_n = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}.$$

Очевидно, что для каждого состояния (номера шага) возможные переходы образуют полную группу событий, т. е. $\sum_{j=1}^{n} p_{ij} = 1$. Переходные вероятности, соответствующие невозможным переходам, равны нулю. Вероятности, расположенные по главной диагонали матрицы, соответствуют тому факту, что система состояния не изменила.

Дискретная марковская цепь называется ***неоднородной***, если переходные вероятности меняются с изменением номера шага.

Отметим, что переход системы из одного состояния в другое в последовательных опытах может происходить непосредственно либо пошагово:

$$p_{ij}^{(1)} = p_{ij}, \quad p_{ij}^{(2)} = \sum_k p_{ik} p_{kj}, \ldots \quad p_{ij}^{(n)} = \sum_k p_{ik} p_{kj}^{(n-1)}.$$

В этом выражении верхний индекс указывает число шагов, за которые система из состояния i переходит в состояние j.

Рассмотрим несколько примеров, иллюстрирующих свойства цепей маркова и матрицы переходных вероятностей.

Пример 1. Имеется конечная марковская цепь с соответствующей матрицей вероятностей переходов

$$P = \begin{bmatrix} \frac{1}{6} & 0 & \frac{1}{6} & 0 & \frac{2}{3} \\ 0 & \frac{1}{4} & 0 & 0 & \frac{3}{4} \\ \frac{1}{4} & \frac{3}{8} & \frac{1}{8} & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{2}{3} \end{bmatrix}.$$

Анализ данной матрицы переходных вероятностей свидетельствует о том, что не для каждого состоянии существует отличная от нуля вероятность перехода в другое состояние. Более того, для данной системы существует тупиковое состояние, из которого ни в какое другое состояние перейти невозможно.

Класс эквивалентности (множество состояний), в котором для каждой пары состояний существует отличная от нуля вероятность перехода из одного состояния в другое.

У нас три класса эквивалентности:

1) $\text{I} \Leftrightarrow \text{III}$, $\rightarrow$ $p_{13} = 1/6$, $p_{31} = 1/4$;
2) $\text{II} \Leftrightarrow \text{V}$, $\rightarrow$ $p_{25} = 3/4$, $p_{52} = 1/3$;
3) IV; $\rightarrow$ $p_{44} = 1$.

Рассмотрим переходы из первого класса эквивалентности во второй и третий. Обратный переход невозможен. Поэтому первый класс называется ***устойчивым***.

Второй и третий классы находятся в равновесии и являются ***замкнутыми***, т. е. если система перешла в один из этих классов, то переход в другой класс невозможен. Эти два класса называются ***эргодическими классами*** эквивалентностей. Третий класс имеет единственное эргодическое состояние, называемое ***поглощающим***. Марковская цепь, в которой каждое эргодическое состояние является поглощающим, называется ***поглощающей цепью***.

Пример 2. «Случайное блуждание».

Рассмотрим случайное блуждание частицы по целочисленным значениям действительной оси (прямой). Частица на каждом шаге с вероятностью p смещается на +1 и с вероятностью $q = 1 - p$ смещается на –1.

Пусть $\xi(n)$ – положение частицы через n – шагов, тогда последовательность $\xi(0)$, $\xi(1)$, $\xi(2)$, … образует марковскую цепь. Если частица находится в какой–то точке i, то ее дальнейшее поведение не зависит от обстоятельств, предшествующих попаданию в точку i. За последующие n шагов частица с вероятностью $P_{ij}(n)$ переходит в соответствующее состояние j. Отметим очевидные свойства.

1. Ясно, что при $|i - j| > n$ переход из состояния i в состояние j невозможен, и тогда $P_{ij}(n) = 0$.

2. Очевидно, что за n шагов частица может перейти лишь в те состояния j, для которых выражение $|i - j|$ имеет ту же четность, что и n. Т. е. возможен переход только в те состояния, для которых число m является целым:

$$m = \frac{n + |i - j|}{2}.$$

3. Если $j \geq i$, то попасть в состояние j можно тогда и только тогда, когда из всех n шагов ровно m шагов совершается в положительном направлении. Вероятность этого события вычислим по формуле Бернулли

$$P_{ij}(n) = C_n^m p^m q^{n-m}.$$

4. Аналогично вычисляется вероятность перехода из i в j, если $j \leq i$:

$$P_{ij}(n) = C_n^m p^{n-m} q^m.$$

4.3. Пуассоновские потоки

Как было отмечено раньше, одними из важнейших параметров (характеристик) СМО являются ***входящий и выходящий потоки***.

Поток случайных событий называется ***пуассоновским***, если число m событий потока, попадаемых на любой участок τ оси времени, распределено по закону Пуассона

$$P_m = \left(\frac{a^m}{m!} \right) \cdot e^{-a},$$

где a – среднее число событий, приходящихся на участок τ.

Пуассоновский поток является ***стационарным***, если плотность потока событий $\lambda = \text{const}$, тогда среднее число событий $a = \lambda\tau$, и ***нестационарным***, если $\lambda = \lambda(t)$, тогда $a = \int_{t_0}^{t_0+\tau} \lambda(t)dt$.

Рассмотрим случайную величину T – интервал времени между соседними событиями в стационарном пуассоновском потоке – и определим ее функцию распределения. Вспомним, что $F(\tau) = P(T < \tau)$. Выражение $T < \tau$ означает, что в интервале времени τ наблюдается хотя бы одно событие потока. Выразим вероятность $F(\tau)$ через вероятность $F_0(\tau)$ того, что в интервале времени τ не наблюдается ни одного события потока:

$$F(\tau) = 1 - F_0(\tau) = 1 - \frac{a^0}{0!} \cdot e^{-a}.$$

Таким образом, для стационарного пуассоновского потока функция распределения времени между соседними событиями и плотность распределения соответственно равны

$$F(\tau) = 1 - e^{-\lambda\tau}; \qquad f(\tau) = \lambda e^{-\lambda\tau}.$$

Из полученного следует, что интервал времени подчинен экспоненциальному (показательному) закону распределения с параметрами

$$m_\tau = \sigma_\tau = 1/\lambda.$$

Если величины T_i являются зависимыми случайными величинами, то поток называется ***потоком с последействием***, ибо для любого момента времени последующее течение потока находится в вероятностной зависимости от предыдущего. Если СВ T_i независимы, то случайный поток называется ***потоком с ограниченным последействием*** и плотность вероятности системы можно представить в виде:

$$f(\tau_1, \tau_2, ..., \tau_n) = f(\tau_1) f(\tau_2) ... f(\tau_n).$$

Таким образом, в случае стационарного пуассоновского потока все интервалы $T_1, T_2, ..., T_n$ имеют одинаковые законы распределения $f_i(\tau_i) = f(\tau_i)$, где $i = 1, 2, ..., n$, что является проявлением отсутствия последействия.

Случайный поток событий, который обладает свойством стационарности, ординарности и не имеет последействия, называется ***простейшим*** *и является стационарным пуассоновским потоком.*

4.4. Задача автоматической телефонии

Рассмотрим широко известную задачу, которая не потеряла актуальность и в настоящее время.

Итак, имеется группа, состоящая из a абонентов, которым предоставлено S коммутаторов. По условию задачи $S << a$. Пусть y – среднее число вызовов за единицу времени в группе из a абонентов. Отметим, что вероятность P_n того, что за время τ произойдет n вызовов, подчиняется закону Пуассона. Предполагаем, что разговор, уже длящийся время t, закончится в момент $t + dt$ с вероятностью dt.

Сформулируем цель исследований: необходимо найти вероятность того, что в момент времени t имеется ровно i занятых коммутаторов. Обозначим искомую вероятность – P_i. Это событие (занято i коммутаторов) может произойти следующим образом:

1. Предположим, что в момент времени $t - dt$ заняты ровно i коммутатора с вероятностью P_i. В момент t их будет все еще i, если только не произойдет вызова (вероятность этого события равна ydt) или какой-нибудь коммутатор не освободится (с вероятностью idt). При нашей гипотезе – это единственно возможные случаи, если пренебречь возможностью занятия/освобождения двух и более коммутаторов за время dt. В этом случае мы пренебрегаем событиями, вероятность которых ~ $(dt)^2$, $(dt)^3$, ... Действительно, вероятность того, что произойдет α вызовов, равна $(ydt)^{\alpha}$, а вероятность того, что освободится β коммутаторов, равна $i(i-1)...(i-\beta+1)(dt)^{\beta}$. Следовательно, суммарная вероятность того, что число занятых коммутаторов не меняется за время dt, равна

$$P_i(1 - ydt - idt). \qquad (4.1)$$

Это же событие может произойти еще двумя способами.

2. Если в момент времени t - dt заняты ровно $i - 1$ коммутатора с вероятностью P_{i-1}, то в момент t их будет i, если за время dt произойдет один вызов (вероятность ydt). Это единственно возможный случай, если пренебречь вероятностями, имеющими порядок малости выше первого. Вероятность этого события равна

$$P_{i-1}(ydt). \qquad (4.2)$$

3. И, наконец, третий случай. Если число занятых коммутаторов в момент $t - dt$ равно $i + 1$ (вероятность P_{i+1}), то в момент t их будет i, если один коммутатор освободится (вероятность $(i+1)dt$. Это также единственно воз-

можный вариант, если пренебречь величинами порядка выше dt. Следовательно, общая вероятность этого случая равна

$$P_{i+1}(i+1)dt. \tag{4.3}$$

Ограничиваясь величинами первого порядка малости, можно записать

$$P_i = P_i(1 - ydt - idt) + P_{i-1}ydt + P_{i+1}(i+1)dt, \tag{4.4}$$

или

$$(y+i)P_i = yP_{i-1} + (i+1)P_{i+1}. \tag{4.5}$$

Учтем, что число занятых коммутаторов не может быть отрицательным, тогда первое уравнение ($i = 0$) будет иметь вид

$$yP_0 = P_1. \tag{4.6}$$

Окончательно получим следующую систему уравнений:

$$\left.\begin{aligned} &yP_0 = P_1,\\ &(y+1)P_1 = yP_0 + 2P_2,\\ &\ldots\ \ldots\ \ldots\ \ldots\ \ldots\\ &(y+i)P_i = yP_{i-1} + (i+1)P_{i+1},\\ &\ldots\ \ldots\ \ldots\ \ldots\ \ldots\\ &(y+S-1)P_{s-1} = yP_{s-2} + SP_s. \end{aligned}\right\} \tag{4.7}$$

Преобразуем систему (4.7), заменив каждое i-е уравнение суммой i первых уравнений:

$$\left.\begin{aligned} &yP_0 = P_1,\\ &yP_1 = 2P_2,\\ &\ldots\ \ldots\ \ldots\ \ldots\ \ldots\\ &yP_i = (i+1)P_{i+1},\\ &\ldots\ \ldots\ \ldots\ \ldots\ \ldots\\ &yP_{s-1} = SP_s. \end{aligned}\right\} \tag{4.8}$$

При решении системы (4.8) можно все P_i выразить через P_0:

$$P_1 = yP_0, \quad P_2 = \frac{y^2}{2}P_0, \quad \ldots, \quad P_i = \frac{y^i}{i!}P_0, \quad \ldots, P_s = \frac{y^s}{S!}P_0. \tag{4.9}$$

Отметим, что в эквивалентных системах (4.7 – 4.9) – имеется S уравнений с $S + 1$ неизвестными. Согласно условию задачи у нас нет ожидающих устройств, тогда занятость 0, 1, 2, ..., S коммутаторов составляет полную группу попарно несовместных событий. Это условие дает нам $S + 1$-е уравнение

$$P_0 + P_1 + \ldots + P_s = 1. \tag{4.10}$$

Введем обозначение

$$\frac{1}{D}=1+\frac{y}{1!}+\frac{y^2}{2!}+\frac{y^3}{3!}+\cdots+\frac{y^s}{s!}. \qquad (4.11)$$

Тогда

$$P_0=D,\quad P_1=D\frac{y}{1!},\quad \ldots,\quad P_i=D\frac{y^i}{i!},\quad \ldots;P_s=D\frac{y^s}{s!}. \qquad (4.12)$$

Формулы (4.12) называются ***формулами Эрланга***. Вспомним, что P_s – вероятность того, что все коммутаторы заняты, тогда в среднем имеем yP_s напрасных вызовов.

Рассмотрим эту же задачу при наличии ожидающего устройства (рассмотрим без выводов). Предварительно введем следующее обозначение: $P_j^{'}$ – вероятность того, что в очереди накопилось j вызовов. Тогда по аналогии с (4.7 – 4.9) получим

$$\left.\begin{array}{lll} yP_0^{'}=SP_1^{'}, & \Rightarrow & P_1^{'}=\dfrac{y}{S}P_0^{'}, \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ldots\ \ldots & & \\ yP_j^{'}=SP_{j+1}^{'}, & \Rightarrow & P_j^{'}=\left(\dfrac{y}{S}\right)^j P_0^{'}. \end{array}\right\} \qquad (4.13)$$

Вероятность $P_0^{'}$ того, что в ожидающем устройстве находится ровно 0 вызовов, очевидно, равна вероятности того, что все S коммутаторов заняты P_s. Тогда

$$P_0+P_1+P_2+\cdots+P_s+P_1^{'}+P_2^{'}+\cdots+P_j^{'}+\cdots=1. \qquad (4.14)$$

Учитывая выводы, полученные в предыдущей задаче, и полагая, что $y < S$, получим

$$P_0=\Delta, P_1=\Delta\frac{y}{1!},\ldots,P_s=\Delta\frac{y^s}{S!},\ldots,P_1^{'}=\Delta\frac{y}{S}\cdot\frac{y^s}{S!},\ldots P_j^{'}=\Delta\left(\frac{y}{S}\right)^j\frac{y^s}{S!}. \qquad (4.15)$$

Здесь

$$\frac{1}{\Delta}=1+\frac{y}{1!}+\cdots+\frac{y^{s-1}}{(S-1)!}+\frac{y^s}{S!}\cdot\frac{S}{S-y}. \qquad (4.16)$$

5. МЕТОДЫ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ

5.1. Основные понятия и структура СМО

Предмет теории массового обслуживания – системы массового обслуживания и сети массового обслуживания.

Под СМО понимаем динамическую систему, предназначенную для эффективного обслуживания случайного потока заявок (требований на обслуживание) при ограничениях на ресурсы системы.

Для современных ЭВМ и систем ЭВМ характерна работа в режиме решения потока случайных по своим характеристикам задач, поступающих в общем случае в случайные моменты времени. Анализ и, главное, синтез подобных систем, с учетом вероятностных характеристик протекающих в них процессов, сложной структуры СМО, подробным учетом функционирования различных подсистем и т. п., аналитическими методами невозможен. Для решения подобных задач чаще всего применяют методы теории массового обслуживания.

Рассмотрим обобщенную структурную схему СМО:

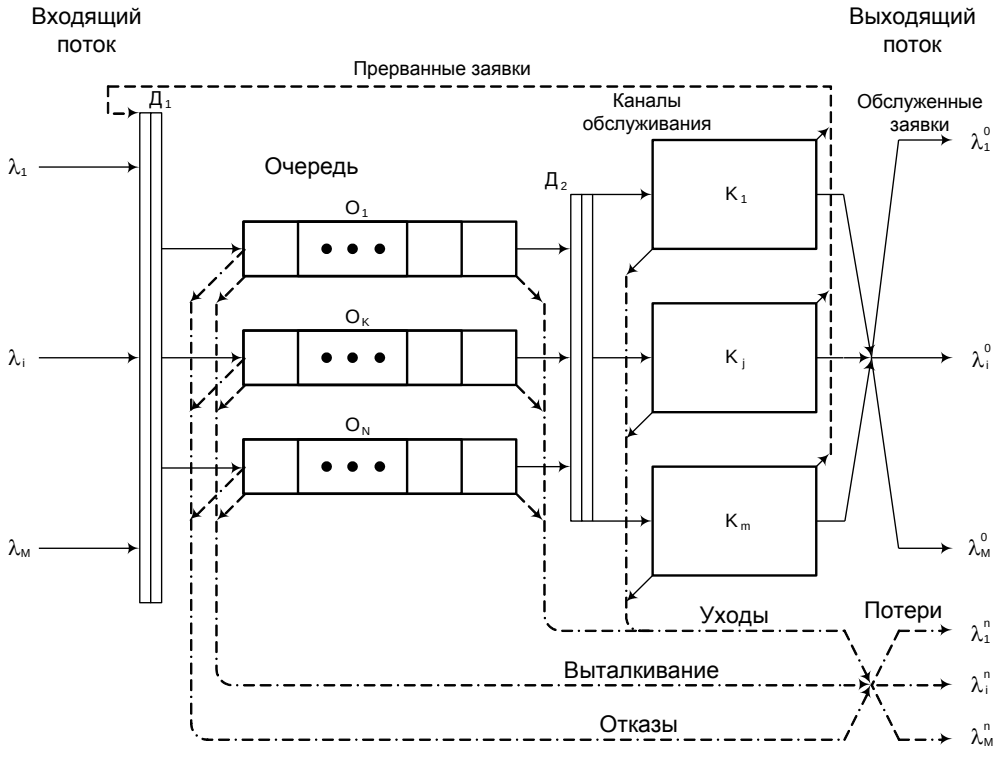

Рис. 13. Структура системы массового обслуживания:

λ_i – интенсивность потока заявок типа i ($i = \overline{1, M}$);

$F_{ij}(\tau)$ – длительность обслуживания каналом типа j ($j = \overline{1, m}$) заявок типа i ($i = \overline{1, M}$)

В произвольный момент времени *канал* может быть занят обслуживанием только одной заявки (запроса), которая в общем случае может быть прервана. Если в момент появления заявки λ_k хоть один канал свободен, ее обслуживание начинается немедленно, без задержки. В противном случае заявка задерживается и занимает место в соответствующей очереди.

