МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Г.Л. Муравьёв

МОДЕЛИРОВАНИЕ СИСТЕМ

**Курс лекций по дисциплине «Моделирование систем»
для студентов специальностей
«Автоматизированные системы обработки информации»
и «Вычислительные машины, системы и сети»**

БРЕСТ 2003

УДК 519.876.5
ББК 65.050я73
М91

Учреждение образования
«Брестский государственный технический университет»
Кафедра электронно-вычислительных машин и систем

Рекомендовано к изданию редакционно-издательским советом
Брестского государственного технического университета

Рецензент:
доцент кафедры математического моделирования
Брестского государственного университета им. А.С. Пушкина,
кандидат технических наук, доцент Пролиско Е.Е.

заведующий кафедрой ПОИТ
Белорусского государственного университета информатики и радиоэлектроники
Кандидат технических наук, доцент Бахтизин В.В.

Муравьев Г.Л.

М91 Моделирование систем: Курс лекций по дисциплине «Моделирование систем» для студентов специальностей «Автоматизированные системы обработки информации» и «Вычислительные машины, системы и сети». В 2 ч. Ч. 1. – Брест: БГТУ, 2003. – ~~74~~ 164, [2] с.

Издание содержит общие вопросы теории моделирования и системного моделирования сложных систем, отражена специфика моделирования сложных систем, компонентов ЭВМ и систем обработки информации на базе ЭВМ.

Курс лекций может быть использован для проведения курсового проектирования и лабораторных работ по соответствующим дисциплинам.

ISBN 985-6[illegible]ического

УДК 519.876.5
ББК 65.050я73

ВВЕДЕНИЕ

1. Предмет, цели и задачи дисциплины

Предметом дисциплины являются: - модели сложных систем; - принципы, методы и средства построения моделей; - технология организации и проведения машинных экспериментов на моделях; - метод Монте-Карло, принципы его машинной реализации; - методы машинной реализации алгоритмов генерации случайных событий, величин и процессов; - особенности организации имитационного моделирования ЭВМ и систем обработки информации на базе ЭВМ; - тенденции в области автоматизации моделирования.

Целью преподавания дисциплины является изучение: - методики имитационного и аналитического моделирования сложных систем с использованием современных средств компьютерной техники; - специфики моделирования ЭВМ, компонентов ЭВМ и систем обработки данных на основе ЭВМ; - получение практических навыков по реализации методики машинного моделирования и расчету характеристик систем и сетей массового обслуживания.

Студенты должны знать: - методику построения математических моделей; - методику проведения аналитического и статистического моделирования сложных систем, обработки результатов моделирования, оценки их точности; - модели случайных величин и процессов; - элементы теории массового обслуживания (МО) в рамках марковских процессов, методы расчета основных характеристик СМО и сетей СМО; - особенности моделирования на разных этапах разработки и использования ЭВМ, комплексов, сетей и систем обработки данных на их основе; - методику исследования процессов функционирования и оценки эффективности реальных систем обработки данных на основе моделирования. Студенты должны уметь применять на практике полученные знания при принятии технических решений на микро- и макроуровнях.

Данный курс опирается на такие дисциплины как "Высшая математика", "Математические модели информационных процессов и управления", "Системный анализ и исследование операций" и др.

2. Моделирование как метод научного познания и решения оптимизационных задач

Практически все математические расчеты, особенно выполняемые с помощью ЭВМ, могут быть отнесены к категории моделирования. Здесь человек, исследователь может изучать свойства какого-либо объекта, в том числе реального, с помощью абстракций, математических соотношений, существующих только в его голове или на бумаге. Такие модели, кстати, так и называются абстрактными.

Моделирование имеет ярко выраженную практическую, прикладную направленность. В зависимости от области приложения, особенностей моделируемого объекта может использовать специфический математический аппарат,

методы и подходы. В то же время вне зависимости от области приложения в моделировании можно выделить общие, базовые методы, приемы, технологии.

Моделирование объекта производится для выявления его свойств: - прогнозирования будущего состояния или поведения объекта; - количественной оценки эффективности объекта; - нахождения аналитической зависимости между характеристиками и параметрами, отыскания оптимальных значений параметров объекта; - обучения специалистов. Это типичные задачи анализа, прогнозирования и синтеза систем.

При этом исходный объект заменяют другим, подобным, аналогичным ему. Все действия по анализу и синтезу производят на новом объекте, а полученные результаты (значения характеристик, выявленные закономерности) переносят на исходный объект. Процесс замещения одного объекта другим, с целью исследования его свойств с последующим переносом их на исходный объект называется *моделированием*, а замещающий объект – *моделью*.

Результаты моделирования тем точнее, чем правдоподобнее модель. Поэтому в основе моделирования лежат предположения, гипотезы о подобии (гомоморфизме) и аналогии (изоморфизме) объекта и модели, их частном и общем сходстве. Эти гипотезы, сведенные к удобным для исследователя логическим, математическим схемам и образуют модель. Поскольку абсолютно подобным любому объекту может быть только сам этот объект, то при моделировании неизбежны ошибки, а оценка адекватности модели, ее точности - одна из важнейших задач моделирования.

Моделирование базируется на принципах информационной достаточности (имеющейся информации должно быть достаточно для построения адекватной модели), осуществимости (модель должна обеспечивать достижение цели исследования с нужным качеством), множественности (полное исследование объекта требует ряда моделей, отражающих его с разной степенью детальности), агрегирования (сложную систему следует представлять в виде агрегатов - подсистем, для описания которых существуют стандартные математические схемы), параметризации (описание в модели процессов функционирования относительно изолированных подсистем можно заменять соответствующими числовыми величинами, таблицами, графиками или формулами).

Человек вынужден прибегать к моделированию, чтобы упростить, удешевить, ускорить процесс исследования по целому ряду веских причин: отсутствует сам объект (например, только проектируется); объект не доступен для исследований; объект доступен, но либо медленно развивается, либо непосредственное манипулирование объектом слишком дорого, сложно.

Таким образом, в узком смысле моделирование – это исследование объекта на модели, в широком и сам процесс построения модели. В философском плане это познавательный процесс, включающий переработку информации, которая поступает из внешней среды об объекте, в результате чего и формируются некоторые образы, которые присущи объекту. "От живого созерцания к абстрактному мышлению и от него к практике – есть диалектический путь познания истины" (В.И. Ленин). Модель не только инструмент, средство, но и объ-

ект, обладающим собственными параметрами и характеристиками, а следовательно тоже объект исследования.

При работе с такими объектами как ЭВМ, компоненты и системы на базе ЭВМ (сети, АРМы, АСУ, АСОИ, АСОЭИ, САПРы и т.п.) наряду с экспериментированием на самих объектах, что ввиду их сложности и дороговизны затруднено, на всех этапах их жизненного цикла, начиная с предпроектного, широко применяют моделирование. В частности выделяют уровень макро- и микромоделирования. На макроуровне моделируется объект управления (обработки), определяются требования к нему, формулируются критерии эффективности. Моделируются алгоритмы управления (обработки), которые будут использованы в системе, оценивается их эффективность. На микроуровне строят модели отдельных компонентов и оцениваются их характеристики.

3. Роль ЭВМ в моделировании. Машинное моделирование

Моделирование стало инженерным инструментом во многом благодаря изобретению и распространение компьютеров, что:

1. Повысило "вычислительную мощность" исследователя и позволило моделировать задачи с известными методами решения, но слишком трудоемкими для "ручного" счета. В частности, широкое распространение получили статистические испытания (метод Монте-Карло), позволяющие заменять объекты случайной и детерминированной природы случайными процессами с аналогичными характеристиками. Процессы имитируются на компьютере, их характеристики накапливаются в виде наборов случайных чисел, которые затем обрабатываются статистически и дают искомые результаты.

2. Привело к появлению существенно нового направления в моделировании, т.е. к машинному, статистическому, имитационному моделированию. Оно напоминает натурный эксперимент, только роль объекта выполняют программы, воспроизводящие его структуру и протекающие в ней процессы на ЭВМ. Поведение этих программ происходит по законам развития объекта. Результаты машинного моделирования также как и в методе Монте-Карло представляют собой наборы случайных чисел, которые затем подвергаются статистической обработке.

4. Место моделирования в системе наук

Общие законы получения, хранения, передачи и преобразования информации в сложных управляющих системах любой природы изучает *кибернетика*, в том числе и *техническая кибернетика*, предшественником которой была теория автоматического управления (ТАУ).

В рамках кибернетики вопросы проектирования, планирования, конструирования и поведения сложных информационных систем, основу которых составляют универсальные средства преобразования информации – ЭВМ, изучаются *системотехникой*. Это направление кибернетики появилось в 60-х годах с развитием АСУ, а ныне применяется в САПР, АСНИ, АСОИ, АСОЭИ и т.п.

Основной принцип принимаемых в системотехнике решений – его максимальная эффективность по соотношению - ценность решения и его стоимость, а основные подходы, включая системный, определены в общей теории систем (ОТС) – *системологии*, которая и составляет теоретическую основу кибернетики и системотехники. Количественное обоснование принимаемых решений, методы оценки их эффективности базируются на аппарате исследования операций (методах аналогий, экспертных оценок, прямых расчетов, в т. ч. и математического моделирования).

ОТС – научное направление, связанное с разработкой совокупности философских, методологических, конкретно-научных и прикладных проблем анализа и синтеза сложных систем произвольной природы. А *теория моделирования* как взаимосвязанная совокупность концепций, методов и средств создания и исследования моделей входит в общую теорию систем как принципиально новый инструмент кибернетики – машинное моделирование (см. рис.1).

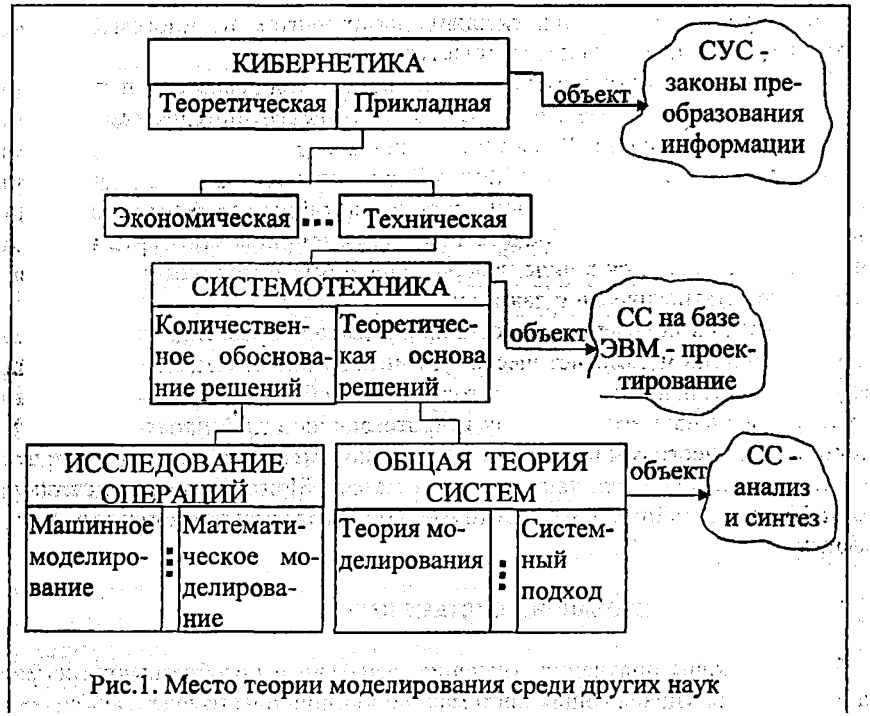


Рис.1. Место теории моделирования среди других наук

5. Основные понятия и определения общей теории систем

Система – совокупность связанных элементов (компонентов), объединённых в целое для достижения цели. Система задается парамстрами структуры и функционирования <*S, F*>, а под *целью* понимается совокупность результа-

тов, рассматриваемых как назначение системы. Основное свойство системы – целостность.

Элемент – минимальный неделимый объект, который рассматривается как целое, "чёрный ящик", внутренний состав и структура которого игнорируются. Описывается на уровне спецификаций, связывающих значения входов и выходов. Понятие элемент относительное и зависит от целей исследования, то есть элемент как бы фиксирует уровень детальности рассмотрения объекта, необходимый и достаточный для моделирования его свойств с нужным качеством.

Система функционирует, преобразуя входные сигналы x в выходные y, и тем *самым*, реализует правило получения результатов в соответствии со своим законом функционирования f_S ($y = f_S(x)$). Например, если система – специализированное устройство, реализующее функцию синуса (то есть $y = sin(x)$), то f_S описывается как функция *sin*. *Закон функционирования* является спецификацией системы, описывает ее как "чёрный ящик" и реализуется соответствующей структурной и функциональной организацией системы. Может быть задан функцией, функционалом, таблицей, системой логических уравнений, алгоритмически и т.п.

Структура – фиксированная совокупность элементов и связей между ними, отображаемая графом, а в инженерной практике схемой.

Организация – целенаправленное изменение структуры или функциональных *процессов* с целью обеспечения максимальной эффективности системы.

Свойства, которыми обладает любая система, делятся на параметры и характеристики. *Параметры* – первичные свойства системы, однозначно задающие ее, а *характеристики* - вторичные свойства, производные от параметров. И те и другие могут быть измерены количественно или качественно.

Степень соответствия системы её назначению называется *эффективностью*. Для ее количественной оценки используют как отдельные характеристики - показатели эффективности одного свойства системы так и *критерии эффективности* - интегральные оценки, построенные путем обобщения характеристик. Характеристики, как правило, противоречивы. Примеры характеристик - стоимость, производительность, надежность системы. Критерии бывают прямые и инверсные, одно и многокритериальные. Пример прямого однокритериального критерия - отношение быстродействия системы к стоимости (критерий В.М. Глушкова).

Система оптимальна, если ей соответствует экстремальное значение критерия эффективности на всём множестве мыслимых вариантов её организации. При работе с системами решают задачи анализа и синтеза. Задача *анализа*: задана структурная и функциональная организация системы, требуется определить ее характеристики. Задача *синтеза:* задана спецификация системы, в ряде случае фрагменты ее структурно-функциональной организации, требуется путем конкретизации структурно-функциональной организации системы добиться ее оптимальности по заданным критериям.

6. Характеристика объектов моделирования как сложных систем

Сложные (большие) системы – это системы, обладающие свойством эмерджентности, когда при объединении элементов у системы появляется новые качества, свойства, которые не выводятся простым способом из свойств составляющих её элементов. Соответственно характеристики сложных систем нельзя получить простой суперпозицией характеристик элементов и необходимо применять системный подход.

Основные причины, делающие систему сложной: - сложность структуры (большое количество элементов и типов элементов, иерархичность структуры, сложные связи между элементами); - сложность реализуемых функций; - сложность функциональной организации (наличие управления, алгоритмов адаптации, обучения, настройки, обеспечения надежности); - наличие случайных факторов, неопределенностей, стохастичности в работе, в т.ч. из-за воздействия внешней среды и присутствия человеческого фактора. Соответственно ЭВМ, компоненты ЭВМ и системы на базе ЭВМ также относятся к категории сложных систем.

7. Системный подход к изучению сложных систем

Выделят классический (индуктивный) и системный подход к анализу систем. *Классический подход* (рис.2) основан на движении от частного к общему. При этом система декомпозируется на отдельные несвязанные подсистемы, которые исследуются (моделируются) независимо. Из полученных на этом этапе характеристик подсистем простой суперпозицией определяют характеристики системы в целом. Например, требуется определить суммарный вес ЭВМ, проектируемой из стандартных блоков двух типов (*k1* блоков первого и *k2* второго типа). В такой постановке задачи на моделирование система является простой, поэтому используется индуктивный подход. По отдельности измеряют веса блоков (*v1* и *v2* соответственно). Суммарный вес определяют их суперпозицией $S = k1 \times v1 + k2 \times v2$. Аналогично, кстати, могут быть вычислены системные показатели надежности ЭВМ.

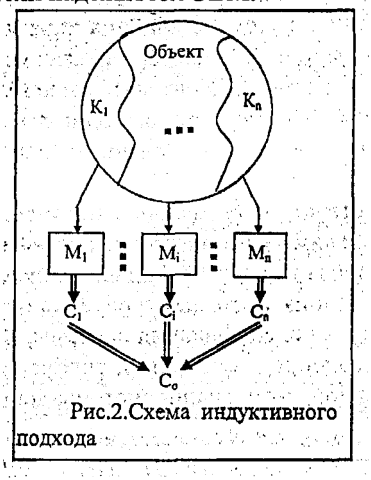


Рис.2.Схема индуктивного подхода

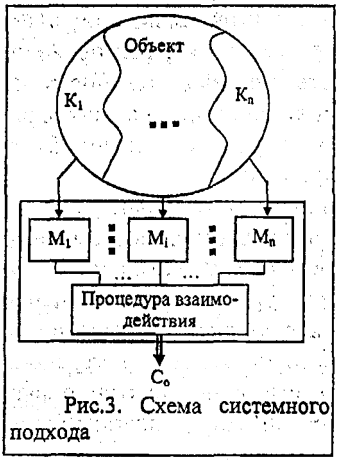


Рис.3. Схема системного подхода

Иначе обстоит дело, если потребуется вычислить среднее время решения задач в указанной системе, при условии, что их параметры, а также закон поступления случайны (например, описываются экспоненциальными законами). Зная среднее время обработки задачи каждым устройством, невозможно определить простой суперпозицией такие системные характеристики, если не учитывать взаимное влияние устройств, возникающие при этом очереди, а следовательно и время, проводимое задачами в очередях. Система здесь превращается в сложную.

Системный подход (рис.3) основан на сохранении целостности объекта на всех этапах его моделирования. Технологически исходный объект также декомпозируется на отдельные подсистемы, в частности с учетом возможности их моделирования. Однако одновременно разрабатывается специальная процедура учета их взаимного влияния. Оцениваются характеристики подсистем и на базе указанной процедуры вычисляются и характеристики всего объекта. Применение системного подхода иллюстрируется в Главе 2, § 3.7 и в Главе 3, § 2.4, модель 2.

Глава 1

ОБЩИЕ ВОПРОСЫ ТЕОРИИ МОДЕЛИРОВАНИЯ

1. Модели: классификация, методы и средства расчета

1.1. Модели: параметры и характеристики

Из-за двойственной природы модели следует различать ее собственные параметры и характеристики и параметры и характеристики объекта, отображаемого в модели. Первые определяют особенности построения модели и соответственно ее эффективность. Вторые универсальность и полезность модели для исследования объекта. Основные параметры модели – тип, способ расчета, способ продвижения модельного времени, подход к реализации псевдопараллельностей и т.п.

Полезность, эффективность модели определяются возможностью просчитывать с ее помощью зависимость характеристик объекта от значений параметров с нужным качеством. Соответственно основные характеристики модели: адекватность (A), которая базируется на понятиях точность расчета характеристик объекта на модели (Δ) и универсальность модели (U); трудоёмкость как разработки модели ($t1$) так и расчёта характеристик объекта на модели ($t2$); стоимость моделирования, включающая стоимость разработки модели ($S1$) и стоимость расчёта на модели характеристик объекта ($S2$).

Трудоёмкость для аналитических моделей может измеряться затратами на однократный расчёт (один прогон) на модели характеристик объекта для одного заданного набора значений параметров, т.е. в одной точке. Либо затратами

на реализацию одного такта модели, обработки одного события, имитации одного состояния объекта для имитационных моделей. В качестве абсолютных единиц измерения используется потребное число команд, в качестве относительных единиц – затрачиваемое время.

Одна из основных задач моделирования – оценка адекватности модели. Внешней оценкой адекватности, не требующей знания и анализа внутреннего строения модели, обычно является точность (погрешность) воспроизведения моделью характеристик объекта. Погрешность может вычисляться с разной степенью детальности. Ее можно рассчитать для одной точки, для множества точек области функционирования объекта, по всей области изменения параметров объекта. Погрешность (абсолютная, относительная) может оцениваться на уровне средних значений, средних значений и дисперсий, на уровне распределений. В зависимости от особенностей модели могут вычисляться погрешности ее чувствительности. В целом сложность оценки адекватности модели схожа с проблемой тестирования программ. И здесь исчерпывающие оценки можно получить, лишь выполнив моделирование для всех возможных наборов параметров, что вряд ли целесообразно и возможно.

В качестве внутренней характеристики адекватности, определяющей в конечном итоге точность модели, используется ее универсальность. Численно универсальность может характеризоваться мощностью класса структур и функций объекта, отображаемых в модели. Пусть исходный объект задается N параметрами, а каждый i-й параметр может принимать h_i число значений. Декартово произведение $H = h_1 \times h_2 \times ... \times h_i \times ... \times h_N$ задаст множество точек в N-мерном пространстве, которые соответствуют возможным состояниям объекта и описывают область его определения. В модели отображаются не все параметры и их значения, соответственно область определения модели описывается как $H^* = h_1^* \times h_2^* \times ... \times h_K^*$. Чем ближе $H^* \to H$, тем модель универсальнее. Численно универсальность можно вычислить как мощность соответствующего множества H^*. Кроме этого, универсальность говорит о полезности, информативности, продуктивности модели для проведения экспериментов над объектом.

Пусть, например, объект (рис.4) задается одним параметром x и обладает одной характеристикой $y = 4 + (x-2)^2$. Представим себе модель с функцией $y = 4$. Эта модель описывает объект абсолютно точно при $x=2$, с некоторой допустимой погрешностью в окрестностях указанной точки, а за ее пределами – с недопустимой погрешностью. К тому же в модели нет параметра x объекта и ее нельзя использовать для изучения влияния значения x на поведение объекта (значение y).

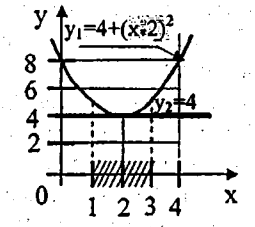


Рис.4. Адекватность модели в точке и в области

1.2. Виды моделей. Классификация

Существует много классификационных признаков, из которых ниже рассмотрены основные. Объекту может соответствовать (рис.5, б) одна модель, группа моделей последовательных приближений, для сложных объектов строят *иерархические (составные) разнотипные* модели. Различают модели объектов и модели среды, в которой объекты существуют. В зависимости от области приложения, характера объектов различают технические, экологические, экономические, социальные, правовые и другие модели. Далее рассматриваются *технические* модели (рис.7).

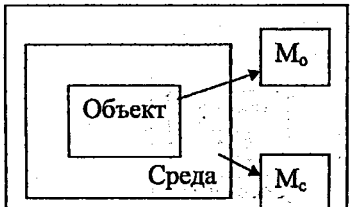


Рис.5. Модели объекта и среды

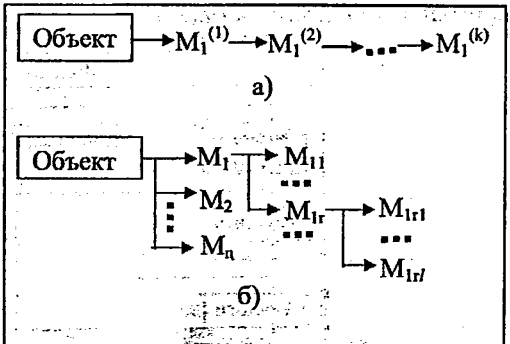


Рис.6. Модель: а) последовательных приближений; б) иерархическая

В зависимости от степени отображения в модели структурно-функциональных свойств объекта различают *структурные, функциональные (поведенческие, кибернетические) и структурно-функциональные модели*, являющиеся *наиболее* полными. В зависимости от того, на каком этапе жизни объекта применяется модель, бывают предпроектные, проектные, эксплуатационные (для обучения персонала, для модернизации и настройки объекта) модели.

В зависимости от принципа построения модели среды выделяют модели, управляемые трассой (*трассовые* модели) и модели, построенные на *уровне средних оценок* параметров среды. В первом случае в модели используют данные мониторинга реального объекта (трассы). Например, при моделировании станка, обрабатывающего детали, производится хронометраж всех существенных процессов, связанных с их обработкой (фиксируется тип детали, время ее поступления, длительность обслуживания, ожидания и т.п.). Эта информация затем напрямую используется в модели. Во втором случае свойства внешней среды описываются как случайные процессы. В указанном выше примере для этого необходимо произвести статистическую обработку трасс и получить описание среды либо на уровне математических моментов, математических ожиданий, дисперсий и т.д., либо на уровне законов распределения. Трассовые модели наиболее адекватны реальной среде, однако их не всегда можно получить, не

всегда удаётся «уловить» типовую трассу, ее параметрами сложно манипулировать при моделировании.

В зависимости от учёта случайных факторов модели бывают *детерминированными и стохастическими* (вероятностными); в зависимости от учёта временного фактора *статическими и динамическими*; в зависимости от характера величин, которые описывают объект, *дискретными (цифровыми), непрерывными (аналоговыми) и смешанными.*

Модели, воспроизводящие структурно-функциональную организацию объекта, его физическую природу (возможно в ином временном и пространственном масштабе) относят к моделям-*подобиям*, а модели, функционирование которых подчиняется тем же законам, что и у объекта, но обладающие иной природой, к моделям-*аналогам*.

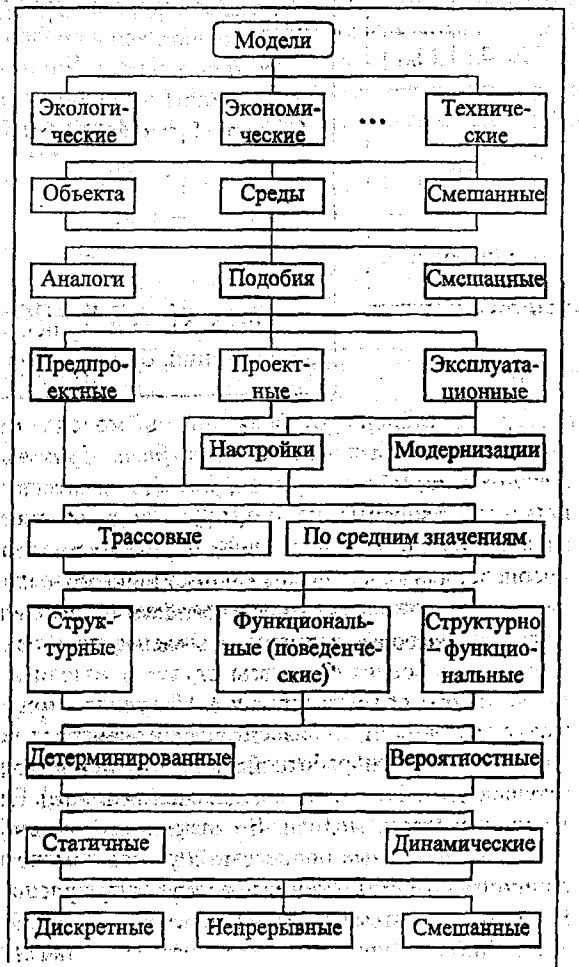


Рис.7. Классификация моделей

В зависимости от природы замещающего объекта, модели делятся на *абстрактные* и *реальные* (рис.8). Реальные модели, использующие объект или его части, называют *натурными* (на их базе проводят комплексные, промышленные и научные испытания).

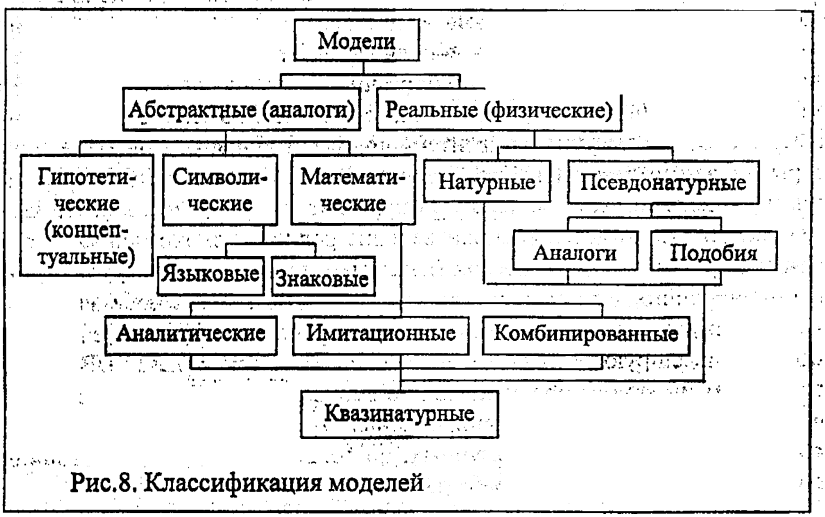


Рис.8. Классификация моделей

В зависимости от степени формализации выделяют концептуальные, символические и математические модели. Наименее формализованы *концептуальные* модели, представляющие собой разнородную информацию об объекте в произвольной форме (в виде таблиц, графиков, формул, текстов и т.д.). Эти модели - результат обследования объекта, они конкретизируют цели исследования, уровень детализации объекта в модели и описание элементов, определяют состав параметров и характеристик объекта, критерии эффективности, основные закономерности, присущие работе объекта. Как правило, абстрактное моделирование начинается с построения концептуальных моделей.

Символические модели являют следующий этап формализации, хотя, как правило, не являются результативными – не позволяют решать модели аналитически. Это знаковые (символьные) и языковые модели. В знаковых вводятся обозначения основных понятий и отношений между ними (операции), а модель представляется цепочкой символов–понятий и символов-операций. В языковых моделях описания строятся на базе фиксированного алфавита и строгих синтаксических и семантических правил (как в языках программирования или языках спецификаций).

В *математических* моделях объект заменяется формализованным описанием, например, совокупностью математических отношений в терминах соответствующего математического аппарата, что делает такую модель максимально формализованной и потенциально результативной. В сравнении с натурным экспериментом и физическими моделями математическое моделирование отли-

чается экономичностью, универсальностью используемого технического и программного обеспечения. Обеспечивает моделирование абстрактных (проектируемых) объектов, опасных или трудновоспроизводимых режимов, позволяет изменять масштаб времени.

Эти модели либо решаются аналитически (дают формульные решения либо перекладывается на имитационное моделирование. Соответственно в зависимости от способа расчёта на модели характеристик объекта выделяют *аналитические, имитационные* и *комбинированные* модели. Комбинированные модели строятся на принципе декомпозиции исходного объекта (модели) и представляют собой совокупность имитационной и аналитической моделей и процедуры их взаимодействия. Это позволяет обеспечить необходимую адекватность и универсальность модели при приемлемой трудоемкости. На имитационную модель возлагается воспроизведение редких и плохо поддающихся аналитическому описанию процессов (например, генерация входного потока заявок, имитирующих задачи пользователя; планирование – выбор заявок из очереди на обслуживание и распределение ресурсов и т.д.). Частые циклические процессы моделируются аналитически (например, счет задач в ЭВМ представляется сетью массового обслуживания с центральным обслуживающим прибором). В имитационных и комбинированных моделях результаты получают путем статистических (как бы натурных, но только на модели) испытаний с последующей статистической обработкой накопленных данных.

1.3. Математические модели сложных систем. Общее описание

Объект и соответствующая ему модель (рис.9) задаются множеством параметров $\langle H,V,X\rangle$, где $H=\langle S,F\rangle$ параметры структуры и функционирования, V - множество параметров внешних воздействий, X - параметры входов. Кроме этого, результаты, выходы системы описываются множеством Y, а эффективность объекта множеством характеристик C, на базе которых могут быть построены критерии $I=I(C)$. Параметры внешних воздействий включают параметры независимых возмущающих воздействий R и управляющих воздействий G.

Множество значений $H=\{\bar{h}(t)\}$ образуют векторы $\bar{h}(t)=(h_1(t),...,h_{nh}(t))\in H$, характеризующие всевозможные наборы параметров объекта. Соответственно декартово произведение $D_H=d_{h_1}\times d_{h_2}\times...\times d_{h_{nh}}$ задает область определения объекта (или модели), а произведение $D_M=D_H\times D_V\times D_X$ - область определения всей системы, включая объект и его среду.

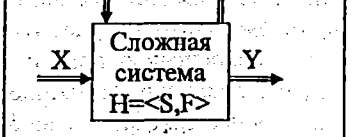


Рис.9. Сложная система

Значения выходов определяются через закон функционирования f_S как $Y=F_S(X,V,H,t)\Leftrightarrow\bar{y}(t)=f_S(\bar{x}(t),\bar{v}(t),\bar{h}(t),t)$. При этом, если $t=const$, то модель статическая и $\bar{y}(t)=f_S(\bar{x},\bar{v},\bar{h})$, в противном случае модель динамическая. У моделей с дискретным временем оно принимает значения из множества $\{t_0,t_1,...\}$, в част-

ном случае меняется с постоянным шагом Δt. При имитационном моделировании время обычно дискретно. Если в ходе моделирования параметры объекта и внешних воздействий не меняются, действие случайных факторов не учитывается, то модель детерминированная и $\vec{y}(t) = f_S(\vec{x}(t), \vec{h}, \vec{v}, t) = f_S(\vec{x}(t))$.