Очередь может быть либо общей, либо раздельной: деление обычно производится по *приоритетам*. Число мест в очереди может быть ограничено, причем ограничиваться может как число мест для каждой очереди, так и для всей совокупности очередей. При этом в системе возможны конфликты:

- как за счет *отказов* системы принять новую заявку;
- так и за счет принятия в очередь более ценной заявки и *выталкивания* из очереди менее ценной для системы в данный момент времени.

Различают СМО с отказами и без отказов. В зависимости от допустимого времени пребывания заявки в системе различают СМО с «*нетерпеливыми*» и «*терпеливыми*» заявками:

- В СМО с «нетерпеливыми заявками» заявка может уйти из системы, если ее время пребывания в СМО превысит некоторое допустимое значение, которое в общем случае может быть случайным.
- «Терпеливые» заявки в СМО непременно дождутся обслуживания.

Процесс продвижения заявки от входа к выходу СМО происходит в соответствии с некоторым законом управления процессами в СМО, который задается дисциплинами ожидания и обслуживания ($Д_1$ и $Д_2$ соответственно, см. рис. 13). В зависимости от принятых в СМО дисциплин ожидания и обслуживания различают СМО с *бесприоритетными* и *приоритетными* дисциплинами:

- В СМО с бесприоритетными дисциплинами все заявки равноправны и никакая заявка не имеет преимуществ перед другой.
- В СМО с приоритетными дисциплинами одни типы заявок имеют более высокий приоритет по отношению к другим типам заявок.

Совокупность *обслуженных* и *потерянных* (полностью необслуженных или недообслуженных) заявок образует выходящий поток СМО. В зависимости от структуры выходящего потока различают СМО без потерь (*чистые* СМО) и СМО с потерями (*смешанные* СМО).

Для чистых СМО характерно отсутствие ограничений на число мест в очереди (бесконечная очередь) и на время пребывания заявки в системе (терпеливые заявки). По этой причине выходящий поток будет состоять лишь из обслуженных заявок.

Обычно в СМО выделяют две основные системные управляющие программы $Д_1$ и $Д_2$, реализующие соответственно дисциплины ожидания и обслуживания.

Модели СМО удобны для описания отдельных подсистем современных вычислительных систем, таких, как подсистема «процессор – основная па-

мять», «канал ввода-вывода» и т. д. Вычислительная система представляет собой совокупность взаимосвязанных подсистем, взаимодействие которых носит вероятностный характер.

Такая совокупность взаимосвязанных СМО называется *сетью массового обслуживания* (стохастической сетью).

Приведем пример стохастической сети, отражающей гипотетическую вычислительную систему и включающей три СМО. На рис. 14 приняты следующие обозначения:

S_0 – источник заявок; S_1 – подсистема «процессор – память»;

S_2 – селекторный канал; S_3 – мультиплексный канал.

Вероятности $P_{ij}(i, j = \overline{0,3})$ характеризуют относительную частоту поступления заявок, обслуженных системой массового обслуживания S_i, на вход системы массового обслуживания S_j.

Основная задача теории массового обслуживания – *задача синтеза* оптимальной структуры СМО или сети (ее состава и функциональных связей) при заданных свойствах, ограничениях на параметры входящего потока и ресурсы системы. Однако решение этой задачи возможно только после решения задачи анализа разнообразных структур СМО.

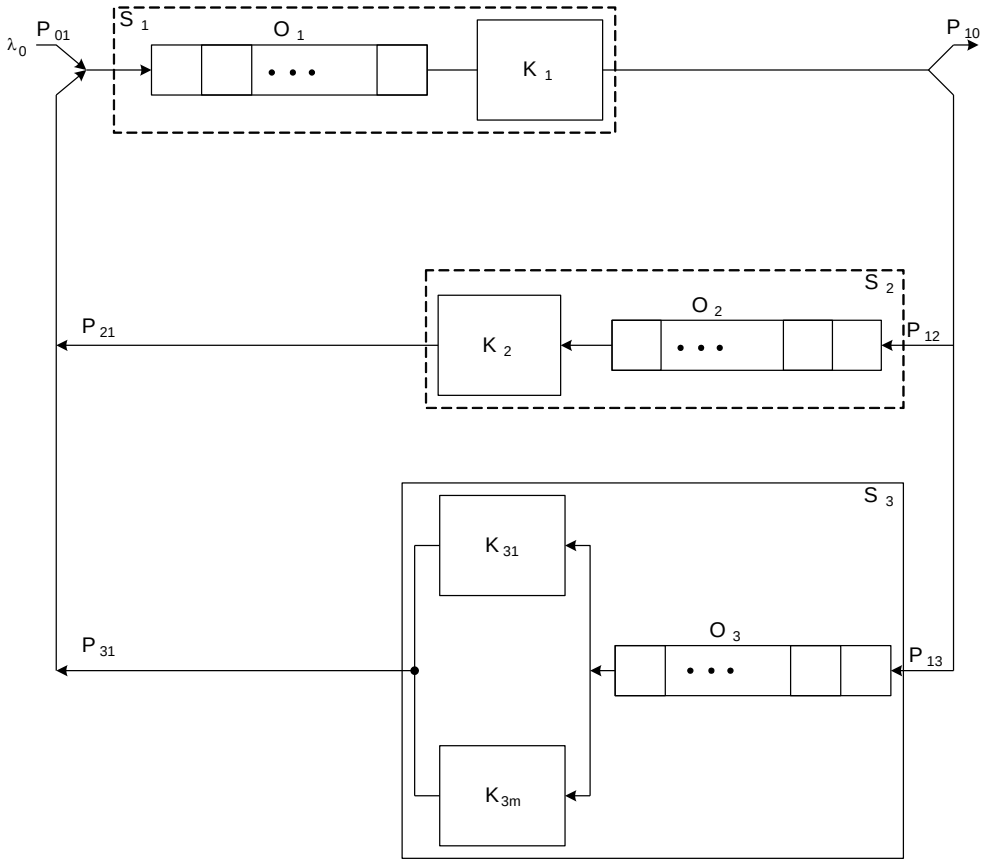

Рис. 14. Сеть массового обслуживания

Задача анализа – определение количественных показателей функционирования систем и сетей массового обслуживания и их зависимости от параметров входящего потока и структуры собственно системы или сети.

5.2. Параметры и характеристики СМО

Решение задачи анализа и синтеза СМО предполагает знание совокупности свойств исследуемой системы или сети, каждое из которых (свойств) может быть оценено количественно. Причем некоторые свойства СМО можно рассматривать как первичные (их количественные оценки называются параметрами СМО). Остальные свойства считаются вторичными, и их количественные оценки называются характеристиками СМО.

Кроме вышеназванных свойств, необходимо также задать (правило, функцию, выражение) критерий эффективности СМО, устанавливающий способ оценки качества системы (сети). Используя данные определения (понятия), СМО можно охарактеризовать совокупностью параметров и характеристик.

Параметры входящего потока

Процесс поступления в СМО заявок на обслуживание является, в общем случае, случайным и может рассматриваться как поток однородных событий, происходящих через случайные промежутки времени. Наибольшее распространение в теории массового обслуживания получил простейший поток заявок, т. е. поток, обладающий свойствами стационарности, ординарности и отсутствия последействия. Распространенность простейшего потока объясняется рядом обстоятельств:

- Допущение о простейшем потоке заявок позволяет получать аналитические зависимости характеристик СМО от параметров входящего потока, что затруднительно для других видов потоков заявок.
- При сложении нескольких независимых, ординарных, стационарных случайных потоков образуется суммарный поток, приближающийся по своим свойствам к простейшему.
- Если СМО обеспечивает желаемую эффективность функционирования при простейшем потоке заявок на входе, то обслуживание системой других случайных потоков с такой же интенсивностью будет выполняться по крайней мере не хуже.

Простейший поток в теории массового обслуживания играет такую же роль, как нормальный закон распределения СВ в теории вероятностей и математической статистике.

Если входящий поток представляет собой совокупность M потоков заявок различных типов с интенсивностями λ_i ($i=\overline{1,M}$), то его можно характеризовать *суммарной интенсивностью* $\lambda=\sum_{i=1}^{M}\lambda_i$.

К параметрам входящего потока относится также допустимое время пребывания заявки в СМО – $\tau_{\text{доп}}$, которое рассматривается в данном случае как свойство заявки. Применительно к вычислительным системам, работающим в режиме «реального времени», превышение $\tau_{\text{доп}}$ приводит к «устареванию» заявки, когда ее дальнейшее пребывание в СМО бесполезно, а в ряде случаев и вредно.

Параметры структуры СМО

Каждая СМО обладает определенной структурой, характеризирующейся совокупностью параметров. По составу СМО можно разделить:

- на СМО с одним каналом обслуживания (одноканальные);
- с M каналами обслуживания (многоканальные).

В свою очередь, многоканальные СМО могут содержать каналы с одинаковой или различной производительностью. Производительность канала обслуживания – величина обратная длительности обслуживания заявки. В общем случае длительность обслуживания – СВ с функцией распределения $F(\tau)$, плотностью $\rho(\tau)$ и математическим ожиданием $\overline{\tau_{\text{об}}}$. Типы заявок различаются либо законом распределения, либо $M[\tau_{\text{об}}]$ при одном и том же законе. При этом предполагаем, что для заявок одного типа длительности времени обслуживания – *независимы*. Наряду с параметром $M[\tau_{\text{об}}]$ используется понятие ***интенсивности потока обслуживания*** $\mu = 1/\overline{\tau_{\text{об}}}$, характеризующее количество заявок, которое может быть обслужено в единицу времени постоянно загруженным каналом.

Важный компонент структуры СМО – очередь, параметром которой является число мест n. В приоритетных системах общая очередь может быть разделена по числу различаемых системой приоритетов (очередей), для каждой из которых должно быть указано число мест

$$n_i, \quad \text{где} \quad i = \overline{1, N}; \quad n = \sum_{i=1}^{N} n_i .$$

Можно ограничивать не только n_i, но и n, увеличивая при этом число мест n_j – с большим приоритетом за счет мест n_k – с меньшим приоритетом. При неограниченном числе мест в очереди (бесконечная очередь) отсутствуют потери заявок за счет отказов и выталкивания.

5.3. Параметры закона управления процессами в СМО

Закон управления процессами в СМО должен выбираться таким образом, чтобы обеспечить оптимальное по отношению к выбранному критерию эффективности функционирование СМО, учитывая ограничения, наложенные на параметры входящего потока заявок и параметры структуры СМО.

Закон управления процессами в СМО можно представить совокупностью двух дисциплин:

- дисциплиной ожидания;
- дисциплиной обслуживания.

В ***бесприоритетных*** дисциплинах заявки какого-либо типа не имеют преимуществ перед заявками других типов ни при постановке в очередь, ни при назначении на обслуживание. Если по каким-то причинам заявки некоторых типов должны обслуживаться СМО быстрее, то они получают преимущество перед заявками других типов, называемое ***приоритетом***. Обычно приоритеты заявок характеризуются целыми положительными числами 1, 2, …, причем более высокому приоритету соответствует наименьшее число (бывает и наоборот). Дисциплины ожидания и обслуживания, учитывающие приоритеты, называются ***приоритетными***.

Дисциплина ожидания определяет правила управления очередью, возникающей в тех случаях, когда каналы обслуживания не справляются с потоком заявок. Рассмотрим ряд примеров бесприоритетной дисциплины ожидания:

– заявка принимается в общую очередь в порядке поступления; при переполнении очереди заявка получает отказ, т. е. теряется системой;

– заявка принимается в общую очередь в порядке поступления; при переполнении очереди вновь прибывшая заявка выталкивает из очереди заявку, дольше всех находящуюся в очереди;

– заявка принимается на свободное место, оставшееся после назначения заявок на обслуживание по случайному закону; при отсутствии свободного места заявка получает отказ.

Рассмотрим некоторые из возможных приоритетных дисциплин ожидания:

– заявка принимается в общую очередь в порядке поступления; при переполнении очереди учитываются приоритеты заявок: вновь поступившая может вытолкнуть из очереди только заявку низшего или аналогичного приоритета, причем внутри приоритета теряется самая старая заявка; если очередь заполнена более приоритетными заявками, то вновь поступившая заявка получает отказ;

– очередь распадается на несколько независимых очередей, каждая из которых предназначена для заявок определенного приоритета. Каждая очередь заполняется в порядке поступления, при переполнении какой-либо очереди вновь поступившая заявка соответствующего приоритета выталкивает из очереди самую старую необслуженную заявку.

Дисциплина обслуживания определяет правило выбора заявки из очереди на обслуживание. При бесприоритетном обслуживании возможны следующие дисциплины:

– выбирается первая в очереди заявка – дисциплина «первым пришел – первым вышел» (FIFO – First Input – First Output);

– выбирается последняя в очереди заявка – дисциплина «последним пришел – первым вышел» (LIFO – Last Input – First Output);

– заявка выбирается из очереди случайным образом.

При приоритетном обслуживании различают *относительные*, *абсолютные* и *смешанные* приоритеты.

Относительные приоритеты – учитываются только в момент назначения заявки на обслуживание, т. е. при освобождении канала обслуживание предоставляется заявке с наибольшим приоритетом. Относительные приоритеты можно рассмотреть на примере работы аэропорта:

– при одновременном запросе на взлет и посадку в первую очередь производится посадка лайнера, а потом взлет;

– при одновременном заходе на посадку нескольких самолетов, первым заходит на посадку аэробус, имеющий больше пассажиров; здесь минимизируется суммарное время ожидания.

Абсолютные приоритеты – предполагают прерывание обслуживания низкоприоритетной заявки в момент поступления в СМО заявки с более высоким приоритетом. Прерванная заявка ставится либо в начало общей очереди, либо в очередь заявок соответствующего приоритета. Обслуживание прерванных заявок может проводиться либо от начала (повторное обслуживание), либо от момента прерывания (дообслуживание). Чаще используют дообслуживание прерванных заявок.

Абсолютный приоритет можно рассмотреть на примере работы крупной больницы, где в поликлиническом отделении обслуживаются посетители разного профиля в соответствии с принятой дисциплиной обслуживания, в хирургических отделениях планируются (проводятся) плановые операции. Но при поступлении информации о крупной аварии (реальной или предполагаемой) плановое обслуживание больных прекращается и все наличные силы направляются на обслуживание травмированных.

Смешанные приоритеты – предполагают сочетание рассмотренных видов приоритетов, причем для отдельных заявок может быть использовано бесприоритетное обслуживание. Смешанные приоритеты реализуются в современных операционных системах, имеющих, как правило, полный набор приоритетов:

– процессы реального времени (real time) имеют максимальный абсолютный приоритет, устанавливаемый администратором;

– системные процессы (system) имеют абсолютные приоритеты, жестко установленные разработчиками системы;

– процессы разделения времени (time-shared) – пользовательские процессы, имеющие относительные приоритеты;

– процессы, выполняемые в фоновом режиме, – бесприоритетное обслуживание.

В общем случае предполагаются различные сочетания дисциплин ожидания и обслуживания.

5.4. Характеристики СМО

Характеристики СМО – вторичны по отношению к параметрам и могут быть разделены на две группы. Характеристики первой группы используются при рассмотрении СМО как элемента более сложной структуры, например сети СМО. Характеристики второй группы позволяют оценить способность конкретной СМО к выполнению возложенных на нее функций и называются ***показателями эффективности***.

Важными характеристиками первой группы являются *характеристики выходящего потока заявок*. Значение их важно, т. к. при рассмотрении сетей массового обслуживания выходящий поток одной СМО в общем случае является составляющей входящего потока другой СМО. Выходящий поток в общем случае распадается на поток обслуженных заявок и поток потерянных заявок. Каждый из них характеризуется законом распределения длительности интервалов между соседними заявками.

Если входящий поток содержит заявки M-типов с интенсивностями λ_i потока заявок i $(i=\overline{1,M})$, то выходящий поток можно охарактеризовать суммарной интенсивностью потока обслуженных заявок и суммарной интенсивностью потока потерянных по различным причинам заявок соответственно:

$$\lambda_{об}=\sum_{i=1}^{M}\lambda_{об.i},\quad \lambda_{п}=\sum_{i=1}^{M}\lambda_{п.i}\,.$$

Здесь $\lambda_{об.i}$ – интенсивность потока обслуженных заявок типа i;

$\lambda_{п.i}$ – интенсивность потока потерянных заявок типа i. Естественно, что выполняется соотношение $\lambda=\lambda_{об}+\lambda_{п}$.

Показатели эффективности СМО

Под показателем эффективности понимают количественный показатель, частично характеризирующий уровень выполнения СМО возложенных на нее функций. На основании показателей эффективности строится некоторый ***критерий эффективности***, совокупно характеризующий эффективность СМО при ограничении на ее параметры. Существуют различные показатели эффективности СМО. Рассмотрим наиболее употребительные из них.

Вероятность обслуживания $P_{об}$ – вероятность того, что произвольно выбранная из входящего потока с интенсивностью λ заявка будет обслужена, т. е. окажется в потоке обслуженных заявок с интенсивностью $\lambda_{об}$:

$$P_{об}=\lambda_{об}/\lambda.$$

Иногда $P_{об}$ называют *относительной пропускной способностью*.

Вероятность потери $P_{\text{п}}$ это вероятность того, что произвольно выбранная из входящего потока с интенсивностью λ заявка окажется в потоке потерянных заявок с интенсивностью λ_n:

$$P_{\text{п}} = \frac{\lambda_{\text{п}}}{\lambda} = \frac{\lambda - \lambda_{\text{об}}}{\lambda} = 1 - P_{\text{об}},$$

$$P_{\text{п}} = P_{\text{отк}} + P_{\text{в}} + P_{\text{у}},$$

где $P_{\text{отк}}$ – вероятность отказа, вследствие переполнения (насыщения) СМО;

$P_{\text{в}}$ – вероятность выталкивания заявки из очереди при некоторых дисциплинах ожидания;

$P_{\text{у}}$ – вероятность *ухода нетерпеливых заявок* из СМО.

Среднее время ожидания $\bar{\tau}_{\text{ож}}$ заявки (среднее время пребывания заявки в очереди) приблизительно равно математическому ожиданию времени ожидания заявки. Время ожидания заявки в системе $t_{\text{ож}}$ – это СВ равная сумме длительности интервалов времени, в течение которых заявка находится в очереди, начиная с момента появления заявки на входе СМО и заканчивая моментом, когда заявка в последний раз покидает очередь по причине назначения на обслуживание:

$$\bar{t}_{\text{ож}} = \overline{t'_{\text{ож}}} + \overline{t''_{\text{ож}}}.$$

Здесь $\overline{t'_{\text{ож}}}$ – среднее начальное время ожидания;

$\overline{t''_{\text{ож}}}$ – среднее время ожидания в прерванном состоянии (при абсолютном приоритете).

Среднее время пребывания заявки в СМО $\bar{t}_c$ – математическое ожидание времени пребывания заявки в СМО, равное промежутку времени от момента появления заявки на входе СМО до момента появления ее в выходящем потоке и связанное с дисциплинами ожидания и обслуживания:

$$\bar{t}_{\text{с}} = \bar{t}_{\text{ож}} + \bar{t}_{\text{об}}.$$

Средняя длина очереди $\bar{l}$ – математическое ожидание числа заявок, находящихся в очереди, т. е. длины очереди l. Для систем без потерь можно записать $\bar{l} = \bar{t}_{\text{ож}} \cdot \lambda$.

Среднее число занятых каналов обслуживания $\overline{K}$ равно математическому ожиданию числа занятых обслуживанием каналов. Эта СВ характеризует степень загрузки обслуживающей системы. Для определения $\overline{K}$ в общем случае необходимо знание совокупности $P_{\text{зан}.i}$, где $i = \overline{1, m}$ – вероятность того, что в произвольный момент времени занято обслуживанием i каналов. Важной характеристикой системы является ***загрузка*** ψ, выражающаяся через $\psi = \overline{K} / m$.

Среднее число заявок в системе $\overline{Z}$ – математическое ожидание числа заявок, находящихся в очереди и в каналах обслуживания, т. е. $\overline{Z} = \overline{l} + \overline{K}$. В эту формулу входит $\overline{K}$, т. к. с каждым каналом может быть связана только одна заявка.

Для СМО без потерь (типа выталкивания и ухода из очереди нетерпеливых заявок) среднее число заявок в системе равно

$$\overline{Z} = \overline{l} + \overline{K} = \lambda \cdot \bar{t}_{\text{ож}} + \lambda \cdot \bar{t}_{\text{об}} = \lambda \cdot (\bar{t}_{\text{ож}} + \bar{t}_{\text{об}}) = \lambda \cdot \bar{t}_{\text{с}}.$$

Критерии эффективности СМО

Критерий эффективности СМО – некоторая функция показателей эффективности, которая служит для совокупной оценки соответствия СМО выполнению возложенных на нее функций.

Выбор того или иного критерия эффективности зависит от назначения СМО, условий ее функционирования, выбора целевой функции и т. п.

Рассмотрим примеры критериев эффективности, применимые к вычислительным системам, работающим в режиме реального времени (цифровые управляющие системы, системы оперативной обработки данных).

Пусть заявка обесценивается пропорционально времени ее задержки в СМО, т. е. времени пребывания заявки в СМО (иногда учитывают лишь время ожидания в очереди). Тогда эффективность СМО определяется как

$$E = \sum_{i=1}^{M} e_{\text{с}.i} \cdot \bar{t}_{\text{с}.i},$$

где $e_{\text{с}.i}$ – штраф за единицу времени пребывания заявки i-го типа в СМО;

$\bar{t}_{\text{с}.i}$ – среднее время пребывания заявки i-го типа в СМО.

Штрафу может подвергаться потеря заявки системой, причем отдельно могут штрафоваться потери за счет отказов, выталкиваний и уходов. Тогда функционал может иметь вид

$$E_{\text{п}} = \sum_{i=1}^{M} [e_{\text{отк}.i} \cdot p_{\text{отк}.i} \cdot \lambda_i + e_{\text{в}.i} \cdot p_{\text{в}.i} \cdot \lambda_i + e_{\text{у}.i} \cdot p_{\text{у}.i} \cdot \lambda_i],$$

где $e_{\text{отк}.i}$ – штраф за отказ СМО принять заявку i-го типа; $e_{\text{в}.i}$ – штраф за выталкивание из очереди заявки; $e_{\text{у}.i}$ – штраф за уход из СМО нетерпеливой заявки; $p_{\text{отк}.i}$, $p_{\text{в}.i}$, $p_{\text{у}.i}$ – вероятности отказа, выталкивания и ухода из очереди заявки i-го типа, соответственно; λ_i – интенсивность входящего потока заявок i-го типа.

Критерии эффективности рассмотренных типов могут, в свою очередь, объединяться в новые, более сложные критерии эффективности.

Как характеристики, так и критерии эффективности СМО могут рассматриваться в одном из двух режимов работы: *переходном* и *установившемся*. В установившемся режиме поведение системы характеризуется обычно средними значениями характеристик.

5.5. Исследование систем массового обслуживания с простейшими потоками событий

Рассмотрим СМО, в которой все события (поступление заявок, уходы нетерпеливых заявок из системы, окончание обслуживания заявок каналами обслуживания и т. п.) можно рассматривать как простейшие потоки событий. Вследствие этого (ординарность, стационарность и отсутствие последействия) все процессы в СМО являются марковскими, что позволяет применять аппарат теории марковских цепей.

Рассмотрим СМО, содержащую m однотипных каналов обслуживания, характеризующихся экспоненциальным распределением времени обслуживания со средним значением $\bar{\tau}_{об}$ или, что эквивалентно, простейшим потокам обслуживания с интенсивностью $\mu = 1/\bar{\tau}_{об}$, независимо от типа обслуживаемой заявки. При загрузке m каналов заявки могут ждать обслуживания в общей очереди с числом мест, равным l. Дисциплины ожидания и обслуживания – бесприоритетные (дисциплина FIFO). При переполнении очереди вновь поступившая заявка получает отказ. Заявки на входе СМО относятся к одному из M типов, причем заявки i-го типа образуют простейший поток с интенсивностью λ_i $(i = \overline{1, M})$. Поскольку исследуется бесприоритетная СМО с общей очередью, целесообразно рассматривать объединенный входящий поток, который можно полагать простейшим с интенсивностью $\lambda = \sum_{i=1}^{M} \lambda_i$.

Будем считать заявки нетерпеливыми, т. е. имеющими право пробыть в СМО не более $\tau_{доп}$ единиц времени. Уходы нетерпеливых заявок возможны как из очереди, если $t_{ож} > \tau_{доп}$, так и из канала обслуживания, если $t_{ож} \le \tau_{доп}$ и $t_{с} > \tau_{доп}$. Предполагается, что $\tau_{доп}$ – СВ с экспоненциальной плотностью вероятности и математическим ожиданием $\bar{\tau}_{доп}$. Для дальнейшего рассмотрения удобно рассматривать простейший поток уходов из СМО с интенсивностью $\nu = 1/\bar{\tau}_{доп}$. Учитывая вышесказанное, удобно рассматривать два потока уходов с соответствующими интенсивностями $\nu_{ож}$ и $\nu_{об}$. Т. к. момент назначения заявки из очереди на обслуживание случайным образом располагается на интервале между соседними событиями в потоке уходов, то как промежуток времени между моментами назначения заявки на обслуживание и возможного ухода заявки из канала обслуживания будет иметь одинаковое экспоненциальное распределение с математическим ожиданием $\bar{\tau}_{доп}$. Причиной этого является свойство простейших потоков – отсутствие последействия. Следовательно, потоки уходов из очереди и канала обслуживания также будут простейшими с одинаковыми интенсивностями:

$$\nu_{ож} = \nu_{об} = \nu = 1/\bar{\tau}_{доп}.$$

Возможные состояния СМО будем связывать с числом заявок, находящихся в системе: S_0 – в СМО нет ни одной заявки; S_1 – в СМО одна заявка, ее обслуживает один канал, остальные m – 1 каналов простаивают, очередь отсутствует; S_m – в СМО m заявок, все каналы заняты, очередь отсутствует; S_{m+1} – в СМО m + 1 заявка, последняя заявка находится в очереди; S_{m+n} – в СМО $m + n$ заявок, все m каналов обслуживания заняты, все n мест в очереди заняты, СМО не способна принять ни одной дополнительной заявки, все вновь приходящие заявки получат отказ. Проиллюстрируем описанные состояния СМО графически (рис. 15).

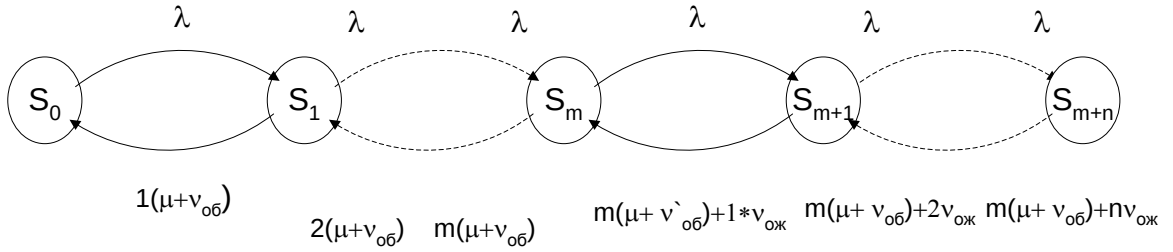

Рис. 15. Граф состояний СМО

Рассмотрим подробнее граф состояний исследуемой СМО. Переходы между состояниями такой СМО будут происходить под действием входящего потока заявок, потока уходов нетерпеливых заявок из очереди или канала обслуживания, потока обслуженных заявок. Тогда получим:

1. Переход в СМО в сторону увеличения индекса происходит под действием входящего потока заявок λ.

2. Для состояний $i = \overline{1,m}$ отсутствует очередь и переходы в сторону уменьшения индекса обусловлены двумя независимыми потоками: поток обслуживания с интенсивностью μ и поток ухода нетерпеливых заявок с интенсивностью $\nu_{об}$. Интенсивность суммарного потока равна $\mu + \nu_{об}$.

3. Т. к. процессы в различных каналах независимы, то интенсивность переходов равна $i \cdot (\mu + \nu_{об})$.

4. В состоянии S_{m+l} ($l = \overline{1,n}$) переход в сторону уменьшения индекса обусловлен двумя потоками:

- суммарная интенсивность переходов за счет потоков обслуживания и потоков ухода нетерпеливых заявок из каналов обслуживания, $\Rightarrow m \cdot (\mu + \nu_{об})$;
- другая составляющая соответствует интенсивности уходов нетерпеливых заявок из очереди и пропорциональна длине очереди l, $\Rightarrow l \cdot \nu_{ож}$.

Обычно исследователей интересует установившийся процесс в СМО, т. е. $t \to \infty$. Учитывая ранее сказанное (свойство цепей Маркова), для получения предельных вероятностей состояний воспользуемся формулами Эрланга

$$\left.\begin{aligned} P_i &= \frac{\lambda^i}{i!\cdot(\mu+\nu_{об})^i}P_0 && \left(i=\overline{1,m}\right);\\ P_{m+l} &= \frac{\lambda^m}{m!\cdot(\mu+\nu_{об})^m}\prod_{i=1}^{l}\frac{\lambda}{[m\cdot(\mu+\nu_{об})+i\cdot\nu_{ож}]}P_0 && \left(l=\overline{1,n}\right). \end{aligned}\right\} \quad (5.1)$$

Приведем интенсивности всех потоков к интенсивности потока обслуживания μ:

$$\left.\begin{aligned} P_i &= \frac{\rho^i}{i!\cdot(1+\alpha_{об})^i}P_0 && \left(i=\overline{1,m}\right);\\ P_{m+l} &= \frac{\rho^m}{m!\cdot(1+\alpha_{об})^m}\prod_{i=1}^{l}\frac{\rho}{[m\cdot(1+\alpha_{об})+i\cdot\alpha_{ож}]}P_0 && \left(l=\overline{1,n}\right). \end{aligned}\right\} \quad (5.2)$$

В формуле (5.2) приняты следующие обозначения: $\rho=\lambda/\mu-$ приведенная интенсивность входящего потока, представляющая собой среднее число заявок, поступающих на вход СМО за среднее время обслуживания одной заявки; $\alpha_{об}=\nu_{об}/\mu-$ приведенная интенсивность потока уходов из канала обслуживания; $\alpha_{ож}=\nu_{ож}/\mu-$ приведенная интенсивность потока уходов из очереди. Вспомним, что

$$\sum_{i=0}^{m+n}P_i=1. \quad (5.3)$$

Теперь с учетом (5.2) и (5.3) получили систему из $m + n + 1$ линейных уравнений с $m + n + 1$ неизвестных. Учитывая вышесказанное, окончательно получим:

$$P_0=\left[1+\sum_{i=1}^{m}\frac{\rho^i}{i!\cdot(1+\alpha_{об})^i}+\frac{\rho^m}{m!\cdot(1+\alpha_{об})^m}\sum_{l=1}^{n}\prod_{i=1}^{l}\frac{\rho}{[m\cdot(1+\alpha_{об})+i\cdot\alpha_{ож}]}\right]^{-1}. \quad (5.4)$$

Один из важных показателей эффективности СМО – среднее число каналов $\overline{K}$, занятых обслуживанием:

$$\overline{K}=0\cdot P_0+\sum_{i=1}^{m}i\cdot P_i+m\cdot\sum_{l=1}^{n}P_{m+l}=\sum_{i=0}^{m}i\cdot P_i+m\cdot\left(1-\sum_{i=0}^{m}P_i\right). \quad (5.5)$$