Поскольку цель моделирования – рассчитать характеристики объекта, то математическая модель есть функционал $\Phi_C = \langle\langle H, V, X, t\rangle, C\rangle$, связывающий значения параметров объекта и его характеристик. Модель, указанный функционал может быть решен аналитически, имитационно либо комбинированным способом.

1.4. Аналитическое решение математической модели

Смысл аналитического решения состоит в построении уравнений, непосредственно связывающих характеристики объекта с его параметрами. Это означает, что функционал Φ_C разрешим относительно характеристик C и $C = F_C(X, V, H, t) \Leftrightarrow \vec{c}(t) = f_C(\vec{x}(t), \vec{v}(t), \vec{h}(t), t)$. Если функционал в явном виде не разрешим или разрешим с низким качеством (точностью, универсальностью и т.п.), то можно воспользоваться численными методами и, в том числе, методом Монте-Карло. Здесь математические операции над переменными заменяют действиями над числами, а для придания получаемым результатам общности надо повторить расчёты многократно.

Достоинства аналитического решения и соответственно аналитических моделей: - высокая общность или универсальность получаемых результатов (в виде формул!), что позволяет в дальнейшем применять аппарат математического анализа для их исследования и методы оптимизации для синтеза наилучших решений; - сравнительно низкая трудоёмкость расчёта (для вычисления характеристик для одного набора значений параметров требуется лишь однократный расчёт полученных формул). Основной недостаток - сравнительно низкая универсальность моделей из-за вносимых упрощений, соответственно их низкая точность и ограниченная познавательная и информационная ценность. Наиболее простые аналитические модели – алгебраические, регрессионные, более сложные – модели теории массового обслуживания, учитывающие действие случайных факторов.

1.5. Имитационное решение математической модели

Если модель не разрешима или ее даже не возможно построить (функционал *Фс)*, либо разрешима, но с неудовлетворительным качеством, то прибегают к имитационным методам. Имитация - наиболее мощный и универсальный метод исследования систем, поведение которых зависит от случайных факторов. Имитационная модель это формальное описание логики функционирования исследуемого объекта, характера взаимодействия его элементов во времени, учитывающее наиболее существенные причинно-следственные связи объ-

екта и обеспечивающее проведение статистических экспериментов. Здесь объект воспроизводится с максимальной адекватностью с сохранением состава и структуры элементов, внутренних процессов и характера их протекания во времени (рис.10). Параметры объекта при этом используются для создания структуры модели и для генерации в ней процессов, адекватных процессам объекта. Имитационные модели чаще реализуются в виде программ в терминах универсальных языков или языков моделирования, реже в виде специализированных апаратно-программных систем. Результаты имитационного моделирования, как и любого численного метода, носят частный характер. Обработка результатов требует применения специальных статистических процедур. А для получения обоснованных выводов необходимо проведение серии модельных экспериментов.

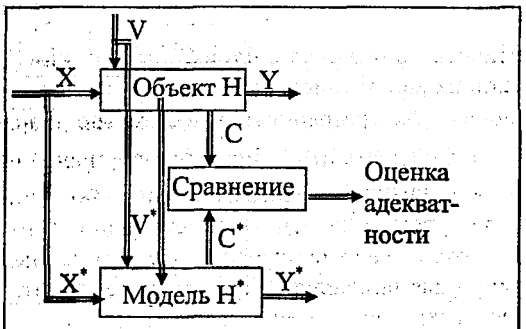


Рис.10. Построение модели объекта

Каждый объект функционирует, переходя под влиянием *событий* из одного состояния множества $S = \{S_i \mid i = \overline{1,K}\}$ в другое. Каждое *состояние* описывается n_z-мерным вектором $\vec{z}(t) = (z_1(t), z_2(t), ..., z_{n_z}(t)) \in Z$ специальных характеристик, однозначно его идентифицирующих. Множество всевозможных векторов состояний $Z = \{z_i(t)\}$ задает точки n_z-мерного фазового пространства объекта, а переходы объекта из одной точки в другую образуют его фазовую траекторию.

Имитационное моделирование состоит в многократном выполнении программы-модели на рассматриваемом отрезке времени $t_0 \le t \le T$, начиная с начального состояния $\vec{z}(t_0) = \vec{z}_0$. В каждый момент времени t_n программа воспроизводит переход объекта в следующее состояние, вычисляя его в зависимости от предыстории - цепочки предыдущих состояний по функции переходов как $\vec{z}(t_n) = \Psi_z(\vec{z}(t_0), \vec{z}(t_1), ..., \vec{z}(t_{n-1}), \vec{X}(t_n), \vec{h}(t_n), \vec{V}(t_n), t_n)$, и протоколирует изменение характеристик объекта. Поэтому результаты статистической обработки характеристик, определяемых на модели, становятся адекватными тем характеристикам объекта, которые нужно вычислить. Причем, чем больше характеристик объекта учтено в векторе состояний модели и чем больше точек присутствует в ее фазовом пространстве, тем детальнее выполняется моделирование. В инженер-

ной практике оказывается приемлемым учитывать только предыдущие состояния объекта. Поэтому в общем виде процесс имитации описывается следующим образом: - задаётся начальное время моделирования $t = t_0$ и модель устанавливается в исходное состояние $\vec{z}(t_0) = \vec{z}_0$; продвигается модельное время и определяют новое состояние, значения выходов и характеристик,

$$\begin{cases} t = t_n; \\ \vec{z}(t_n) = \Psi_z(\vec{z}(t_{n-1}), \vec{X}(t_n), \vec{h}(t_n), \vec{V}(t_n), t_n); \\ \vec{y}(t_n) = \Psi_y(\vec{z}(t_n), t_n); \\ \vec{c}(t_n) = \Psi_c(\vec{y}(t_n)); \end{cases}$$

повторяя это многократно (для тактов $n=1,2,3...$).

Число тактов (длительность прогона) выбирается из необходимости обеспечения требуемой статистической точности характеристик либо достижения заданного T. На следующем этапе производится статистическая обработка значений характеристик, полученных на каждом такте, и рассчитываются критерии.

достоинства имитационного решения и соответственно имитационных моделей: многопараметричность, потенциально высокая универсальность и соответственно высокая адекватность, точность результатов и информационная, познавательная ценность. Теоретически статистическая составляющая погрешности моделирования может быть снижена до нужного предела увеличением его длительности (числа тактов). Недостаток моделей: сравнительно высокая сложность и трудоемкость. Сравнительная оценка характеристик расчета различных математических моделей приведена на рис.11.

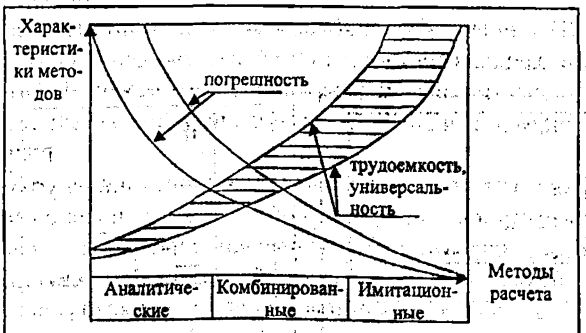


Рис.11. Сравнительная характеристика методов расчета математических моделей

1.6. Пример построения математической модели

Объект исследования – система, которая состоит из одного обслуживающего канала ($k = 1$) и в каждый момент времени может обслуживать только од-

ну заявку (например, автоматическая телефонная станция АТС, которая последовательно обслуживает заявки на соединение, поступающие от абонентов). Заявки поступают случайным образом. Число команд, требуемых для их обслуживания, также варьируется случайным образом. Скорость канала составляет $V = 200$ команд в минуту. Система работает стационарно, потерянных заявок нет – интенсивности входного и выходного потоков заявок совпадают. Цель моделирования: анализ характеристик качества обслуживания заявок и использования канала системы; прогнозирование характеристик при изменении параметров системы (например, как изменятся характеристики АТС, если входной поток вырастет на 30%. Что нужно изменить в системе, чтобы среднее время обслуживания осталось на прежнем уровне?).

Концептуальная модель. В поцессе мониторинга и обследования системы получены трассы заявок, фиксирующие время τ, через которое появлялась заявка, и количества команд Θ, потраченных на ее обслуживание. Это позволило построить соответствующие законы распределения - функции плотности f_τ и f_t (рис. 12, 13). Здесь времена обслуживания заявок определялись как $t=\Theta/V$. По функциям плотности могут быть рассчитаны моменты параметров системы, например, среднее время между заявками в потоке составляет $m_\tau = 7c$, а среднее время обслуживания $m_t = 3c$. Соответственно средняя интенсивность потока заявок $\lambda_0 = 10$ заявок/мин, а средняя скорость обслуживания $\mu = 20$ заявок/мин. Таким образом, система задается параметрами $\bar{h} = (h_1, h_2) = (V, k)$, входной поток заявок параметрами $\bar{x} = (x_1, x_2) = (f_\tau, f_\Theta)$. А искомые характеристики составляют вектор $\bar{c} = (\rho; l; m; \omega; u)$, где ρ - коэффициент загрузки канала в долях единицы; l - средняя длина очереди заявок; $m = l + \rho$ - среднее количество заявок в системе; ω - среднее время ожидания в очереди; $u = \omega + m_t$ - среднее время пребывания заявки в системе.

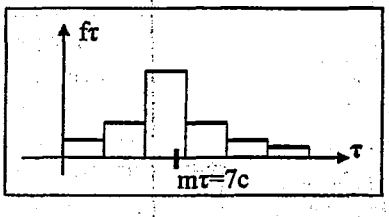


Рис.12. Закон распределения времени поступления заявок τ

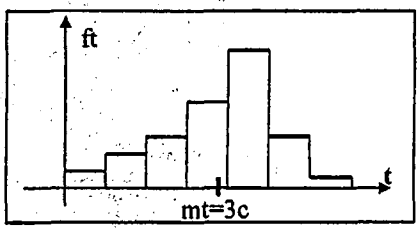


Рис.13. Закон распределения времени обслуживания заявок t

Математическая модель (аналитическая, алгебраическая). В целях упрощения учтем только средние значения параметров системы (математические ожидания) и сделаем допущение, что все заявки одинаковы, поступают ровно через 7 с и обслуживаются за 3 с. Модель станет статичной, детерминированной и ее решение будет описываться следующими соотношениями:

$$\begin{cases} \omega = 0\,c \\ u = \omega + \mu_\tau = 3\,c \\ l = 0 \\ m = l + \rho = 3/7 \end{cases}$$

Математическая модель (аналитическая, вероятностная). Учтем случайный характер поступления и обслуживания заявок. Однако реальные законы f_τ и f_t заменим теоретическими, для которых известно аналитическое решение модели. Сохраним лишь их совпадение на уровне математических ожиданий. Будем считать, что заявки во входном потоке распределены по закону Пуассона (тогда значения τ подчиняются экспоненциальному закону), а время обслуживания, значения t подчиняются экспоненциальному закону. Соответственно средняя интенсивность потока заявок λ_0 и средняя скорость обслуживания μ останутся прежними. При таких допущениях моделью системы будет СМО типа M/M/1 (рис.14) и можно использовать известные формульные решения. Коэффициент загрузки $\rho = \lambda_0 / \mu = 0{,}5 \le 1$, что означает, что СМО работает в стационарном режиме и очередь в системе конечна. Значения других характеристик:

$$l = \frac{\rho^2}{1-\rho} = 0{,}5 \quad , \quad m = \frac{\rho}{1-\rho} = 1.$$

Из закона Литтла ($\lambda_0 \omega = l$ и $\lambda_0 u = m$) определяют временные характеристики системы

$$\omega = \frac{l}{\lambda_0} = 3c \qquad u = \frac{m}{\lambda_0} = 6c$$

Математическая модель (имитационная). Учтем в модели все знания об объекте, в том числе реальные законы поступления и обслуживания заявок. Модель будет строиться как динамическая, вероятностная. Модельное время меняется дискретно, алгоритм продвижения модельного времени – событийный. Структура модели представлена на рис. 15, где модули М1 и М2 выполняют роль генераторов. Первый модуль генерирует последовательность случайных чисел τ, подчиняющихся закону распределения f_τ, а второй имитирует длительности времени обслуживания заявок t в соответствии с заданным законом распределения f_t. Модуль М3 имитирует функционирование объекта, реагируя на происходящие в нем события и изменяя его состояния. Кроме того, он собирает информацию о прошлых, текущем и будущих состояниях объекта. Модуль М4 выполняет статистическую обработку накопленных резуль-

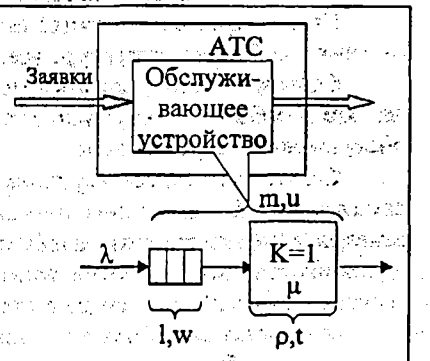


Рис.14. Структура объекта и его модель в виде СМО M/M/1

татов и вычисляет характеристики, а модуль М5 управляет моделью и ходом моделирования.

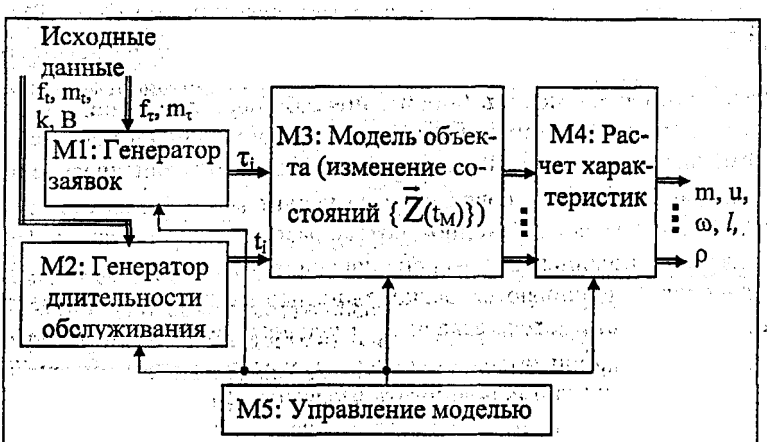


Рис.15. Структура имитационной модели

Алгоритм работы модели на уровне модулей включает следующие этапы: ввод исходных данных (параметров среды и объекта), задание режимов *моделирования* (описание набора вычисляемых характеристик и требований к ним; задание длительности моделирования); настройка и инициализация модели; моделирование (многократный чередующийся запуск модулей М1-М3, М5); вычисление итоговых результатов.

При построении имитационной модели должны быть определены: - ее состояния; - события; - структура временных цепей (списков).

Событие – факт, который может привести к изменению состояния модели. Здесь события: поступление на обслуживание новой заявки и завершение обслуживания текущей заявки.

Состояние модели укрупнённо описывается числом заявок *z(t)*, находящихся в ней. Для детального описания необходимо фиксировать временные параметры и характеристики заявок (времена появления заявок, их типы, номера, трудоемкости, время дообслуживания текущей заявки и т.п.), позволяющие прогнозировать последующие состояния при наступлении новых событий.

Временная цепь (список) – описывает распределение во времени последовательности событий и может храниться, например, в виде списковой структуры. Включает цепь прошлых, настоящих и будущих - прогнозируемых событий. По мере продвижения модельного времени настоящие события становятся прошлыми, а будущие настоящими. Текущие события обрабатываются, могут изменять состояние модели, приводить к новым событиям в будущем и следовательно изменять цепь будущих событий. Цепь прошлых событий не меняется и используется для управления работой модели (поэтому цепь часто реализует-

ся в виде списковых структур, а эффективность модели во многом определяется эффективностью построения цепи и операций с нею).

Одна из основных задач модуля М5 – анализировать временные цепи и определять время наступления ближайшего события, идентифицировать его тип, инициировать обработку события (запуск модулей М1-3), корректировать временные цепи, изменять состояние модели. Алгоритм моделирования показан на рис. 16.

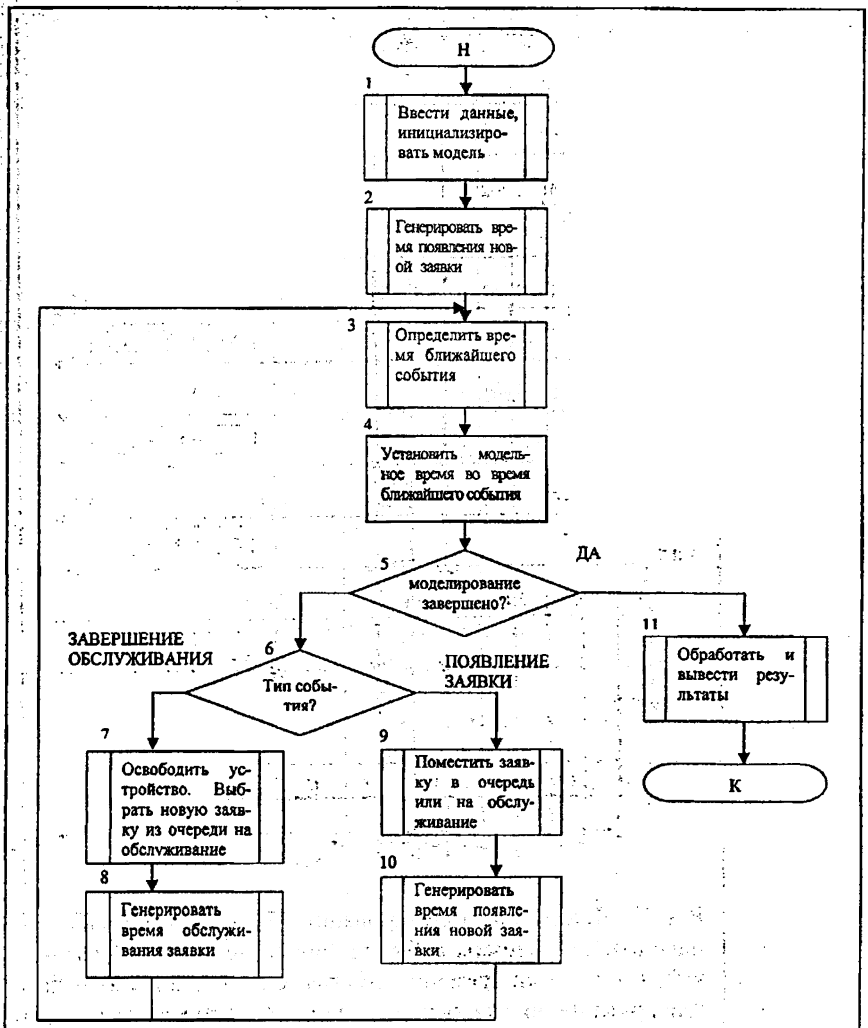


Рис. 16 Моделирование одноканальной СМО типа G/G/1

Несколько тактов работы модели представлены на рис. 17, где временная цепь представлена двумя списками C_1 и C_2 . При инициализации модели (такт 0), модельное время *tм* устанавливается в ноль, запускаются модули М1-2 (например, генерируются числа 1 и 4) и в список C_1 заносится прогнозируемое событие со временем наступления *tм+1=1*. В начальном состоянии очередь и канал пусты.

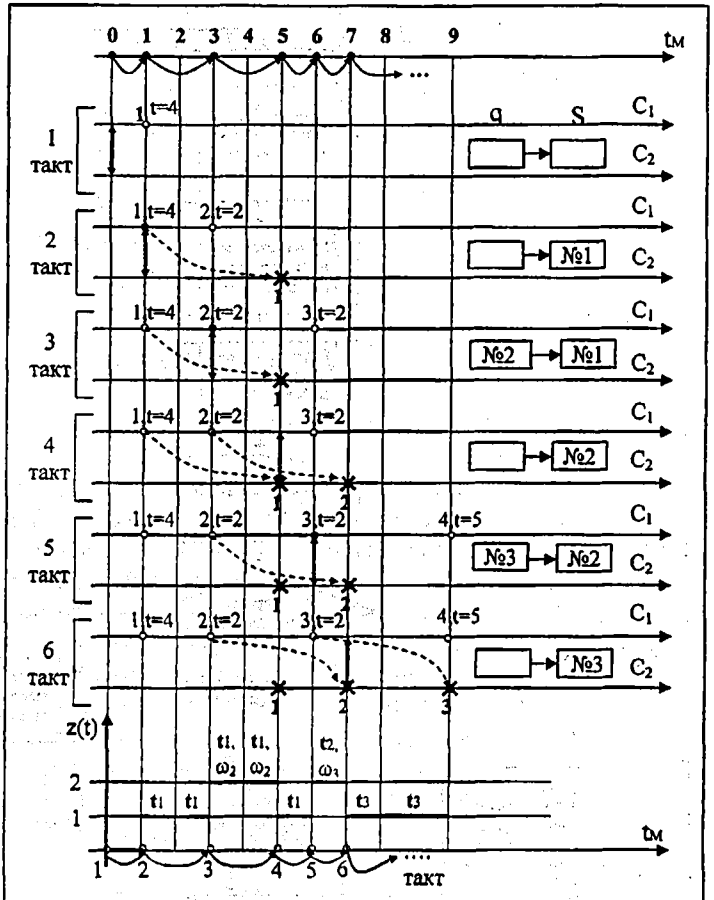


Рис.17. Диаграмма моделирования объекта (АТС). Обозначения: C_1 – список событий "поступление новой заявки"; ∘ – отметка времени события; C_2 – список событий "завершение обслуживания заявки"; × – отметка времени наступления события; q – содержимое очереди (номера заявок); S – содержимое обслуживающего устройства (номер обслуживаемой заявки); ↕ -модельное время

F-модели используют для исследования цифровых систем (элементов и узлов ЭВМ, систем контроля, устройств управления и т.д.). В частности, это модели раздела кибернетики - теории автоматов (рис.18) $F = \langle X, H, Y, t \rangle = \langle X, Z, \varphi, \Psi, z_0, Y, t \rangle$. Например, конечные автоматы Мили описывают системы с K возможными состояниями $z_0, z_1, ..., z_K$, преобразующими в допустимые такты времени слова $x(t_0), x(t_1), ..., x(t_n)$ из входного алфавита X в слова $y(t_0), x(t_1), ..., x(t_n)$ выходного алфавита Y. Правила преобразования информации (слов), определяемые законом функционирования f_S, задаются функцией переходов φ и функцией выходов Ψ:

$$\begin{cases} y(t_n) = \Psi(z(t_n);\ x(t_n)); \\ z(t_{n+1}) = \varphi(z(t_n), x(t_n)). \end{cases}$$

В частном случае – для автоматов Мура, значение выходного слова определяется только текущим состоянием автомата:

$$\begin{cases} y(t_n) = \Psi(z(t_n)) \\ z(t_{n+1}) = \varphi(z(t_n), x(t_n)). \end{cases}$$

Если у автомата только одно состояние, то он относится к категории автоматов без памяти или комбинационных, логических схем, моделью которых в двоичном алфавите служат булевы функции $\{y(t_n) = \Psi(x(t_n))\}$.

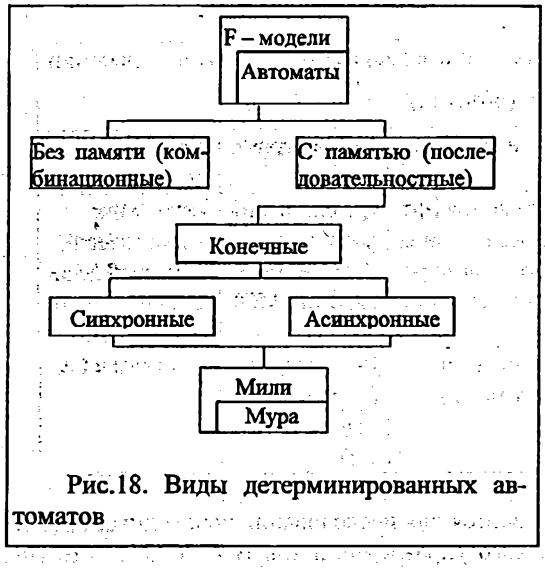


Рис.18. Виды детерминированных автоматов

P-модели используют для исследования вероятностного поведения систем, в т.ч. цифровых, описываемых, в частности, марковскими цепями. Например, вероятностные автоматы (рис.19) $P = \langle X, H, Y, t \rangle = \langle X, Z, B, z_0, Y, t \rangle$ описывают дискретный, потактовый преобразователь информации с памятью. Его функционирование схоже с работой конечного автомата, также в каждом такте зависит

от состояния памяти, но описывается статистически. Здесь закон функционирования f_S (функции φ и Ψ) описывается матрицей B вероятностей переходов из одного состояния модели в другое. Матрица для каждой пары текущих значений входного сигнала и состояния задает вероятность пары новых значений состояния и выходного сигнала. В частном случае, в зависимости от того, какой аспект модели (переходы или выходы) описывается как в F-модели, выделяют соответственно Z-детерминированные и Y-детерминированные вероятностные автоматы.

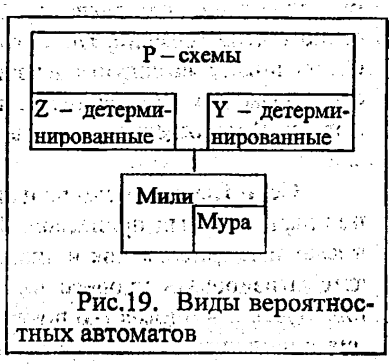


Рис.19. Виды вероятностных автоматов

Q-модели - класс моделей $Q = \langle X, H, Z, Y, t \rangle = \langle X, S, F, Z, z_0, Y, t \rangle$, широко используемых в инженерной практике системного моделирования, в частности, для анализа производительности, надежности систем. Их подмножество составляют стохастические сетевые модели (ССМ), в которых, как правило, упрощенно отражен аспект управления в системах. Подмножество ССМ – сети массового обслуживания (сети МО), состоящие из систем массового обслуживания (СМО) и ориентированные на исследование систем с ограниченными ресурсами и очередями, в т.ч. аналитически точно и приближенно. Для реализации Q-схем существует целый класс языков моделирования (например, Симула, Симскрипт, GPSS и т. д.).

Здесь выделяют неподвижные объекты-узлы, включающие обслуживающие узлы (устройства, памяти), маршрутные, управляющие и др., и подвижные объекты-заявки (транзакты), которые в процессе обслуживания движутся по сети неподвижных узлов, образуя очереди. В сетях МО текущее состояние описывается через состояния отдельных СМО $Z = \{\vec{z}_i\}$, где состояние отдельной СМО $\vec{z}_i = (\overline{z_i^H}; \overline{z_i^K})$ складывается из состояния очереди-накопителя $z_i^H = \overline{0, V_H}$ (задается числом ожидающих заявок) и состояния обслуживающих каналов $z_i^K = \overline{0, K}$ (задается числом обслуживаемых заявок).

A-модели применяют, в частности, для объединения в рамках одного описания моделей разных типов (например D-, Q-, F- и P-моделей). Они основаны на агрегатном подходе к описанию сложных систем, предложенном академиком Н. П. Бусленко. Исходная система рассматривается как совокупность подсистем-агрегатов, чье поведение описывается в едином стиле похожем на описание автоматов. A-модель описывается как $A = \langle X, H, V, Y, t \rangle = < X, H, Z, \varphi, \psi, \varphi_R, Z^{(r)}, Y, T, T_z >$, где X - описание входных сигналов; H - параметров объекта; Z - состояний; φ - функция переходов, определяющая при изменении входов следующие состояния модели, исходя из текущих; ψ - функция выходов; φr - функция переходов, определяющая следующие состояния в результате случайного перехода при неизменных входных сигналах; $Z^{(r)}$ - состояния, на которых автомат генерирует выходной сигнал; Y -

описание выходных сигналов; *T* - описание тактов времени объекта; *Tz* – описание тактов времени, где допустимо изменение состояния объекта. А-модели обеспечивают высокую адекватность и являются формальным описанием достаточным для дальнейшего программирования или аппаратной реализации модели. В упрощённых случаях они позволяют исследовать модели аналитически.

Сети Петри - одно из наиболее мощных средств описания и моделирования параллельных процессов. Задаются как в графической форме, что обеспечивает наглядность, так и аналитически как $P=<B, D, I, O, M>$, что позволяет автоматизировать процесс их анализа методами имитационного моделирования. Здесь *B* - множество позиций; *D* - множество переходов; *I* - входная функция инцидентности, которая для каждого перехода задает множество его входных позиций; *O* - выходная функция инцидентности, которая для каждого перехода задает множество его входных позиций; *M* - функция разметки, которая каждой позиции сети ставит в соответствие неотрицательное целое число. Соответственно граф сети состоит из вершин-переходов (соответствуют событиям) и вершин-позиций (соответствуют условиям их возникновения), соединенных ориентированными дугами. Каждая дуга может связывать лишь разнотипные вершины. Сеть описывает присущие системе причинно-следственные связи в статике. Для отображения динамики используют специальный объект - фишки или метки позиций. Расположение фишек в позициях сети называется ее разметкой. Переход активен (событие может произойти), если в его каждой входной позиции есть хотя бы одна фишка. Срабатывание конкретного перехода изменяет разметку сети по следующему правилу: изымается по одной метке из каждой из его входных позиций и добавляется по одной метке в каждую из его выходных позиций. Основной недостаток сетей Петри – нулевое время срабатывания переходов, что не позволяет исследовать временные характеристики объектов. Расширение сетей Петри – "оценочные" Е-сети (evaluation), где используется несколько типов вершин-позиций (простые позиции, позиции-очереди, разрешающие позиции); фишки имеют атрибуты; с каждым переходом может быть связана ненулевая задержка и функция преобразования атрибутов фишек.

2. Технология моделирования

2.1. Формализация проблемы

Основные этапы моделирования схематично представлены на рис. 20.

Моделирование начинается с постановки проблемы, под которой понимается несовпадение желаемого и действительного. Формулируются цели моделирования, выявленная проблема сводится к перечню задач, подлежащих решению. Дальнейшая формализация проблемы связана с построением модели.

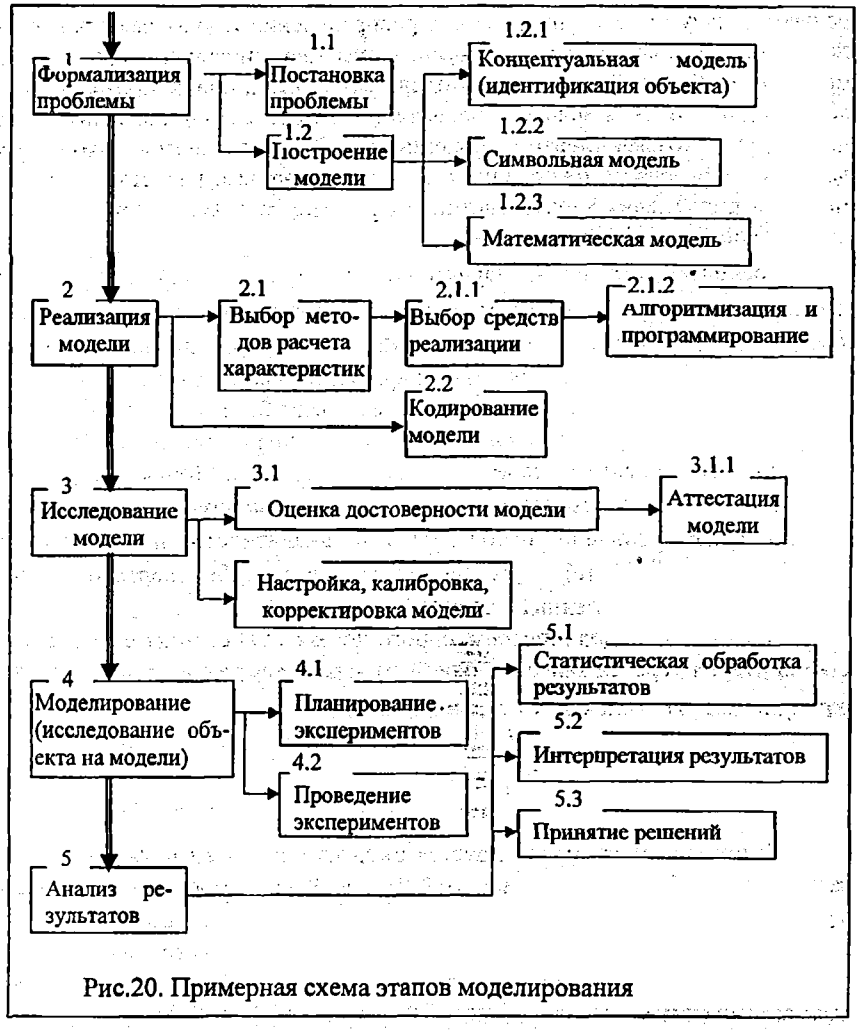


Рис.20. Примерная схема этапов моделирования

Построение концептуальной модели включает: - определение типа объекта; - описание внешней среды (рабочей нагрузки); - декомпозицию объекта на элементы. Объект, если это возможно, обследуют, собирают сведения о его структуре и функционировании. Выявляют характер взаимодействия его элементов, алгоритмы управления. Формулируют набор параметров объекта, выделяют первичные, наиболее влияющие на его характеристики, которые и следует отображать в модели. Выявляют характерные состояния объекта. Определяют, будет ли модель детерминированной или вероятностной. Иногда детерминированную среду заменяют стохастической с целью избежания полного перебора параметров при экспериментах с моделью. И напротив, в целях упрощения модели, решения ее аналитическими методами вероятностную среду и объект могут трактовать как детерминированные. Если модель вероятностная, то

решают задачу определения законов распределения случайных процессов среды и объекта, подбора соответствующих им теоретических законов. В зависимости от требований к точности модели могут быть ориентированы на вычисление только средних значений характеристик, или средних значений и их дисперсий, учет коэффициентов вариации случайных процессов, законов распределения. Определяют требования к исходным данным, качеству задания параметров и вычисления характеристик объекта (по составу, по точности), формулируют критерии эффективности.