Аналогично определяется средняя длина очереди $\bar{l}$:

$$\bar{l}=0\cdot\sum_{i=1}^{m}P_i+\sum_{l=1}^{n}l\cdot P_{m+l}=\sum_{l=1}^{n}l\cdot P_{m+l}. \quad (5.6)$$

Из выражений (5.5) и (5.6) можно получить среднее число заявок находящихся в системе:

$$\overline{Z} = \bar{l} + \overline{K} = \sum_{l=1}^{n} l \cdot P_{m+l} + \sum_{i=0}^{m} i \cdot P_i + m \cdot (1 - \sum_{i=0}^{m} P_i). \quad (5.7)$$

В нашей СМО потери заявок возможны вследствие переполнения системы либо вследствие ухода нетерпеливых заявок. Вероятность отказа можно определить как вероятность нахождения системы в состоянии S_{m+n}:

$$P_{\text{отк}} = P_{m+n} = \frac{\rho^{m+n}}{m! \cdot (1+\alpha_{\text{об}})^m \cdot \prod_{i=1}^{n} [m \cdot (1+\alpha_{\text{об}}) + i \cdot \alpha_{\text{ож}}]} P_0. \quad (5.8)$$

Введем обозначения $P_{\text{у.ож}}$, $P_{\text{у.об}}$ – вероятность ухода нетерпеливой заявки во время ожидания в очереди и во время обслуживания в канале соответственно. Учитывая то, что рассматриваемые события (уходы) несовместны, вероятности суммируются:

$$P_{\text{у}} = P_{\text{у.ож}} + P_{\text{у.об}}. \quad (5.9)$$

Вероятность ухода заявки во время обслуживания определяется

$$P_{\text{у.об}} = \overline{K} \cdot \nu_{\text{ух}} / \lambda. \quad (5.10)$$

Аналогично определяем вероятность ухода при ожидании в очереди:

$$P_{\text{у.ож}} = \bar{l} \cdot \nu_{\text{ух}} / \lambda. \quad (5.11)$$

Т. к. рассматриваемые процессы несовместны, то из этого следует

$$P_{\text{п}} = P_{\text{отк}} + P_{\text{у.ож}} + P_{\text{у.об}}, \quad (5.12)$$

$$P_{\text{об}} = 1 - P_{\text{п}}, \quad (5.13)$$

$$\lambda_{\text{об}} = \lambda \cdot P_{\text{об}} = \lambda \cdot (1 - P_{\text{п}}). \quad (5.14)$$

6. МОДЕЛИРОВАНИЕ ВНЕШНИХ ВОЗДЕЙСТВИЙ

6.1. Стохастическое моделирование

В общем случае исследуемая система содержит ряд элементов, обладающих некоторой неопределенностью. Подобные системы обычно называют стохастическими (вероятностными), т. к. их поведение не может быть полностью предсказано заранее. При имитации стохастических систем требуется описывать изменчивость элементов в терминах теории вероятностей. Не следует также забывать, что результаты, полученные с помощью имитационной модели, носят вероятностный характер и требуют статистической интерпретации.

В основе стохастического моделирования лежит метод статистических испытаний – Метод Монте-Карло (ММК).

Следует сразу выделить два основных класса задач, решаемых ММК:

1. Во-первых, ММК позволяет моделировать любой процесс, на протекание которого влияют случайные факторы. (Различные задачи этого класса мы позднее рассмотрим подробно).

2. Во-вторых, для многих математических задач – детерминированных, т. е. не связанных с какой-либо вероятностью, можно искусственно придумать вероятностную модель (и даже не одну), позволяющую решать эту задачу.

Примером такой задачи является вычисление определенного интеграла $I = \int_a^b f(x)dx$, который геометрически равен площади заштрихованной фигуры (рис. 16).

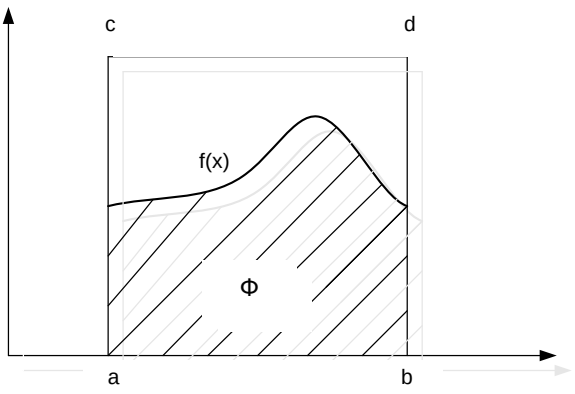

Рис. 16. Интегрирование методом Монте-Карло

Рассмотрим теперь следующую вероятностную модель:

1. Допустим, мы умеем моделировать на декартовой плоскости точки, равномерно распределенные в прямоугольнике *abcd*.

2. Пусть мы проводим этот опыт N раз, из них n точек попали в заштрихованную фигуру Φ, тогда из геометрического определения вероятности и чисто интуитивно можно записать

$$\frac{S(\Phi)}{S_{abcd}} \approx \frac{n}{N}, \quad \Rightarrow \quad I = S(\Phi) \cong S_{abcd} \cdot \lim_{N\to\infty} \frac{n}{N}.$$

Очевидно, что аналогичным образом можно решить задачу об объеме некоего тела и т. п.

История развития метода Монте-Карло

Название «Монте-Карло» происходит от города Монте-Карло (княжество Монако), известного своим казино, т. к. одним из простейших приборов для генерирования случайных чисел служит рулетка.

Официальной датой рождения ММК считают 1949 г., когда появилась статья под заглавием «Метод Монте-Карло». Возникновение метода связывают с именами Дж. Фон Неймана, С. Улама, Н. Метрополиса, а также Г. Канна и Э. Ферми. Все эти ученые в 40-х гг. XX в. работали в Лос-Аламосе (США). Необходимо сразу же подчеркнуть, что теоретические основы метода Монте-Карло были известны значительно раньше. Более того, фактически такие методы не раз использовались для расчетов в математической статистике. Однако до появления ЭВМ эти методы не могли стать универсальными и широко распространенными, т. к. моделирование случайных величин вручную – весьма трудоемкий процесс.

Первоначально ММК использовался для решения задач нейтронной физики, где традиционные численные методы оказались малопригодными. В дальнейшем его влияние распространилось на широкий класс задач статистической физики, теории массового обслуживания, задачи теории игр и математической экономики, задачи теории передачи информации при наличии помех и т. п.

Основы метода Монте-Карло

Теоретической основой ММК являются закон больших чисел и предельные теоремы теории вероятности.

Действительно, свойства устойчивости массовых случайных явлений известны человечеству с давних пор. Именно эта устойчивость средних и составляет физическое содержание «закона больших чисел» вместе с предельными теоремами теории вероятности. Принципиально важно то, что предельные теоремы теории вероятностей гарантируют высокое качество статистических оценок при большом числе реализаций (испытаний) N. А именно это, т. е. большое число испытаний мы и имеем при моделировании на ЭВМ.

Неравенство Чебышева

Имеется случайная величина (СВ) $X \Rightarrow m_x, D_x$. Для любого $\varepsilon > 0$ выполняется неравенство

$$P(|X - m_x| > \varepsilon) \le \frac{D_x}{\varepsilon^2}. \tag{6.1}$$

Теорема Чебышева

Говорят, что СВ X_n сходится по вероятности к величине a, если, при увеличении n, вероятность того, что X_n и a будут сколь угодно близки, неограниченно приближается к 1, т. е. при достаточно большом n:

$$P(|X_n - a| < \varepsilon) > 1 - \delta. \tag{6.2}$$

Здесь $\varepsilon, \delta -$ произвольно малые положительные числа.

Имеется случайная величина (СВ) $X \Rightarrow m_x, D_x$. Пусть над этой СВ X проделано n независимых опытов и получено n значений СВ X: $x_1, x_2, \ldots, x_i, \ldots, x_n$ – значение X в i-ом опыте. Необходимо найти характеристики среднего арифметического СВ – математическое ожидание и дисперсию. Вспомним, что среднее арифметическое равно

$$Y = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{6.3}$$

СВ Y – линейная функция независимых СВ $X_1, X_2, \ldots, X_n$. Тогда

$$m_y = M[y] = \frac{1}{n}\sum_{1}^{n} M[x] = \frac{1}{n} \cdot n \cdot m_x = m_x, \tag{6.4}$$

$$D_y = \frac{1}{n^2} \cdot \sum_{1}^{n} D[x_i] = \frac{D_x}{n} = \frac{\sigma^2}{n}. \tag{6.5}$$

Таким образом, m_y не зависит от числа опытов, в то время как дисперсия среднего D_y может стать сколь угодно малой при $n \to \infty$.

При достаточно большом числе независимых опытов среднее арифметическое значений случайной величины сходится по вероятности к ее математическому ожиданию (теорема Чебышева):

$$P\left(\left|\frac{1}{n} \cdot \sum_{1}^{n} x_i - m_x\right| < \varepsilon\right) > 1 - \delta. \tag{6.6}$$

Обобщенная теорема Чебышева

Пусть имеются $X_1, X_2, \ldots, X_n$ – независимые СВ с соответствующими математическими ожиданиями и дисперсиями $m_{x_1}, D_{x_1}; m_{x_2}, D_{x_2}; \ldots m_{x_n}, D_{x_n}$. Допустим, что дисперсия СВ – ограничена, т. е. $\max\limits_{i=1,n} D[x_i] \le L$.

Тогда при возрастании n среднее арифметическое значений величин $X_1, X_2, \ldots X_n$ сходится по вероятности к среднему арифметическому их математических ожиданий:

$$P\left(\left|\frac{1}{n}\cdot\sum_1^n X_i - \frac{1}{n}\cdot\sum_1^n m_{x_i}\right| < \varepsilon\right) > 1-\delta. \qquad (6.7)$$

Теорема Маркова

Пусть имеются зависимые СВ $X_1, X_2, \ldots, X_n$ с соответствующими математическими ожиданиями и дисперсиями $m_{x_1}, D_{x_1}; m_{x_2}, D_{x_2}; \ldots m_{x_n}, D_{x_n}$. Мы не требуем, чтобы дисперсии были ограничены. Однако если при $n \to \infty$ выполняется соотношение

$$\frac{D[\sum_{i=1}^n X_i]}{n^2} \to 0,$$

то среднее арифметическое значений СВ $X_1, X_2, \ldots, X_n$ сходится по вероятности к среднему арифметическому их математических ожиданий:

$$P\left(\left|\frac{1}{n}\cdot\sum_1^n X_i - \frac{1}{n}\cdot\sum_1^n m_{x_i}\right| < \varepsilon\right) > 1-\delta. \qquad (6.8)$$

Теорема Бернулли

Пусть производится n независимых опытов, в каждом из которых с вероятностью p может произойти событие A. Тогда при неограниченном возрастании n частота события A сходится по вероятности к его вероятности p

$$P\left(\left|p^* - p\right| < \varepsilon\right) > 1-\delta. \qquad (6.9)$$

Эту теорему можно рассматривать как следствие теоремы Чебышева, т. к. частота события А определяется выражением

$$p^* = \frac{1}{n}\sum_1^n X_i .$$

Центральная предельная теорема

Если $X_1, X_2, \ldots, X_n$ – независимые, одинаково распределенные случайные величины, имеющие математическое ожидание a и дисперсию σ^2, то при $n \to \infty$ закон распределения суммы $\sum_{i=1}^n x_i$ неограниченно приближается к нормальному:

$$\lim_{n\to\infty} P\left\{\alpha < (\sum_{i=1}^{n} x_i - Na)/\sqrt{N}\sigma < \beta\right\} = \frac{1}{\sqrt{2\pi}}\int_{\alpha}^{\beta} e^{-\frac{t^2}{2}}\,dt. \qquad (6.10)$$

Очевидно, что при моделировании стохастических процессов необходимо уметь моделировать значения случайных величин, имеющих различные функции распределения.

6.2. Моделирование распределений

При использовании метода Монте-Карло исходной случайной величиной является одномерная, равномерно распределенная на промежутке [0, 1] СВ α. Плотность распределения СВ α имеет вид

$$\rho_\alpha(x) = \begin{cases} 1, & x \in [0,1]; \\ 0, & x \notin [0,1]. \end{cases} \qquad (6.11)$$

Если α равномерно распределена на интервале [0, 1] с плотностью (6.11), то СВ $\beta = a + \alpha\cdot(b-a)$ – равномерно распределена на промежутке $[a, b]$, где $(b > a)$. Плотность распределения СВ β имеет вид

$$\rho_\beta(x) = \begin{cases} \dfrac{1}{b-a}, & x \in [a,b]; \\ 0, & x \notin [a,b]. \end{cases} \qquad (6.12)$$

Моделирование дискретных случайных величин

Пусть имеется дискретная СВ ξ с распределением:

X	x_1	x_2	…	x_n
P	p_1	p_2	…	p_n

(6.13)

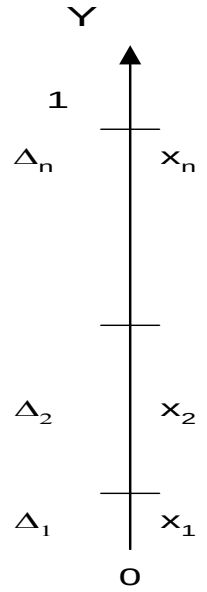

Рис. 17. Моделирование дискретных СВ

Здесь $p_i = P\{\xi = x_i\}$. Чтобы вычислить (смоделировать) значение этой величины, разделим интервал $0 \le y \le 1$ на подынтервалы Δ_i такие, что длина Δ_i равняется p_i.

Теорема: СВ ξ, определенная формулой

$$\xi = x_i, \quad \text{когда} \quad \gamma = \Delta_i \qquad (6.14)$$

имеет распределение вероятностей (6.13).

Доказательство: Действительно,

$$P\{\xi = x_i\} = P\{\gamma = \Delta_i\} = \text{длина } \Delta_i = P_i.$$

Для практической реализации формулы (6.14) на ЭВМ строят функцию (рис. 17).

x_1	x_2	x_3	…	x_n
p_1	$p_1 + p_2$	$p_1 + p_2 + p_3$	…	1

(6.15)

При моделировании рекомендуется функцию (6.15) строить таким образом, чтобы первые места слева занимали значения СВ, имеющие максимальные вероятности P_i, тогда скорость работы программы будет максимальной.

Моделирование непрерывных случайных величин

Пусть СВ ξ определена на интервале $a \le x \le b$ и имеет плотность распределения $\rho(x)$ при $x \in [a,b]$. Тогда $F(x)$ – интегральная функция распределения СВ ξ, которая при $x \in [a,b]$ равна:

$$F(x) = \int_a^x \rho(u)du. \qquad (6.16)$$

Здесь случай $a = -\infty$ и (или) $b = \infty$ не исключается.

Теорема: СВ ξ, удовлетворяющая уравнению

$$F(\xi) = \gamma, \qquad (6.17)$$

имеет плотность распределения $\rho(x)$.

Доказательство: Т. к. функция $F(x)$ строго возрастает в интервале $[a, b]$ от $F(a) = 0$ до $F(b) = 1$ (рис. 18), то уравнение (6.17) имеет единственный корень при каждом γ. При этом равны вероятности

$$P\{x < \xi < x + dx\} = P\{F(x) < \gamma < F(X + dx\}. \qquad (6.18)$$

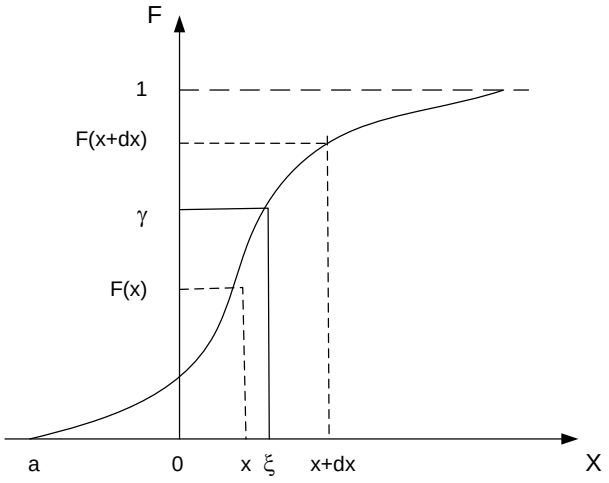

Рис. 18. Функция распределения

И т. к. СВ γ равномерно распределена в интервале $[0, 1]$, то

$$P\{x < \xi < x + dx\} = F(x + dx) - F(x) = \rho(x)dx. \qquad (6.19)$$

Что и требовалось доказать.