Поскольку отдельные характеристики $\vec{c} = (c_1, c_2, ..., c_m)$ системы обычно противоречивы и разнотипны, то на их основе строят критерии эффективности $\bar{I} = \bar{I}(\vec{c})$. В практическом применении проще однокритериальные модели (например, система описывается одной характеристикой - стоимостью S, $m = 1$ и критерий $I = C = S$). Если $m > 1$, то система может быть многокритериальной и для упрощения работы ее пытаются свести к однокритериальной. При этом, если получено аналитическое выражение для критерия эффективности $I = I(\vec{c}) = I(\vec{h}, \vec{x}, \vec{v}, t)$, то это позволяет решать оптимизационные задачи, в частных случаях решать уравнение относительно h аналитически и находить оптимальные решения $\vec{h} = I^{-1}(\vec{c})$. Существуют разные способы свертывания частных характеристик в обобщенный критерий.

Один из подходов (способ главного показателя) состоит в том, что в качестве критерия берется значение одной из характеристик c_k, а на остальные характеристики накладываются ограничения λ

$$I = I(c_k)$$
$$\lambda \begin{cases} \forall f(c_i) \\ i \neq k \end{cases}$$

Например, пусть проектируется система обработки информации, описываемая системой характеристик $\vec{c} = (S, U)$, где S - стоимость, U - среднее время ответа на запрос. Тогда возможны следующие варианты: $I = \min S$ и $U \le U^*$ или $I = \min U$ и $S \le S^*$, где значение со звездочкой – заданное ограничение.

Иногда удается построить единый критерий на базе основных характеристик естественным способом (способ "затраты - эффект"). Например, таким является критерий В.М. Глушкова $I = I(\vec{c}) = S / \lambda_0$ для систем обработки информации с характеристиками $\vec{c} = (S, \lambda_0)$, где S - стоимость, а λ_0 - производительность системы.

Единый критерий можно построить искусственным путём, используя метод экспертных оценок, например, в виде аддитивного критерия $I = \sum_{i=1}^{m} \alpha_i \times \beta_i(c_i)$. Здесь α_i - веса характеристик ($\sum_{i=1}^{m} \alpha_i = 1$), а β_i - функциональное преобразование, устраняющее разнотипность (разную размерность) характеристик. Например, если $0 \le c_i < c_{i\max}$, то преобразование $\beta_i = c_i / c_{i\max}$ превращает любые размерные величины в безразмерные. В аналогичном способе целевого

программирования строится критерий $I = \sum_{i=1}^{n} \alpha_i (c_i - c_{i0})^2$, имеющий смысл расстояния до "идеальной" точки c_{i0} в пространстве значений характеристик эффективности. Если свертку вектора характеристик эффективности не удается получить, то можно использовать один из аксиоматических методов, основанных на использовании понятия парето-оптимальности. Стратегия парето-оптимальна, если по всем характеристикам она не хуже любой стратегии из допустимого множества и лучше хотя бы по одной из них.

Указанные подходы относят к случаю, когда характеристики вычисляются на уровне средних значений. Если характеристики - случайные величины, то для стационарно функционирующего объекта на каждую характеристику можно наложить ограничение $\lambda = \{P[c_{i\min} \le c_i \le c_{i\max}] = \int_{c_{i\min}}^{c_{i\max}} f_{c_i}(x)dx \ge \beta_i\}$, а для нестационарного объекта $\lambda = \{1/(T - t_0) \int_{t_0}^{T} \int_{c_{i\min}}^{c_{i\max}} f_{c_i}(x)dx \ge \beta_i\}$.

Если объект, среду, либо подобные объекты невозможно обследовать, то их параметры и поведение прогнозируются, например, на основе экспертных оценок. При этом пытаются подобрать такие значения параметров внешней среды, которые создавали бы наиболее тяжёлые, напряженные режимы работы объекта. Это позволяет получать на модели граничные значения - наихудшие оценки характеристик. Тогда в реальной обстановке объект обеспечит качество функционирования не хуже смоделированного.

2.2. Реализация и исследование модели

Модель имеет смысл, если ее можно решить. Если для модели существуют средства реализации, то её нужно закодировать на соответствующем языке. В противном случае выбранный метод расчёта математической модели реализуют программно.

Разработанную математическую модель аттестуют – например, определяют ее качество путем расчёта характеристик на модели и сравнения модельных результатов с эталонными либо на основе их экспертной оценки. Достоверность модели проверяется, в частности, анализом возможности решения поставленной задачи на модели с заданным качеством, анализом точности и корректности этапов моделирования, в т.ч. перевода концептуальной модели в математическую, правильности математических соотношений, заложенных в модели. Требования к адекватности зависят от характера решаемых задач (см. таблицу ниже).

Область применения модели		Допустимая погрешность, %	Типы моделей
Проектирование новых объектов (системный уровень)		30-50	Аналитические, имитационные, модели чувствительности
Проектирование новых объектов на базе серийных		20-30	Аналитические, имитационные
Эксплуатация	Настройка объекта	10-30	Аналитические, имитационные модели, вычисляющие характеристики в абсолютных единицах
	Модернизация объекта	20-30	

При оценке адекватности нужно учитывать разнотипность характеристик, их вероятностный характер, сложность определения допустимых отклонений $\Delta_\partial, P_{\Delta\partial}, \delta_\partial, P_{\delta\partial}$, отсутствие эталонных значений (особенно при проектировании новых объектов). Оценка может проводиться: - сравнением средних значений характеристик модели и системы; - по дисперсии отклонения характеристик модели от среднего значения характеристик системы; - по максимальному значению относительных отклонений характеристик модели от характеристик системы. При оценке адекватности могут рассчитывать абсолютные $\Delta C = | C_Э - C_M |$ и относительные $\delta C = \Delta C / C_Э$ погрешности моделирования средних значений характеристик, используя эталонные характеристики $C_Э$ и модельные характеристики C_M. Для учета случайного характера оценок рассчитываются и оцениваются реальные значения вероятностей отклонения $P[\Delta C < \Delta_\partial] \geq P_\partial$ и $P[\delta C < \delta_\partial] \geq P_{\delta\partial}$.

Если характеристики реального объекта нельзя использовать в качестве эталона $C_Э$, то используют похожие объекты, либо сравнивают модели одного объекта, но разной сложности и построенные разными способами, прибегают к косвенной и экспертной оценке моделей, оценивают адекватность подмоделей или оценивают чувствительность модели. Кроме точностных свойств, оценивают также достоверность, трудоемкость модели.

При неудовлетворительных результатах выполняют калибровку модели, т.е. ее коррекцию с целью приведения в соответствие предъявляемым требованиям. Процесс калибровки может носить итеративный характер и включать следующие этапы: - перепроектирование модели, глобальные изменения (например, введение новых процессов, типов событий и т.д.); - настройку модели, локальные изменения (например, изменение законов распределения случайных величин и т.д.); - изменение специальных параметров, называемых калибровочными.

2.3. Моделирование и анализ результатов

После аттестации модель можно использовать для моделирования и исследования объектов. Если модели сложные, трудоемкие, число экспериментов велико, то они планируются специальным образом. Особое значение это имеет в случае использования имитационных моделей, моделей Монте-Карло, проведения статистических экспериментов. Цель планирования экспериментов - получить максимальный объем информации при минимальных затратах на его проведение. При моделировании стационарных объектов средние значения их характеристик не зависят от времени и вычисляются по потактным данным одного прогона. Соответственно трудоёмкость расчетов составляет $k \times n$ тактов и складывается из количества прогонов модели k (для стационарных объектов соответствует числу экспериментов, один эксперимент для одного набора характеристик) и длительности одного прогона – числа тактов моделирования n (для статистических экспериментов определяется точностью вычислений – необходимым объемом статистического материала). Для нестационарного объекта (рис.35) характеристики зависят от времени и для каждого временного среза усредняются по прогонам. Поэтому каждый из k экспериментов потребует r прогонов длительностью n тактов, определяемых заданной длительностью моделирования. Всего потребуется $r \times k \times n$ тактов. План эксперимента строится в так называемом факторном пространстве, образуемом множеством параметров модели, значениями которых можно управлять в ходе эксперимента. Часто план строится относительно одной характеристики, называемой наблюдаемой переменной. Ее значение, полученное в ходе эксперимента, складывается из значения неслучайной функции отклика факторов-параметров и случайной величины – статистической ошибки эксперимента.

Снижение трудоёмкости экспериментов возможно за счет уменьшения количества прогонов r, числа экспериментов k и длительности одного прогона n. Стратегическое планирование направленно на выбор оптимального количества экспериментов-прогонов, компромиссного плана (сочетания значений – уровней и рангов параметров), обеспечивающего получение наиболее полной и достоверной информации о поведении системы при фиксированном или минимальном числе опытов. Тактическое направлено на выбор оптимальной длины прогона.

В стратегическом планировании: - идентифицируют факторы, ранжируя их по степени влияния на показатели эффективности и выделяя первичные и вторичные (количественные и качественные), для количественных определяют уровни возможных значений, для качественных – ранги значений; - выбирают уровни факторов, учитывая, что они должны перекрывать возможный диапазон изменения каждого из них, а общее число уровней по всем факторам не должно приводить к чрезмерному объему моделирования. Два полюсных плана эксперимента - полнофакторный (полный перебор, в котором реализуются все возможные сочетания уровней факторов) и однофакторный план с изменением факторов по одному (исследование в экспериментах влияния одного параметра

при фиксации всех остальных). Все остальные варианты планов лежат между указанными полюсами, относятся к неполным, частичным факторным планам и являются наиболее робастными. Это: - рандомизированный, когда сочетания уровней факторов для каждого прогона выбираются случайным образом; - латинский план («латинский квадрат»), когда эксперимент проводится с одним первичным фактором и несколькими вторичными; - дробный факторный план, когда каждый фактор принимает только два уровня - нижний и верхний.

Например, система включает процессор (возможны модификации с быстродействием b_1 в 100, 200 и 300 тысяч операций/с), оперативную память (доступны два типа, $b_2 = 2$), внешнюю память (один из четырех типов, $b_3 = 4$). Тогда полнофакторный эксперимент потребует 3x2x4=24 эксперимента, а однофакторный 3+2+4=9 экспериментов.

При тактическом планировании уменьшают длительность эксперимента за счет предварительного расчета числа тактов, исходя из требуемой точности результатов, контролируют достижение точности в ходе эксперимента. Поскольку при статистическом эксперименте значения характеристик рассчитываются как значения случайных величин, то необходимое число опытов зависит от вида распределения наблюдаемой характеристики, ее дисперсии, коррелированности между собой элементов выборки, наличия и длительности переходного процесса моделируемого объекта.Используют математические методы для уменьшения дисперсии моделируемых процессов. Для исключения из длительности моделирования переходных процессов, длительность которых существенно зависит от начального состояния объекта, он предварительно просчитывается на "грубой", аналитической модели, которая отображает его работу в установившемся состоянии, а полученные данные используют затем для начальной установки имитационной модели.

Результаты, полученные при моделировани, обрабатываются, в т.ч. статистически, интерпретируются и служат основой принятия управленческих, проектных решений.

3. Аналитическое моделирование сложных систем

В инженерной практике многие аналитические модели основываются на аппарате математического программирования и исследования операций. На базе математического программирования решают задачи синтеза, а на основе исследования операций – анализа систем. При этом задачи и соответственно модели традиционно разделяют на: стандартные, однокритериальные, с однозначными целями, рассчитываемые по формулам, аналитически; хорошо структурированные, однокритериальные, сформулированные количественно, с определенными исходными данными, многовариантные; слабо структурированные, многокритериальные, с неясными целями, с элементами неопределенности в поведении объекта, многовариантные. Здесь типовые модели (задачи): модели распределения; управления запасами; модели теории массового обслуживания;

модели упорядочивания (сетевого планирования); модели выбора оптимальных маршрутов; замены оборудования; игры и т.п..

Виды машинного моделирования представлены на рис. 21. Аналитические модели рассчитываются формульно или численно. Часто применяемая разновидность численного метода - статистические испытания (метод Монте-Карло). В качестве стандартных средств реализации следует отметить средства ЭТ (Excel), математических и статистических пакетов (Mathematica, MathCad, Statistica и др.).

Рис.21. Виды машинного моделирования

4. Имитационное моделирование сложных систем

4.1. Общая характеристика имитационного моделирования

Имитационное моделирование состоит в замене объекта аналогичным математическим с сохранением его структуры и состава и проведением над ним статистических опытов, экспериментов. Пусть характеристика X, которую надо оценить в ходе моделирования, отличается вероятностной природой. Например, это время ответа на запрос. Тогда можно вычислять среднее значение времени ответа, оценивать его вероятностный разброс – дисперсию, вычислять закон распределения времени ответа и т.п. Чтобы вычислить оценку $\tilde{m}_x$ математического ожидания m_x (например, времени ответа), надо провести n опытов (тактов имитационного моделирования) и определить в каждом значение этой характеристики. На основе полученного ряда $x_1,...,x_n$ из n значений вычисляют оценку $\tilde{m}_x$ как их среднее значение $\sum x_i / n$. При этом каждая новая серия опытов приведет к новому значению оценки, т.е. они сами ведут себя как случайные величины $\tilde{m}_x(n)$.

Теоретическую основу опытов и имитационного моделирования составляют предельные теоремы, которые доказывают, что при неограниченном увеличении числа опытов средние значения наблюдаемых характеристик перестают быть случайными и совпадают с их реальными значениями. Для нашего примера это означает, что для повышения точности оценки надо увеличивать длительность моделирования, число опытов n и тогда в пределе $\lim_{n\to\infty} \tilde{m}_x \to m_x$. Поскольку большинство других статистических оценок, включая оценки начальных и центральных моментов произвольного порядка, также вычисляются как средние значения, то этот принцип применим и для них. В практическом применении предельные теоремы позволяют оценивать необходимую длительность имитационного моделирования исходя из требуемой точности вычисления характеристик.

Предельные теоремы основываются на неравенстве Чебышева $P\left\{\left|X-m_x\right|\geq\alpha\right\}\leq D_x/\alpha^2$, где α - величина отклонения (погрешность), а D_x дисперсия X и каждая теорема применима для своих условий проведения статистических опытов.

Теорема Чебышева применяется для случая, когда значения величины X, полученные в опытах, независимы и принадлежат одной и той же выборке, т.е. одинаково распределены, обладают одинаковыми средними и дисперсиями. Теорема формулируется как $P\left[\left|\sum_{i=1}^{m} x_i/n - m_x\right|\geq\alpha\right]\leq D_x/n\alpha^2$ или для противоположного события $P\left[\left|\sum_{i=1}^{m} x_i/n - m_x\right|<\alpha\right]\geq 1-D_x/n\alpha^2$. Указанное выражение оценивает вероятность того, что отклонение (погрешность) $\left|\sum_{i=1}^{m} x_i/n - m_x\right|$ оценки $\tilde{m}_x(n)$, определяемой как среднее $\sum_{i=1}^{m} x_i/n$ по набору полученных в опытах значений $x_1,\ldots,x_n$, от их реального (неизвестного) математического ожидания m_x составит не более величины α. Как видно, эта вероятность ограничена величиной $1-D_x/n\alpha^2$, что означает, что для любой сколь угодно малой погрешности α всегда можно подобрать число опытов n, которое ее обеспечит. При $n\to\infty$ теоретически обеспечивается абсолютно точный результат (α=0), т.к. $\lim P\left[\left|\sum_{i=1}^{m} x_i/n - m_x\right|\leq\alpha\right]\to 1$ и $\lim_{n\to\infty} \tilde{m}_x(n)\to m_x$.

Обобщённая теорема Чебышева применяется, если значения величины X, полученные в опытах, независимы, но принадлежат к разным выборкам, обладают разными средними и дисперсиями, ограниченными величиной L. Тогда

$$P\left[\left|\sum_{i=1}^{m} x_i/n - \sum_{i=1}^{m} x_i/n\right|\geq\alpha\right]\leq L/n\alpha^2.$$

Теорема Маркова применяется, если значения величины X, полученные в опытах, произвольно распределены и зависимы.

Основные задачи, решаемые при организации имитационного моделирования: - разработка структуры модели и выбор средств ее реализации; - выбор и реализация способа продвижения модельного времени; - выбор и реализация алгоритмов имитации параллельных процессов (реализация псевдопараллельностей); - выбор способа организации временных цепей (списков), алгоритмов их просмотра; - генерация случайных чисел и объектов с заданными распределениями; - планирование экспериментов; - сбор результатов и их статистическая обработка; - автоматизация имитационных моделей, в т.ч. разработка пользовательских интерфейсов.

4.2. Сбор результатов моделирования и их статистическая обработка

Результаты имитационного моделирования, как правило, представляют собой наборы случайных чисел $\{x_i \mid i=\overline{1,n}\}$ - реализаций случайных процессов, описывающих качество функционирования моделируемого объекта. Например, это такие характеристики как сведения о поминутной загрузке устройств, времена обслуживания деталей в процессе обработки на станке, длительности ожидания заявок на установление связи с абонентом и т.д. Соответственно характеристики надо правильно измерять, накапливать, хранить, обрабатывать и анализировать.

Поскольку исходных данных может быть очень много, то их можно обрабатывать по ходу эксперимента. Иногда это диктуется и структурой модели – так в составных, иерархических моделях функционирование одних моделей может потребовать в качестве исходных данных текущие результаты других моделей. Это может существенно увеличить трудоемкость моделирования, к тому же часто заранее не ясно, какие оценки характеристик надо вычислять. Как правило, до эксперимента не известны законы поведения вычисляемых характеристик, часто приходится использовать непараметрические оценки. Поэтому, если это возможно, лучше накапливать данные, а затем обрабатывать их.

К типовому набору обычно оцениваемых характеристик относят статистические оценки $\{\tilde{a}_k \mid k=\overline{1,r}\}$ точечных характеристик, моментов $\{a_k \mid k=\overline{1,r}\}$. Чаще всего это математическое ожидание $a_1 = m_X$, дисперсия $a_2 = D_X$, корреляционный момент $a_3 = K_{XY}$ для системы двух случайных величин. Кроме этого, для детального описания наблюдаемых случайных величин (характеристик) строят законы их распределения, гистограммы. При анализе данных выдвигают и подвергают проверке различные гипотезы.

Если характеристика стационарна, ее изменения с течением времени носят случайный характер при неизменном законе распределения $F_X(x,t) = P[X(t) < x] = P[X < x] = F_X(x)$, то ее оценки вычисляют на выборке значений одного прогона $\{x(t_i) \mid i=\overline{1,n}\} = \{x_i \mid i=\overline{1,n}\}$.

В качестве статистической оценки $\tilde{a}_k = \tilde{a}_k(x_1, x_2, ..., x_n) = \tilde{a}_k(n)$ измеряемой величины a_k используют результаты вычислительных процедур, формул, обладающих свойствами несмещённости, состоятельности и эффективности. Несмещённость означает, что оценка не содержит методическую ошибку, т.е. $M[\tilde{a}_k(n)] = a_k$. Состоятельность означает, что точность оценки растет с увеличением числа опытов $\lim_{n\to\infty} \tilde{a}_k(n) = a_k$, а эффективность, что оценка обладает лучшей "сходимостью" - минимальным разбросом значений по сравнению с другими оценками той же величины, т.е. $D[\tilde{a}_k(n)] \to \min$.

Эти требования приводят к тому, что для перечисленных выше моментов в качестве таких оценок используют

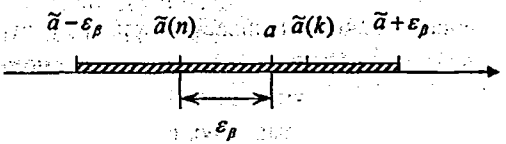
$$\tilde{a}_1 = \tilde{m}_X = \sum_{i=1}^{n} x_i / n,$$

$$\tilde{a}_2 = \tilde{D}_X = \sum_{i=1}^{n} (x_i - \tilde{m}_X)^2 / (n-1),$$

$$\tilde{a}_3 = \tilde{K}_{XY} = \sum_{i=1}^{n} (x_i - \tilde{m}_X)(y_i - \tilde{m}_Y) / n.$$

Пусть при моделировании вычисляется *оценка* $\tilde{a}(a_1, ..., a_n) = \tilde{a}(n)$ характеристики A. Между ней и ее реальным значением a всегда существует погрешность ε_β, т.е. оценке можно доверять с некоторой доверительной вероятностью $\beta = P\left[|\tilde{a} - a| < \varepsilon_\beta\right]$. Если произвести статистические опыты, рассчитать оценку $\tilde{a}$ и задаться погрешностью ε_β, то можно построить ее доверительный интервал $I_\beta = (\tilde{a} - \varepsilon_\beta; \tilde{a} + \varepsilon_\beta)$. Он накрывает, локализует местоположение реального значения a с вероятностью $\beta = P\left[\tilde{a} - \varepsilon_\beta < a < \tilde{a} + \varepsilon_\beta\right]$, как показано на рисунке ниже. Таким образом, оценка случайной величины должна описываться системой

$$\begin{cases} I_\beta = (\tilde{a} - \varepsilon_\beta; \tilde{a} + \varepsilon_\beta); \\ \beta. \end{cases}$$



Здесь не определена количественная связь между доверительной вероятностью и величиной погрешности, хотя качественная зависимость ясна. Известно, что для величин X, распределенных по нормальному закону, вероятности попадания их значений в интервал (α, β) определяется через функцию Лапласа Φ как $P[\alpha < X < \beta] = \left(\Phi\left((\beta - m_x)/\sigma_x\sqrt{2}\right) - \Phi\left((\alpha - m_x)/\sigma_x\sqrt{2}\right)\right)/2$. Тогда формула применима для любой оценки, сводящейся к вычислению среднего значения. (в силу центральной предельной теоремы такие оценки распределяются по закону близкому нормальному); и для вычисления вероятности попадания оценки в дове-

рительный интервал ($\alpha=\tilde{a}-\varepsilon_\beta, \beta=\tilde{a}+\varepsilon_\beta$). Так для оценки среднего $\tilde{m}_x=\sum_{i=1}^{n} x_i/n$ β $=P[\tilde{m}_x-\varepsilon_\beta<X<\tilde{m}_x+\varepsilon_\beta]=\left(\Phi\left(\varepsilon_\beta\right)/\sigma_{m_x}\sqrt{2}\right)-\Phi\left(\left(-\varepsilon_\beta\right)/\sigma_{m_x}\sqrt{2}\right)\Big)/2=\Phi\left(\varepsilon_\beta/\sigma_x\sqrt{2}\right)$. Можно решать обратную задачу – по значению вероятности β найти ε_β и построить доверительный интервал. Тогда из уравнения $\Phi\left(\varepsilon_\beta/\sigma_x\sqrt{2}\right)=\beta$ получим обратную функцию $\Phi^{-1}(\beta)=\left(\varepsilon_\beta/\sigma_{m_x}\sqrt{2}\right)$ и $\varepsilon_\beta=\sigma_{\tilde{m}_x}\cdot\sqrt{2}\cdot\Phi^{-1}(\beta)=\sigma_{\tilde{m}_x}\cdot t_\beta$. Соответственно доверительный интервал записывается как $I_\beta=(\tilde{m}_x-\sigma_{\tilde{m}_x}t_\beta;\tilde{m}_x+\sigma_{\tilde{m}_x}t_\beta)$. Аналогично для оценки $\tilde{D}_x=\sum_{i=1}^{n}(x_i-\tilde{m}_x)^2/(n-1)$ можно получить выражение $I_\beta=(\tilde{D}_x-\sigma_{\delta_x}t_\beta;\tilde{D}_x+\sigma_{\delta_x}t_\beta)$.

Формулы для расчета оценок, приведенные выше, годятся при их вычислении по завершении моделирования, когда в ходе моделирования только накапливаются значения величин $\{x_i\,|\,i=\overline{1,n}\}$, $\{y_i\,|\,i=\overline{1,n}\}$ и т.д. Если же требуется, с одной стороны, минимизировать количество накапливаемых и хранимых в ходе моделирования данных, а с другой стороны, вычислять оценки прямо в ходе моделирования, то для оценки $\tilde{a}_1$ можно просто накапливать сумму значений (одно число!), а конечное значение определять по завершении моделирования. Формулы для $\tilde{a}_2$ и $\tilde{a}_3$ в этом случае не пригодны, так как требуют знания $\tilde{m}_X$ и $\tilde{m}_Y$, которые в свою очередь вычисляются по завершении моделирования. Можно использовать формулы пошагового вычисления оценки через ее предыдущее значение. Обозначим через $\tilde{m}_X(n)=\sum_{i=1}^{n} x_i/n$ оценку, вычисленную по данным n первых шагов. Преобразуем формулу $\tilde{m}_X(n)=\sum_{i=1}^{n} x_i/n=(n-1)\cdot\sum_{i=1}^{n-1} x_i/(n\cdot(n-1))+x_n/n=(n-1)\cdot\tilde{m}_X(n-1)/n+x_n/n$ к виду, пригодному для вычисления скользящего значения математического ожидания и применяемому на каждом такте моделирования. Аналогично может быть получена формула вычисления скользящего значения дисперсии: $\tilde{D}_X(n)=(n-2)\cdot\tilde{D}_X(n-1)/(n-1)+(x_n-\tilde{m}_X(n-1))^2/n$.

Для построения закона распределения наблюдаемой случайной величины (характеристики) X диапазон ее возможных значений разбивается на интервалы одинаковой длины Δx. В ходе моделирования для каждого зафиксированного значения x_i определяется номер интервала и его значение, фиксирующее число попаданий величины X в интервал, увеличивается на единицу. По завершении моделирования по значениям интервальных счетчиков строится гистограмма.

Если результаты наблюдения характеристики X представляются парами – значение x_i и время τ_i, в течение которого оно наблюдается, то применяется формула для получения средне взвешенного значения X по времени: $\tilde{m}_X(n)=\sum_{i=1}^{n}(x_i\cdot\tau_i)/\sum_{i=1}^{n}\tau_i$.

Собранный при моделировании статистический материал может использоваться для выдвижения и проверки различных гипотез. Степень обоснованности гипотез определяется методами математической статистики путем вычисления соответствующих ситуации критериев согласия. Так справедливость гипотезы о том, что гистограмма наблюдаемой характеристики соответствует какому-либо теоретическому закону распределения, проверяется на основе критерия Пирсона. Справедливость гипотезы о том, что несколько статистических выборок принадлежат одной и той же генеральной совокупности, проверяется на основе критерия Смирнова. Справедливость гипотез о равенстве математических ожиданий или дисперсий двух статистических выборок проверяется на основе критериев Стьюдента и Фишера соответственно. Для анализа стационарности процессов используют критерий Вилкоксона.

Если моделируемые процессы и соответствующие характеристики нестационарны, то они описываются законами распределения, задающими их поведение во времени: $F_X(x_1,t_1;x_2,t_2;\ldots;x_n,t_n)=P[X(t_1)<x_1;X(t_2)<x_2;\ldots;X(t_n)<x_n]$. Это означает, что с течением времени закон поведения характеристики $F_X(x,t)=P[X(t)<x]$ меняется, а вместе с ним меняются моменты и их оценки $\tilde{a}_k(t)=\tilde{a}_k(x_1,x_2,\ldots,x_n,t)=\tilde{a}_k(n,t)$. Поэтому нельзя рассчитывать их по одной выборке значений. Соответственно при моделировании проводят k прогонов и получают разные выборки $X_1,X_2,\ldots,X_k$, где $X_i=X_i(t)$. Далее фиксируют значения времени и для каждого временного среза (см. рис.36) вычисляют значение оценки, например как $\tilde{a}_1=\tilde{a}_1(t)=m_X(t)=m_X(t_j)=\sum_{i=1}^{k}X_i(t_j)/k$.

4.3. Выбор длительности статистического эксперимента

Длительность имитационного эксперимента также как и численных методов определяется как эмпирически так и теоретически. Эмпирические подходы базируются на контроле и анализе непосредственно в ходе моделирования тенденций изменения текущих значений средних характеристик. Теоретические методы состоят в прогнозировании необходимой длительности моделирования, исходя из необходимости обеспечения заданной точности. Это сопряжено со сложностями, вызываемыми незнанием реальных законов распределения характеристик и отсутствием другой априорной информации.

Пусть в ходе имитационного эксперимента в качестве характеристики требуется вычислить вероятность моделируемого события. Обозначим реальное значение искомой вероятности через $p(x)$. В качестве ее оценки возьмем $\tilde{p}(x)=\tilde{m}_x(n)=\sum x_i/n$, где x_i - реализация соответствующей случайной величины.

Будем считать, что все величины x_i независимы, одинаково распределены принимают единичное значение, если событие произошло, и нулевое в противном случае. Поскольку закон их распределения биномиальный, то дисперсия $D_x = p(x)(1-p(x))$. В силу центральной предельной теоремы закон распределения самой оценки $\tilde{p}(x)$ близок к нормальному, поэтому, применяя соотношение из Главы 1, § 4.2, получим $\varepsilon_\beta = t_\beta \sigma_{\tilde{m}_x} = t_\beta \cdot \sqrt{D_{\tilde{m}_x}} =$

$$= t_\beta \cdot \sqrt{D\left(\sum_{i=1}^{n} x_i / n\right)} = t_\beta \cdot \sqrt{D\left(\sum_{i=1}^{n} x_i\right) / n^2} = t_\beta \cdot \sqrt{nD(x_i)/n^2} = t_\beta \cdot \sqrt{(p(x)(1-p(x))/n}\,.$$

Отсюда $n = t_\beta^2 p(x)(1-p(x))/\varepsilon_\beta^2$ и нетрудно заметить, что при $\varepsilon_\beta \to 0$ значение $n \to \infty$, а длительность моделирования обратно пропорциональна квадрату погрешности. Поэтому для увеличения точности моделирования в два раза число экспериментов придется учетверить.

Если случайные значения наблюдаемой характеристики C не коррелированы и их распределение не изменяется от прогона к прогону, то выборочное среднее характеристики можно считать нормально распределенным. В этом случае число опытов n, необходимое для того, чтобы истинное среднее m_C с заданной доверительной вероятностью лежало в интервале $\tilde{m}_C \pm \varepsilon_\beta$, определится как $n = t_\beta^2 \cdot D_C / \varepsilon_\beta^2$, где D_C - дисперсия характеристики. Если D_C до начала эксперимента неизвестно, то выполняют пробную серию опытов и вычисляют ее оценку.

5. Метод Монте-Карло. Основные принципы

Метод Монте-Карло основан на замене исходного объекта независимо от его природы случайными процессами, чьи характеристики (например, m_X, D_X) совпадают с характеристиками объекта. Метод используется как для моделирования вероятностных так и детерминированных объектов (например, для решения моделей, описанных дифференциальными уравнениями в частных производных или n-кратными интегралами). Широко применяется для генерации случайных объектов (событий, процессов) с заданными вероятностными характеристиками, необходимых для организации имитационного моделирования.

Достоинства метода: - инвариантность к объекту исследования, однотипность схемы организации моделирования и соответственно универсальность применения; - сравнительно низкая трудоёмкость, так как сами вычисления, как правило, однотипны и относительно несложны, а их объёмы зависят от точности почти линейно.

Общая схема применения метода включает: подбор и замену объекта адекватной вероятностной схемой (моделью), характеристики которой совпадают с вычисляемыми характеристиками объекта; выполнение вычислений по схеме необходимое количество раз и накопление статистических данных; выполнение статистической обработки результатов и их оценки.

Пример 1 иллюстрирует применение метода для реализации простейшего умножителя двух чисел, например, A и B. Пусть их значения принадлежат диапазону [0;1], например, $A = 0{,}3; B = 0{,}7$. Результат умножения $Y = AB$. Если числа промасштабированы в диапазоне [0;1], то их можно трактовать как вероятности соответствующих независимых событий A и B. Тогда результат Y можно рассматривать как вероятность соответствующего события $P(Y) = P(A \cdot B) = P(A)P(B) = AB$. Вероятностная схема умножителя представлена на рис.21. Здесь блоки 1,2 – независимые генераторы чисел ω_i и ω_{i+1}, равномерно распределенных в диапазоне [0;1]. Числа необходимы для имитации указанных событий A и B с соответствующими вероятностями $p(A) = A = 0{,}3$ (блок 3) и $p(B) = B = 0{,}7$ (блок 4). Сами события представляются в схеме тактируемыми импульсами с частотой 0,3 и 0,7. Умножитель реализован блоком 5 - это схема И с частотой результата, равной $P(A \cdot B)$. Блок 6 служит для вычисления частоты выходного потока импульсов, дающей значение результата умножения.

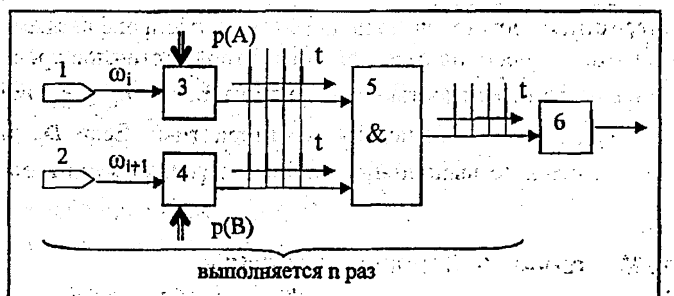


Рис.22. Вероятностная схема умножителя

Пример 2 является классическим примером моделирования детерминированного объекта, который описывается интегралом. Пусть интеграл не берется аналитически. Для простоты полагаем, что интегрируемая функция может быть вписана в единичный квадрат, а интеграл равен $\int f(x)dx$. Идея подхода состоит в случайном, независимом и равномерном выборе точек внутри этого квадрата. Соответственно значения координат x и y подчиняются равномерному закону распределения в диапазоне [0;1]. При каждом выборе точки отмечается событие – точка попала в часть площади квадрата, ограниченную сверху функцией, снизу осью X. Значение интеграла численно равно отношению m/n, где n – общее число точек, а m – число точек, попавших в указанную площадь, ограниченную сверху функцией. Адекватная вероятностная схема представлена на рис.23, где блоки 1,2 – генераторы чисел, равномерно распределенных в диапазоне [0;1] ($x_i = \omega_i$). Блок 3 реализует функцию $f(x_i) = f(\omega_i)$; блок 4 определяет, произошло ли событие

$$h_i = \begin{cases} 1, если \omega_{i+1} \le f(\omega_i) \\ 0, если \omega_{i+1} > f(\omega_i) \end{cases},$$

блок 5 определяет число таких событий $m = \sum h_i$ и блок 6 вычисляет результат $F = m/n$.

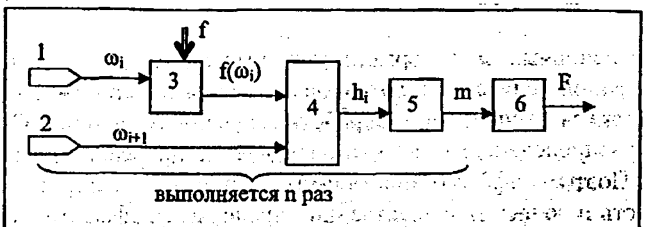


Рис.23. Вероятностная схема вычисления интеграла

Пример 3 модели Монте-Карло для вероятностного объекта – специализированного вычислителя, реализующего функцию $y = \sqrt{x + \varphi^2}$, где значения ее аргументов x и φ - случайные величины с заданными законами распределения f_x, f_φ. Вероятностная схема представлена на рис.24, где блоки 1,2 – независимые генераторы чисел ω_i и ω_{i+1}, равномерно распределенных в диапазоне [0;1]. Они необходимых для имитации пар случайных величин φ_i (блок 3) и x_i (блок 4) в соответствии с заданными законами распределения f_x, и f_φ. Блок 5 вычисляет значение φ^2, блок 6 реализует функцию объекта, а блок 7 определяет результат как оценку математического ожидания y.

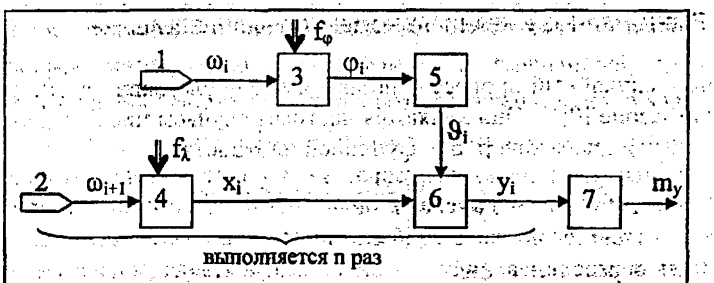


Рис.24. Вероятностная схема специализированного устройства

В приведенных примерах все блоки схемы (кроме последнего) выполняются многократно, пока не будет накоплен необходимый статистический мате-

риал. Последний блок выполняет статистическую обработку данных и вычисляет результаты.

6. Теоретические основы моделирования случайных объектов

6.1. Формирование случайных чисел с равномерным распределением

Одна из основных задач организации имитационного моделирования генерация (имитация) случайных объектов с заданными вероятностными свойствами. Она сводится к генерации и обработке случайных величин (СВ) - чисел с адекватными распределениями, на что расходуется значительная доля машинного времени. Поэтому эффективность имитационного моделирования в целом, его длительность и точность существенно определяется эффективностью работы генераторов случайных величин.

Применяют три основных подхода к генерации СВ: аппаратные (физические), табличные (файловые) и программные (алгоритмические). При алгоритмическом подходе в качестве источника для генерации СВ используют стандартные или базовые случайные процессы - чаще всего числа ω_i, равномерно распределенные в диапазоне [0;1]. Они подвергаются необходимому функциональному преобразованию в соответствии с имитируемыми законами распределения. В аппаратном подходе для генерации случайных чисел используют шумы различных физических объектов (электронных или полупроводниковых устройств, явления радиоактивного распада и т.д.). При табличном подходе используют результаты программной или аппаратной генерации чисел. Ранее сгенерированные и накопленные значения случайных величин образуют готовую к использованию базу данных. Сам базовый процесс также может быть создан программно или аппаратно.

6.2. Равномерное и квазиравномерное распределение

Базовый случайный процесс - независимые числа, равномерно распределенные в диапазоне [0;1]. Они являются частным случаем чисел X, равномерно распределенных в диапазоне [a;b] с функцией плотности

$$f_X(x) = \begin{cases} \frac{1}{b-a}, & \text{если } a \le x \le b; \\ 0, & \text{в остальных случаях,} \end{cases}$$

и функцией распределения

$$F_X(x) = \int_0^x f_X(t)dt = \begin{cases} 1, & x > b; \\ \frac{x-a}{b-a}, & b \ge x \ge a; \\ 0, & x < a. \end{cases}$$

Значения моментов $m_x = (a+b)/2$ и $D_X = (b-a)^2/12$. Для диапазона [0;1] (a = 0 и b = 1)

$$f_X(x) = \begin{cases} 1, a \le x \le b \\ 0, \text{в остальных случаях} \end{cases}$$

$$F_X(x) = \begin{cases} 1, x > b \\ x, b \ge x \ge a \\ 0, \text{x < a} \end{cases}$$

и $m_x = 0{,}5$, $D_x = 1/12$.

Если значения величины X вычислять на цифровой аппаратуре, которая обладает ограниченной n–разрядной сеткой, то это приведет к естественной ошибке перехода из непрерывной области в дискретную, а полученные числа будут обладать квазиравномерным распределением. Сравнительные характеристики идеальных и квазиравномерных СВ приведены в табл. ниже

Характеристика	Равномерно распределенные числа в диапазоне [0, 1]	Квазиравномерные числа в диапазоне [0, 1]
Диапазон	0÷1	0÷1
Количество чисел	∞	2^n
Значения	непрерывные в диапазоне 0+1	*дискретные* $\omega_i = i/(2^n - 1), i = \overline{0,2^n - 1}$
Вероятности значений	0	$1/2^n$
Среднее значение	0,5	0,5
Дисперсия	1/12	$(2^n + 1)/12 \cdot (2^n - 1)$

Значение n-разрядного квазиравномерного числа ω_i можно представить в виде двоичной комбинации $\omega_i = 0, \varepsilon_1 \varepsilon_2 \varepsilon_3 \ldots \varepsilon_n$, где ε_j - двоичный разряд числа. Покажем, что если каждый ε_j является случайной независимой величиной, распределенной равномерно, то каждое ω_i является реализацией случайной квазиравномерной величины. Действительно так как $p(\varepsilon_j = 1) = p(\varepsilon_j = 0) = 1/2$, то $p(\omega_i) = p[\varepsilon_1 = \varepsilon_{1k}; \varepsilon_2 = \varepsilon_{2k} \ldots] = p(\varepsilon_1 = \varepsilon_{1k}) p(\varepsilon_2 = \varepsilon_{2k}) \ldots = 1/2^n$. Соответственно $p(\omega_1) = p(\omega_2) = \ldots = p(\omega_n) = Const = 1/2^n$, т.е. числа подчиняются квазиравномерному распределению, а указанные допущения можно использовать для их генерации.

6.3. Аппаратный подход

Пример схемы аппаратной генерации представлен на рис. 25, где генераторы Г1-Гn имитируют в каждом такте i одно из двух значений ε_r^i (0 или 1), распределенное по равномерному закону. Из них как из двоичных разрядов составляется квазиравномерная случайная величина ω_i.

Основу каждого генератора составляет источник "белого шума" Y и значения ε определяют по схеме, показанной на том же рисунке. На каждом периоде наблюдения T определяют значение случайной величины X – число раз, когда значения Y превышают пороговое значение шума V, и определяют

$$\varepsilon = \begin{cases} 0, & \text{если } x \text{ - чётное;} \\ 1, & \text{если } x \text{ - нечётное.} \end{cases}$$

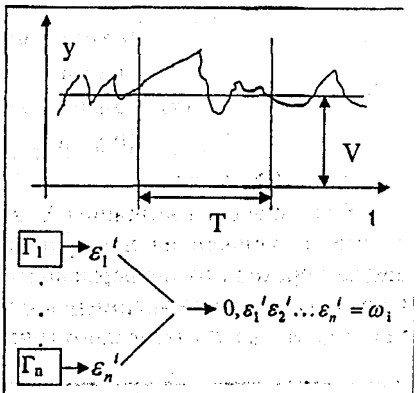


Рис.25. Схема аппаратной генерации чисел

А для настройки генератора используют значения V и T, которые подбирают таким образом, чтобы ε_j' обладало свойствами равномерности, т.е. $p(\varepsilon_j = 1) = p(\varepsilon_j = 0) = 1/2$.

Другая схема генерации, которая может быть построена на базе тех же величин Y, V и T состоит в следующем. На каждом периоде наблюдения T определяются значения ω'_i – длины отрезков, образованные на оси V соседними точками пересечения функции Y и *линии* V. Сами квазиравномерные числа получают их приведением к длине T, т.е. $\omega_i = \omega'_i / T$. Настройка генератора производится с помощью тех же величин.

6.4. Алгоритмические методы

Алгоритмические методы ориентированы на программную реализацию и строятся на базе рекуррентных соотношений первого порядка $X_{i+1} = \Phi(X_i)$, позволяющих получать каждое следующее случайное равномерно распределенное число X_{i+1} через значение предыдущего числа. В зависимости от вида Φ выделяют мультипликативный метод, где $X_{i+1} = (\lambda X_i) \bmod M$ и смешанный, где $X_{i+1} = (\lambda X_i + \mu) \bmod M$. Последняя формула может быть выражена через начальное значение X_0 как $X_i = (\lambda^i X_0 + \mu(\lambda^i - 1)/(\lambda - 1)) \bmod M$. Здесь λ и μ - коэффициенты, выбор которых основывается как на теоретических исследованиях так и на результатах, полученных экспериментально. А значение $M = 2^n$, где n – разрядность цифровой сетки ЭВМ, на которой генерируются числа. Поэтому значения X_i равномерно распределены в диапазоне $[0; M-1]$, а максимальная длина периода получаемой последовательности случайных чисел не превышает значения M (после чего она повторяется). Соответственно для приведения чисел к диапазону $[0; 1]$ полученные значения масштабируют - $\omega_i = x_i / M$.

Исчерпывающие рекомендации по выбору начальных значений параметров генератора приведены в [2]. Отметим только, что X_0, λ и μ - целые положительные числа; выражение $\lambda \cdot X_i$ необходимо вычислять точно, без округ-

ения; коэффициент λ должен удовлетворять соотношениям $\lambda \bmod 8 = 5$ и $\sqrt{M} \le \lambda \le M - \sqrt{M}$; коэффициент μ должен быть нечётным и не кратным 5 и приближенно удовлетворять равенству $\mu / M \approx 0.5 - \sqrt{3/6}$.

Пример генерации чисел для $n=4$, $X_0 = 7$ (0111) и $\lambda = 5$ (0101) описан ниже. Так как $M = 2^n = 16$, то первом шаге вычисления получим $\lambda \cdot X_0 = 35 \rightarrow$ 00100011, четыре младших разряда результата составят значение $X_1 = (\lambda X_0) \bmod M = 35/16 = 3 \rightarrow 0011$ и первое сгенерированное число $v_1 = x_1 / M = 3/16 = 0{,}1875$ и т.д.

Мультипликативный метод менее трудоемок из-за отсутствия операции ложения, но качество генерируемой последовательности хуже, т.к. между числами может быть корреляционная зависимость.

6.5. Требования к генераторам и сравнение подходов

К идеальному генератору предъявляют следующие требования: - числа олжны быть квазиравномерными и независимыми; генерируемая последовательность должна быть воспроизводимой; комбинации чисел не должны повторяться; затраты компьютерных ресурсов, стоимость и трудоёмкость генерации олжна быть минимальной. Сравнительная оценка ранее рассмотренных методов приведена в таблице ниже.

Метод	Достоинства	Недостатки
Аппаратный	▪Быстродействие ▪Неограниченность генерируемых последовательностей ▪Незначительное потребление ресурсов	▪Высокая стоимость ▪Необходимость периодической проверки качества генерируемой последовательности ▪Невозможность воспроизведения последовательностей
Программный	▪Низкая стоимость ▪Необходимость толь- ко однократной верификации ▪Возможность воспроизведения последовательностей ▪Незначительное потребление памяти	▪Запас чисел и период генерируемой последовательности ограничен ▪Большие затраты времени
аблич-ный	▪Минимальные затраты времени ▪Необходимость однократной проверки ▪Возможность воспроизведения последовательностей	▪Запас чисел ограничен ▪Большее потребление памяти

6.6. Проверка качества генерируемой последовательности

Квазиравномерные числа, полученные тем или иным методом, должны ыть проверены на соответствие предъявляемым требованиям. Для алгоритми-

ческих методов проверка выполняется однократно, а для аппаратных производиться периодически, так как сами источники "белого шума" могут функционировать нестабильно и менять свои характеристики с течением времени. Генерируемая последовательность чисел проверяется на равномерность, стохастичность и независимость.

При анализе равномерности на основе статистических методов оценивают степень расхождения эмпирического закона распределения генерируемой последовательности чисел (гистограммы) и эталонного, теоретического закона, в качестве которого берется равномерный. При этом оценивают правдоподобность гипотезы о том, что расхождение эмпирического и теоретического законов обусловлено случайными факторами, недостаточной представительностью анализируемой выборки. В качестве критерия согласия может использоваться критерий χ^2.

При анализе стохастичности оценивают характер распределения нулей и единиц в двоичных последовательностях, соответствующих сгенерированным числам. Для этого анализируют закон распределения длин участков между соседними единицами или нулями в числе либо закон распределения длин серий единиц или нулей, стоящих в числе рядом. Так как для чисел, распределенных по равномерному закону, вероятность появления единицы (нуля) в каждом двоичном разряде неизменна и равна 0,5, то эталонным, теоретическим законом распределения длин серий служит соответствующий биномиальный закон. Проверяемые гипотезы и используемые критерии аналогичны предыдущим.

При анализе независимости исследуется независимость двоичных разрядов сгенерированных квазиравномерных чисел. В частности, выполняется корреляционный анализ соседних разрядов.

6.7. Моделирование случайных объектов

При имитации в качестве случайных объектов чаще всего используют события и группы событий (зависимые и независимые), величины (дискретные и непрерывные), системы случайных величин и процессы.

Моделирование случайного события. Пусть требуется имитировать случайное событие X, происходящее с вероятностью $P[X]=p$. Заменим событие X новым событием $Z=\omega_i<p$, состоящим в том, что случайное равномерно распределенное в диапазоне [0;1] число $\omega_i<p$ и покажем, что они равносильны, т.е. их вероятности совпадают. Действительно, вероятность события Z (с учетом того, что $f_\Omega(\omega)=1$) $P(z)=P(\omega<p)=\int_0^p f_\Omega(t)dt=p$. Это означает, что вместо значений X можно генерировать значения Z по следующему алгоритму: - генерируют равномерно распределенное число ω_i; - сравнивают ω_i со значением p; - если $\omega_i<p$, то отмечают, что событие произошло, в противном случае – нет, соответственно получают одну реализацию события. Схема иллюстрируется на рис.26. Здесь на отрезке единичной длины, например, от его начала откладывается отрезок длины p. Затем берется случайное число ω_i и "набрасыва-

ется на единичный отрезок. Считается, что событие произошло, если ω_i попадает на отрезок длины p.

Моделирование группы независимых элементарных событий. Пусть требуется имитировать случайные события $X_1, X_2, ..., X_n$, образующие полную группу событий $\Omega = \{X_i\}$ и происходящие соответственно с вероятностями $p_1, p_2, ..., p_n$. Сумма $\sum_{i=1}^{n} p_i = 1$. Аналогично предыдущему методу каждое X_r заменяется соответствующим равносильным $Z_r = \omega_i < p_r$. Схема генерации иллюстрируется рис.27, где на отрезке единичной длины, например, от его начала последовательно откладываются отрезки длиной $p_1, p_2, ..., p_n$. Затем берется случайное число ω_i и "набрасывается" на единичный отрезок. Считается, что произошло событие X_r, если ω_i попадает на r-ый отрезок, т.е. если $\sum_{i=1}^{r-1} p_i < \omega_i < \sum_{i=1}^{r-1} p_i + p_r$.

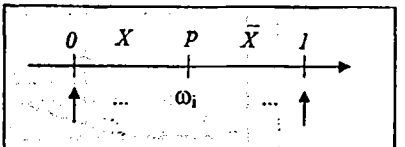


Рис.26. Схема генерации события с заданной вероятностью p

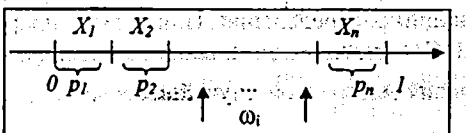


Рис.27. Схема генерации событий,образующих полную группу

Моделирование системы произвольных событий. Пусть требуется имитировать систему событий A и B с заданными вероятностями $P(A)$, $P(B)$ и $P(B|A)$. Указанные события приводят к полной группе событий $X_1 = A \cdot B$, $X_2 = A \cdot \bar{B}$, $X_3 = \bar{A} \cdot B$ и $X_4 = \bar{A} \cdot \bar{B}$. Их вероятности могут быть рассчитаны по формуле $P(AB) = P(A) \cdot P(B|A) = P(B) \cdot P(A|B)$. Тогда $P(X_1) = P(A) \cdot P(B|A)$, $P(X_2) = P(A) \cdot P(\bar{B}|A) = P(A) \cdot (1 - P(B|A))$, $P(X_3) = P(B) \cdot P(\bar{A}|B)$, $P(X_4) = 1 - \sum_{i=1}^{3} P(X_i)$ соответственно и задача сводится к предыдущей для случая $n = 4$. Если события независимы (т.е. $P(B) = P(B|A)$), то их вероятности примут значения $P(A) \cdot P(B)$, $P(A) \cdot P(\bar{B})$, $P(\bar{A}) \cdot P(B)$ и $P(\bar{A}) \cdot P(\bar{B})$ соответственно. И задача также сводится к предыдущей.

$$P(A/B) = \frac{P(A) \cdot P(B/A)}{P(B)}$$

Моделирование случайных величин. Пусть требуется имитировать значения случайной величины X, для которой задан закон распределения, например, в виде функции распределения $F_X(x) = P(X < x)$. Если величина непрерывная и функция распределения задана аналитически, в виде формулы, то можно применить метод обратной функции $F_X^{-1}(\omega)$, позволяющей через ее значение ω определить значение аргумента, т.е. $F_X(x) = \omega \Rightarrow x = F_X^{-1}(\omega)$. Если бу-

дет получено аналитическое выражение обратной функции, то алгоритм генерации (рис.28) состоит в следующем: генерируют значение ω_i и рассчитывают очередное значение x_i по формуле $x_i = F_x^{-1}(\omega_i)$. Например, для равномерно распределенных величин из функции распределения по формуле $(x-a)/(b-a) = \omega$ получают выражение $\Rightarrow x_i = \omega_i(b-a)+a$ для аналитического расчета значений x_i.

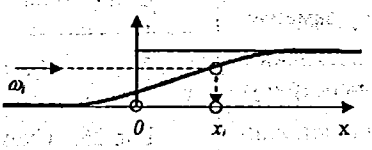


Рис.28. Генерация чисел методом обратных функций

Если же решить уравнение не удается или величина является дискретной, то для применения этого метода используется график (табличные значения) функции распределения. При этом непрерывная величина сводится к дискретной. Схема генерации иллюстрируется на рис.29, где показана функция распределения дискретной случайной величины ($\sum_{i=1}^{n} p_i = 1$). Действительно, значениям величины X - $x_1, x_2, ..., x_n$ можно поставить в соответствие вероятности $p_1, p_2, ..., p_n$, рассчитанные по значениям функции распределения как $p_k = F_x(x_k) - F_x(x_{k-1})$. Указанные значения X образуют полную группу событий $X = x_1, ..., X = x_n$, а задача генерации сводится к задаче моделирования полной группы независимых элементарных событий и графически (рис.29) означает "набрасывание" случайного числа ω_i на отрезок единичной длины по оси *0-Y*.

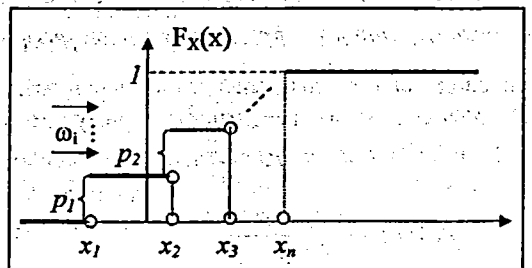


Рис.29. Генерация чисел методом обратных функций

Если закон распределения случайной величины X задан (рис.30) в виде функции плотности $f_X(x) = F'_X(x)$, то для имитации ее значений может быть применен метод, описанный в [1]. Интервал *[a; b]* допустимых значений функции разбивается на n отрезков разной длины, таким образом, чтобы все отрезки были равновероятны при случайном выборе. Так как вероятность выбора $=1/n$, то длину любого k-го отрезка определяют из уравнения $\int_{a_k}^{a_{k+1}} f_X(x)dx = 1/n$.

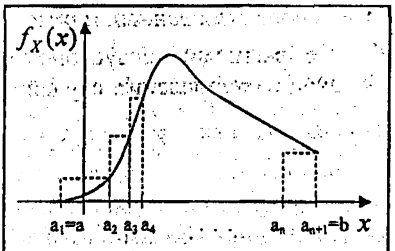


Рис.30. Генерация случайных чисел по заданной $f_X(x)$

Если величина непрерывная, то она сводится к дискретной и для выбранных отрезков выполняется ступенчатая аппроксимация функции плотности (рис.30). События, состоящие в случайном выборе одного из отрезков, образуют полную группу из n событий (каждое происходит с вероятностью $1/n$). Схема генерации включает два этапа: - имитируют выбор отрезка (для этого применяется подход, рассмотренный ранее); - на выбранном k-ом отрезке (a_k, a_{k+1}) определяют искомое значение x_i, сделав, например, допущение о том, что его распределение внутри отрезка подчиняется равномерному закону. Соответствующий алгоритм включает следующие шаги: а) генерируют ω_i и выбирают отрезок (например, с номером k); б) генерируют ω_{i+1} и определяют (масштабируют) значение $x_i = \omega_{i+1}(a_{k+1} - a_k) + a_k$.

Для многих законов распределения случайных величин существуют специальные алгоритмы генерации их значений, основанные на свойствах и особенностях этих распределений. Качество таких алгоритмов, как правило, выше качества универсальных.

Моделирование систем случайных величин. Пусть требуется имитировать значения системы случайных величин (X,Y), заданной законом распределения. Для дискретных величин это может быть таблица значений $p_{ij} = P[X = x_i; Y = y_j]$,

$x \backslash y$	y_1	...	y_j	...	y_k
x_1	p_{11}	...	p_{1j}	...	p_{1k}
...	...	...	...	...	...
x_i	p_{i1}	...	p_{ij}	...	p_{ik}
...	...	...	...	...	...
x_n	p_{n1}	...	p_{nj}	...	p_{nk}

где случайные величины принимают значения из множеств $\{x_i \mid i=\overline{1,n}\}, \{y_j \mid j=\overline{1,k}\}$. На основе этих данных можно определить безусловный закон распределения P^*_X, т.е. вычислить безусловные вероятности любой из возможных величин. Так закон распределения случайной величины X (вероятности ее значений, вычисляемые как $p_{Xi} = P[X=x_i] = \sum_{j=1}^{k} p_{ij}$) представлен в таблице ниже

x	x_1	...	x_r	...	x_n
p_{Xi}	p_1	...	p_r	...	p_n

Аналогично, фиксируя выбранное значение x_i, можно определить условный закон распределения $P^*_{Y|x}$ величины Y, т.е. вычислить безусловные вероятности $p_{Y|x}$

$y \mid x_i$	y_1	...	y_r	...	y_k
$p_{Y\mid x}$	p_{i1}	...	p_{ir}	...	p_{ik}

Тогда алгоритм генерации этой системы величин может базироваться на моделировании значений двух отдельных случайных величин, описываемых законом P^*_X и $P^*_{Y|x}$ соответственно. В каждом i-ом такте: - генерируют пару чисел ω_i и ψ_i, равномерно распределенных в диапазоне 0-1; - по значению ω_i и закону P^*_X определяют $x_{i,r} = x_r$; - по значению ψ_i для случая $x=x_r$ и закону $P^*_{Y|x}$ определяют значение $y_{i,l} = y_l$. Процедура повторяется необходимое число раз.

Если задана система непрерывных случайных величин $\langle X,Y \rangle$ с известной совместной функцией плотности $f_{XY}(x,y)$ или функцией распределения $F_{XY}(x,y)$, то определяют безусловную функцию плотности $f_X(x) = \int_{-\infty}^{\infty} f_{XY}(x,y)dy$ величины X, а для величины Y из соотношения $f_{XY}(x,y) = f_X(x) \cdot f_Y(y|x)$ определяют условную функцию плотности $f_Y(y|x_i) = f_{XY}(x_i,y)/f_X(x_i)$. Алгоритм генерации такой системы величин базируется на моделировании значений двух отдельных случайных величин, описываемых функциями плотности $f_X(x)$ и $f_Y(y|x)$ соответственно. Для этого может быть использован выше описанный метод моделирования СВ по ее функции плотности. В каждом i-ом такте: - ге-

нерируют значение $f_X(x) \Rightarrow x_i$; - генерируют значение $f_Y(y|x_i) = f_{XY}(x,y)/f_X(x) \Rightarrow y_i$.

Моделирование марковских процессов, заданных множеством возможных состояний $S = \{S_i \mid i = \overline{1,n}\}$, вектором распределения вероятностей начальных состояний $\vec{p}(0) = (p_1(0); p_2(0); ...; p_n(0))$ ($p_i(t) = P[S(t) = s_i]$) и матрицей P вероятностей перехода из состояния i в состояние j

$$P = [p_{ij}] = \begin{Vmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{Vmatrix}$$

Для вектора и матрицы справедливы соотношения $\sum_{i=1}^{n} p_i(t) = 1$ и $\sum_{j=1}^{n} p_{ij} = 1$.

Схема моделирования основывается на многократной генерации события, состоящего в выборе номера j нового состояния процесса при известном номере i текущего состояния. Эти события, соответствующие всем возможным состояниям процесса, образуют полную группу событий. Их вероятности для i-го текущего состояния заданы значениями i-ой строки матрицы P. А для имитации начального состояния используют вектор $\vec{p}(0)$. Таким образом задача сводится к генерации события из полной группы событий, а алгоритм включает следующие шаги: - генерируют ω_i и по значениям вектора $\vec{p}(0)$ "разыгрывают" номер начального состояния $S_k(0)$ (например, $S_3(0)$); - генерируют ω_{i+1} и по значениям вектора $\vec{p_k}$, образованного соответствующей k-ой строкой матрицы P, "разыгрывают" новое состояние $S_i(1)$ (например, по третьей строке матрицы P - вектору $\vec{p_3} = (p_{3,1}; p_{3,2}; ...; p_{3,n})$ определяют $S_i(1) = S_7(1)$). Последний пункт повторяется многократно (например, далее по 7 строке - вектору $\vec{p_7} = (p_{7,1}; p_{7,2}; ...; p_{7,n})$ определяют состояние $S_i(2)$ и т.д.).

7. Имитационные модели: состав, структура, алгоритмы

7.1. Компоненты, функциональные действия, активности, события

При моделировании объект (систему) можно рассматривать как набор компонентов (элементов), подсистем, взаимодействующих друг с другом в определённой среде. Они могут находиться в различных состояниях, например, в активном состоянии или в состоянии ожидания. Активное состояние соответствует некоторой деятельности в системе, процессу, отображаемому в модели выполнением соответствующего алгоритма. Изменение состояний системы происходит под влиянием событий. События в исследуемой системе наступают, как правило, в непредсказуемые моменты физического времени, происходят мгновенно в системном времени и требуют конечных затратат машин-

ного времени. Системное, модельное время необходимо, в частности, для синхронизации происходящих в модели событий и процессов. В отличие от непрерывного физического времени системное время - дискретное.

В системах с параллельно функционирующими элементами, подсистемами может иметь место одновременное выполнение ряда процессов. Соответственно в моделях таких систем в некоторые моменты системного времени в активном состоянии оказывается одновременно несколько объектов, выполняющих свои функции одновременно. Параллельные процессы могут быть асинхронными и синхронными, подчиненными и независимыми.

Поведение каждого компонента (рис.31) аппроксимируется последовательностью отрабатываемых им функций - *функциональных действий* ФДij. Функциональное действие характеризуется алгоритмом АЛij и временем его выполнения τi2, образующими активность, а также набором событий Cij, к которым приводит его выполнение. *Активность*, работа - это единичное действие системы по обработке, преобразованию входных данных (информационных или какие-либо материальных ресурсов). Результаты обработки событий учитываются компонентами при запуске и выполнении очередного функционального действия.

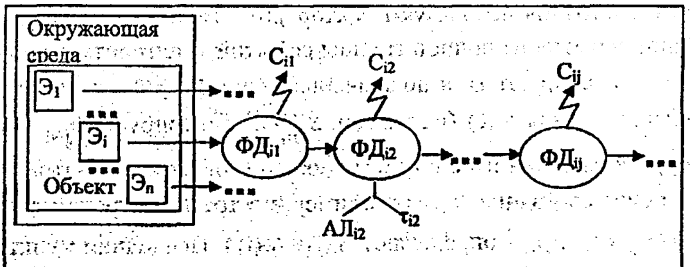


Рис.31. Объект моделирования как набор функциональных действий

Условно процесс функционирования *i*-го компонента показан в виде графика на рис.32, где по оси *X* откладываются значения модельного времени t_M, а по оси *Y* выполняемые действия. При переносе в имитационную модель реальные функциональные действия $ФД_{ij}$ с некоторой погрешностью заменяются модельными $ФД^*_{ij}$, а сам процесс моделирования компонента включает следующие многократно повторяющиеся такты: - для текущего модельного времени $t_M = tj$ имитируется функциональное действие $ФД^*_{ij}$; - модельное время увеличивается на значение $\tau_{i,j}$; - прогнозируются события $C_{i,j}$.

Кроме того, здесь события обрабатываются не тогда, когда происходят, что может приводить к ошибкам моделирования. Заниженный шаг приводит к росту пустых тактов, завышенный – к искажению реальной картины развития процессов в объекте. Подбором величины Δt можно сохранить приемлемую точность, сократить число тактов по сравнению с событийным способом и следовательно уменьшить накладные расходы и трудоемкость моделирования. Шаг проще подобрать, если известно распределение событий во времени. Метод приращения целесообразен при большом числе событий, при их тенденции к группированию. Реализация метода относительно проста. В событийный способе пустых тактов нет, модули обработки событий активизируются только тогда, когда они реально происходят.