Особый интерес представляют случаи, когда уравнение (6.17) аналитически разрешимо относительно ξ. Тогда получается явная формула $\xi = \Psi(\gamma)$, где $\Psi(\gamma)$ – функция обратная по отношению к $y = F(x)$.

Пример: Дана СВ ξ, распределенная по экспоненциальному закону на интервале $x_0 \le x \le \infty$ с плотностью

$$\rho(x) = a \cdot e^{a(x-x_0)}. \tag{6.20}$$

Известно, что

$$F(x) = \int_{x_0}^{x} a \cdot e^{-a(u-x_0)} du = 1 - e^{-a(x-x_0)}. \tag{6.21}$$

Для данного случая уравнение (6.17) примет вид:

$$1 - e^{-a(\xi - x_0)} = \gamma. \tag{6.22}$$

Отсюда в явном виде получим

$$\xi = x_0 - \frac{1}{a} \cdot \ln(1-\gamma). \tag{6.23}$$

Замечание I*:* Т. к. СВ $1-\gamma$ имеет то же распределение, что и γ, то во многих выражениях, например (6.23), можно одну СВ заменить другой. Тогда формула (6.23) примет вид

$$\xi = x_0 - \frac{1}{a} \cdot \ln(\gamma). \tag{6.24}$$

Замечание II: Несмотря на то, что метод обратной функции позволяет записать формулу практически для любой СВ ξ. Порой он приводит к сложным или просто неудобным вычислениям. Примером таких вычислений является моделирование СВ, распределенной по нормальному (гауссову) закону $N(0,1)$.Метод обратной функции приводит к уравнению

$$\int_{-\infty}^{\xi} e^{-\frac{t^2}{2}} dt = \sqrt{2 \cdot \pi} \cdot \gamma. \tag{6.25}$$

Интеграл в левой части уравнения (6.25) – не берущийся.

Метод отбора (метод Неймана)

Этот остроумный способ чрезвычайно удобен и нагляден, т. к. в принципе соответствует плотности вероятности.

Теорема: Пусть γ_1 и γ_2 – независимые СВ равномерно распределенные на интервале $[a, b]$ и пусть $\xi' = a + \gamma_1(b-a), \quad \eta' = c \cdot \gamma_2$. Случайная величина ξ, определенная условием

$$\xi = \xi', \quad \text{если} \quad \eta' < \rho(\xi'). \tag{6.26}$$

Доказательство: Во-первых, следует отметить, что точка (ξ', η') равномерно распределена в прямоугольнике $a \le x \le b, \quad 0 \le y \le c$. Теперь вычислим условную вероятность

$$P(\xi < z) = P\{\xi' < z / \eta' < \rho(\xi')\} = \frac{P\{\xi' < z, \eta' < \rho(\xi')\}}{P\{\eta' < \rho(\xi')\}}. \tag{6.27}$$

Знаменатель выражения (6.27) – это вероятность попадания точки (ξ', η') под кривую $y = \rho(x)$ (рис. 19).

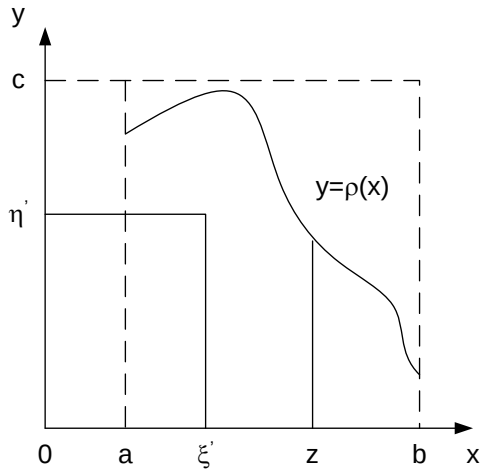

Рис. 19. Метод Неймана

Т. к. плотность распределения СВ (ξ', η') – постоянна и равна $[c \cdot (b-a)]^{-1}$, то получим

$$P\{\eta' < \rho(\xi')\} = \int_a^b dx \int_0^{\rho(x)} [c \cdot (b-a)]^{-1} dy = [c \cdot (b-a)]^{-1}. \qquad (6.28)$$

Числитель выражения (6.27) равен вероятности того, что точка (ξ', η') окажется под кривой и в то же время $\xi' < z$:

$$P\{\zeta' < z, \eta' < \rho(\xi')\} = \int_a^z dx \int_0^{\rho(x)} [c \cdot (b-a)]^{-1} dy =$$

$$= [c \cdot (b-a)]^{-1} \int_a^z \rho(x) dx. \qquad (6.29)$$

Теперь подставим полученные выражения (6.28) и (6.29) в (6.27):

$$P(\xi < z) = \int_a^z \rho(x) dx. \qquad (6.30)$$

Что и требовалось доказать.

Пример: СВ ξ определена на интервале $-R \le x \le R$ с плотностью $\rho(x) = (2\pi R^2)^{-1} \sqrt{R^2 - x^2}$.

Согласно доказанной теореме выбираем два независимых значения, равномерно распределенных на $[0,1]$ СВ γ_1 и γ_2. Затем вычисляем $\xi' = -R + \gamma_1 (R+R) = R \cdot (2\gamma_1 - 1)$ и $\eta' = (2\pi R)^{-1} \gamma_2$. Если $\eta' < \rho(\xi')$, то $\xi = \xi'$, т. е.

$$(2\pi R)^{-1} \gamma_2 < (2\pi R^2)^{-1} \sqrt{R^2 - R^2 (2\gamma_1 - 1)^2},$$

$$\gamma_2 < \sqrt{1 - (2\gamma_1 - 1)^2}.$$

После дальнейших упрощений получим

$$\xi = R(2\gamma_1 - 1), \text{ если } (2\gamma_1 - 1)^2 < 1 - \gamma_2^2. \qquad (6.31)$$

Эти формулы очень просты для реализации, правда, не для каждой пары γ_1 и γ_2 решение удовлетворяется. Оценивая эффективность, получим $Э = \pi/4$, т. е. для одного вычисления (моделирования) СВ необходимо взять $4/\pi$ пар значений γ_1 и γ_2.

Если в этом примере сразу применить метод обратной функции, то получим

$$F(\xi) \equiv \frac{1}{2} + \frac{1}{\pi}\arcsin\frac{\xi}{R} + \frac{\xi}{\pi R^2}\sqrt{R^2 - \xi^2} = \gamma. \qquad (6.32)$$

Понятно, что решение трансцендентного уравнения (6.32) – задача более сложная и трудоемкая, чем метод Неймана.

Правда, если в исходной задаче перейти к полярным координатам, т. е. дополнить второй СВ η так, чтобы точка в декартовых координатах зависела от $r = \sqrt{x^2 + y^2}$. А затем от двумерной плотности перейти к одномерной, и в явном виде получим

$$\xi = R\sqrt{\gamma_1}\cos 2\pi\gamma_2. \qquad (6.33)$$

6.3. Генерирование равномерно распределенных случайных чисел на ЭВМ

Узловой проблемой метода Монте-Карло является моделирование случайных чисел ξ, равномерно распределенных в интервале $[0, 1]$. Идея о возможности генерирования случайных чисел (СЧ) с помощью детерминированного процесса впервые возникла где-то в 1946 г. До этого их получали с помощью «случайного выбора» цифр из данных переписи населения, телефонных справочников, а также с помощью специально сконструированных устройств. Так, для составления таблицы СЧ использовалась специально сконструированная электронная рулетка. Делались также попытки оборудовать первые ЭВМ физическими датчиками СЧ. Их недостаток заключается в следующем:

- нельзя гарантировать постоянную удовлетворительную работу датчика;
- невозможно повторить расчеты с той же последовательностью СЧ, а это не позволяет использовать корреляционные методы для оценки тонких эффектов, затрудняет нахождение ошибок в программах.

Как правило, для решения различных задач методом Монте-Карло используют рекуррентные формулы, производящие последовательности *псевдослучайных чисел*. Задание начального числа определяет такую последова-

тельность полностью, но многие ее свойства аналогичны свойствам случайно выбранных значений, что и обеспечивает успех моделирования.

Числа $\gamma_1,\gamma_2,\ldots,\gamma_n$, которые вычисляются по какой-либо заданной формуле и могут быть использованы вместо СЧ при решении задач, называются *псевдослучайными*.

Конкретная последовательность псевдослучайных чисел сходна с таблицей СЧ и обладает рядом свойств:

- все числа легко воспроизводятся,
- запас чисел в этой последовательности тоже ограничен,
- ее можно один раз тщательно проверить и затем многократно применять.

Алгоритм 1

Большинство алгоритмов, используемых на практике для получения псевдослучайных чисел, представляют собой рекуррентные формулы первого порядка

$$\gamma_{n+1} = \Phi(\gamma_n), \qquad (1)$$

где начальное значение γ_0 задано. Иллюстрация алгоритма (1) представлена на рис. 20, а.

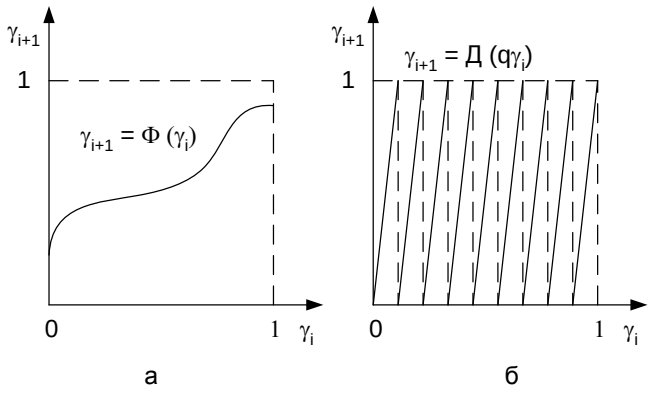

Рис. 20. Алгоритмы датчика СЧ

Очевидно, что эта функция в качестве датчика случайных чисел плохая. Действительно, СЧ $\{\gamma_1,\gamma_2,\ldots,\gamma_n\}$ должны плотно покрывать единичный квадрат. А мы имеем: $\{(\gamma_1,\Phi(\gamma_1));(\gamma_3,\Phi(\gamma_3));(\gamma_5,\Phi(\gamma_5));\ldots\}$, т. е. эти точки располагаются на кривой $y=\Phi(x)$. Отсюда и вытекает первое требование – выбранная нами функция в (1) должна возможно плотнее покрывать единичный квадрат.

Алгоритм 2

В основе данного алгоритма лежит выражение

$$y = Д(qx). \qquad (2)$$

Здесь $q-$ очень большое целое число, $Д(x)-$ дробная часть x.

При правильном выборе формула (2) может дать более удачную последовательность СЧ, покрывающую единичный квадрат (см. рис. 20, б).

Однако независимо от того, насколько удачно реализована рекуррентная формула (1), при реализации алгоритма на ЭВМ всегда порождаются периодические последовательности.

Действительно, т. к. любая ЭВМ оперирует ячейками конечной разрядности, а, следовательно, в ней можно записать только N (конечное) различных чисел, заключенных между 0 и 1. Тогда рано или поздно какое-то значение γ_L совпадет с γ_l. А тогда в соответствии с (1)

$$\gamma_{L+i} = \gamma_{l+i} \qquad (i = 1,2,\ldots). \tag{3}$$

Пусть $L-$ наименьшее число, удовлетворяющее выражению (3) при некотором значении $l(l < L)$. Множество чисел $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_{L-1}$ называется *отрезком апериодичности* последовательности (1). Число $L-$ *длиной апериодичности*, а $P = L - l -$ *длиной периода* (рис. 21).

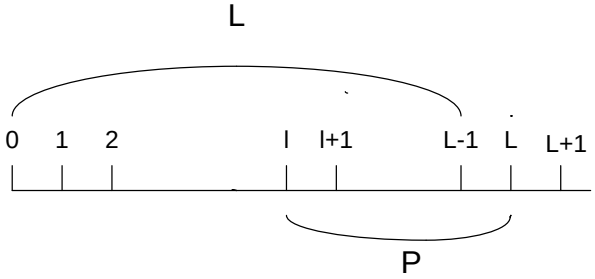

Рис. 21. Последовательность
псевдослучайных чисел

Алгоритм 3

Метод был предложен Дж. Нейманом и его называют методом середины квадрата. Суть метода состоит в следующем, число $\gamma_n -$ предполагается $2k -$ значным:

$$\gamma_n = 0, \alpha_1\alpha_2\alpha_3\ldots\alpha_{2k}.$$

Чтобы получить γ_{n+1}, необходимо возвести γ_n в квадрат, т. е.

$$\gamma_n^2 = 0, \beta_1\beta_2\beta_3\ldots\beta_{4k},$$

и затем оставляем средние $2k -$ цифры:

$$\gamma_{n+1} = 0, \beta_{k+1}\beta_{k+2}\beta_{k+3}\ldots\beta_{3k}.$$

Очевидно, если вспомнить выражение (1), этому алгоритму соответствует функция

$$\Phi(x) = Д[10^{-2k}\, Ц(10^{3k} x^2)]. \tag{4}$$

Недостатком метода является получение слишком большого числа малых чисел.

Алгоритм 4

Алгоритм, предложенный Д. Лемером, который называют методом вычетов, или методом сравнений, является частным случаем алгоритма (1) и получил в настоящее время большое распространение:

$$m_{n+1} = q \cdot m_n (\mathrm{mod} M). \qquad (5)$$

В формуле (5) новое значение m_{n+1} равно остатку, полученному при делении $q \cdot m_n$ на M. Известна реализация алгоритма (5) при $q = 5^{17}$, $M = 2^{42}$, $m_0 = 1, \Rightarrow (L = P = 2^{40})$. При другом выборе q, M и m_0 можно получить очень неудачные последовательности с очень короткими длинами апериодичности.

В конечном итоге вопрос о пригодности полученной последовательности псевдослучайных чисел решается с помощью статистических тестов.

6.4. Тестирование датчика случайных чисел

Эффективность стохастического моделирования на ЭВМ и достоверность получаемых результатов в большой степени зависит от качества используемой последовательности случайных чисел. Поэтому прежде чем приступать к собственно моделированию, необходимо удостовериться в качестве датчика случайных чисел (ДСЧ).

Оценка характеристик ДСЧ

Для уверенности в том, что в процессе моделирования мы не встретимся с циклом (использование одних и тех же СЧ), необходимо оценить длину апериодичности L и длину периода P.

Шаг 1: Задаем достаточно большое целое число, которое заведомо превышает длину последовательности СЧ, используемых в одном расчете. Обычно $k = 10^5 - 10^8$.

Шаг 2: В цикле запускается ДСЧ, который генерирует последовательность $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_k$. Число γ_k запоминаем.

Шаг 3: Повторно запускается в цикле ДСЧ, начиная с γ_0, при этом на каждом шаге проверяется условие $\gamma_i = \gamma_k$. При этом возможны два случая:

- если равенство ни разу не выполнилось, то $L > k$;
- если равенство выполнилось сначала для γ_j, а затем для γ_l, причем $j < l < k$, то $P = k - l$.

Шаг 4: Для нахождения L тестируемый ДСЧ дублируется. Теперь два датчика запускаются параллельно, но первый датчик запускается с числа γ_0, а второй с числа γ_P. Теперь находим минимальное число m такое, что выполняется соотношение $\gamma_m = \gamma_{m+P}$. Тогда $L = m + P$.

Проверка равномерности плотности распределений

Пусть $\gamma_1, \gamma_2, \ldots$ – последовательность выборочных значений СЧ, равномерно распределенных в интервале $[0,1]$, и векторы $\eta_1^{(k)}, \eta_2^{(k)}, \ldots, \eta_n^{(k)}, \ldots$ получены из нее так:

$$\eta_1^{(k)} = (\gamma_1, \ldots, \gamma_k),$$
$$\eta_2^{(k)} = (\gamma_{k+1}, \ldots, \gamma_{2k}),$$
$$\eta_n^{(k)} = (\gamma_{k(n-1)+1}, \ldots, \gamma_{nk}).$$

С вероятностью 1 такие векторы равномерно заполняют единичный $k-$ мерный куб. А это значит, что частота попадания вектора в любую прямоугольную область куба стремится к объему этой области при $n \to \infty$. Проверку будем проводить с помощью статистического критерия χ^2. Для этого разобьем $k-$ мерный куб на $s = r^k$ одинаковых кубиков с объемом r^{-k}. Теперь $m_1, \ldots, m_s -$ количество точек из последовательности векторов, попавших в соответствующие части (кубики): $m_1 + \ldots + m_s = N$. Известно, что для достаточно большого числа N величина

$$X^2_{s-1} = \frac{1}{N \cdot s^{-1}} \sum_{i=1}^{s} (m_i - N \cdot s^{-1})^2$$

(при выполнении гипотезы о равномерности и независимости векторов $\eta_1^{(k)}, \ldots, \eta_N^{(k)}$) приближенно распределена как χ^2_{s-1} (хи-квадрат с $s-1$ степенями свободы). Отметим, что при $k > 3$ такую проверку осуществляют очень редко из за большой трудоемкости.