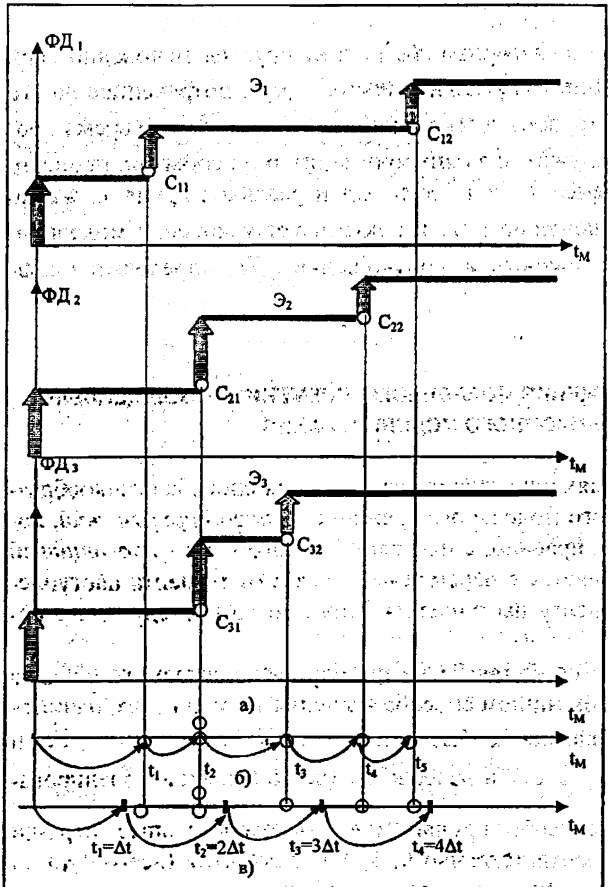


Рис.33. Пример вариантов продвижения модельного времени: а) функциональные действия; б) событийный способ; в) способ Δt

Обобщённые алгоритмы моделирования стационарных объектов представлены на рис.34, 35. Здесь блоки 1-3 служат для подготовки модели к моделированию, в том числе для ввода исходных данных, генерации и инициализации модели. Блоки 4,6,7 (рис.34) и блоки 4,5,7,8 (рис.35) отрабатывают очередной такт (итерацию) имитационного моделирования. Для нестационарных объектов при каждом эксперименте надо выполнять несколько прогонов моделирования, а результаты усреднять по их временным срезам (рис.36). Соответствующий алгоритм приведен на рис.37, где составляющие инициализации модели представлены более подробно блоками 2-4, а блок 5 представляется в соответствующей редакции в зависимости от принятого способа продвижения мо-

дельного времени.

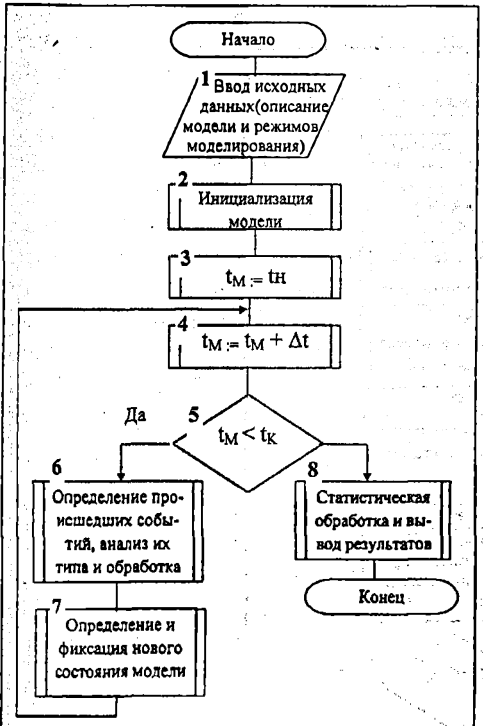


Рис. 34. Алгоритм моделирования на основе постоянного приращения модельного времени

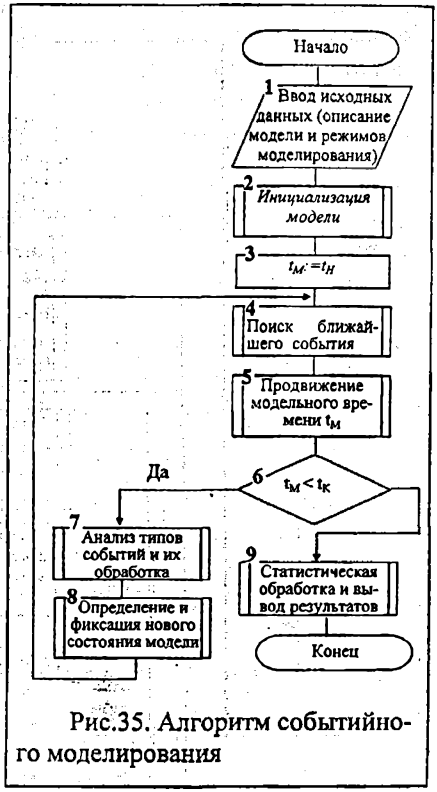


Рис.35. Алгоритм событийного моделирования

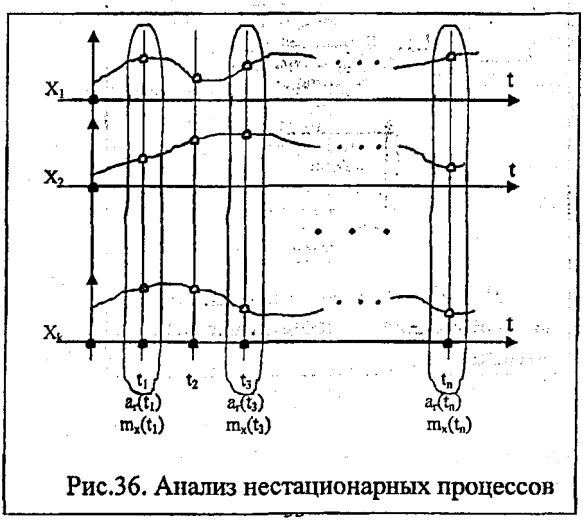


Рис.36. Анализ нестационарных процессов

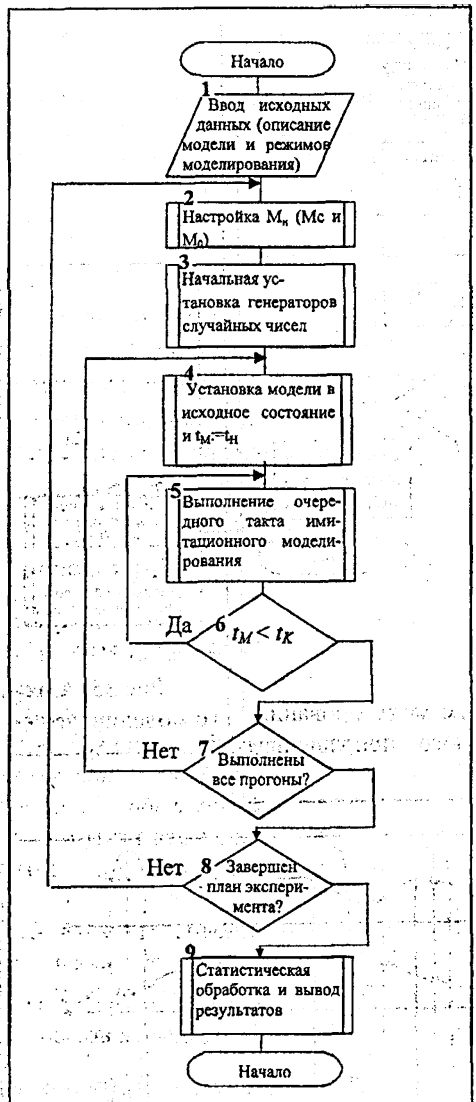


Рис.37. Алгоритм имитационного моделирования нестационарных объектов

7.3. Типовая структура имитационной модели

Структура типовой имитационной модели представлена на рис.38. Она включает подсистему моделей ПМ и подсистему организации моделирования *ПОМ*, состоящую из модуля пользовательского интерфейса ИП, модуля инициализации модели ИнМ, модуля статистической обработки результатов моделирования СО, модуля ЗЗМ загрузки и запуска модулей подсистемы моделей и др. Подсистема ПМ включает модули модели объекта *Мо*, модели окружающей среды *Мс* и модуль *АУМ*, реализующий алгоритм управления их взаимодействием. Модули *Мо* и *Мс* образуют переменную часть подсистемы моделей *Мн*, настраиваемую на конкретный объект, а *АУМ* ее фиксированную часть *Мф*, в которой "зашит" выбранный способ продвижения модельного времени и реализации псевдопараллельностей.

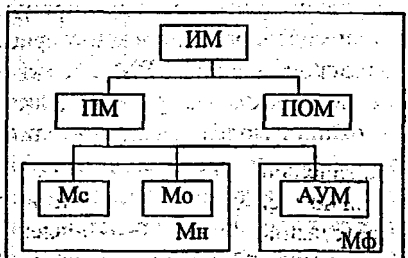


Рис.38. Состав типовой имитационной модели

Характер взаимодействия подсистем и модулей приведен на рис.39, где сплошными линиями показаны управляющие, а двойными линиями информационные связи. Модули Мн взаимодействуют друг с другом через общую информационную базу ИБ, внося в нее изменения и анализируя ее состояние во время выполнения после активизации. Основу ИБ составляет информация, отображающая процесс изменения состояния модели (объекта) и хранящаяся в виде временных списков (цепей) и других информационных структур.

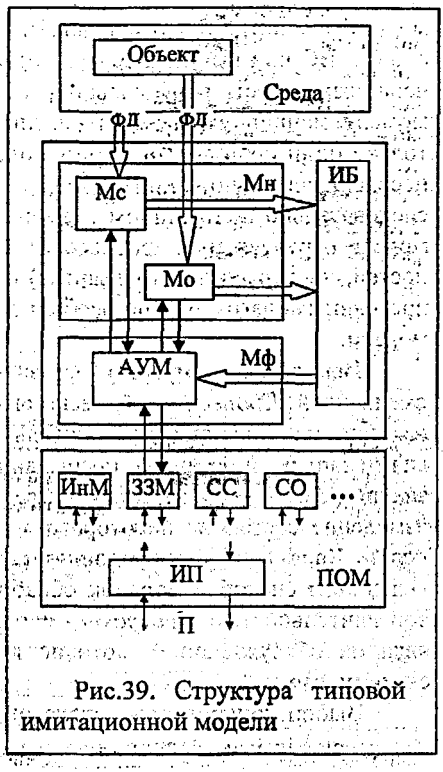


Рис.39. Структура типовой имитационной модели

Пользователь П через модуль ИП вводит исходные данные: - описание объекта в табличной, графической или языковой форме для последующей генерации или настройки модулей Мн; - описание режимов моделирования, включая план эксперимента, число прогонов, длительность моделирования; - требования к результатам моделирования, их полноте, составу, точности. Модуль ИнМ готовит систему к моделированию: - по исходному описанию объекта

при необходимости генерирует и настраивает модуль Мн; - устанавливает модель и соответствующие модули в заданное исходное состояние. Модуль ЗЗМ передает управление АУМ. Модуль АУМ реализует один прогон, имитируя нужное количество тактов моделирования. В каждом такте модуль АУМ: - анализирует состояние информационной базы ИБ; - продвигает модельное время; - определяет порядок запуска модулей Мн в следующем такте и инициирует их выполнение. По завершении моделирования или в его процессе может инициироваться работа и других модулей, в частности СО для подготовки и возвращению пользователю результатов моделирования в определенном им виде.

7.4. Методы реализации псевдопараллельностей

Реализация параллельных процессов на однопроцессорных ЭВМ имеет свои особенности, поскольку на уровне задач такие вычислительные процессы могут быть истинно параллельными только при моделировании в многопроцессорных ВС или вычислительных сетях. В однопроцессорной ЭВМ, где исключена возможность параллельной реализации нескольких алгоритмов, в любой момент машинного времени в активном состоянии в модели может находиться только один объект. Поэтому в моделях параллельные процессы реализуются псевдопараллельно - они протекают параллельно в системном времени, но последовательно в машинном времени. Соответственно способ представления в модели параллельных процессов и их динамики (посредством событий, активностей, процессов или транзактов) и способ изменения модельного времени (с постоянным шагом или по особым состояниям) определяют тип имитационной модели.

Используются методы: активностей, событий, транзактный, процессный и смешанный. *Процесс* - логически связанный набор работ. Характеризуется совокупностью статических (длительностью, потребляемыми ресурсами, условиями запуска и останова, прерывания) и динамических характеристик, описывающих состояние процесса (активен, неактивен и т.д.). *Событие* - мгновенное изменение состояния некоторого элемента системы или состояния системы в целом. Характеризуется условиями (или законом) возникновения, типом, определяющим способ и порядок обработки, нулевой длительностью. *Транзакт* - сообщение (заявка на обслуживание), которое поступает в систему извне.

Выбор конкретного метода определяется характеристиками функциональных действий моделируемого объекта, соответствующая классификация показана на рис.40. Функциональные действия могут быть одинаковыми (1.1) либо разными (1.2). Разные действия отличаются алгоритмами (2.1) и (или) условиями запуска (2.2). Кроме этого, действия могут быть неза-

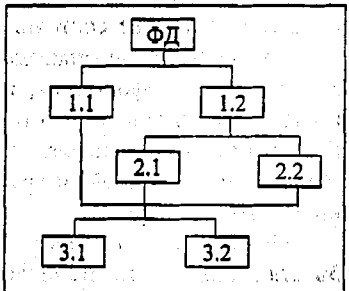


Рис.40. Классификация типов функциональных действий

висимыми (3.1) или зависимыми (3.2), что определяется характером их взаимодействия друг с другом (наличием только информационных связей, либо и связей по управлению). Соответственно на графе классификации могут быть выделены пути, определяющие предпочтительный метод. Методу событий соответствует путь 1.1-3.1; методу активностей путь 1.2-3.1; транзактному путь 1.1-3.2; процессному - 1.2-3.2. Как видно, для простых объектов больше подходит событийный или транзактный метод, а сложные объекты с разнотипными и сильно зависимыми функциональными действиями, в том числе связанными по управлению, моделируются процессным методом.

7.5. Примеры реализации псевдопараллельностей

Описание объекта и концептуальной модели. Объект (пример взят из [7]) представляет собой производственную систему - цех, в котором на 4 станках обрабатываются детали двух типов. Структура цеха и маршруты обработки деталей показаны на рис.41. Предполагается, что параметры деталей и станков известны. Детали появляются в случайные моменты времени и параметры обработки также могут носить случайный характер. Станки могут рассматриваться как отдельные компоненты системы, не связанные друг с другом по управлению.

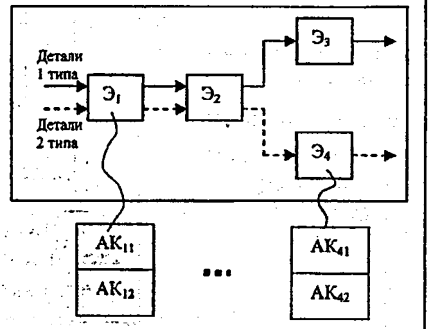


Рис.41. Структура объекта моделирования

Функционирование i-го станка предполагает поочередное выполнение активностей (рис.41, 42): AK_{i1} - занятие станка деталью и начало ее обработки; AK_{i2} – освобождение станка при завершении обработки детали. Выполнение AK_{i1} приводит к прогнозированию в будущем появления события типа C_2 - завершение обработки детали, что может привести к занятию следующего станка. Выполнение активности AK_{i2} приводит к реализации этого события. Функционирование внешней среды, поставляющей детали в цех, может быть представлено выполнением активностей (рис.43) AK_{01} и AK_{02}, представляющих появление деталей соответствующего типа. Они приводят к событию C_1 - приход новой детали в цех. Каждое из указанных событий меняет состояние цеха и требует отработки некоторых действий.

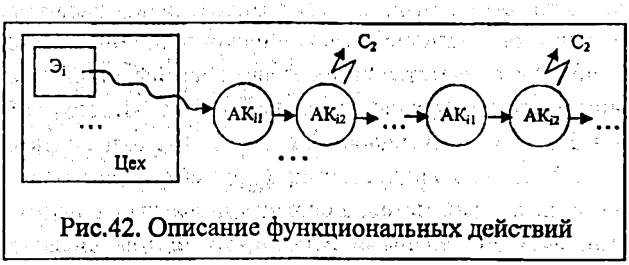


Рис.42. Описание функциональных действий

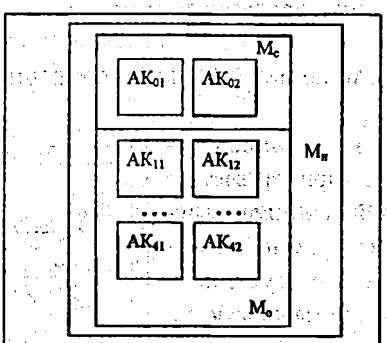


Рис.43. Структура настраиваемой части модели (метод активностей)

Состояние цеха в целом характеризуется распределением деталей по станкам. Детальное описание состояния цеха, необходимое для организации моделирования, может включать: описание состояния каждой i-ой очереди q_i = (α_i – *длина очереди,* {*№ заявки, тип, время поступления в очередь, и др.*}); предполагаемые времена завершения обработки детали на каждом i-ом станке $\{t_{i,3}\}$; признаки активности станков $\pi_i = \begin{cases} 0, \text{станок свободен;} \\ 1, \text{станок занят.} \end{cases}$ $i = \overline{1,4}$. Эти описания вместе со списками событий C_1 и C_2 образуют информационную базу модели (ИБ), показанную на рис.44. В ИБ на каждом такте моделирования вносятся изменения, отображающие смену со-

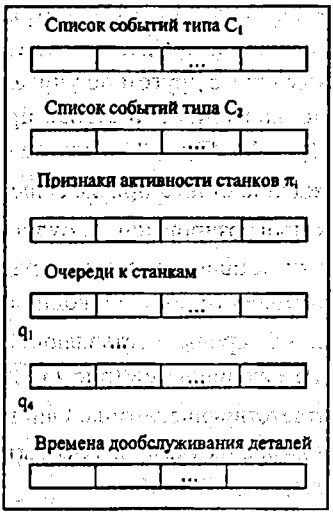


Рис.44. Информационная база модели

стояний объекта. Состояние самой ИБ анализируется модулем управления АУМ и модулями моделей для реализации очередного такта имитационного моделирования.

Метод активностей применяется, если функциональные действия компонентов существенно отличаются и слабо связаны друг с другом. Структура настраиваемой части модели $M_н$ приведена на рис.43 и включает 10 активностей. Это 4 пары AK_{i1}, AK_{i2} и пара AK_{01}, AK_{02}, образующие соответственно модель объекта M_o и модель среды M_c. Структура всей модели приведена на рис.45, где модуль $M_ф$ реализует алгоритм моделирования по методу активностей и определяет порядок запуска модулей активностей. Блок 1 в каждом такте: - анализирует временные списки событий в ИБ и определяет список наступивших событий, которые должны быть обработаны; - если такие события есть, то блок 5 поочередно, пока не будет исчерпан их список, анализирует условия запуска активностей и инициирует выполнение тех модулей, чьи условия запуска "сработали"; - управление передается модулю активности, он выполняется, корректирует состояние ИБ и может создать условия для запуска других активностей; по завершении выполнения все модули активностей возвращают управление в модуль $M_ф$; - если все события обработаны, все активности выполнены, то блок 3 продвигает модельное время t_M; - если прогон завершился, то блок 6 производит статистическую обработку результатов, в противном случае выполняется переход на блок 1 и начинается следующий такт моделирования. Условие запуска AK_{i1} формулируется как $(\alpha_i \neq 0) \& (\pi_i = 0)$, а условие запуска AK_{i2} как $t_{i,э} \geq t_M$.

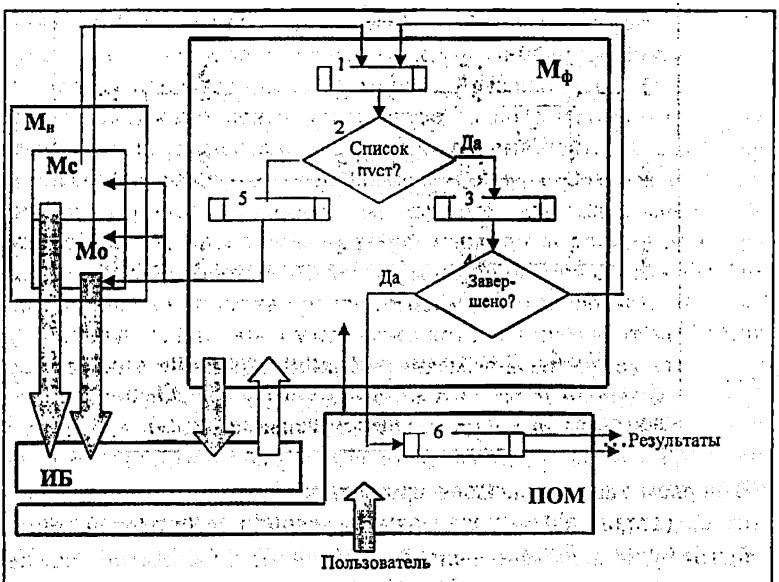


Рис.45. Структура модели (метод активностей). Обозначения: ⇨ - анализ БД; ⟶ - передача управления; ⇨ - воздействие и корректировка ИБ

Состав типовых действий AK_{11}: - фиксация момента начала обработки детали; генерация длительности обслуживания детали τ_i и фиксация момента завершения ее обработки $t_{i,3} := t_M + \tau_i$; - установка признака занятости станка $\pi_i := 1$ и корректировка очереди к нему (выбор детали на обслуживание из i-ой очереди и уменьшение ее длины $\alpha_i := \alpha_i - 1$); - корректировка ИБ и временных списков (внесение в список для момента времени $t_{i,3}$ нового события типа C_2).

Состав типовых действий AK_{12}: - определение дальнейшего маршрута движения детали; - установка признака занятости станка $\pi_i := 0$ и корректировка соответствующей j-ой очереди (помещение в нее детали и увеличение длины $\alpha_j := \alpha_j + 1$); - корректировка ИБ и временных списков (внесение в список для текущего модельного времени t_M нового события типа C_1).

Порядок функционирования модели иллюстрируется рис.46, где временные списки событий двух типов объединены в одну и изменение их состояний показано для трех модельных тактов, не считая начального (1-3). Тип события указан видом значка (символа), а его номер соответствует номеру детали. В исходном состоянии (такт 0) в списке находится 5 будущих событий первого типа C_{11}-C_{15}, соответствующих появлению в цехе новых деталей.

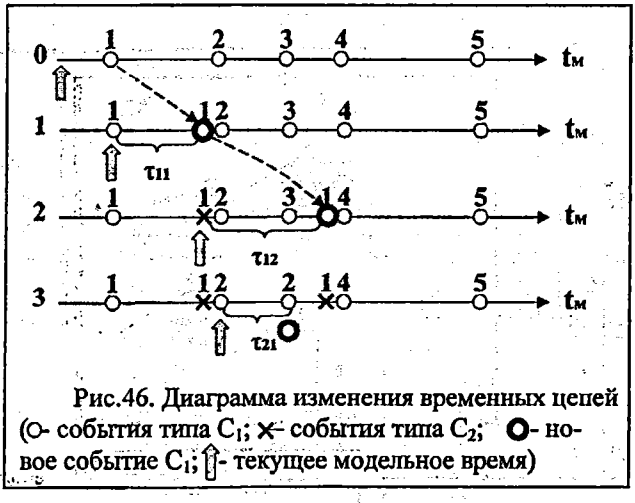


Рис.46. Диаграмма изменения временных цепей (○- события типа C_1; ×- события типа C_2; ●- новое событие C_1; ⇧- текущее модельное время)

В первом такте модельное время перемещается к времени наступления события C_{11} (деталь №1) – к моменту появления в цехе первой детали. Только это событие будет включено в число обрабатываемых событий. Так как условия запуска AK_{11} $(\alpha_1 \neq 0) \& (\pi_1 = 0)$ = ИСТИНА, то активность AK_{11} выполняется, что приводит к внесению в список событий нового, шестого будущего события C_{21}

(деталь №1) - завершение обслуживания первой детали на первом станке в момент времени $t_{i,з} = t_M + \tau_{11}$.

Во втором такте модельное время перемещается к времени наступления события C_{21} (деталь №1) - завершение обслуживания первой детали на первом станке. Оно включается в список обрабатываемых событий. Так как условия запуска активности AK_{12} освобождение первого станка первой деталью $t_{1,з} \geq t_M$ = ИСТИНА, то активность AK_{12} выполняется. Это приводит к освобождению первого станка и перемещению первой детали ко второму станку. Соответственно в этом же такте при следующем выполнении блоков 1, 2 и 5 (рис.44) будут выявлены условия запуска активности AK_{21} захват второго станка первой деталью $(\alpha_2 \neq 0) \& (\pi_2 = 0)$=ИСТИНА. Выполнение активности AK_{21} приводит к внесению в список нового, седьмого будущего события C_{21} (деталь №1) - завершение обслуживания первой детали на втором станке в момент времени $t_M + \tau_{12}$.

В третьем такте модельное время перемещается к времени наступления события C_{12} (деталь №2) - появление второй детали. Оно включается в список обрабатываемых событий. Будут выявлены условия запуска активности AK_{11} - захват первого станка второй деталью $(\alpha_1 \neq 0) \& (\pi_1 = 0)$=ИСТИНА. Выполнение активности AK_{11} приводит к внесению в список нового, восьмого будущего события C_{21} (деталь №2) - завершение обслуживания второй детали на первом станке в момент времени $t_M + \tau_{21}$ и т.д.

Если считать, что события C_{21} (деталь №1) и C_{12} (деталь №2) происходят одновременно (рис.46), то во втором такте модельное время перемещается к времени их наступления и они включаются в список обрабатываемых событий. Так как условие запуска активности AK_{12} - освобождение первого станка первой деталью $t_{1,з} \geq t_M$ = ИСТИНА, то она выполняется, что приводит к освобождению первого станка для занятия второй деталью и к перемещению первой детали для занятия второго станка. Соответственно при последующем выполнении блоков 1, 2 и 5 (рис.45) будут выявлены условия запуска активности AK_{21} - захват второго станка первой деталью $(\alpha_2 \neq 0) \& (\pi_2 = 0)$ = ИСТИНА и условия запуска активности AK_{11} - захват первого станка второй деталью $(\alpha_1 \neq 0) \& (\pi_1 = 0)$=ИСТИНА. Выполнение активности AK_{21} приводит к внесению в список событий нового, седьмого будущего события C_{21} (деталь №1) - завершение обслуживания первой детали на втором станке в момент времени $t_M + \tau_{12}$. Аналогично выполнение активности AK_{11} приводит к внесению в список событий нового, восьмого будущего события C_{21} (деталь №2) - завершение обслуживания второй детали на первом станке в момент времени $t_M + \tau_{21}$ и т.д.

Метод событий применяется, если число типов функциональных действия не велико, либо они одинаковы, либо отличаются только условиями запус-

ка и слабо связаны друг с другом. В этом случае одинаковые активности разных компонентов объединяются в одну группу, которая и называется событием. Каждому такому событию соответствует модуль, а условия запуска модулей строят путем комбинирования условий запуска образующих их активностей. Это позволяет уменьшить размер модели и сократить накладные расходы, связанные с многократным запуском и завершением выполнения модулей активностей.

Структура настраиваемой части модели $M_н$ приведена на рис.47, где модули $СОБ_1$ и $СОБ_2$ образуют модель $M_ф$, а модуль $СОБ_0$ модель среды M_c. Модуль события $СОБ_1$ построен объединением активностей AK_{11}-AK_{14}; модуль события $СОБ_2$ объединением активностей AK_{21}-AK_{24}, а модуль события $СОБ_0$ получен объединением AK_{01} и AK_{02}. Структура модели такая же как и на рис.44, только в модуле $M_ф$ должен быть заложен алгоритм анализа и запуска модулей событий. Так условие запуска $СОБ_1$ $(\alpha_1 \neq 0)\&(\pi_1 = 0)+(\alpha_2 \neq 0)\&(\pi_2 = 0)+...$, а условие запуска $СОБ_2$ $(t_{1,3} \geq t_M) + ... + (t_{4,3} \geq t_M)$. Соответственно все модули $СОБ$ должны идентифицировать тип наступившего события.

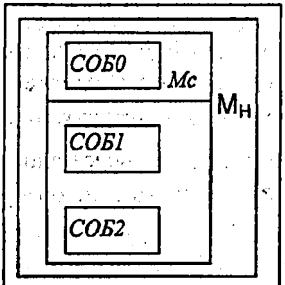


Рис.47. Структура настраиваемой части модели (метод событий)

Транзактный метод применяется, если, например, математической моделью объекта служат Q-схемы и в особенности ССМ и сети МО (см. Глава 1, § 1.7, Глава 2, §1). Это означает, что в объекте используется ограниченное количество типов неподвижных компонентов (узлов), которые управляют движением и обслуживанием подвижных компонентов (транзактов, заявок). Соответственно число типов функциональных действий ограничено, а компоненты слабо связаны друг с другом. В качестве узлов, как правило, используют генераторы и приемники заявок, очереди, устройства и памяти, маршрутные узлы. Сама модель представляется схемой соединения узлов и схемой маршрутов движения заявок в процессе обслуживания, обработки. Алгоритм моделирования состоит в том, что в каждом такте делается попытка продвинуть в модели в соответствии с ее схемой каждую из заявок. При этом узел, через который проходит заявка, корректирует ее временные характеристики, сохраняемые в атрибутах заявки. И так до тех пор, пока подвижных заявок не останется. Затем модельное время продвигается к следующему событию и процесс повторяется. Основные типы событий: приход в систему новой заявки, разблокирование продвижения заявки и завершение ее обслуживания. Такой подход отличается ограниченной универсальностью, но позволяет упростить сам процесс построения модели, создаваемой по принципу конструктора из типовых узлов и состоящий в описании схемы модели.

Пример структуры модели цеха приведен на рис.48. Здесь: - G_1 и G_2 генераторы заявок, имитирующие появление в цехе новых деталей двух типов; - S_1-

S_i обслуживающие узлы, устройства, имитирующие станки и задерживающие заявки на время обработки детали; - R_1 маршрутный узел, определяющий дальнейший маршрут заявки в зависимости от ее типа; - Pr_1 и Pr_2 приемники обслуженных заявок, деталей. На входах в узлы S_1-S_4 изображены очереди заявок, ожидающих обслуживания.

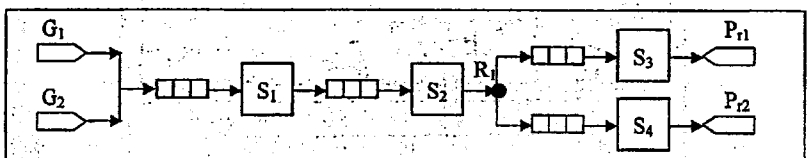


Рис.48. Структура настраиваемой части модели (транзактный подход)

Процессорный метод применяется, если у компонентов функциональные действия разные, а сами компоненты сильно связаны, в том числе и по управлению. В этом случае функционирование каждого компонента рассматривается как единое целое – процесс, полученный объединением его активностей, а вся модель представляется множеством взаимодействующих процессов. Процессы описываются алгоритмически. Используют как стандартные так и специфические средства изображения - для описания и управления специальными типами объектов и для синхронизации и управления параллельными процессами. В частности операторы типа ЖДАТЬ, блокирующие развитие процесса на заданное время, в зависимости от заданного условия и т.д., что позволяет описывать достаточно сложные объекты и обеспечивает высокую универсальность подхода.

Моделирование состоит в том, что в каждом такте поочередно делается попытка продолжить выполнение процессов. Процессы запускаются с той точки (с оператора ЖДАТЬ), в которой их выполнение было приостановлено в предыдущих тактах моделирования, т.е. с точки текущего состояния. Процесс выполняется, пока опять не будет приостановлен каким-либо оператором ЖДАТЬ. И так до тех пор, пока активных процессов не останется. Далее модельное время продвигается к следующему событию и выполняется следующий такт. Соответственно состояние модели задается состояниями процессов, а состояния процессов задаются номерами операторов ЖДАТЬ, где они были приостановлены. Основные события: изменения управляющих сигналов либо истечение времени задержки, приводящие к разблокированию процессов и их дальнейшему выполнению. В ходе выполнения процессы вносят изменения в будущие значения сигналов управления, что находит отражение в ИБ.

Общая структура модели такая же как и на рис.45, где в модуле M_ϕ заложен алгоритм запуска модулей активных процессов, составляющих модуль M_*. Структура настраиваемой части модели цеха M_* приведена на рис.50, где $Пр_1$-$Пр_4$ - процессы, моделирующие станки. Структура процесса показана на рис.51, где для его описания использованы 4 команды. Нетрудно заметить, что для рас-

сматриваемого примера лучше всего подходит метод событий или транзактный метод.