Проверка коррелированности случайных чисел

В идеальном датчике генерируемые случайные числа независимы. Некоррелированность генерируемых случайных чисел проверяют с помощью автокорреляционной функции. Известно, что для случайных чисел, равномерно распределенных в интервале [0,1], $M[x] = \frac{1}{2}$, $D[x] = \frac{1}{12}$. Пусть мы сгенерировали достаточно длинную последовательность СЧ $x_1, x_2, \ldots, x_N$. Теперь вычислим

$$r_k = \frac{12}{N-k} \sum_{i=1}^{N-k} (x_i - 0.5)(x_{i+k} - 0.5) \quad (k = 1, 2, 3, \ldots).$$

Потом для каждого из этих r_k проверяется гипотеза о равенстве коэффициентов корреляции нулю. Стандартным является вычисление величин

$$t_k = \frac{r_k}{\sqrt{1 - r_k^2}} \sqrt{N - k - 2}$$

и использование критерия Стьюдента. Т. к. обычно $N-k>>1$, то граничные значения для t_k можно брать как для нормальной случайной величины, т. е. 1.96, 2.58, 3.29.

При необходимости более тщательного тестирования ДСЧ можно применить тест $k-$ нормальности, тест серий и т. п.

ЛИТЕРАТУРА

1. Советов Б. Я., Яковлев С. А. Моделирование систем: Учебник для вузов. – М.: Высш. шк., 2001. – 343 с.
2. Вероятностные методы в вычислительной технике: Учеб. пособие для вузов по спец. ЭВМ/ А. В. Крайников, В.А. Курдиков и др. – М., 1986. – 312 с.
3. Марков А.А. Моделирование информационно вычислительных процессов: Учеб. пособие для вузов. – М.: Изд–во МГТУ им. М.Э. Баумана, 1999. – 360 с.
4. Ермаков С. М., Михайлов Г. И. Статистическое моделирование. – М.: Наука, 1982. – 296 с.
5. Советов Б.Я., Яковлев С.А. Моделирование систем: Лабораторный практикум: Учеб. пособие для вузов по спец. «Автом. сист. обраб. инф. и управл.» – М.: Высш. шк., 1989. – 80 с.
6. Тузовский А. Ф. Программное моделирование систем: Учебное пособие.- Томск: Изд-во ТПУ, 1988. – 96 с.
7. Основы имитационного и статистического моделирования: Учеб. пособие / Ю.С. Харин и др. – М.: Дизайн ПРО, 1997. – 288 с.
8. Голованов О. В., Дуванов С. Г., Смирнов В. Н. Моделирование сложных дискретных систем на ЭВМ третьего поколения (опыт применения GPSS).- М.: Энергия, 1978. – 160 с.
9. Гультяев А.К. MATLAB 5.2. Имитационное моделирование в среде Windows: Практическое пособие. – СПб.: КОРОНА принт, 1999. – 288 с.
10. Прицкер А. Введение в имитационное моделирование и язык СЛАМ II / Пер. с англ. – М.: Мир, 1987. – 646 с.
11. Бенькович Е.С., Колесов Ю.Б., Сениченко Ю.Б. Практическое моделирование динамических систем – СПб.: БХВ-Петербург, 2002. – 464 с.

ПРИЛОЖЕНИЯ

Приложение 1

Моделирование систем на базе *Q* – схем

Непрерывно-стохастический подход, рассмотренный на примере математических схем *систем массового обслуживания*, может быть реализован в виде *Q* – схем.

В качестве процесса обслуживания могут быть представлены различные по своей физической природе процессы функционирования экономических, производственных, технических и других систем. Примерами таких процессов являются потоки самолетов, прибывающих в аэропорт, поток клиентов в банке, поток запросов к *SQL*-серверу от удаленных пользователей и т. п. Общим для работы таких систем является случайное появление запросов (клиентов) на обслуживание и завершение обслуживания в случайные моменты времени.

В любом элементарном акте обслуживания можно выделить две основные составляющие: *ожидание обслуживания* заявки и собственно *обслуживание* заявки. Это можно изобразить в виде некоторого i-го прибора обслуживания (рис. П1.1). Прибор обслуживания $П_i$ состоит из накопителя заявок H_i, в котором может одновременно находиться $l_i = \overline{0, L_i^H}$ заявок, где L_i^H – емкость i-го накопителя и канала обслуживания заявок (или просто канала) K_i. На каждый элемент прибора обслуживания $П_i$ поступают потоки событий: в накопитель H_i – поток заявок w_i, на канал K_i – поток обслуживаний u_i. Заявки, обслуженные каналом K_i, и заявки, покинувшие прибор $П_i$ по различным причинам необслуженными (например, из-за переполнения накопителя), образуют выходной поток y_i.

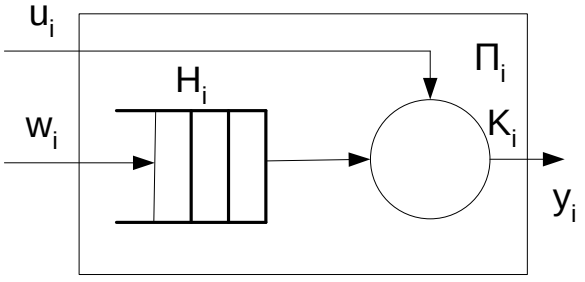

Рис. П1.1. Прибор обслуживания заявок

При моделировании реальных систем, имеющих более сложные структурные связи и алгоритмы поведения, для формализации используются не отдельные приборы обслуживания, а *Q-схемы*, образуемые композицией многих элементарных приборов обслуживания. Если каналы K_i различных приборов обслуживания соединены параллельно, то имеет место многока-

нальное обслуживание (многоканальная *Q-схема*). Если приборы $П_i$ и их параллельные композиции соединены последовательно, то имеет место многофазное обслуживание (многофазная *Q-схема*). Таким образом, для задания *Q-схемы* необходимо использовать оператор сопряжения R, отражающий взаимосвязь элементов структуры (каналов и накопителей) между собой. Связи между элементами *Q-схемы* изображаются в виде стрелок.

Для задания *Q-схемы* необходимо также описать алгоритмы ее функционирования, которые определяют набор правил поведения заявок в системе в различных ситуациях. Различают алгоритмы (дисциплины) ожидания в накопителе H_i и обслуживания заявок в канале K_i каждого элементарного обслуживающего прибора $П_i$.

В качестве элементов структуры *Q-схем* ограничимся рассмотрением элементов трех типов: *И* – источники; *Н* – накопители; *К* – каналы обслуживания заявок. Кроме связей, отражающих движение заявок в *Q-схеме* (сплошные линии), возможны различные управляющие связи. Примером таких связей являются различные блокировки обслуживающих каналов (по входу и по выходу): «клапаны» изображены в виде треугольников, а управляющие связи – пунктирными линиями. Блокировка канала по входу означает, что этот канал отключается от входящего потока заявок. Блокировка канала по выходу указывает, что заявка, уже обслуженная блокированным каналом, остается в этом канале до момента снятия блокировки (открытия «клапана»). Рассмотрим *Q-схему* общего вида (рис. П1.2).

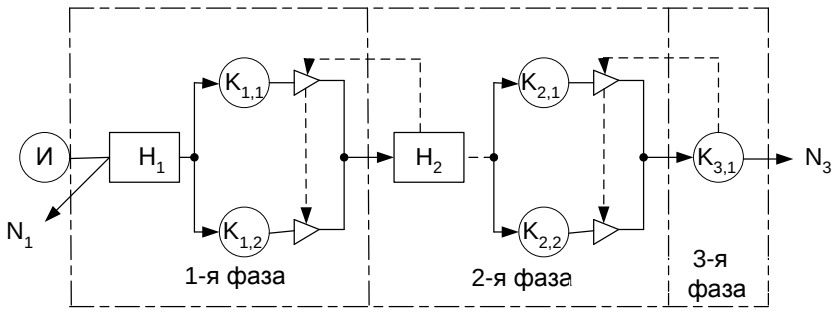

Рис. П1.2. *Q*-схема общего вида

На рис. П1.2 представлена трехфазная *Q-схема* с блокировкой каналов по выходу в первой и второй фазах обслуживания. В качестве выходящих потоков можно рассмотреть (оценить) поток потерянных заявок в накопителе H_1 и поток заявок, обслуженных в $K_{3,1}$.

Приложение 2

Система имитационного моделирования GPSS

GPSS (General Purpose Simulating System) – процессно - ориентированный язык имитационного моделирования, предназначен для моделирования сложных дискретных систем с учетом логических правил взаимодействия и функционирования ее элементов во времени. Модели на GPSS компактны, часто состоят из меньшего числа операторов, чем такие же модели, написанные на языках высокого уровня (C, C++, Pascal и т. п.).

GPSS очень удобен при программировании, поскольку интерпретатор языка многие функции выполняет автоматически. Например, GPSS без специального на то указания разработчика модели собирает статистические данные, описывающие поведение модели, и автоматически выдает итоговую статистику по завершении моделирования. Пользователю нет необходимости включать в программу модели операторы для сбора и накопления этих данных или задавать формат, указывающий, в каком виде должны быть распечатаны итоговые данные. В язык включены и многие другие полезные элементы. Например, GPSS обслуживает таймер модельного времени, планирует события, которые должны произойти позднее в течение времени моделирования, вызывает их своевременное появление и управляет очередностью поступления.

Основные концепции моделирования на GPSS

GPSS представляет собой язык и машинную (компьютерную) программу. Как любой язык, он содержит словарь и грамматику, с помощью которых легко могут быть разработаны корректные модели систем определенного типа. Машинная программа интерпретирует модель, написанную на языке GPSS, предоставляя тем самым разработчику возможность проведения экспериментов с этой моделью на ЭВМ. Эта машинная программа и называется интерпретатором.

В GPSS имеется основное подмножество блоков (объектов), выбранных таким образом, чтобы можно было создавать законченные, относительно простые модели систем. Предполагается, что для определенного класса моделируемых систем можно выделить ограниченный набор абстрактных элементов – ***объектов***, причем набор логических правил также ограничен и может быть описан небольшим числом стандартных подпрограмм (блоков) на языке GPSS.

Объекты GPSS описывают взаимодействия элементов сравнительно несложным набором операций, и достаточно просто и наглядно представить процесс функционирования исследуемой системы *S*, формализуемой в виде *Q*-схемы, с помощью этих объектов языка.

Объекты GPSS подразделяются на семь категорий и четырнадцать типов (см. табл. П2.1).

Таблица П2.1

Категории объектов	Типы объектов
Динамическая	Транзакты
Операционная	Блоки
Аппаратная	Устройства. Памяти. Ключи
Вычислительная	Переменные (арифметические, булевы). Функции
Статистическая	Очереди. Таблицы
Запоминающая	Матрицы ячеек. Ячейки
Группирующая	Списки пользователя. Группы

Кроме того, для облегчения пользователю процесса построения модели системы в GPSS разработан язык блок – диаграмм, позволяющий упростить переход от алгоритма к программе модели системы *S*. Таким образом, каждый блок GPSS имеет свой графический аналог. С помощью блок – диаграмм отображается пространственная конструкция модели, упрощающая дальнейшую линеаризацию программы.

Построение блок – диаграмм знакомит пользователя с набором операторов языка, который однозначно соответствует графическому набору блоков. Вследствие этого, очевидно, что построение блок-диаграммы не является самоцелью, а лишь промежуточный этап, помогающий разработать имитационную модель исследуемой системы с использованием операторов языка GPSS. Этапы разработки модели можно представить в виде схемы (рис. П2.1).

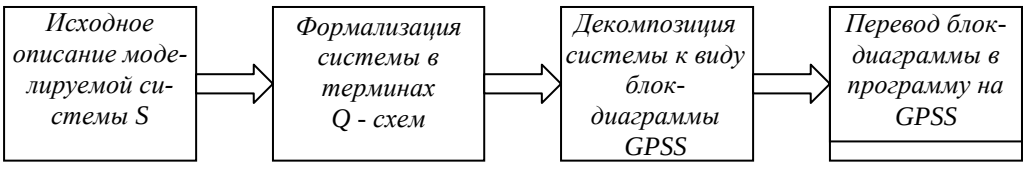

Рис. П2.1. Этапы создания имитационной модели

Конечным этапом создания модели системы является программа на языке GPSS, описывающая функционирование выделенного конечного набо-

ра объектов, и специальная диспетчер – программа – симулятор, обладающая следующими основными свойствами:

- обеспечением заданных программистом маршрутов продвижения динамических объектов, называемых далее транзактами (сообщениями);
- планированием событий, происходящих в модели, путем регистрации времени наступления каждого события и выполнения их в нарастающей временной последовательности;
- регистрацией статистической информации о функционировании модели;
- продвижением модельного времени в процессе моделирования системы.

Объекты *динамической категории* – транзакты – представляют собой единицы исследуемых потоков. Они производят последовательность определенных действий, продвигаясь по фиксированной блок – диаграмме, представляющей собой совокупность объектов других категорий.

В самом начале в модели нет ни одного транзакта. В процессе моделирования транзакты входят в модель в определенные моменты времени, продвигаются по модели и покидают ее в соответствии с логикой функционирования исследуемой системы. В общем случае в модели существует большое число транзактов, но в каждый момент времени движется только один. Если транзакт начал свое движение, он перемещается от блока к блоку по пути, предписанному блок-схемой. Такое продвижение транзакта продолжается до тех пор, пока не произойдет одно из следующих возможных событий:

- транзакт входит в блок, функцией которого является задержка транзакта на некоторое определенное время;
- транзакт входит в блок, функцией которого является удаление транзакта из модели;
- транзакт «пытается» войти в следующий блок в соответствии с блок-схемой, но блок «отказывается» принять его.

Если возникло одно из описанных условий, то транзакт остается на месте и в модели начинается перемещение другого транзакта. Таким образом, выполнение моделирования в системе продолжается. Модель на GPSS состоит из одного или нескольких независимых сегментов. В процессе моделирования активным является тот из сегментов, в котором находится перемещающийся в настоящий момент транзакт. Когда он блокируется, начинает двигаться следующий транзакт и может оказаться, что этот следующий транзакт принадлежит другому сегменту модели. Таким образом, происходит переключение активности между различными сегментами модели.

Объекты *операционной категории* – блоки – задают логику функционирования модели системы и определяют пути движения транзактов между объектами аппаратной категории.

Объекты *вычислительной категории* служат для описания таких ситуаций в процессе моделирования, когда связи между компонентами системы наиболее просто и компактно выражаются в виде математических (аналитических и логических) соотношений. С этой целью в качестве объектов вычислительной категории введены арифметические и булевы переменные и функции.

Объектами *статистической категории* являются очереди и таблицы. Они вводятся для оценки характеристик поведения системы *S*.

Очереди вводятся в моделирующую программу для регистрации статистической информации о процессе ожидания. Транзакт помещается в очередь в том случае, если некоторое устройство не в состоянии обслужить его немедленно, например, устройство занято либо память переполнена. Статистические данные об очередях могут быть получены в разных точках модели.

В процессе моделирования системы различные объекты взаимодействуют друг с другом. В результате этого происходят изменения и преобразование арифметических или логических значений этих атрибутов. Такие преобразования в системе называются *событиями*.

Транзакты моделируют прохождение по системе соответствующих единиц исследуемого потока. Такое движение может быть разбито на цепь элементарных событий, происходящих в определенные моменты времени. Основной задачей модели на языке GPSS является определение моментов наступления этих событий, расположение их в правильной временной последовательности (в соответствии с логикой функционирования модели) и выполнение соответствующих действий при наступлении каждого события. В теории моделирования этот механизм называется «календарем событий». В GPSS аналогом этого механизма являются «системные часы».

***Внимание*:** *В GPSS все временные отрезки описываются целыми числами, поэтому перед разработкой модели необходимо провести временное масштабирование всех* (*временных*) *параметров и характеристик системы.*

Каждому объекту в модели соответствуют *атрибуты*, описывающие его состояние в данный момент времени. Большая часть атрибутов недоступна для программистов. Атрибуты, к которым может обращаться программист, называются *стандартными числовыми атрибутами* (СЧА). Все стандартные числовые атрибуты имеют одно- или двухбуквенное мнемоническое обозначение. Мнемонические обозначения указывают на тип атрибута, а целочисленное значение – на конкретный атрибут. Список основных стандартных числовых атрибутов приведен в табл. П2.2.

Таблица П2.2

Объект	СЧА	Назначение
Блоки	N W	Счетчик входов. Счетчик текущего содержимого
Генераторы случайных чисел	RN	При использовании в качестве аргумента функции представляются действительным числом в диапазоне 0.000000 – 0.999999. В остальных случаях – целым числом в диапазоне 000 – 999
Многоканальные устройства	R S SA SC SR SM ST	Остающаяся емкость. Текущее содержимое. Среднее содержимое. Счетчик числа входов. Коэффициент использования. Максимальное содержимое. Среднее время задержки на единицу емкости
Очереди	Q QA QC QM QZ QT QX	Текущее содержимое. Среднее содержимое. Счетчик общего числа входов. Максимальное содержимое. Счетчик числа нулевых входов. Среднее время пребывания (на основе QC). Среднее время пребывания (на основе QZ)
Приборы	F FC FR FT	Состояние прибора (1 – занят, 0 – свободен). Счетчик числа занятий. Коэффициент использования. Среднее время задержки на одно занятие
Таблицы	TB TC TD	Среднее значение взвешенных входов. Количество не взвешенных входов. Стандартное отклонение не взвешенных входов
Таймер	C1	Значение системного времени
Транзакты	P PR M1 MP	Значение параметра. Значение приоритета. Время пребывания в модели. Время с момента входа в блок

Синтаксис элементов языка GPSS

Алфавит. Алфавит языка состоит из латинских букв от A до Z, цифр – от 0 до 9 и специальных символов. Буквы русского алфавита (кириллица) используются в программах только в комментариях.

Числа. В языке GPSS различают два типа чисел: целые и действительные. Числа в памяти могут занимать два байта (полслова) и четыре байта (слово).

Идентификаторы. Идентификаторы используются для задания имен объектов и блоков. Они должны содержать не менее трех и не более пяти алфавитно-цифровых символов, причем первые три символа должны быть буквами.

Правила записи программы. Программа на языке GPSS оформляется в виде текстового файла, имеющего строгую структуру:

- поле имени блока или объекта (метка) занимает со 2 по 6 позиции (колонки);
- название блока или карты описания занимают с 8 по 18 позиции;
- операнды (параметры блока) занимают с 19 по 71 позиции; эти позиции разбиты на подполя, которые обозначены A, B, C, ..., H; операнды друг от друга отделяются запятыми; если какой-то (не последний) операнд отсутствует, то соответствующая ему запятая остается;
- комментарии располагаются после описания операндов через пробел; если в первой позиции строки стоит символ «*», то это строка комментариев.

Управляющие карты и блоки, используемые в GPSS-моделях

Программа на GPSS состоит из набора карт и блоков, записанных соответствующим образом (см. выше). Рассмотрим подробнее несколько карт, управляющих процессом моделирования.

Карта SIMULATE является первой картой программы и указывает на необходимость проведения моделирования. При отсутствии этой карты моделирование проводиться не будет.

Блок GENERATE (ГЕНЕРИРОВАТЬ)

GENERATE – это блок, который создает транзакты в модели. В одной модели может быть несколько таких блоков. Необходимую для данного блока информацию задает разработчик с помощью операндов. Сведения об этих операндах сведены в табл. П2.3.

Таблица П2.3

GENERATE A, B, C, D, E

Операнд	Описание операнда	Значение по умолчанию
A	Среднее время между последовательными прибытиями транзактов	0
B	Половина размаха интервалов прибытия	0
C	Момент времени, в который в модели должен появиться первый транзакт	1
D	Ограничитель (максимальное число транзактов, которые могут войти в систему)	∝
E	Уровень приоритета, который задается числами от 0 до 127	0

Операнды A и B в блоке являются основными (их наличие обязательно), операнды C, D и E – дополнительными. Все эти операнды не обязательно

должны быть заданы явно. Но когда операнды заданы в виде констант, то они должны быть неотрицательными целыми числами.

Примеры использования блока GENERATE

GENERATE 5,3	Операнды A и B означают, что интервал времени прибытия равен 5 плюс-минус 3, т. е. интервал может быть выражен числами 2,3,4,5,6,7 и 8.
GENERATE 10	Здесь интервал времени прихода равен 10, а операнд B равен 0, т. е. интервал времени – величина не случайная.
GENERATE 3,3,10,5	Здесь интервал времени прихода изменяется от 0 до 6, причем первый транзакт появляется с момента времени 10. Всего 5 транзактов войдут в систему через этот блок.
GENERATE 8,fn$expon	Здесь интервал времени прихода изменяется по показательному закону со средним временем прихода, равным 8.

Блок TERMINATE (*ЗАВЕРШИТЬ*)

TERMINATE – блок уничтожения транзактов. В модели может быть любое число этих блоков. Информация для этого блока задается с помощью одного операнда A, значение которого должно вычитаться из специального счетчика завершений каждый раз, когда транзакт входит в блок TERMINATE. Если операнд A не задан, то по умолчанию подразумевается значение 0. В этом случае вход транзакта в такой блок не вызывает уменьшения содержания счетчика. При достижении содержания счетчика нуля моделирование завершается. Значение счетчика задается в начале моделирования при помощи оператора START следующим образом: START A , где значение операнда A соответствует начальному значению счетчика. Окончание моделирования производится при обнулении счетчика (поле A).

В модели существует некоторая точка, перед которой помещены все команды управления прогонами модели. Это и есть оператор END.Он побуждает интерпретатор вернуть управление в операционную систему. Оператор END не имеет операндов и стоит после оператора START.

Пример

```
SIMULATE
GENERATE  18,6     ! Интервал прибытия транзактов 12 - 24 мин.
TERMINATE          ! Удаление транзакта из модели
GENERATE 480       ! Моделирование в течение 8 часов
TERMINATE 1        ! Уменьшение счетчика завершений на 1
START 1
END
```

Блок ADVANCE (*ЗАДЕРЖАТЬ*)

ADVANCE – блок для реализации задержки продвижения транзакта в течение некоторого интервала времени. Обычно транзакт занимает прибор

для того, чтобы немедленно начать в нем обслуживаться. Время обслуживания задается, как правило, случайной переменной с помощью операндов.

ADVANCE A, B С помощью операнда А задается среднее время обслуживания транзакта. Операнд В, как и в блоке GENERATE, равен половине размаха интервала обслуживания.

Пример использования блока ADVANCE

ADVANCE 30,5	Операнды А и В означают, что для каждого транзакта, входящего в этот блок, возможные значения интервала времени обслуживания лежат в пределах 25 – 35.
ADVANCE 18,fn$expon	Здесь интервал времени изменяется по показательному закону со средним временем обслуживания, равным 18.

Блоки SEIZE (*ЗАНЯТЬ*) и RELEASE (*ОСВОБОДИТЬ*)

SEIZE – блок занятия прибора. Для того чтобы транзакт в момент своей активности мог занять прибор, соответствующий блок должен обладать следующими свойствами:

- Если прибор занят другим транзактом, активный транзакт не может войти в блок, и должен ждать в очереди.
- Если прибор свободен, транзакт входит в блок, статус которого изменяется со «свободен» на «занят».

SEIZE A, B Для этого блока обязательно должен быть указан операнд А, значением которого является имя занимаемого прибора. Имя прибора может быть:

- символическим – состоять из трех-пяти алфавитно-цифровых символов, причем первые три символа должны быть буквами;
- числовым – являться положительным целым числом.

В качестве операнда В может указываться символ 'Q', который указывает на необходимость сбора статистики. Если в качестве операнда В указан символ 'L', то блок становится *логическим блоком* SEIZE. Это значит, что если прибор А занят, то транзакт не ждет освобождения прибора, а проходит через этот блок.

Примеры использования блока SEIZE

SEIZE sar43	Транзакт занимает прибор с символическим именем «sar43».
SEIZE 3	Транзакт занимает прибор с числовым именем «3».
SEIZE sal,Q	Транзакт занимает прибор с символическим именем «sal», очередь не будет создаваться, только собирается и выводится статистика.

Вход транзакта в прибор моделируется блоком SEIZE, что соответствует занятию прибора. Вход того же транзакта в другой прибор соответствует изменению статуса первого прибора с «занят» на «свободен».

RELEASE – блок освобождения прибора. Для этого блока также должен быть указан операнд A, который обозначает имя (символическое или числовое) освобождаемого прибора.

Замечания:

1. При моделировании нет необходимости определять (описывать) имя прибора перед тем, как использовать эти два блока.

2. При попытке освободить свободный прибор печатается сообщение об ошибке и прекращается выполнение моделирования. Система аналогично реагирует, если транзакт пытается освободить прибор, занятый другим транзактом.

3. Когда нужно получить информацию о средней длине очереди или о среднем времени ожидания, то используют операнд B. В этом случае в качестве операнда B используют символ "Q".

Пример

```
SIMULATE
GENERATE  18,6
SEIZE SAL          ! Занятие прибора SAL.
ADVANCE 25,5       ! Время обслуживания транзакта 20 - 30 мин.
RELEASE SAL        ! Освободить прибор SAL.
TERMINATE
GENERATE 480       ! Моделирование в течение 8 часов.
TERMINATE 1        ! Уменьшение счетчика завершений на 1.
START 1
END
```

Блоки ARRIVE (*СТАТЬ В ОЧЕРЕДЬ*) и DEPART (*ПОКИНУТЬ ОЧЕРЕДЬ*)

ARRIVE и DEPART – блоки регистрации очереди. Эти блоки обеспечивают возможность автоматического сбора статистических данных при вынужденном ожидании в различных точках модели. Собранная информация должна максимально полно отразить следующие характеристики системы (модели):

- Сколько транзактов вошло в очередь?
- Сколько транзактов сразу заняли прибор (число нулевых вхождений) и сколько транзактов присоединились к очереди?
- Каково максимальное значение длины очереди?
- Каково среднее число ожидающих транзактов?
- Каково среднее время ожидания для транзактов, которым пришлось ждать?

В программе может быть несколько регистраторов очереди, различающихся именами. Правила задания имен регистраторов такие же, как и для приборов. Операнд A в блоках ARRIVE и DEPART используют для указания имени соответствующей очереди. Крайне редко возникает необходимость использовать в этих блоках второй операнд – B, который указывает, на какую величину должен быть изменен счетчик содержимого очереди. По умолча-

нию для этого операнда подразумевается значение, равное 1. В конце моделирования автоматически распечатываются следующие элементы статистики: «счетчик входов», «максимальное содержимое», «среднее значение содержимого», «счетчик текущего содержимого», «среднее время пребывания в очереди», «счетчик нулевых вхождений».

Замечание: Регистратор очереди не обязательно использовать в модели везде, где могут возникать очереди. Если он не используется, то просто не собирается статистика.

Пример

```
SIMULATE
GENERATE  18,6
ARRIVE TIM          ! Транзакт становится в очередь с именем TIM.
SEIZE SAL
ADVANCE 25,5
RELEASE SAL
DEPART TIM! Транзакт покидает очередь.
TERMINATE
GENERATE 480
TERMINATE 1
START 1
END
```

Блок GOTO (*ПЕРЕЙТИ*)

GOTO – блок перехода транзакта в блок, отличный от последующего. Блок можно использовать в одном из двух режимов:

- в режиме безусловной передачи (перехода);
- в режиме статистической передачи.

Безусловный режим блока GOTO подобен оператору goto во многих языках программирования высокого уровня. Но статический режим блока GOTO имеет некоторые особенности. Рассмотрим эти режимы подробнее.

Если в режиме безусловной передачи необходимо передать транзакт в блок, отличный от последующего, то блок имеет вид GOTO A. Здесь операнд A указывает адрес (т. е. положение, занимаемое блоком), в который транзакт должен сделать попытку входа. В качестве адреса обычно используют символическое имя.

Пример

```
BACK ADVANCE 30,5
     SEIZE JVEN
     GOTO BACK          ! Транзакт пытается войти в блок с меткой BACK.
```

Режим статистической передачи используется, когда необходимо передать транзакт в один из блоков случайным образом. Блок в режиме статистической передачи имеет следующий формат: GOTO A, B.

Здесь операнд B – это частота, с которой транзакт должен попадать в блок с адресом, указанным в операнде A. Операнд B – это число от 0 до 1, которое может иметь максимум 4 цифры после запятой. Таким образом, минимальное значение операнда B равно 0.0001, а максимальное значение – 0.9999.

Пример

GOTO PLAY,.25	Транзакты, входящие в этот блок, в 25 % случаев будут передаваться в блок с именем PLAY. В остальных 75 % случаев они будут переданы в следующий блок.

Блоки PREEMPT (*ЗАХВАТИТЬ*) *и* RETURN (*ВЕРНУТЬ*)

PREEMPT – блок захвата прибора у еще не обслуженного до конца транзакта вновь пришедшим транзактом, который немедленно занимает прибор. В это время замещенный (прерванный) транзакт может либо ждать, пока прибор не освободится вновь, либо может быть потерян.

Структура и операнды блока имеют следующий вид:

PREEMPT A, B, C, D, E

A	Имя захватываемого прибора
B	Необязательный операнд. Если операнд не используется, то захват происходит, когда обслуживаемый транзакт сам не является захватчиком. Если операнд используется, в нем должна стоять двухбуквенная последовательность PR. Захват разрешен, когда возможный захватчик имеет более высокий уровень приоритета
C	Имя блока, в который будет послан прерванный транзакт
D	Номер параметра прерванного транзакта, в который помещается значение времени, оставшегося транзакту до окончания обслуживания в приборе
E	Если задан параметр RE, то прерванный транзакт теряет право на автоматическое восстановление обработки в приборе

Транзакт, захвативший прибор, может освободить его (т. е. возвратить прибор ранее прерванному транзакту) только при дальнейшем вхождении в блок RETURN. Операнд A этого блока указывает имя прибора, подлежащего освобождению.

Пример

```
SIMULATE
GENERATE  120,30,,,1
ARRIVE NEWSQ          ! Встать в очередь.
PREEMPT SAL           ! Захватить прибор.
DEPART NEWSQ
ADVANCE 45,30
RETURN SAL            ! Возвратить прибор.
TERMINATE
GENERATE 360
TERMINATE 1
START 1
END
```

Блоки SPLIT (*СОЗДАТЬ КОПИЮ*) и ASSEMBLE (*УНИЧТОЖИТЬ КОПИЮ*)

В процессе моделирования систем может возникнуть необходимость создать (уничтожить) копии имеющихся транзактов.

SPLIT – блок создания копий транзактов, уже участвующих в моделировании. Созданные транзакты имеют точно такие же характеристики (параметры, приоритет...), как и оригинал.

SPLIT A, B, C Значение операнд A – целое положительное число, равное количеству копий транзакта. Операнд B – адрес блока, в который следует послать копию транзакта. Операнд C – значение так называемого серийного параметра. Когда транзакт входит в блок SPLIT, значение этого параметра увеличивается на 1 для транзакта-оригинала, на 2 – для его первой копии, на 3 – для второй копии и т. д.

ASSEMBLE – блок уничтожения всех копий, которые были сделаны с одного транзакта-оригинала.

ASSEMBLE A Операнд A равен числу уничтожаемых копий плюс 1. Таким образом, если A = 2, то уничтожится только одна копия.

Пример

```
        SIMULATE
        GENERATE  ,,,100
        SEIZE MACH1
        ARRIVE TOTTI
        ADVANCE 32,6
        RELEASE MACH1
        SPLIT 1, MAC3          ! Сделать одну копию транзакта и послать ее в блок
        SEIZE MACH2            ! с адресом MAC3.
        ADVANCE 28,5
        RELEASE MACH2
        GOTO MAC4
MAC3    SEIZE MACH3
        ADVANCE 30,15
        RELEASE MACH3
MAC4    ASSEMBLE 2             ! Уничтожить сделанную копию.
        SEIZE MACH4
        ADVANCE 22,9
        RELEASE MACH4
        DEPART TOTTI
        TERMINATE 1
        START 100
        END
```

Блоки моделирования многоканальных устройств

Карта STORAGE служит для описания накопителя (многоканального устройства). Поле метки используется для задания имени накопителя (символического или числового), а в поле A указывается его максимальная емкость.

```
MAIN STORAGE 5          Накопитель с именем «MAIN» имеет емкость – 5.
```

Блоки ENTER (*ВОЙТИ*) *и* LEAVE (*ВЫЙТИ*)

Элементом, который занимает и использует накопитель, является транзакт. При этом происходят следующие события:

- транзакт ожидает своей очереди (если необходимо);
- транзакт занимает устройство;
- устройство осуществляет обслуживание;
- транзакт освобождает устройство.