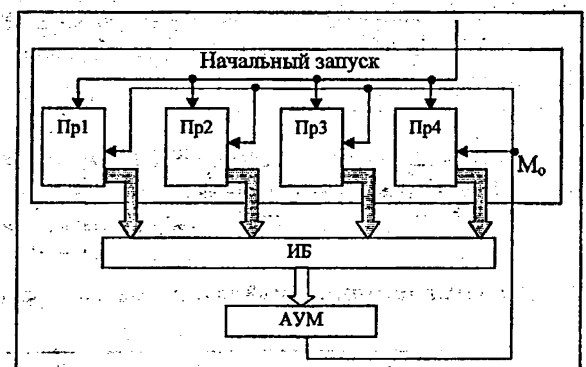


Рис.50. Структура настраиваемой части модели (процессный метод)

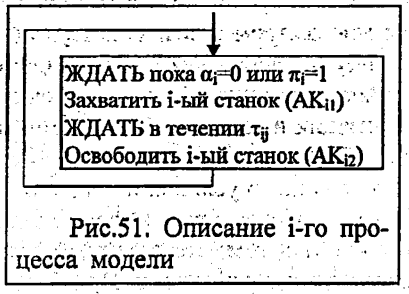


Рис.51. Описание i-го процесса модели

7.6. Классификация средств имитационного моделирования

Средства моделирования можно разделить на программные и аппаратные (рис.52). К программным средствам относят: - универсальные языки программирования высокого уровня; - проблемно-ориентированные языки моделирования; - предметно-ориентированные языки (системы) моделирования. К аппаратным средствам относят: - ЭВМ ОН, системы и комплексы на их основе; - суперЭВМ; - специализированные средства (АВМ; - СВМ; - языковые процессоры; - МИМ).

Универсальные языки, позволяя моделировать любой объект, чье поведение представимо алгоритмом, как правило обладают ограниченным набором встроенных вычислительных объектов, не имеют средств управления параллельными процессами, средств статистического анализа результатов и, как следствие, кроме описания самой модели требуют программирования всех задач, необходимых для организации имитационного моделирования. Среди них следует выделить языки реального времени, обладающие средствами управления па-

раллельными процессами, например, АДА, Модула, Элъ-76 (язык многопроцессорных вычислительных комплексов "Эльбрус").

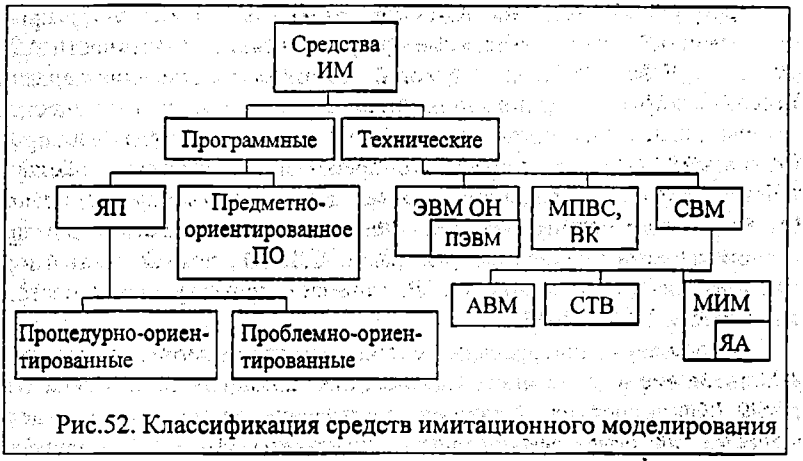


Рис.52. Классификация средств имитационного моделирования

Появление проблемно-ориентированных языков системного моделирования тесно связано с осуществлением крупных проектов в сфере управления промышленностью, финансами и обороной. Их особенность в том, что они полностью реализуют ту или иную технологию моделирования (модули подсистемы M_ϕ и ПОМ), создают методологическую основу для описания и исследования систем различной природы с единых системных позиций. Предоставляют разработчику инструмент мышления в виде понятий языка, что способствует направлению основных усилий на выявление системных свойств объекта исследования. Они ориентированны на псевдопараллельную обработку параллельных процессов, содержат средства динамического управления памятью и системным временем, генерации случайных объектов, задания режимов моделирования, обработки результатов и т.п. Кроме того, они предоставляют пользователю входной язык – средства описания модели, что упрощает и экономит время на программирование и отладку модели. Для пользователя процесс моделирования состоит в разработки модели соответствующего типа (M_x) и записи ее структуры и порядка функционирования на входном языке. В зависимости от типа модели, заложенного в языке, выделяют языки моделирования дискретных (GPSS, SIMULA, Simscript, SOL, СЛЭНГ, VHDL, ОККАМ, АДА и др.) и непрерывных систем (СИМФОР, GASP и др.). **GPSS** ориентированн на обработку транзактов и подробно описан в Главе 2, § 2. **SOL** (Simulation Orientated Language) разработан в 1964 г., ориентирован на описание асинхронных, параллельных процессов. Имеет развитые общеалгоритмические возможности, т.к. является расширением языка АЛГОЛ. Включает все средства описания систем, присущие GPSS. Базируется на взаимосвязанных понятиях класса подвижных объектов и процессов - алгоритмов, описывающих поведение объектов некаждого класса. Функционирование системы описывается в виде совокупности,

взаимодействующих процессов. Процессы оперируют ресурсами типа память и устройство. Недостатки: фиксированная дисциплина обслуживания; взаимодействие процессов только через глобальные переменные; необходимость навыков программирования на АЛГОЛе. **SIMULA** (SIMULA-67) процессно-ориентированный язык, сочетающий алгоритмические возможности АЛГОЛА, развитые средства обработки списков и возможности описания параллельных процессов. Базируется на понятиях объекта как набора данных определенной структуры и класса как описания структуры данных и алгоритма их преобразования. В языке отсутствуют объекты с заранее определенными свойствами. Все объекты и операции по ведению очередей должны быть определены пользователем. Язык более ориентирован на применение специалистами для программирования пакетов прикладных программ. **СЛЭНГ**, разработанный в 1967 г. Институтом кибернетики АН УССР, занимает промежуточное место между языками SOL и SIMULA-67.

Современные универсальные системы моделирования, ориентированные на использование в среде многозадачных операционных систем типа Windows, частично используют их механизмы управления процессами, что делает их применение еще более эффективным. Это пакеты MATLAB, Scientific Workplace, MathCAD, Mathematica и другие. **Scientific Workplace** ориентирован на проведение математических исследований, в особенности аналитических. Пакет **Mathematica** ориентирован на математические исследования как в символьной, так и в численной форме. **MathCAD** является полноценным Windows-приложением со встроенными средствами обмена (Clipboard, OLE). Его прикладные дополнения ориентированы на методы численного анализа, статистики, теории очередей. **MatLab** (Matrix Laboratory) - наиболее мощное средство, эксплуатируется более двух десятков лет. Базируется на матричных и векторных вычислениях, отличается легкостью расширения. Прилагаемые инструментальные пакеты ориентированы на решение численных задач, аналитическое и имитационное моделирование. Это методы и модели теории автоматического управления, дифференциальные уравнения и интегральное исчисление, модели нечеткой логики, нейронные сети. Имитационное моделирование реализовано системой визуального моделирования SIMULINK. **OSS** - пакет оптимизации, ориентированный на линейное, нелинейное, целочисленное и динамическое программирование, оптимизацию на графах, игры, аналитическое и имитационное моделирование (марковские процессы и теория очередей).

Специализированные системы моделирования ориентированы на конкретную предметную область, реализуют заранее определенную обобщенную модель и соответственно предназначены для моделирования объектов только определенного типа. Пользователю не приходится описывать алгоритм функционирования объекта, а достаточно доопределить параметры модели. Описание выполняется в терминах не входного языка моделирования, как в предыдущем случае, а в терминах соответствующей, привычной пользователю предметной области. Например, в системе CSS и ее аналоге ППП МВС, предназначенных для моделирования производительности вычислительных систем,

пользователь описывает параметры не абстрактных узлов и заявок, а процессора, шин, задач и т.п. Недостаток систем – их узкая специализация.

Обычные ЭВМ как правило однопроцессорны, не оптимизированы на выполнение задач моделирования и являются средством реализации моделей, описанных на универсальных, проблемно- или предметно-ориентированных языках моделирования. Тем не менее, и здесь ускорение может быть достигнуто применением более мощных ЭВМ, рабочих станций, вычислительных комплексов, сверхбыстродействующих ЭВМ. Но радикального ускорения моделирования можно достичь лишь использованием дорогостоящих специализированных аппаратных средств. Причем иногда это единственная возможность моделировать сложные объекты с нужным качеством. Технологически специализированная аппаратура может быть выполнена в виде отдельной специализированной ЭВМ, устройств, плат.

Среди наиболее ранних устройств такого типа следует отметить АВМ. Аналоговые вычислительные машины применяются для решения *D*-моделей, исследования дифференциальных уравнений. В них числовые значения, используемые при моделировании, кодируются значениями токов и напряжений, и являются непрерывными величинами. В состав АВМ входит ограниченный набор типовых блоков, реализующих элементарные операции над непрерывными величинами. Число блоков каждого типа также ограничено. Соответственно моделирование состоит в разработке математической модели и ее вводу в АВМ. Сама модель строится по структурному принципу - пользователь набирает ее путем коммутации типовых блоков. Моделирование происходит со скоростью переходных процессов в электронных блоках. Однако аналоговый характер вычислений приводит к низкой точности, а ограниченное количество типовых элементов ограничивает допустимую сложность моделируемых объектов.

Для моделирования дискретных объектов в классе *Q*-моделей применяют СВМ. Стохастические вычислительные машины также как и АВМ построены по структурному принципу и предоставляют пользователю наборы типовых элементов – генераторов заявок, очередей, устройств и т.п. Пользователь вводит модель, коммутируя и настраивая параметры типовых элементов. Подвижные объекты, транзакты имитируются потоками импульсов. Основной недостаток: ограниченность набора типовых элементов и следовательно сложности моделируемых объектов.

Машины имитационного моделирования (МИМ) – многопроцессорные системы, ориентированные на моделирование сложных и нестационарных объектов. Ускорение достигается естественным распараллеливанием выполнения моделируемых процессов, опережающим выполнением основных операций имитационного моделирования. В составе МИМ отдельные процессоры используют для аппаратной поддержки генерации случайных объектов; обработки статистических данных; управления временными цепями и т.д. Функции процессоров, как правило, могут оперативно перераспределяться. МИМ позволяют ускорять моделирование на один-два порядка. Модели описываются и вводятся в МИМ с использованием специальных входных языков. Отдельный класс МИМ аппаратно реализует и ускоряет выполнение стандартных конструкций

языков моделирования, например GPSS, VHDL и др. Такие МИМ относят к классу языковых процессоров (акселераторов). Например, для обеспечения приемлемого сочетания гибкости и скорости моделирования создаются аппаратно-программные системы VHDL-моделирования, дополненные оптимизированным средством моделирования на вентильном уровне. В его качестве могут использоваться аппаратные ускорители, сопроцессоры. Такая система, выполняя "грамматический разбор" входного VHDL-описания, выделяет в нем "ускоряемые" (структурные конструкции), которые передаются в ускоряющий процессор, а смешанные структурно-поведенческие конструкции реализуются программно.

Глава 2

СИСТЕМНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ

1. Структурно-функциональное моделирование систем. Стохастические сетевые модели (ССМ)

Определение и состав. ССМ – совокупность узлов и заявок. Узлы (приборы) определяют маршруты движения заявок и отображают обслуживающие ресурсы типа "*память*" и "*устройство*". Заявки имитируют подвижные объекты (детали, задачи и т.п.), обслуживаемые в узлах сети. Поскольку обслуживающие ресурсы ограничены, то к ним могут возникать очереди из задержанных (ожидающих обслуживания) заявок. Типовые узлы ССМ представлены на рис.1 и 2. Здесь одиночный генератор имитирует однородный поток заявок (заявки одного класса) с заданными вероятностными характеристиками. Заявки появляются в соответствии с заданным законом распределения f_t независимо от состояния других узлов и состояния сети в целом. Групповой генератор имитирует однородный поток заявок с заданными вероятностными характеристиками. Его параметр M определяет количество заявок, которое сразу появляется в сети в начале моделирования. При завершении обслуживания любой из этих заявок вместо нее в сеть отправляется новая заявка. Таким образом в сети поддерживается постоянное количество заявок и создается напряженный режим ее работы. Приемники служат для изъятия заявок из сети. Узел типа "устройство" обладает конечной скоростью работы (числом операций в единицу времени) и числом обслуживающих каналов. Может обрабатывать одновременно только одну заявку в каждом канале и задерживает движение заявки на время ее обработки. Узел типа "память" обладает конечной емкостью и может захватываться одновременно несколькими заявками, которые впоследствии могут освобождать память, проходя через узлы освобождения памяти. Если заявка получила требуемую долю памяти, то она продолжает движение по сети без задержки. В противном случае, как и в устройстве, заявка помещается в очередь. Маршрутный узел типа M

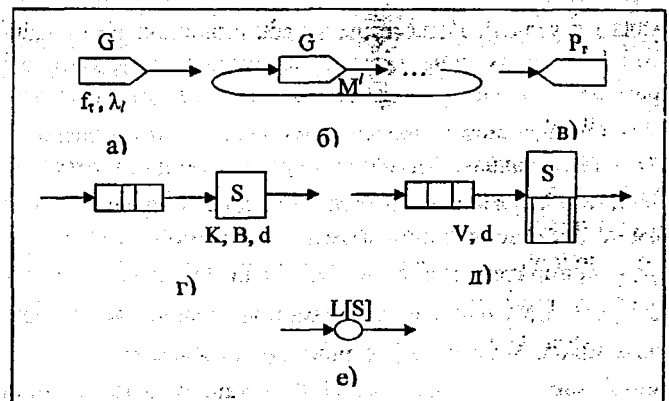


Рис.1. Типовые узлы ССМ: а) одиночный генератор; б) групповой генератор; в) приемник; г) устройство; д) память; е) узел освобождения памяти

позволяет менять параметры потока заявок. Вероятностный узел типа P определяет, разыгрывает дальнейший маршрут движения заявок в соответствии с заданными вероятностями. Узел типа R определяет дальнейший маршрут движения заявок в зависимости от их атрибутов, например имен. Узел типа U определяет основной маршрут движения заявки и альтернативный на случай, если узел основного маршрута занят.

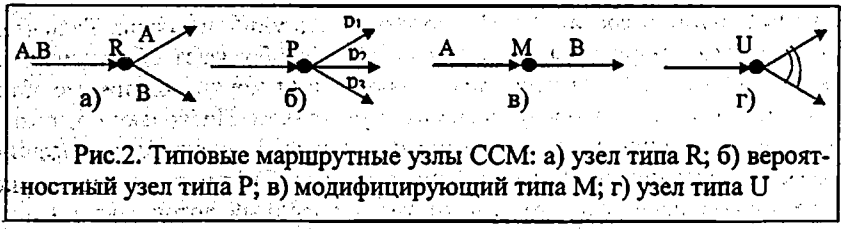


Рис.2. Типовые маршрутные узлы ССМ: а) узел типа R; б) вероятностный узел типа P; в) модифицирующий типа M; г) узел типа U

Параметры. ССМ задается как $X=\langle C,Z\rangle$. Здесь система $C=\langle C_B,C_Q,C_D\rangle$ описывает: - состав узлов $C_B=\{b_i\}$ (число узлов $|C_B|=N$); - состав классов однородных потоков заявок $C_Q=\{q_i\}$ (число классов $|C_Q|=Q$); - порядок соединения узлов в сети и маршруты движения заявок каждого q-го класса $C_D=\{d_{ij}^q\}$ (значение d_{ij}^q = 1, если есть путь движения заявки q-го класса от узла i к узлу j). Параметры заявок одного класса распределяются по одним и тем же законам, в частном случае имеют одинаковые параметры. Перечень параметров заявок q-го класса может включать $((\lambda^q$ или $M^q);\{\Theta_{ij}\};pr_q;\ldots)$, где λ^q - интенсивность поступления заявок (число заявок в единицу времени), а M^q - число заявок, постоянно циркулирующих в сети, $\{\Theta_{ij}\}$ - параметры трудоемкости обслуживания заявок в узлах сети (среднее число операций в каждом j-ом устройстве, средний размер занимаемой памяти v_i^q в каждой i-ой памяти), pr_q - приоритет заявок и т.д. Если Q=1, то сеть однородная. Множество $Z=\{z_i\,|\,i=\overline{1,N}\}$ описывает параметры генераторов, маршрутных и обслуживающих узлов. Узел типа "устройство" описывается как $z_i=(K_i,d_i,\{\upsilon_i^q\},s_i)$, а "память" как $z_j=(V_j,d_j,s_j)$. Здесь K_i - канальность устройства, V_j - объем памяти, d_i - принятая дисциплина обслуживания заявок (в т.ч. извлечения заявок из очереди), υ_i^q - скорость обслуживания (число операций в единицу времени), s_i - стоимость узла.

Характеристики. Работа ССМ оценивается набором *узловых* (по каждому обслуживающему узлу) и *системных* (по всей сети) характеристик. Системные характеристики рассчитываются через узловые. Характеристики вычисляются как по каждому классу заявок так и усредненные по всему потоку заявок в сети и включают временные и натуральные характеристики. Основные узловые характеристики — интенсивность потока заявок на входе в узел λ_i, коэффици-

ент загрузки узла $\rho_i = \lambda_i / \mu_i$, коэффициент использования памяти, средняя длина очереди к узлу l_i, среднее количество заявок в системе очередь-узел m_i, среднее время ожидания обслуживания в узле ω_i, среднее время пребывания в узле u_i. Основные системные характеристики - интенсивность потока заявок в сети $\lambda_0 = \sum_{q=1}^{Q} \lambda^q$, среднее число заявок в сети $M_0 = \sum_{q=1}^{Q} M^q$, среднее число заявок, обслуживаемых в сети $R = \sum_{q=1}^{Q} R^q$, среднее число заявок в очередях L, среднее время ожидания W, среднее время обслуживания заявки в сети U.

Узловые характеристики без учета классов заявок могут быть вычислены через аналогичные характеристики отдельных классов заявок $l_i = \sum_{q=1}^{Q} l_i^q$, $\omega_i = \sum_{q=1}^{Q} \omega_i^q \lambda_i^q / \lambda_i$. Аналогично системные характеристики каждого класса заявок $L^q = \sum_{i=1}^{N} l_i^q$, $W^q = \sum_{i=1}^{N} \omega_i^q \alpha_i^q$ и системные характеристики без учета классов заявок $L = \sum_{q=1}^{Q} L^q = \sum_{i=1}^{N} l_i$ и $W = \sum_{q=1}^{Q} W^q \lambda^q / \lambda_0 = \sum_{i=1}^{N} \omega_i \alpha_i$.

Классификация приведена на рис.3. Сети массового обслуживания (сети МО) в отличие от ССМ допускают использование из обслуживающих узлов только узлов типа "устройство", а из маршрутных только вероятностных. В зависимости от структуры, номенклатуры узлов, классов заявок и степени детализации описания законов функционирования узлов можно выделить девять классов стохастических сетей и связанные с ними типы стохастических сетей. Тип стохастической сети определяется перечислением классов, к которым относится сеть, т.е. множество типов сетей соответствует множеству различных путей на графе классификации, ведущих от начальной вершины в конечные вершины графа.

Сеть МО линейная (Л), если интенсивность потока заявок на входе в любую СМО $\lambda_i = \sum_j \lambda_j p_{ji}$ определяется линейной суперпозицией выходных потоков других СМО. Здесь p_{ji} - вероятность перехода заявки с выхода j-ой СМО на вход i-ой СМО, задаваемая для каждого класса заявок матрицей вероятностей переходов. В такой сети нельзя создавать копии заявок и обслуживать их параллельно. В противном случае сеть нелинейная (НЛ). Сеть замкнутая (З), если в ней используются только групповые источники заявок, и разомкнутая (Р), если применяются только одиночные генераторы. Сеть смешанная (СС), если применяются и те и другие. Если в сети все заявки одного класса (применяется только один генератор), то она однородная (О), в противном случае сеть неоднородная (Н). Если законы поступления заявок в сеть пуассоновские, а законы обслуживания заявок в узлах – экспоненциальные, то сети называют экспоненциальными (Э). Приведенная классификация стохастических сетей сопрягается

с классификацией методов, используемых для расчета их характеристик. Для исследования сетей массового обслуживания используется метод имитационного моделирования, если они относятся к классу смешанных, или приближенные численные методы для остальных классов сетей массового обслуживания. Точные аналитические методы существуют для экспоненциальных однородных замкнутых и разомкнутых сетей массового обслуживания, а также для некоторых типов неоднородных сетей массового обслуживания. Примеры сетей МО представлены на рис.4.

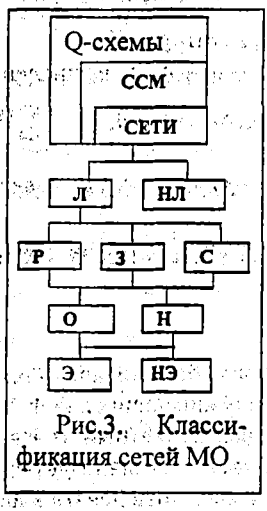


Рис.3. Классификация сетей МО

Модели одного типа можно упорядочить в зависимости от их сложности, характеризуемой размерностью - числом параметров, используемых для задания законов функционирования узлов.

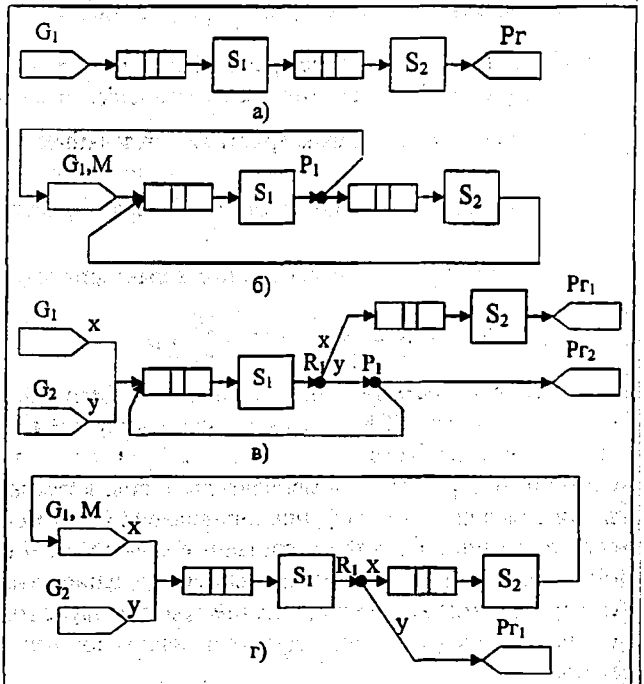


Рис.4. Примеры сетей МО: а) однородная разомкнутая; б) однородная замкнутая; в) неоднородная разомкнутая; г) смешанная сеть

Так закон функционирования узла i для каждого из l_i классов заявок, обслуживаемых в нем, может задаваться с различной степенью детализации, характеризуемой числом параметров k_i, используемых для его описания, например, только средними значениями, с точностью до средних и дисперсий или с точностью до распределений (гистограмм). При использовании приоритетных дисциплин обслуживания число k_i увеличивается на единицу. Тогда для описания закона функционирования узла i (i = 1,...,N) потребуется $k_i' = l_i k_i$ параметров, а количество параметров $C_N = \sum_{i=1}^{N} k_i' = \sum_{i=1}^{N} l_i k_i$ охарактеризует сложность всей сети.

2. Система моделирования общего назначения GPSS/PC

2.1. Общая характеристика языка GPSS

Язык GPSS - система моделирования общего назначения (General Purpose Simulation System), создан фирмой IBM в 60-х годах. Относится к числу проблемно-ориентированных языков моделирования, предназначенных для описания и имитации дискретных объектов. Тип моделей, подход к описанию – функциональный, способ продвижения модельного времени – событийный, метод реализации псевдопараллельностей – транзактный. Наиболее удобен для моделирования Q-схем, стохастических сетевых моделей, сетей МО. Система выпускалась в разных версиях (GPSS-2, GPSS-3, GPSS/360 для ЭВМ серии IBM/360, затем GPSS V и т.д.). В 70-х годах функциональные возможности GPSS расширили, объединив его, в частности с языком программирования APL, с 90-х годов существует DOS-ориентированная версия GPSS/PC для ПЭВМ. С 80-х годов в США GPSS наряду с языками GASP, Simscript входил в число основных языков моделирования. Популярность языка связана с минимальными требованиями к опыту программирования, с возможностью быстрого освоения моделирования на начальных стадиях. Все стандартные задачи имитационного моделирования автоматизированы (скрыты в интерпретаторе GPSS). В то же время разработка сложных моделей требует детального знания внутренней организации языка и его интерпретатора, а также методов статистики. Модели сложных объектов мало читабельны и соответственно недостаточно удобны в использовании, в частности из-за преобладания функционального подхода к описанию моделей. Далее рассматривается версия 2 GPSS/PC (1991 г.). Состав системы моделирования приведен на рис.5. Система включает входной язык для описания моделей и задания режимов моделирования и соответствующее программное обеспечение, обеспечивающее интерфейс, моделирование и статистическую обработку результатов.

2.2. Внутренняя организация языка GPSS

Объекты. В соответствии с транзактным подходом (см. Главу 1, § 7.4) модель строится из взаимосвязанных динамических, подвижных объектов – транзактов и статичных, неподвижных объектов – узлов (по терминологии GPSS – блоков, которым в тексте GPSS-модели соответствуют операторы) и представляет собой схему перемещения транзактов от узла к узлу. Среди неподвижных узлов (см. рис.5) следует выделить обслуживающие (устройства, многоканальные устройства – памяти и задержки), маршрутные и статистические, управляющие сбором статистики (очереди, таблицы). Кроме этого в GPSS-моделях используются объекты-данные. Это обрабатываемые, вычисляемые объекты, такие как переменные (значения арифметических и логических выражений), функции, сохраняемые величины (ячейки) и их матрицы. Особый объект – временные цепи. Это списки событий транзактов (текущих, будущих, прерванных, синхронизируемых и пользовательских), образующие информаци-

онную базу (ИБ) GPSS.

Интерпретатор. Язык реализован по принципу интерпретатора, поэтому каждый оператор (узел, блок) модели рассматривается как точка вызова соответствующей процедуры. В каждый такт модельного времени интерпретатор: - пытается продвигать активные транзакты (из списка текущих событий) от узла к узлу в соответствии со схемой модели, пока это возможно; - обрабатывает внутренние события, происходящие в процессе продвижения транзактов (изменение состояний обслуживающих и маршрутных узлов; завершение или прерывания в обслуживании заявок; события синхронизации транзактов и т.п.); - перемещает модельное время к времени наступления ближайшего события (в списке будущих событий) и т.д.

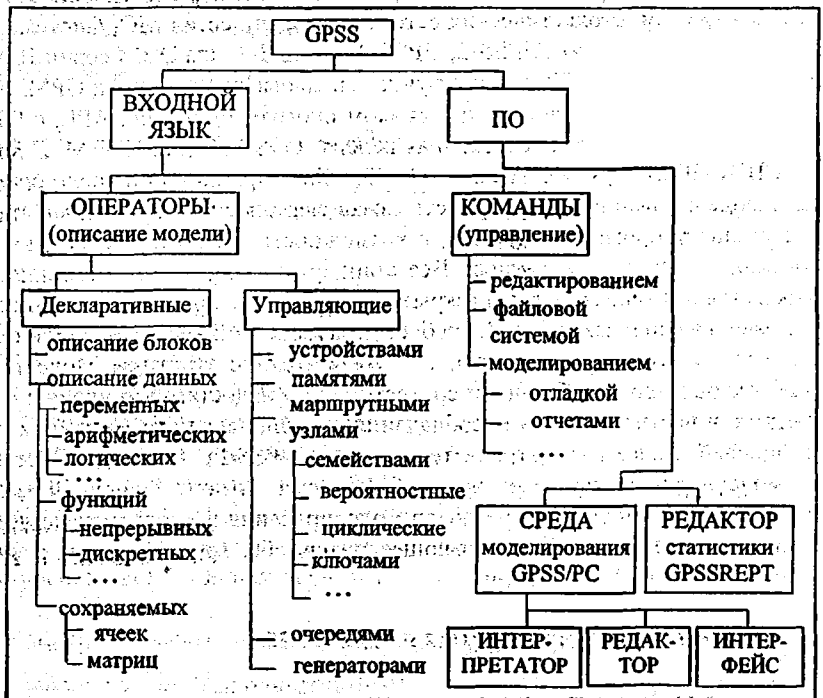


Рис.5. Состав системы моделирования GPSS/PC

Транзакт продвигается, пока не будет задержан, например, на время обслуживания (помещается в список будущих событий); пока не попадет в приемник транзактов, завершивших обработку (изымается из модели); пока не будет заблокирован на входе в очередной узел из-за его занятости (теряет активность, оставаясь как в списке текущих событий, так и в списке транзактов, задержанных к соответствующему узлу).

Модельное время. Синхронизация событий выполняется с использованием модельного времени (системных часов), которое в GPSS представляется целым, безразмерным числом. Соответственно все временные параметры в модели должны быть предварительно приведены разработчиком к одной единице измерения и установлен масштаб модельного времени. Системные часы включают таймер абсолютного t_{MA} и относительного модельного времени t_{MO}. Таймер управляется (см. рис.6) командами START и RESET. Контроль за t_{MA} выполняется так же командой SIMULATE, которая определяет его максимально возможное значение.

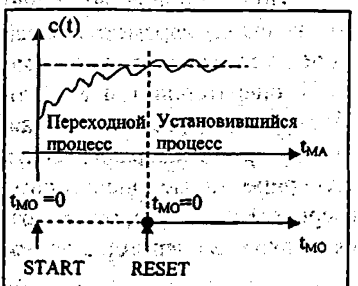


Рис.6. Управление модельным временем

2.3. Примеры GPSS-модели

Пример 1. Объект моделирования - станок, обслуживающий поток деталей. Детали поступают через случайное время, распределенное по равномерному закону в диапазоне 4-16 с (со средним значением 10 с). Время обслуживания детали – случайная величина, распределенная по равномерному закону в диапазоне 3-9 с (со средним значением 6 с). Требуется промоделировать работу объекта в течение получаса, т.е. 1800 с. Модель объекта в виде ССМ представлена на рис.7, г. А графичекая схема GPSS-модели, включающая сегмент описания объекта (операторы 1-7) и сегмент описания системных часов (операторы 8-9), изображена на рис.7, а и б соответственно. Текст GPSS-модели приведен ниже

```
 1 Generate  10,6   - генерация заявок (моментов появления деталей) через
                      время, распределенное по равномерному закону в диа-
                      пазоне 10±6 с
 2 Queue     A      - отметка заявки в очереди А (фиксация времени появле-
                      ния детали в системе)
 3 Seize     1      - занятие одноканального устройства №1 (станка)
 4 Depart    A      - отметка заявки в очереди А (фиксация начала
                      обработки детали)
 5 Advance   6,3    - задержка заявки на время, распределенное равномерно
                      в диапазоне 6±3 с (имитация обработки детали)
 6 Release   1      - освобождение устройства №1 (завершение обработки
                      детали)
 7 Terminate        - изъятие заявки из сети (завершение обслуживания)
 8 Generate  1800   - генерация очередной заявки через 1800 единиц
                      модельного времени
 9 Terminate 1      - изъятие заявки из сети и уменьшение значения мо
                      дельного таймера (счетчика) на 1
10 START     1      - команда начала моделирования и установки счетчика
                      таймера в 1
11 END              - команда завершения работы с системой GPSS
```

По команде START счётчик завершения устанавливается в соответствии со значением параметра команды, здесь в 1. Начинается собственно моделирование, состоящее в одновременном выполнении всех сегментов модели, начиная с операторов 1 и 8 соответственно. Первый транзакт, пройдя операторы (блоки) 1-4, перестает быть активным и его дальнейшее продвижение блокируется на время задержки обслуживания оператором 5. Все остальные транзакты, сгенерированные оператором 1, пока имитируется обслуживание текущего (первого) транзакта, отмечаются в очереди А (оператор 2) и приостанавливаются на входе в оператор 3 до момента его освобождения и т.д.

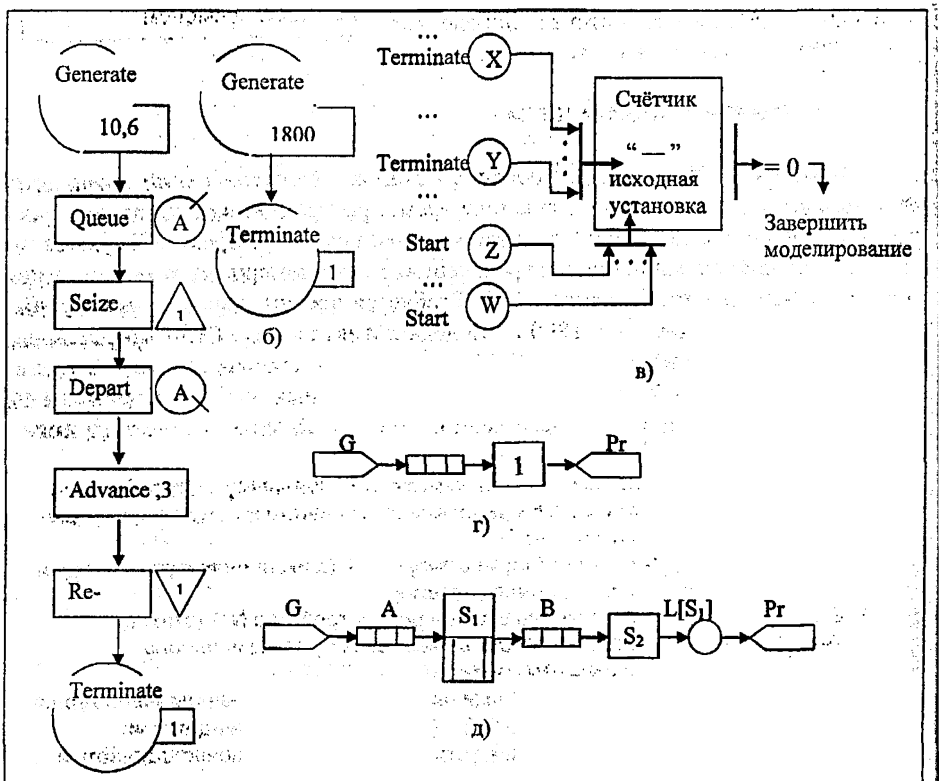


Рис.7. Модель объекта на языке GPSS: а) блок-схема модели; б) модельные часы; в) схема управления модельными часами; г) объект моделирования; д) объект моделирования с памятью

Во втором сегменте первая заявка появится ровно через 1800 единиц модельного времени с начала моделирования. В соответствии с рис.7, в текущее значение счетчика завершений будет уменьшаться при выполнении каждого оператора TERMINATE на значение его параметра, здесь на 1. Его обнуление приведет к завершению моделирования, которое можно продолжить новой командой START. Другой способ управления модельным временем состоит в явном задании количества заявок, которые должны пройти через модель объекта за все время моделирования, например 10000. Для этого команда START 1 заменяется на START 10000, 7-й оператор TERMINATE на TERMINATE 1, а второй сегмент исключается из модели. Тогда по команде START счётчик завершения будет установлен в значение 10000, а каждый обслуженный транзакт, проходя через оператор TERMINATE 1, уменьшит его значение на 1.

Пример 2. Объект моделирования - станок, обслуживающий поток деталей и описанный выше. В отличие от предыдущей модели здесь учтено наличие в цехе склада и использован узел S типа "память". Соответствующая ССМ показана на рис.7,д, а текст GPSS-модели представлен ниже

```
Generate    10,6
Queue       S      - фиксация момента появления детали в цехе
Queue       A      - фиксация момента появления детали в цехе на входе в   склад
Enter       1,1    - захват одного места на складе
Depart      A      - фиксация момента появления детали на складе
Queue       B      - фиксация момента появления детали в очереди к станку
Seize       2
Depart      B      - фиксация момента начала обработки детали
Advance     6,3
Leave       1,1    - освобождение одного места на складе
Depart      S      - фиксация момента завершения обработки детали в цехе
Terminate   1
START       10000
END
```

Здесь таймер построен по второму варианту. Статистика, собираемая в очереди S, фиксирует среднее время пребывания детали в цехе и среднее число деталей, обрабатываемых в цехе. Статистика, собираемая в очереди A и B, фиксирует среднее время пребывания детали в ожидании места на складе и ожидания занятия станка.

2.4. Внутренняя последовательность событий GPSS

Состояния транзактов и узлов. Каждый транзакт в процессе обслуживания и движения по сети последовательно переходит из одного состояния в другое в соответствии с графами, представленными на рис.8, а и б. Это состояния: А – активен (находится в списке текущих событий), П – пассивен (находится в других списках), И – исключен из сети. В свою очередь, состояние активен А включает состояние А&П - активен и продвигается по сети и А&Ж - ак-

тивен и ждет, например, на входе в какой-то узел, блок, оператор. А состояни пассивен П включает состояние П&З - пассивен и задержан на время, П&П пассивен и прерван, П&С - пассивен и синхронизируется, П&ЗП - пассивен и задержан пользователем. Узлы-устройства могут находиться в одном из двух состояний Fs: С – узел свободен и З – занят.

Списки, образующие ИБ GPSS, и их структура представлены на рис.8, 9. ИБ включает 5 типов списков событий, а также списки транзактов, задержанных к узлам сети.

Список будущих событий содержит записи о пассивных транзактах, соответствующих событиям, чье время наступления t_i еще не пришло ($t_i > t_M$). Например, это события, вызванные задержкой транзактов явным образом оператором ADVANCE. Они упорядочиваются в порядке роста времени наступления без учета приоритетов. Интерпретатор просматривает список, начиная с ближайшего события по времени наступления, и использует это время в качестве следующего значения модельного времени.

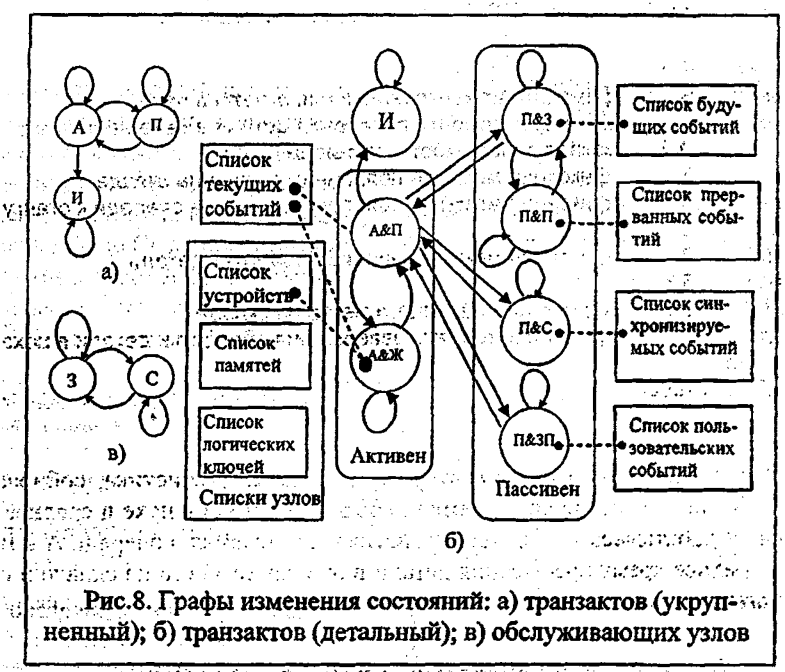


Рис.8. Графы изменения состояний: а) транзактов (укрупненный); б) транзактов (детальный); в) обслуживающих узлов

Список текущих событий содержит записи об активных транзактах, соответствующих событиям, чье время уже наступило ($t_i \le t_M$). Для каждого события (транзакта) эти записи хранят время его наступления t_i, приоритет pr_i, признак его активности I_i и т.д. Событие с признаком активности А&П соответствует активному транзакту, который в текущее время может продвигаться по сети. Событие с признаком активности А&Ж соответствует активному транзакту, который заблокирован на входе в узел (оператор) из-за его недоступности, за-

нятости другими транзактами, и не может продвигаться по сети. События упорядочиваются по убыванию приоритетов, а при их равенстве по времени поступления в список. Интерпретатор просматривает список, начиная с наиболее приоритетного события, и продвигает по сети, пока это возможно, каждое событие с признаком активности A&П.

Список транзактов, задержанных к узлам сети, содержит транзакты, заблокированные при попытке войти в узел (оператор). Они упорядочиваются в соответствии с принятыми в узле дисциплинами обслуживания с учетом приоритетов и ждут изменения состояния соответствующего узла. При освобождении узла просматривается его список задержанных транзактов, выбирается транзакт – кандидат на занятие узла, в списке текущих событий его признак меняется на A&П и он начинает продвигаться по сети.

Список прерываний содержит записи о транзактах с признаком П&П, соответствующих событиям, вызванным прерыванием обслуживания транзактов на узлах типа "устройство" другими более приоритетными транзактами. Это происходит при захвате устройства оператором PREEMPT. Такие события упорядочиваются в соответствии с принятой дисциплиной обслуживания по убыванию приоритетов. Интерпретатор просматривает список, начиная с наиболее приоритетного события, транзакта.

Список синхронизируемых событий содержит записи о транзактах с признаком П&С, чьи копии, созданные оператором SPLIT, находятся в состоянии взаимной синхронизации в операторах MATCH, ASSEMBLE и

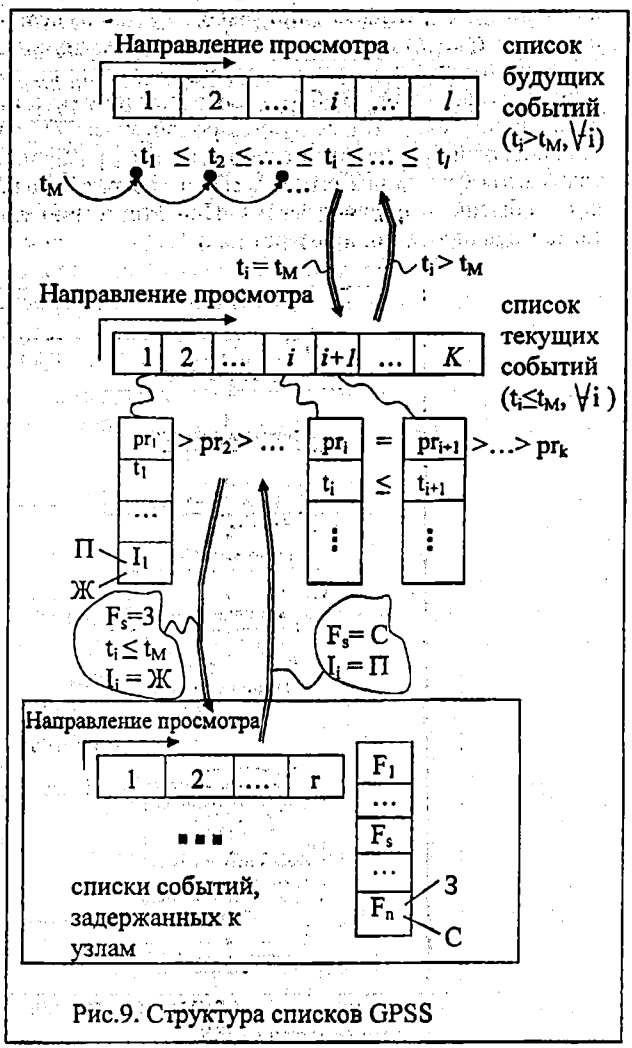


Рис.9. Структура списков GPSS

GATHER.

Список пользовательских событий содержит записи о транзактах с признаком П&ЗП, которые были удаленны пользователем из списка текущих событий с помощью оператора LINK и помечаются как временно не активные.

Укрупненный алгоритм работы интерпретатора представлен на рис.9 и 10. В каждом такте он включает следующие шаги: 1) просмотр списка будущих событий и поиск ближайших событий (их может быть несколько); 2) корректировку списка текущих событий, перенос найденных событий из списка будущих в список текущих событий с признаком А&П; 3) продвижение модельного времени; 4) просмотр списка текущих событий, определение порядка продвижения транзактов в сети, обработка ситуации – одновременное наступление событий (пропуск в списке транзактов с признаком А&Ж и продвижение далее по сети транзакта с признаком А&П); 5) если список текущих транзактов не просмотрен до конца, то возврат на п.4 и продвижение очередного транзакта; 6) просмотр списков транзактов, задержанных на входах в узлы, у которых изменились состояния. Выбор в соответствии с принятой в узле дисциплиной обслуживания тех, которые будут продвигаться; 7) корректировка в списке текущих событий признаков активности у всех разблокированных на входах в узлы транзактов (установка его в А&П) и возврат к началу просмотра списка текущих событий – переход на п.4. При отсутствии таких транзактов завершении такта моделирования и возврат на п.1.

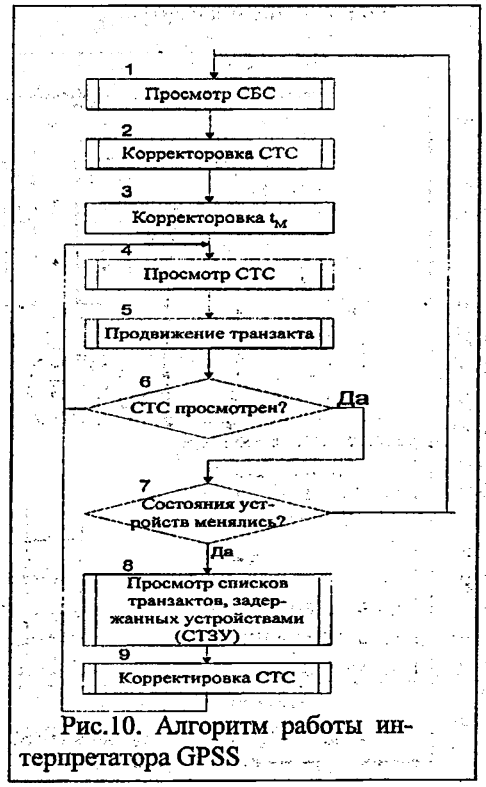


Рис.10. Алгоритм работы интерпретатора GPSS

Алгоритм продвижения транзакта представлен на рис.11. и состоит в последовательном выполнении операторов (блоков) на его пути. При этом: - транзакты, заблокированные на время, переносятся в список будущих событий с признаком П&З; - транзакты, заблокированные на входах в узлы, помечаются в списке текущих событий признаком активности А&Ж и копируются в соответствующий список задержанных транзактов; - транзакты, ставшие пассивными по другим причинам, помещаются в соответствующие списки; - полностью обслуженные транзакты, попавшие в блоки терминации, извлекаются из сети.

2.5. Общая структура системы GPSS

Состав системы моделирования приведен на рис.5. Входной язык GPSS включает операторы для описания модели объекта и команды для организации и управления моделированием. Декларативные операторы предназначены для описания объектов модели (задания параметров узлов, переменных, сохраняемых величин и их матриц, функций и т.п.) и управления процессами обработки транзактов в сети. Соответствующее программное обеспечение образует инструментальную среду разработки, отладки и реализации моделей.

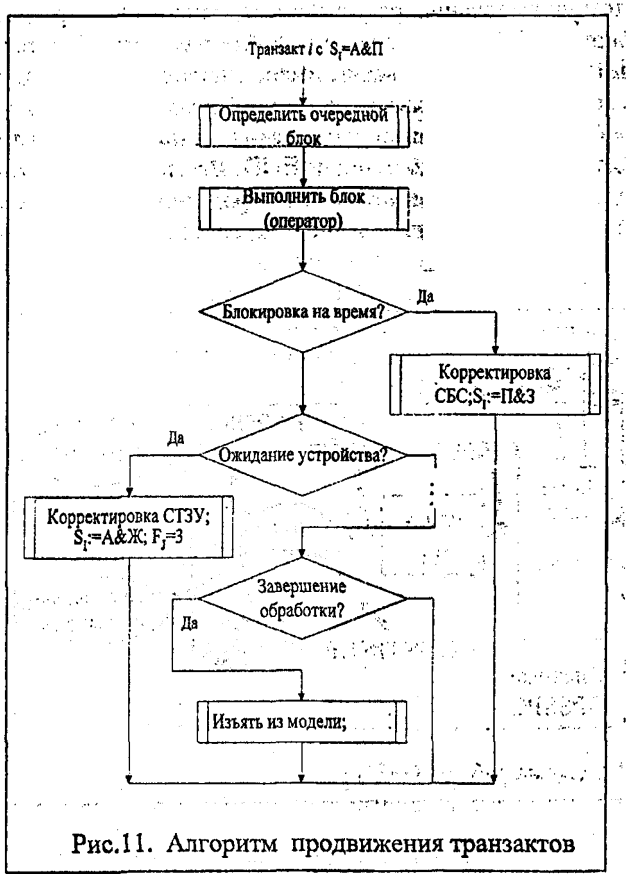


Рис.11. Алгоритм продвижения транзактов

2.6. Характеристика инструментальной среды

Состав инструментальной среды GPSS представлен на рис.12. Она включает систему моделирования GPSSPC и систему подготовки результатов моделирования – редактор отчетов GPSSREPT. Работа начинается запуском системы моделирования командой gpsspc. Появляется заставка, а затем и окно текстового редактора GPSS, позволяющего редактировать загруженный текст или создавать новый. При необходимости текст gpss-модели загружается в редактор (буферную память GPSS) командой @<имя_файла>. Визуализация текста на экране редактора выполняется командой DISPLAY. Сохранение текста модели выполняется командой SAVE. Подготовленный текст модели может выполняться интерпретатором по команде START (поэтому не рекомендуется сохранять в тексте модели команды управления моделированием). Если моделирование начато, то появится сообщение "Simulation in progress". Моделирование можно приостановить командами ESC, STEP, STOP (две последние используются в процессе отладки) и возобновить командой CONTINUE. По окончании моделирования появится сообщение "Simulation complete", а затем "Writing report". При этом по умолчанию результаты моделирования, представленные во внутреннем формате GPSS, сохранятся в файле report.gps. Для сохранения результатов в файле с пользовательским именем используют команду REPORT. Для перевода результатов в пользовательский формат и вывода в нужном виде (на экран, в файл, на печать) командой gpssrept инициируют редактор отчетов. Работа со средой завершается командой END. Все настройки среды, включая установки звука и цвета, состав статистики, формируемой по умолчанию, название файла для ее хранения и т.п. хранятся в файле-конфигураторе settings.gps.

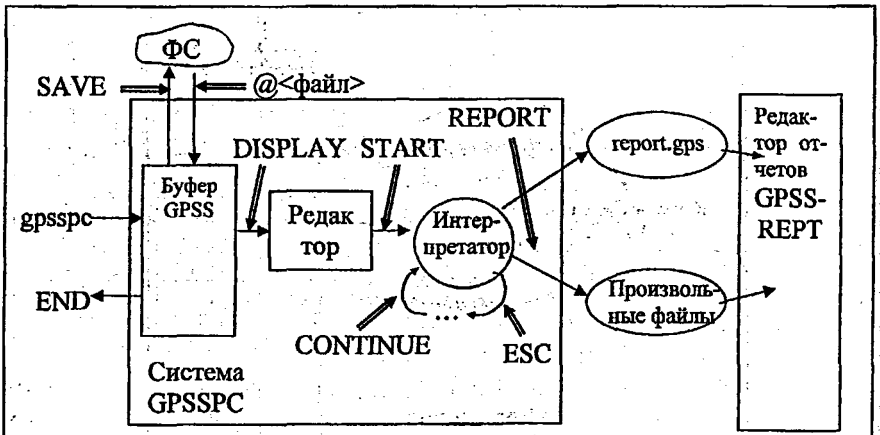


Рис. 12. Схема работы GPSS

Редактор GPSS является текстовым построчным редактором. Он предоставляет пользователю экран просмотра текста модели и нижнюю строку для управления редактированием текста и моделированием. Каждая строка GPSS-текста должна иметь номер, который используется командами редактора. Основные команды работы с файловой системой: SAVE – сохранение, @<имя_файла> – загрузка текста модели. Команды редактирования текста: DELETE – удаление строк, DISPLAY – визуализация строк, EDIT – модификация строки, RENUMBER – перенумерация строк и т.п. Текст скролируется командами PgUp и PgDown. Команды работы со средой: END – завершение работы и выход из среды, START – вызов интерпретатора и начало моделирования. Команды управления интерпретатором: CONTINUE – возобновление моделирования с точки приостановки, STEP и STOP – команды пошаговой отладки, HOME и ESC – команды приостановки моделирования. Команды работы с генератором отчетов: REPORT и PLOT. Для вызова справочной системы в командной строке вводится знак вопроса, для получения справки по команде или ее опции знак вопроса набирается после названия команды или вместо соответствующей опции.

2.7. Стандартный набор статистики GPSS

Интерпретатор автоматически собирает стандартный набор статистики по каждому объекту модели. В то же время, используя средства языка, можно управлять составом статистики, способом ее обработки и формой конечного представления пользователю. В частности, можно расставлять точки сбора статистики, используя операторы QUEUE и DEPART. По окончании моделирования собранные данные из внутреннего формата GPSS преобразуются подсистемой gpssrept в пользовательский формат и предоставляются пользователю в виде отчета. Отчет содержит комментарий (заголовок, который берется из строки текста, предшествующей описанию GPSS-модели), название файла с исходной статистикой, дату и время его создания и сами результаты моделирования, сгруппированные в секции (см. таблицу ниже).

Секция/величина	Назначение	СЧА
Комментарий	Описание модели (берется из строки, предшествующей декларативной части модели)	
СЕКЦИЯ 1 Start time End time Blocks Facilities Storages	ОБЩАЯ СТАТИСТИКА начальное значение времени моделирования конечное значение времени моделирования число блоков, использованных при моделировании число устройств, использованных при моделировании число многоканальных устройств, использованных при моделировании	
СЕКЦИЯ 2 Line	СТАТИСТИКА ИСПОЛЬЗОВАНИЯ БЛОКОВ (ОПЕРАТОРОВ) номер соответствующей строки в программе	

Loc	номер, имя оператора	
Block Type	тип, название оператора	
EntryCount	количество транзактов, прошедших через оператор с начала моделирования либо после последних команд RESET или CLEAR	Nj
Curr.Count	число транзактов в операторе в конце моделирования	Wj
СЕКЦИЯ 3	СТАТИСТИКА ИСПОЛЬЗОВАНИЯ ИМЁН	
СЕКЦИЯ 4	СТАТИСТИКА ИСПОЛЬЗОВАНИЯ ОДНОКАНАЛЬНЫХ УСТРОИСТВ	
Facility	номер, имя устройства	
Entries	количество входов в устройство	FCj
Util.	коэффициент использования (ρj)	FRj
Ave.Time	среднее время обслуживания одного транзакта	FTj
Available	признак доступности (0-занято, 1-свободно)	Fj
СЕКЦИЯ 5	СТАТИСТИКА ОЧЕРЕДЕЙ	
Queue	номер, имя очереди	
Max	максимальная длина очереди	QMj
Count	текущая длина очереди	Qj
Entries	число транзактов, вошедших в очередь	QCj
Entries(0)	число транзактов, прошедших очередь без ожидания (с нулевым ожиданием)	QCj-QZj
Ave.count	средняя длина очереди (lj)	QAj
Ave.time	среднее время ожидания в очереди (ωj)	QTj
Ave(-0)	среднее время ожидания в очереди без учета транзактов с нулевым ожиданием	QXj
СЕКЦИЯ 6	СТАТИСТИКА ИСПОЛЬЗОВАНИЯ МНОГОКАНАЛЬНЫХ УСТРОИСТВ (ПАМЯТЕЙ)	
Storage	номер, имя памяти	
Cap	емкость памяти	
Remain	количество оставшейся свободной памяти	Rj
Min	минимальный объем занятой памяти	SMj
Max	максимальный объем занятой памяти	
Entries	число транзактов, вошедших в память	SCj
Avl.	признак доступности (0-нет, 1-да)	
Ave.c	средний объем занятой памяти	SAj
Util.	коэффициент использования памяти	SRj
	объем занятой памяти (текущий)	Sj(Sj+Rj=Cap)
	среднее время пребывания заявки в памяти	STj
	признак отсутствия транзактов в памяти	SEj
	признак полной занятости памяти	SFj
СЕКЦИЯ 7	СВЕДЕНИЯ О ЗНАЧЕНИЯХ СОХРАНЯЕМЫХ ВЕЛИЧИН	
	содержимое ячейки	Xj
	содержимое элемента матрицы ячеек	MXj(a,b)
СЕКЦИЯ 8	СВЕДЕНИЯ О ВРЕМЕННЫХ ЦЕПЯХ	

2.8. Структура GPSS-модели. Формат операторов

Структура GPSS-модели представлена на рис.13. Она включает описание объекта - текст модели в терминах операторов GPSS и набор команд GPSS,

управляющих запуском, остановкой, приостановкой моделирования, сбором и обработкой статистики. Текст модели может включать описательную, декларативную часть и нескольких исполняемых сегментов (как минимум один). Описательная часть нужна для задания свойств узлов (объектов), используемых в модели, например, оператор STORAGE для задания числа каналов многоканального узла, оператор FUNCTION для описания функций, оператор VARIABLE для описания переменных (выражений) и т.п. Эти описания используются как при инициализации модели так и в ходе моделирования, например, для получения значения ранее описанной функции, для вычисления значения выражения и т.п. Сегменты состоят из операторов, задающих процессы перемещения и обработки транзактов в узлах модели. Они многократно исполняются в ходе моделирования. Каждый сегмент, как правило, начинается оператором генерации транзактов GENERATE и завершается их терминацией, извлечением из сети оператором TERMINATE и описывает маршрут обработки транзактов одного класса.

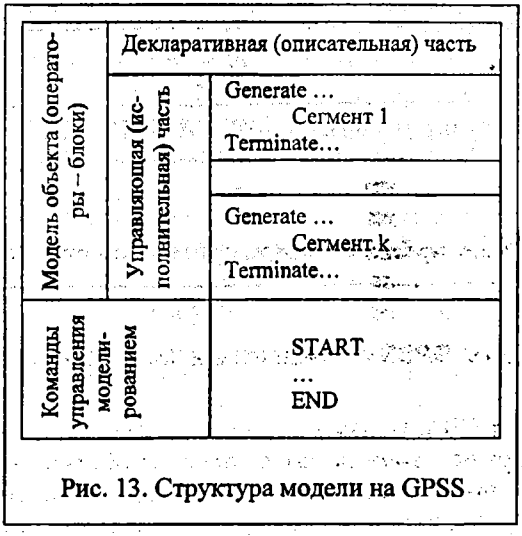


Рис. 13. Структура модели на GPSS

Формат операторов. Операторы GPSS нумеруются, могут иметь метку, состоят из полей (обязательное поле - код оператора) и операндов. Поля отделяются пробелами, операнды, как правило, запятыми. При необходимости, через точку с запятой в строку оператора добавляют комментарий (как правило, в латинском алфавите). Поля и операнды операторов строго форматированы. Это (см. рис.14): L – метка оператора; V – название (мнемонический код) оператора; X – описание проверяемого логического отношения и признаков; Z – описание данных (например, табличное задание функции); A , B, ... , H – описание операндов оператора. Состав полей и операндов в каждом конкретном случае определяется как типом оператора так и контекстом его использования. При редактировании переход от поля к другому полю выполняется по клавише Space. Примеры записи операторов, их полей и операндов приведены ниже:

```
№    L      V          X     A   B ... H          Z
Номер Метка Код оператора        Операнды
 20          Generate           10,  2
 30    S1    Queue              1
 50    HH    Gate       SNF     sklad,  otkaz
 40    EX1   Function           P1,  C4
                0 0/2 1/4 2/5 4
```

Допустимые формы задания значений операндов (обычно это имя, номер, СЧА или СЧА*номер - см. следующий параграф) в каждом конкретном случае следует уточнять по справке.

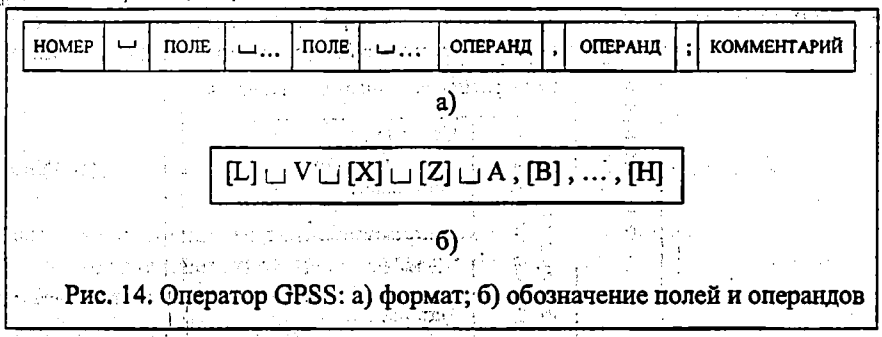


Рис. 14. Оператор GPSS: а) формат; б) обозначение полей и операндов

2.9. Объекты GPSS и стандартные числовые атрибуты

Объекты и СЧА. Основные *объекты GPSS*: - операторы; - узлы, описываемые операторами; - вычислимые и хранимые объекты; - транзакты и др. Состояние любого объекта GPSS в каждый момент модельного времени описывается набором арифметических и логических значений его атрибутов, параметров. Большая часть этих атрибутов используется интерпретатором для организации моделирования, скрыта и недоступна пользователю. Часть атрибутов, называемых **стандартными** *числовыми атрибутами* (СЧА), доступна пользователю и может применяться им в качестве операндов и частей выражений в операторах GPSS для управления моделированием. Основные СЧА операторов и узлов приведены в таблице типовой статистики GPSS в секции 2 и секциях 4-6 соответственно (см. параграф 2.7), а СЧА транзактов описаны ниже.

Каждый транзакт обладает десятками атрибутов, называемых параметрами P. В параметрах могут храниться числовые значения, смысл которых определяется пользователем в соответствии с контекстом модели, в зависимости от характера их использования в модели. Например, в параметрах транзакта можно хранить значение его приоритета, среднюю длительность обработки в ка-

ком-либо узле, номер захватываемого узла и т.д. Так как сам пользователь придает параметрам транзакта определенный смысл, то он же должен заботиться о правильном использовании значений параметров. Обращение к значению i-го параметра текущего транзакта выполняется по его имени как Pi. Например, в операторе SEIZE P1 номер захватываемого узла определяется значением первого параметра транзакта, вошедшего в текущее модельное время в этот узел (оператор). Занесение в параметры транзакта значений выполняется операторами ASSIGN, PRIORITY и др.

Особые объекты – значения сохраняемых (СЧА - Xj) и вычисляемых по заданному выражению величин (СЧА - Vj). Кроме СЧА отдельных объектов используются и системные, относящиеся ко всей модели, например, текущее значение абсолютного модельного времени AC1 и относительного времени C1.

Обращение к СЧА. Для обращения к СЧА конкретного объекта надо указать название СЧА и название (адрес) объекта в виде <название_СЧА><адрес _объекта>. Если объект задан прямой адресацией путем указания его номера или имени, то обращение к СЧА примет вид <название_СЧА><номер_объекта> либо <название_СЧА>$ <имя_объекта> соответственно. Например, P1 – адресует первый параметр текущего транзакта, FN3 – текущее значение функции №3, FN$S1 - текущее значение функции с именем S1, FR10 - текущее значение коэффициента использования устройства №10, FR$QWE - текущее значение коэффициента использования устройства с названием QWE.

При косвенной адресации сам объект указывается через значение параметра текущего транзакта, т.е. транзакта, вошедшего в узел, которому соответствует оператор, где используется этот СЧА. Например, FR*1 - текущее значение коэффициента использования устройства с номером, определяемым значением первого параметра текущего транзакта; оператор SEIZE P*3 – вызывает захват устройства, чей номер задан в третьем параметре пришедшего транзакта; оператор ADVANCE 5,FN*1 – задерживает транзакт на время, распределенное случайно со средним значением 5 по закону, описанному функцией FN с номером, указанным в первом параметре этого транзакта.

2.10. Вычислимые и хранимые объекты GPSS и их СЧА

К числу **вычислимых** и хранимых объектов в GPSS относят встроенные генераторы, сохраняемые величины (ячейки, матрицы), переменные (арифметические и логические), функции (непрерывные, дискретные и т.д.).

Встроенные генераторы, датчики (СЧА RN1÷RN999) имитируют последовательности случайных чисел равномерно распределенных в диапазоне от 0 до 999. А при использовании этих СЧА в качестве аргументов функций случайные значения масштабируются в диапазоне 0-0,999999. Обращение к очередному значению j-ого генератора выполняется соответственно как RNj.

Переменные представляют собой вычислимые объекты, часто используемые в операторах в качестве значений операндов. Обращение к j-ой перемен-

ной или переменной с именем X выполняется соответственно как Vj или V$X. Каждой переменной в GPSS соответствует выражение и при каждом выполнении оператора, в котором упоминается переменная, ее значение является результатом вычисления этого выражения. Связь переменной и выражения устанавливается декларативным оператором VARIABLE. *Выражение* представляет собой совокупность других переменных, параметров объектов и модели (СЧА), целых констант, соединенных знаками операций. Основные знаки операций: +, -, # (умножить), /, @ (взять по модулю), ^ (возвести в степень), \ (делить нацело), 'G' (больше), 'L' (меньше), 'E' (равно), 'NE' (не равно), 'LE' (меньше или равно), 'GE' и др.