Как и в случае с прибором, используются два блока:

ENTER – блок, соответствующий состоянию «занято»;

LEAVE – блок, соответствующий состоянию «свободно».

В этих блоках используются три операнда. Операнд A – используется для указания имени многоканального устройства. Операнд B – задает число мест, которое должно быть занято (или освобождено). Это значит, что транзакт может войти в накопитель, если его оставшаяся емкость больше или равна B. Этот операнд редко используется, и его значение по умолчанию равно 1. Операнд C используется тогда, когда нужно собрать статистику об очереди. В этом случае пишется символ «Q». В общем случае использование блоков ENTER и LEAVE аналогично использованию блоков ARRIVE и DEPART.

Пример

```
        SIMULATE
stor    STORAGE  2              ! Емкость накопителя stor = 2.
        GENERATE 18,6
        ENTER stor             ! Занятие многоканального накопительного
        ADVANCE 25,5           ! устройства (МНУ) stor.
        LEAVE stor             ! Освобождение МНУ.
        TERMINATE
        GENERATE 480
        TERMINATE 1
        START 1
        END
```

Блок IF (*ЕСЛИ*)

Ранее был рассмотрен блок GOTO, который в своем простейшем режиме осуществляет безусловный переход к блоку, отличному от последующего. Иногда возникает необходимость совершить переход с учетом какого-либо условия, т. е. условный переход. Для этого используют блок IF. Этот блок имеет два режима:

- SNA – режим (или режим отношения), при котором операнды A и B связаны отношением.
- SERVER – режим (или режим присвоения), при котором имеется дополнительный операнд D.

Рассмотрим эти режимы подробнее.

Режим отношения

Формат блока имеет следующий вид:

IF A * B,C

A, B	Операнды A и B являются константами или СЧА
*	Знак проверяемого отношения между операндами A и B. Допустимые отношения: > , >= , = , <> , < , <=
C	Операнд C – это адрес блока, в который транзакт должен перейти, если проверка отношения дает результат true (истина), в противном случае транзакт переходит в следующий блок

Пример

```
IF Q$lin=4,bye      ! Если в очереди lin есть 4 транзакта, то перейти к блоку
                    ! с адресом bye.
```

Отношение может быть выражено также одним из следующих кодов:

U	(The facility is) in Use – прибор занят
NU	(The facility is) Not in Use – прибор не занят
E	(The storage is) Empty – МНУ пустое
NE	(The storage is) Not Empty – МНУ не пустое
F	(The storage is) Full – МНУ заполнено
NF	(The storage is) Not Full – МНУ не заполнено

В этом случае операнд A – это имя прибора или МНУ, операнд B содержит один из вышеперечисленных кодов и формат блока будет следующим:

IF A = код,C

Пример

```
IF sal=U,bye        !Если прибор sal занят, то перейти к блоку bye.
```

Режим присвоения

Формат блока –

IF A*B,C=D

A, B	Операнды A и B являются константами или СЧА
*	Знак отношения (код) между операторами A и B
C	Операнд C – это S-величина или параметр, которой будет присвоено значение операнда D в случае, если проверка отношения между операндами A и B даст результат true (истина)
D	Операнд D – константа, СЧА или выражение

Пример 1

```
IF X$norm<0,X$norm=0    !Если величина X$norm < 0,то ее нужно установить
                        !в «0», в противном случае она сохранит свое
                        !прежнее значение.
```

Пример 2

```
        SIMULATE
        GENERATE  8,5
        IF  q$lin1>q$lin2,serv2      ! Если длина очереди lin1 больше длины очереди lin2,
        ARRIVE lin1                  ! то перейти к блоку с адресом serv2.
        SEIZE  sal1
        ADVANCE  18,10
        RELEASE  sal1
        DEPART lin1
        TERMINATE
serv2   ARRIVE lin2
        SEIZE  sal2
        ADVANCE  18,10
        RELEASE  sal2
        DEPART lin2
        TERMINATE
        GENERATE 480
        TERMINATE 1
        START 1
        END
```

Блок WAITIF (*ЖДАТЬ, ЕСЛИ* ...)

Блок является разновидностью блока IF. Операнды A, B и отношения между ними задаются так же, как и в блоке IF, операнд C отсутствует.

Пример 1

```
        WAITIF stor=E        ! Транзакт должен ждать до тех пор, пока МНУ stor занято.
                             ! Как только это условие перестает выполняться (т. е. МНУ
                             ! освободится), транзакт переходит в следующий блок.
```

Пример 2

```
        SIMULATE
 stor   STORAGE 10
        GENERATE 18,6
        WAITIF lock=u        ! Транзакт должен ждать до тех пор, пока прибор lock
        ENTER stor           ! занят.
        SEIZE sal,q
        ADVANCE 25,5
        RELEASE sal
        LEAVE stor
        TERMINATE
        GENERATE 480
        SEIZE  lock
        WAITIF stor=ne       ! Транзакт должен ждать до тех пор, пока МНУ stor не
        TERMINATE 1          ! пустое.
        START 1
        END
```

Блок и оператор LET

GPSS позволяет использовать одно и то же обозначение и для блока, и для оператора. Блок и оператор LET используются для создания С – величин (переменных) и работы с ними.

Блок LET (*СОЗДАТЬ ПЕРЕМЕННУЮ*)

Блок используется при создании С-величин. При этом значение, данное каждой переменной, остается таким же до конца моделирования, если только его не изменит какой-нибудь транзакт, вошедший через этот или другой блок, относящийся к данной С – величине. Имя переменной записывается как X$n, где n – символическое имя. В общем случае формат блока имеет вид

LET A=B

Здесь операнд А – имя переменной, которой присваивается значение операнда В. Операнд В может быть константой, СЧА или выражением. Блок LET может работать в нескольких режимах:

- *режим назначения*.

Примеры:

```
1. LET X$cost=27          ! C - величина X$cost принимает значение 27.
2. LET X$fval=FN$val      ! C - величина X$fval получает текущее значение
                          ! функции val.
```

Блок LET можно использовать для распечатки значений СЧА не в конце моделирования, как обычно, а в любой момент времени. Например, используя такой сегмент, можно распечатать stor:

```
GENERATE 80,,,1
LET X$stor80=S$           ! Текущая емкость МНУ stor в момент времени 80.
TERMINATE
```

Блок LET может работать с параметрами транзактов. Значения параметров транзактов могут назначаться и изменяться при использовании одного из следующих трех режимов блока LET:

- *режим замещения*.

Старое значение параметра заменяется новым независимо от того, каким было это значение. Формат: LET A=B, где операнд А – номер модифицируемого параметра, который обозначается как P_j (j = 1,...,12). Операнд В – значение, присваиваемое параметру А.

Пример

```
LET P2=7
```

- *режим приращения*.

Новое значение параметра вычисляется путем сложения значения операнда В со старым значением параметра. Формат:

LET+ A,B или LET A=A+B,

где операнд A – номер модифицируемого параметра Pj. Операнд B – величина, на которую увеличивается текущее значение параметра Pj.

Пример

```
        LET+ P4,5           ! 4-ый параметр увеличивается на значение, равное пяти.
```

- *режим уменьшения - аналогичен режиму приращения и имеет формат*

LET- или LET A=A-B

Пример

```
        SIMULATE
        GENERATE 18,6
        LET+ x$cust,1           ! C-величина x$cust увеличивается на 1.
        LET P1=x$cust           ! Величина P1 принимает значение C-величины x$cust.
        ARRIVE lin
        SPLIT  1,reneg
        WAITIF sal=u
        IF x(P1)=1,bye
        SEIZE sal
        LET x(P1)=1
        DEPART lin
        ADVANCE  25,5
        RELEASE  sal
bye     TERMINATE
reneg   ADVANCE 40
        IF x(P1)=1,out
        LET x(P1)=1
        DEPART lin
out     TERMINATE
        GENERATE 480
        PRINT q
        TERMINATE 1
        START 1,np
        END
```

Блоки LET+ (*УВЕЛИЧИТЬ СЧЕТЧИК*) и LET- (*УМЕНЬШИТЬ СЧЕТЧИК*)

Блок LET (либо оператор LET) может использоваться в качестве счетчика. В GPSS предусмотрена простая конструкция, а именно:

LET+ A,B (LET- A,B для уменьшения значения счетчика). Здесь операнд A – это имя C-величины, операнд B – величина, на которую изменяется значение счетчика.

Пример

```
        LET+ X$cvb,1            ! Величины X$cvb увеличивается на 1.
```

Оператор LET

LET может использоваться не только как блок, но и как операнд. Следует вспомнить, что большинство СЧА, в том числе и С-величины, имеют в качестве начального значения «0». Иногда бывает необходимо, чтобы в начале моделирования С-величина (или другой СЧА) имела начальное значение, отличное от 0. Это можно сделать с помощью оператора LET, который имеет следующий формат:

LET A,B

Здесь операнд А – имя С-величины (или другого СЧА), операнд В в общем случае – положительная константа, которая присваивается С-величине в качестве начального значения.

Пример

```
LET X$cvx=25          ! Начальное значение С-величины X$cvx равно 25.
```

Следует заметить, что оператор LET должен располагаться обязательно перед первым блоком GENERATE. Отметим также, что операнд В не может быть выражением или СЧА, которые ранее нигде не были использованы. Это основное отличие между блоком LET и оператором LET. Оператор LET используется в основном тогда, когда нужно запустить программу несколько раз с разными значениями некоторых величин.

Построение таблиц

С помощью таблиц могут быть собраны и автоматически табулированы статистические данные о работе модели. Это легко сделать, используя блок TABULATE и оператор TABLE.

Оператор TABLE (*ТАБЛИЦА*)

Этот оператор используют при работе с таблицами. В модели может быть несколько таблиц. Каждая таблица сначала должна быть определена (см. ниже). Существует разновидность оператора TABLE – это оператор QTABLE, который отличается тем, что используется только для табулирования информации об очереди. Структура этих двух операторов имеет следующий вид:

N TABLE A, B, C, D

N	Имя таблицы
A	Имя переменной, значение которой табулируется
B	Левая граница значения переменной. Обычно это "0"
C	Ширина интервала (число интервалов времени в интервале)
D	Общее число интервалов таблицы, включая левый и правый

Замечания:

- Следует запомнить, что эти операторы пишутся после оператора SIMULATE, но перед первым блоком GENERATE.
- Табулируемые значения (в общем случае) могут иметь размах от $[-\infty, +\infty]$. При использовании таблиц этот размах должен быть разделен (на оси действительных чисел) на ряд последовательных интервалов. Интерпретатором подсчитывается частота, с которой табулируемое значение попадает в каждый из этих интервалов. При определении таблицы указывается, в какой части оси действительных чисел нужно расположить эти интервалы. Заметим, что левый интервал включает значение от $-\infty$ до значения первой (левой) границы включительно. Правый интервал включает все значения, большие, чем последняя (правая) граница.
- В программе может быть несколько таблиц, различающихся именами, причем для QTABLE имя не обязательно, а для TABLE – обязательно.
- Главное отличие между этими двумя операторами заключается в операнде A, который для TABLE – это имя СЧА, который будет табулироваться, например: SSAD, Qlin и т. п. Для оператора QTABLE операнд A – это имя очереди, для которой составляется таблица.

Пример

```
SIMULATE
QTABLE sal,0,10,20        ! Составляется таблица для очереди sal с шириной
GENERATE 18,6             ! интервалов, равной 10, и имеющая 20 таких интервалов.
SEIZE sal,q
ADVANCE 25,5
RELEASE sal
TERMINATE
GENERATE 480
TERMINATE 1
START 1
END
```

Блок TABULATE (*ТАБУЛИРОВАТЬ*)

Для того чтобы собиралась статистика, вместе с оператором TABLE нужно использовать специальный блок TABULATE (а для оператора QTABLE блоки ARRIVE и DEPART). Формат блока –

TABULATE A,

где операнд A – это имя таблицы, для которой собираются данные.

Оператор FUNCTION

Оператор FUNCTION определяет функцию, имя которой записано в поле адреса. Оператор имеет два операнда: операнд A – это СЧА, определенный как независимая переменная функции; операнд B – код, который состоит из двух частей:

• буква – это символ «C», если описывается непрерывная функция, или «D» – с дискретной функцией;

• цифра – число пар данных, используемых для определения функции.

Рассмотрим это на примере.

Пример

time FUNCTION RN2,D4
.15,2/.35,5/.75,8/1,12

Здесь time – имя функции. Следовательно, значение функции будет в СЧА FN$TIME. RN2 – независимая переменная, случайное число, полученное вторым генератором случайных чисел. D4 – количество пар (4) дискретных (D) значений. Как мы видим, данные, определяющие функцию, состоят из пар. Первое значение в паре – это независимая переменная, второе значение – значение функции. Оператор ставится перед первым блоком GENERATE, но после оператора SIMULATE.

Значения выходных параметров модели

По завершении моделирования интерпретатор GPSS автоматически распечатывает статистическую информацию о модели. Эта информация включает статистические данные по каждому элементу, используемому в модели, т. е. по каждому из приборов, очередей и т. д. Для того чтобы иметь представление о работе модели на основании результатов моделирования, ниже приводится описание всех выходных параметров и русский эквивалент их названий.

Описание значений выходных параметров

RELATIVE CLOCK 480	Относительное время	Данные параметры означают, что моделирование завершилось в момент модельного времени, равный 480
RELATIVE CLOCK 480	Абсолютное время	
FACILITY joe	Прибор	Здесь указывается перечень всех символических имен приборов (или присвоенные им числовые эквиваленты), используемых в модели
QUEUE joeq	Очередь	Полностью аналогично FACILITY

Выходные данные прибора joe

AVARAGE UTILIZATION .860	Средняя загрузка прибора	В течение 86 % модельного времени прибор был занят
NUMBER ENTRIES 26	Число входов в прибор равно 26	В прибор вошло 26 транзактов для обслуживания
AVARAGE TIME/TRAN 15.884	СРЕДНЕЕ ВРЕМЯ/ТРАН	Среднее время обслуживания одного транзакта прибором joe равна 15.884 мин.

Выходные данные для очереди joeq

MAXIMUM CONTENTS 8	Максимальное содержимое	В течение времени моделирования в очереди никогда не было более 8 клиентов
AVARAGE CONTENTS 3.78	Среднее содержимое	Среднее число клиентов, находившихся в очереди
TOTAL ENTRIES 27	Общее число входов	Всего в очередь вставало 27 клиентов (транзактов)
ZERO ENTRIES 1	Нулевые входы	Число входов в очередь без последующего ожидания
PERCENT ZEROS 3.70	Процент нулевых входов	Из общего числа входов в очередь 3.7% было нулевых
AVARAGE TIME/TRAN 67.25	Среднее время/транз	Среднее время, проведенное в очереди, с учетом нулевых входов
$AVARAGE TIME/TRANS 69.83	$ Среднее время/транз	Среднее время нахождения в очереди на один ненулевой вход
CURRENT CONTENTS 8	Текущее содержимое	В момент завершения моделирования в очереди находились 8 транзактов

Выходные данные для МНУ

STORAGE stor	Имя МНУ	Имя накопителя stor
CAPACITY 2	Емкость МНУ	В накопителе 2 места
AVARAGE UTILIZATION 0.70	Средняя загрузка МНУ	В течение 70 % модельного времени 2 места в накопителе были заняты
ENTRIES 27	Число входов	Общее число транзактов, вошедших в накопитель
AVARAGE TIME/TRAN 24.73	Среднее время/транз	Среднее время, проведенное в накопителе
MAXIMUM CONTENTS 2	Максимальное содержимое	В процессе моделирования были моменты, когда все 2 места в накопителе были заняты одновременно
AVARAGE CONTENTS 1.39	Среднее содержимое	В среднем 1.39 мест в МНУ были заняты
CURRENT CONTENTS 1	Текущее содержимое	В момент завершения моделирования одно место в накопителе было занято

Юлий Янович Кацман

МОДЕЛИРОВАНИЕ

Учебное пособие

Научный редактор

доктор технических наук, профессор
В. М. Дмитриев

Редактор Н. Т. Синельникова

Подписано к печати
Формат 60x84/16. Бумага ксероксная.
Плоская печать. Усл. печ. л. 5,29.Уч.-изд. л. 4,74.
Тираж экз. Заказ . Цена свободная.
ИПФ ТПУ. Лицензия ЛТ № 1 от 18.07.94.
Типография ТПУ. 634034, Томск, пр. Ленина, 30.