Сохраняемые величины могут рассматриваться как частный случай переменных, чье значение инициализируется заранее декларативным оператором INITIAL, а затем может меняться операторами SAVEVALUE и MSAVEVALUE в ходе моделирования. Применяются для хранения и использования значений СЧА, а также исходных параметров модели. Последнее позволяет варьировать значения параметров в ходе экспериментов с моделью, не меняя ее описание. Обращение к j-ой ячейке Xj, а к ячейке j-ой матрицы или матрицы с именем X с номерами m и n соответственно MXj(m,n) и MX$X(m,n).

Функции позволяют описывать табличные зависимости величин с помощью декларативных операторов FUNCTION. Обращение к j-ой функции выполняется как FNj, а к функции с именем X - FN$X.

2.11. Типы адресации и правила образования имен

Типы адресации и типы имен, используемые в качестве операндов операторов GPSS, приведены в таблице ниже.

№	Типы имен	Тип адресации	Формат имени	Пример
1	Числовые	прямая	<Число>	Seize 1
2	Символьные	прямая	<Имя>	Seize ABC
3	Числовые (на базе СЧА)	прямая	СЧА <Число> = СЧА <Номер_СЧА>	Seize FN1
4	Символьные (на базе СЧА)	косвенная	СЧА$ <Имя> = СЧА$ <Имя_СЧА>	Seize FN$ABC
5	На базе параметров транзактов	косвенная	{СЧА*<Число>*<Число>} = {СЧА*<Номер_параметра_транзакта>*<Номер_параметра_транзакта>} СЧА*<Число>*<Число> СЧА*<Номер_параметра_транзакта>*<Номер_параметра_транзакта>	Seize P*1 Seize *1

В GPSS применяют как прямую так и косвенную адресацию объектов. *Прямая адресация* не зависит от параметров пришедшего в узел (оператор) тра-

нзакта и непосредственно указывает на адресуемый объект, так как использует его номер или название (числовое или символьное имя). Соответственно прямая адресация задается числовыми или символьными именами, а также их разновидностью - числовыми и символьными именами на базе СЧА. Например, FN3, P1, FN$S1.

Косвенная адресация для получения реального адреса объекта требует выполнения некоторых промежуточных манипуляций с параметрами пришедшего в узел (оператор) транзакта и соответственно ее результат может меняться в зависимости от конкретной ситуации – от значения параметра транзакта. По универсальности это напоминает использование индексированных переменных и позволяет создавать компактные и универсальные описания моделей. Соответственно косвенную адресацию задают имена на базе параметров транзактов, например FR*1, P*3 , FN*1 и т.д.

Имена. Соответственно бывают: - числовые; - символьные; - числовые на базе СЧА; символьные на базе СЧА; - на базе параметров транзактов.

Числовые имена - это целые положительные числа. Вне зависимости от разработчика модели все объекты GPSS, включая операторы (узлы), нумеруются интерпретатором в соответствии с установками конфигуратора. *Символьные* имена представляют собой комбинации не более чем из 20 цифр, букв латинского алфавита и символа подчеркивания и начинаются с буквы. Такие же правила действуют и при формировании имен меток. *Числовые и символьные имена на базе СЧА* представляют собой комбинацию соответственно числового или символьного имени объекта и названия его СЧА. Так в операторе SEIZE FN1 в качестве захватываемого адресуется устройство с номером, задаваемым значениями функции № 1, а в операторе SEIZE FN$ABC захватывается устройство с номером, задаваемым значениями функции с названием ABC.

Кроме указанных, используются имена вида СЧА*<номер_параметра _транзакта>, формируемые на базе параметров транзактов и задающие косвенную адресацию объектов. Здесь номер объекта вычисляется как значение СЧА с номером, заданным содержимым указанного параметра текущего транзакта, т.е. СЧА*<X>=СЧА[X], где X– номер параметра текущего транзакта, а [X] его содержимое. Например, номер адресуемого объекта (устройства) в операторе SEIZE P*1 определяется числом, которое содержится в P[P1]-ом параметре пришедшего в оператор транзакта, где [P1] – содержимое первого параметра этого транзакта.

Пусть [P1]=5 и [P5]=7, тогда получим из SEIZE P*1 => SEIZE P5 => SEIZE 7; SEIZE P1 эквивалентно SEIZE 5; SEIZE *1 эквивалентно SEIZE 5. Пусть [P3]=1, [P7]=3, [P8]=4, тогда ENTER P*7,*8 эквивалентно оператору ENTER 1,4. Пусть [P1]=5 и [X5]=7, тогда SEIZE X*1 эквивалентно оператору SEIZE X5 и соответственно SEIZE 7, где X5 - сохраняемая величина GPSS.

2.12. Команды и операторы GPSS

Классификация операторов представлена на рис.15. Форматы операторов

и команд с указанием наиболее часто используемых в них операндов представлены ниже.

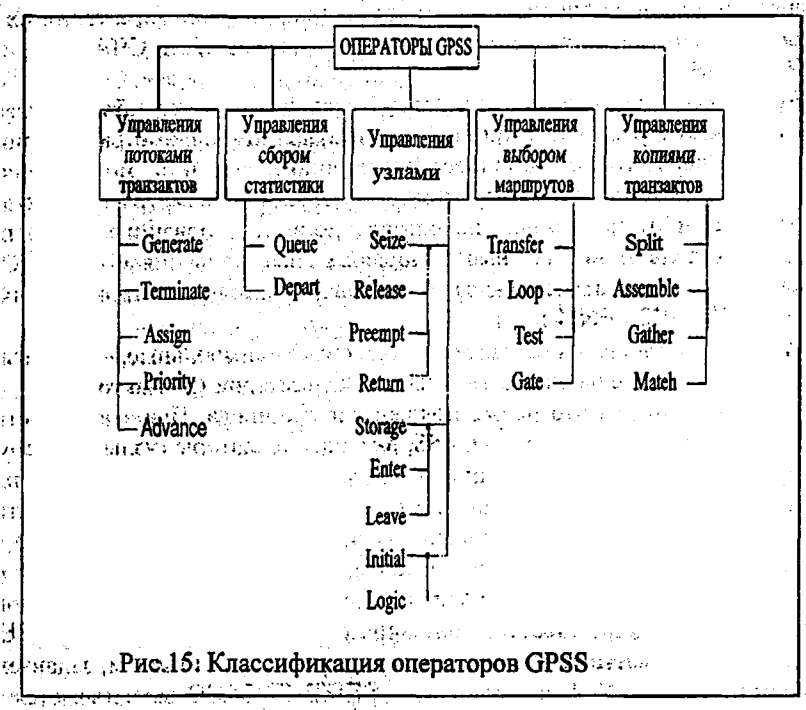


Рис.15. Классификация операторов GPSS

2.12.1. Команды GPSS. Предназначены для управление моделью, созданной на основе операторов. Это команды START, RESET, CLEAR, SIMULATE и др.

START A передает управление интерпретатору, для реализации моделирования текущей модели. Значение операнда A (целое) используется для задания длительности моделирования - начальной установки счетчика завершения модели.

RESET позволяет обнулить собранные статистические данные и значения системных атрибутов. При этом сохраняется текущее значение модельного времени и текущие состояния генераторов случайных чисел (операторов GENERATE, СЧА Rni и т.д.), что позволяет при необходимости исключить статистику переходных процессов.

CLEAR работает так же как предыдущая команда, обнуляя всю статистику и устанавливая генераторы и датчики случайных чисел в исходное состояние.

SIMULATE A задает предельную длительность моделирования A в минутах. Моделирование может быть завершено ранее этого времени при обнулении счетчика завершения модели.

2.12.2. Декларативные операторы. Используются для описания параметров узлов, **вычислимых** объектов: сохраняемых величин (ячеек и матриц), переменных (арифметических и логических), функций.

Переменные описываются декларативным оператором VARIABLE

<имя_переменной> VARIABLE <выражение> ,

который устанавливает связь между переменной и ее выражением. Например, оператор RRR VARIABLE 3#P1 - описывает переменную RRR, значение которой вычисляется как произведение константы 3 на значение первого параметра транзакта (3x[P1]). Оператор RRR1 VARIABLE Q$LLL+30+P12^2 – описывает переменную RRR1, значение которой вычисляется как сумма текущей длины очереди LLL, константы 30 и квадрата значения 12-го параметра транзакта. Пусть имитируется задержка сообщения на время его передачи по линии связи, а ее длительность прямо пропорциональна длине сообщения. Тогда фрагмент модели выглядит как

```
RRR     Variable     3#P1
DLINA   Function ...
          ...
        Generate     10,3
          ...
        Assign       1,FN$DLINA
          ...
        Advance      RRR
          ...
```

Здесь в первом параметре транзакта запоминается длина сообщения, задаваемая функцией DLINA. А в момент входа транзакта в оператор ADVANCE RRR величина задержки вычисляется как 3#P1.

Сохраняемые величины инициализируются оператором INITIAL и изменяются операторами SAVEVALUE и MSAVEVALUE.

INITIAL A, [B] – декларативный оператор, задающий начальное значение B (по умолчанию 1) объекта A (A:=B). В качестве объектов могут использоваться логические ключи (LS), ячейки (Xj), ячейки матриц (MXj(m,n) или MX$X(m,n)).

SAVEVALUE A[±], B – оператор, изменяющий текущее значение объекта A (A:=B или A:= A± B). В качестве объектов могут использоваться ячейки сохраняемых величин (Xj).

MSAVEVALUE A[±], B – аналогичный оператор, где в качестве объектов могут использоваться ячейки матриц (MXj(m,n) или MX$X(m,n)).

Пример

```
Initial     X12, 120
Initial     MX3(1,1), -33
Savevalue   1+, 1
```

Функции позволяют описывать табличные зависимости величин с помощью декларативных операторов FUNCTION

<имя_функции> FUNCTION A, B Z .

Здесь операнд A задает аргумент функции, операнд B описывает тип функции (тип аппроксимации) и задает число пар значений (точек), используе-

мых при ее описании в виде конструкции <тип_аппроксимации><число_точек>. Допустимо задание функций: - с кусочно-линейной (непрерывной); ступенчатой (дискретной) аппроксимацией; табличное, точечное задание функции без аппроксимации; задание дискретных атрибутивных; и табличных атрибутивных функций, с <типом_аппроксимации> = *C / D / L / E / M* соответственно. Пары значений задаются в виде <Z> = $\{x_i, f_{X_i}\}$. На рис.16 приведены примеры функций GPSS с разными типами аппроксимации. Так функция EX1 с кусочно-линейной аппроксимацией (тип C) отображает зависимость времени передачи сообщения FN$EX1 от его длины P1

```
EX1   Function  P1, C4
      0,0/2,1/4,2/5,4
```

Ступенчатая (тип D) функция EX2 отображает зависимость приоритета FN$EX2, устанавливаемого транзакту, от текущей длины очереди Q$AB

```
EX2   Function  Q$AB, D4
      0,0/2,1/4,2/5,4
```

Функция EX3 без аппроксимации (тип L) отображает зависимость значения FN$EX3 от значения 5-го параметра текущего транзакта

```
EX3   Function  P5, L4
      0,0/2,1/4,2/5,4 .
```

Одно из типовых применений оператора FUNCTION состоит в описании функций распределения случайных величин, используемых в модели, с целью генерации их значений. Генерация основана на методе обратных функций (см. параграф 6.7) и иллюстрируется на рис.17. Пусть требуется генерировать случайные числа X, подчиняющиеся заданному таблицей закону распределения. По табличным значениям можно построить функцию распределения FX(x) и соответствующую ей обратную функцию $F_X^{-1}(x)$. Обратная функция описана ниже как функция EX4, у которой в качестве аргументов используются числа ω_i (значения генератора RN1), равномерно распределенные в диапазоне 0-1. Значения этой функции FN$EX3, вызываемые и используемые в операторе задержки, и дают случайные числа с заданным распределением

```
EX4     Function  RN1, C5
        0,0/0.2,2/0.5,5/0.7,8/0.999,12
        . . .
        Advance   FN$EX3
```

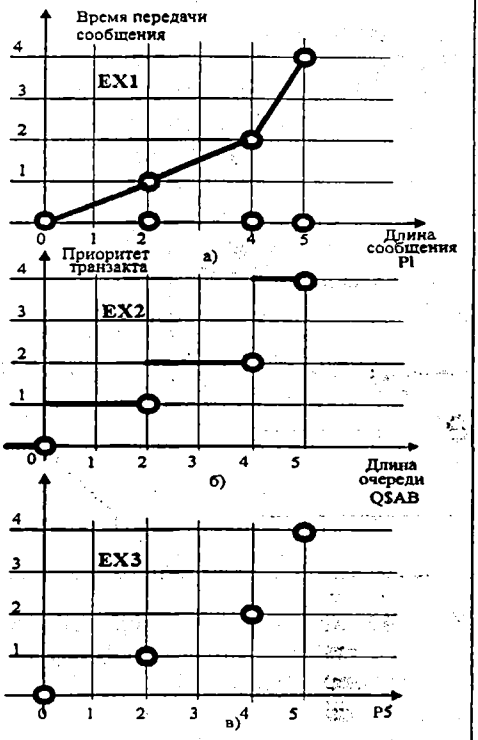


Рис.16. Примеры графиков GPSS с разными типами аппроксимации: а) кусочно-линейная (тип C); б) ступенчатая (тип D); в) без аппроксимации (тип L)

На рис.18 показана обратная функция для генерации чисел, равномерно распределенных в диапазоне 2÷6. Она описана как функция EX5, у которой в качестве аргумента используются значения генератора RN2. Ее значения FN$EX4 - случайные числа с заданным распределением

```
EX5    Function  RN2, C2
       0,2/0.999,6
         . . .
       Assign    1,FN$EX4
         . . .
```

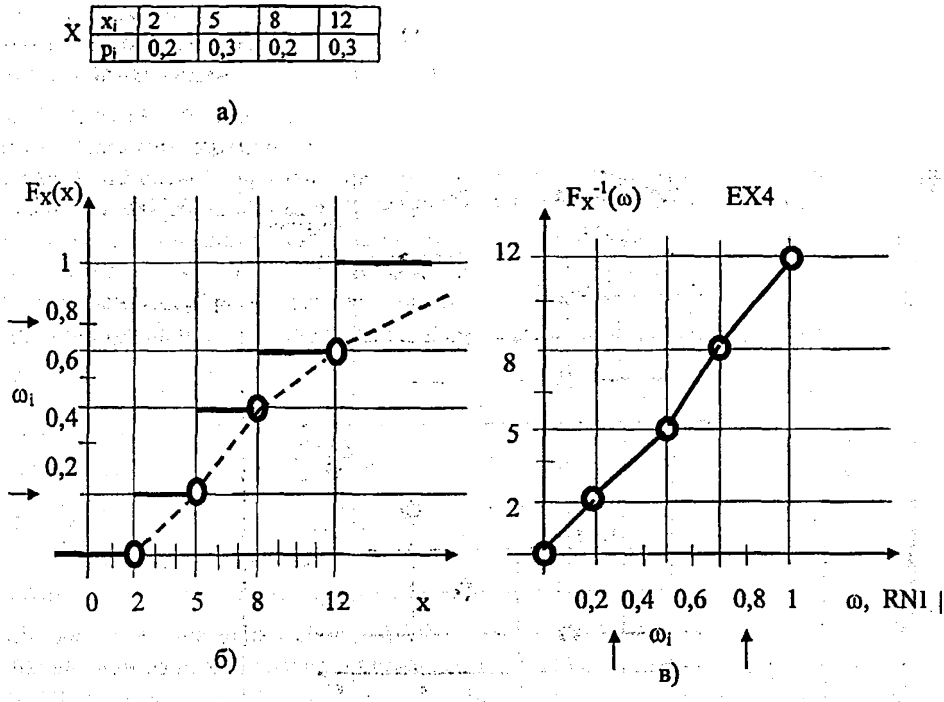


Рис.17. Иллюстрация метода обратных функций: а) таблица распределения случайной величины X; б) закон распределения $F_X(x)$; в) обратная функция $F_X^{-1}(x)$

2.12.3. Операторы управления потоками транзактов. Включают операторы GENERATE, TERMINATE, ASSIGN, PRIORITY, ADVANCE и др.

GENERATE [A], [B], [C], [D], [E] – генерирует поток однородных заявок (транзактов) с заданным пользователем законом поступления - законом распределения промежутков времени между их появлениями. Операнд A задает время или среднее время, через которое появляется очередной транзакт, B – модифицирует это время, придавая ему случайный характер. Если B – целое число, то время появления транзактов подчиняется равномерному закону распределения в диапазоне значений A ± B. Если B – ссылка на функцию (например, функцию распределе-

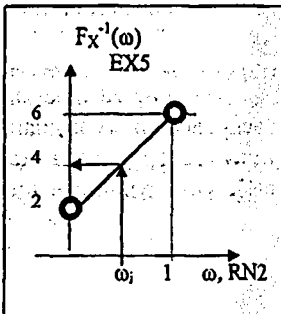


Рис.18. Обратная функция для генерации чисел, равномерно распределенных в диапазоне 2÷6

ния), то время, через которое появится очередной транзакт, определяется произведением A*B среднего значения A на текущее значение функции B, например, A*FN1. Это позволяет использовать метод обратных функций для генерации соответствующих значений. При отсутствии модификатора имитируется регулярный поток. Операнд C (по умолчанию ноль) задает время задержки появления первого транзакта или время, с которого начинается работа GENERATE. Операнд D – предельное число транзактов, генерируемых в ходе моделирования (по умолчанию не ограничено), E – приоритет транзактов (по умолчанию ноль). При отсутствии операндов A и B генерируется серия из D транзактов, т.е. генератор работает как групповой. Следующим после GENERATE должен быть оператор, невносящий задержек, т.к. в противном случае генерация всех последующих транзактов приостанавливается. На оператор GENERATE нельзя ссылаться.

TERMINATE [A] – оператор терминации, выполняющий функции приемника заявок. Он извлекает транзакт из модели и уменьшает значение счетчика завершений на значение A.

ASSIGN A, B, [C] – позволяет задавать или модифицировать текущее значение параметра транзакта с номером A. Новое значение задается операндом B и его модификатором C.

PRIORITY A – присваивает каждому пришедшему транзакту приоритет A (целое число – 0...127).

ADVANCE A, [B] – выполняет задержку транзакта на время, подчиняющееся заданному пользователем закону распределения. Транзакт становится пассивным и помещается в цепь будущих событий. Как и в операторе GENERATE операнд A задает время или среднее время, на которое задерживается транзакт, а B – модифицирует это время с использованием метода обратных функций.

Примеры использования операторов приведены ниже: GENERATE 5,,10 – по истечении 10 единиц времени начинает генерироваться регулярный поток заявок (новая заявка приходит ровно через 5 единиц времени); GENERATE 5,3,,,2 – транзакты распределены по равномерному закону в диапазоне от 2 до 8 единиц времени с приоритетом 2; GENERATE ,,,10,5 – групповой генератор серии из 10 заявок с приоритетом 5; GENERATE 5,FN$EXP,,,2 – транзакты распределены по закону, задаваемому значениями FN$EXP функции EXP, и появляются через отрезки времени 5*FN$EXP с приоритетом 2; ASSIGN 1,2 – записывает в первом параметре текущего транзакта значение два; ASSIGN 1+,2 – увеличивает значение первого параметра текущего транзакта на значение два; ASSIGN 2,20,FN$DELAY – записывает в втором параметре текущего транзакта случайное значение 20*FN$DELAY (например, время задержки, распределенное по закону, описанному функцией DELAY; среднее значение 20). Фрагмент модели, представленный ниже, иллюстрирует использование косвенной адресации и параметров транзактов. Первый параметр хранит номер устройства, второй – длительность обслуживания в устройстве

```
Generate    5,3
Assign      1,2
```

```
Assign      2,3
Seize       *1
Advance     *2
Release     *1
Terminate   .
```

2.12.4. Операторы управления сбором статистики. В GPSS для сбора статистики используется специальный объект – очередь (не путать с очередью в ТМО!). Очередь GPSS – это пара контрольных точек в модели, в которых будут отмечаться все проходящие через них транзакты и будут собираться статистические данные. Пользователь сам расставляет точки (очереди) с помощью операторов

QUEUE A – войти в очередь A (отметиться в первой контрольной точке очереди A).

DEPART A – покинуть очередь A (отметиться во второй контрольной точке очереди A).

Особенности использования очередей: один и тот же транзакт может находиться (отмечаться) одновременно в нескольких очередях; транзакты входят в очереди без ограничений; очереди не создают задержек и не влияют на логику работы модели. Основная статистика, связанная с очередями, представлена в таблице, секция 5 (см. параграф 2.7). Это: число транзактов QCj, вошедших в очередь; число транзактов QZj, прошедших очередь без ожидания; средняя длина очереди Qaj; среднее время ожидания в очереди QTj (ωj); среднее время ожидания в очереди без учета транзактов с нулевым ожиданием QXj.

На рис.19 приведен фрагмент модели однородной одноканальной СМО с очередями. В модели выделены три точки a, b, c и собирается статистика трех очередей. Первой очереди соответствует пара точек - a, b. Она фиксирует статистику пребывания заявок в состоянии ожидания обслуживания. Второй очереди соответствует пара точек - b, c. Она фиксирует статистику пребывания заявок в состоянии обслуживания. Третьей очереди соответствует пара точек - a, c. Она фиксирует статистику пребывания заявок в системе, включая и ожидание и обслуживание. Фрагмент модели представлен ниже

```
Generate    10,2
Queue       1
Queue       3
Seize       S
Depart      1
Queue       2
Advance     3,1
Depart      2
Depart      3
Terminate   1   .
```

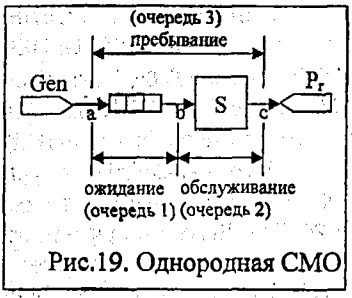


Рис.19. Однородная СМО

2.12.5. Операторы управления обслуживающими узлами. В GPSS используют два основных типа обслуживающих узлов: одноканальный узел типа "устройство" с возможностью организации приоритетного обслуживания и многоканальный узел типа "память". Узлы можно захватывать и освобождать,

а задержки транзактов в процессе обслуживания имитируют оператором ADVANCE.

Одноканальные узлы. Их математическим эквивалентом служит одноканальная система массового обслуживания. Практикуют два режима использования устройств, две дисциплины обслуживания: - без приоритетную дисциплину FIFO; - приоритетную дисциплину с возможность прерывания обслуживания текущей заявки (т.е. с учетом относительных и абсолютных приоритетов). В одной модели можно использовать оба режима.

Без приоритетный режим реализуется операторами:

SEIZE A – захватить устройств номером A;

RELEASE A – освободить устройство с номером A;

Приоритетное обслуживание реализуется операторами:

PREEMPT A – захватить устройство с номером A;

RETURN A – освободить устройство с номером A.

Состояние каждого устройства отображается значениями СЧА FCj, FRj, FTj, Fj и т.д., а также набором признаков U, NU, I, NI, FV, FNV и др. (см. параграф 2.7, таблица - секция 4). Поэтому косвенно устройством можно также управлять с помощью операторов GATE – маршрутных узлов, позволяющих проходящему транзакту анализировать состояние устройства и выбирать дальнейший маршрут.

Многоканальные узлы. Их математическим эквивалентом может служить многоканальная система массового обслуживания. Все каналы считаются идентичными. При работе с ними используют декларативный и управляющие операторы. Декларативный оператор предшествует операторам SIMULATE или первому оператору GENERATE и описывает емкость памяти в виде числа каналов A

<имя_памяти> STORAGE A

Управление реализуется операторами:

ENTER A, [B] – в многоканальном узле A захватить B (по умолчанию один) каналов одновременно;

LEAVE A, [B] – в многоканальном узле A освободить B (по умолчанию один) каналов одновременно.

Многоканальный узел может применяться для имитации как многоканальных СМО так и памятей. Процесс имитации СМО иллюстрируется рис.20, а. Соответствующий фрагмент модели приведен ниже

```
S       Storage     K
  ...
Enter       S,1
Advance     Δt
Leave       S,1
  ...
```

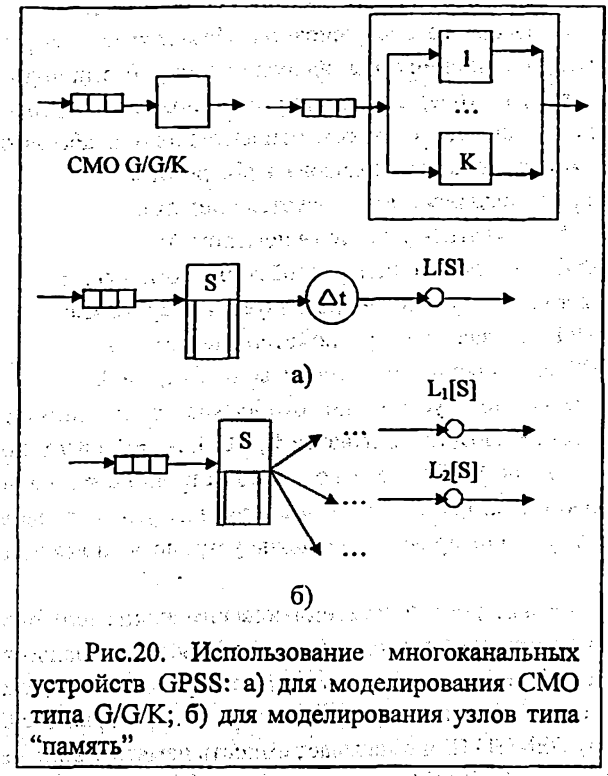


Рис.20. Использование многоканальных устройств GPSS: а) для моделирования СМО типа G/G/K; б) для моделирования узлов типа "память"

Процесс имитации памяти иллюстрируется рис.20, б. Сама память в отличие от СМО задержек не вносит. Задержки зависят от параметров других узлов модели. Соответствующий фрагмент приведен ниже

```
S       Storage     K
 ...
Enter       S,r
 ...
Leave       S,...
 ...
  Leave     S,...
   ...
```

Особенности работы многоканального устройства: - транзакт не может войти в память, если ему не хватает каналов или память недоступна; - транзакт, которому хватает каналов, может "обойти" тот транзакт, которому их не хватает; - транзакты могут освобождать не то количество каналов, которое они захватили; - нельзя освободить каналы, которые не были заняты.

Состояние устройства отображается значениями СЧА Rj, SMj, SCj, Saj, SRj, Sj, STj, SEj и т.д., а также набором признаков SF, SNF, SE, SNE и др. (см.

параграф 2.7, таблица - секция 6). Поэтому косвенно устройством можно также управлять с помощью операторов GATE.

2.12.6. Операторы управления маршрутами. Транзакты перемещаются в пределах своего сегмента от одного узла модели к другому в соответствии с последовательностью расположения в тексте модели операторов, соответствующих этим узлам. При необходимости, используя средства управления движением транзактов, пользователь может нарушить последовательное выполнение операторов друг за другом и направить транзакт по нужному маршруту.

Управление маршрутизацией в GPSS включает средства: - безусловного перемещения транзактов в узел с указанной меткой (оператор TRANSFER); - вероятностного выбора маршрута на базе имеющейся или предполагаемой статистики движения транзактов (оператор TRANSFER); - организации циклических маршрутов с детерминированным или вероятностным числом повторений (операторы TRANSFER, LOOP и др.); - выбора маршрута в зависимости от текущего состояния модели, узлов модели и транзактов, отображаемых в СЧА, сохраняемых величинах и переменных (оператор TEST и в частных случаях оператор TRANSFER); - выбора маршрута в зависимости от текущего состояния основных обслуживающих узлов, например одноканальных и многоканальных (оператор GATE как частный случай оператора TEST); - "семафорные" средства управления (логические ключи в роли переключаемого семафора и операторы GATE в роли анализатора выбора маршрута в зависимости от текущего состояния "семафора"); - средства управления копиями (семействами) транзактов (операторы SPLIT, ASSEMBLE, GATHER, MATCH).

TRANSFER [A], [B], [C], [D], [E] – используется в разных модификациях, в частности, определяемых значением первого операнда. Основные из них описаны ниже.

TRANSFER ,B – маршрутный узел, безусловно отправляющий транзакт к оператору с меткой, указанной операндом B.

TRANSFER .<число>, B, C – вероятностный маршрутный узел, отправляющий транзакт с вероятностью, заданной в операнде A как .<число>, в узел с меткой, указанной операндом C. С вероятностью 1-.<число> транзакт отправляется в узел с меткой, указанной операндом B. Значение вероятности .<число> может быть задано непосредственно как доля единицы или с использованием косвенной адресации.

TRANSFER BOTH, B, C – оператор, пытающийся провести транзакт через узел с меткой, указанной операндом B. В случае неудачи безусловно отправляет транзакт в узел с меткой, указанной операндом C.

Примеры использования операторов: TRANSFER ,next – переход транзакта к оператору с меткой next; TRANSFER .7, AA, AA1 – переход транзакта с вероятностью 0,7 к оператору с меткой AA1 или с вероятностью 0,3 к оператору с меткой AA; TRANSFER .P1, AA, AA1 – переход транзакта с вероятностью, задаваемой его первым параметром, к оператору с меткой AA1 или с вероятностью 1-,[P1] к оператору с меткой AA; TRANSFER BOTH, B1, C1 – попытка перехода

транзакта к оператору с меткой B1, а в случае невозможности к оператору с меткой C1. Фрагмент модели замкнутой однородной сети из двух одноканальных СМО (рис.21) приведен ниже

```
        Generate    ,,,10
WW      Seize       S1
        Advance     5, 1
        Release     S1
        Seize       S2
        Advance     5, 2
        Release     S2
        Transfer    ,ww .
```

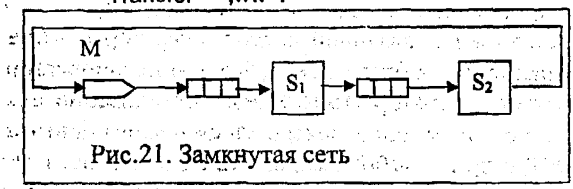


Рис.21. Замкнутая сеть

Фрагменты двух вариантов построения модели неоднородной одноканальной СМО (рис.22) приведены ниже. Первый фрагмент включает два сегмента, каждый сегмент описывает процесс обслуживания одного однородного потока транзактов, а параметры обслуживания присутствуют в модели как константы. Во втором фрагменте параметры обслуживания хранятся в параметрах транзактов (первый параметр хранит номер очереди для сбора статистики, второй - длительность обслуживания).

```
Generate   10,2
Queue      1
Seize      1
Depart     1
Advance    5
Release    1
Terminate  1

Generate   5,1        S1
Queue      2
Seize      1
Depart     2
Advance    2
Release    2
Terminate  1
```

```
Generate   10, 2
Assign     1, 1
Assign     2, 5
Transfer   ,S1

Generate   5, 1
Assign     1, 2
Assign     2, 2
Queue      *1
Seize      1
Depart     *1
Advance    *2
Release    1
Terminate  1
```

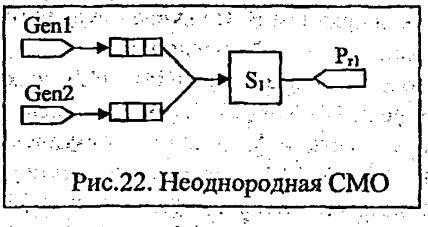


Рис.22. Неоднородная СМО

Ниже представлен фрагмент модели разомкнутой однородной сети из вух последовательно соединенных одноканальных СМО. После обслуживания первой СМО заявки могут двигаться по одному из трех маршрутов: - с вероятностью 0,5 возвращаются в первую СМО на дообслуживание ; - с вероятностью 0,45 переходят на обслуживание во вторую СМО; с вероятностью 0,05 завершают обслуживание. Вероятностный выбор маршрутов реализуется двумя операторами TRANSFER .5, S1 и TRANSFER .1, S2, PR

```
          Generate  10, 2
S1        Queue     1
          Seize     1
          Depart    1
          Advance   5
          Release   1
          Transfer  .5, S1
          Transfer  .1, S2, PR
S2        Queue     2
          Seize     2
          Depart    2
          Advance   5, 1
          Release   2
          Transfer  ,S1
PR        Terminate       1
```

С помощью этого оператора организуют и циклическое продвижение транзакта по некоторому участку модели. Для этих целей используют и специальный оператор.

LOOP A, B - служит для организации циклов. Операнд А задает число повторений (номер параметра транзакта), а В – метку начального оператора цикла. Во фрагменте ниже вызывается 10-кратный повтор задержки транзакта

```
          Assign    4, 10
RRR       Advance   5, 3
          Loop      4, RRR
```

TEST X A, B, [C] – определяет направление дальнейшего движения транзакта в зависимости от результатов проверки заданного условия ([A] X [B] = ИСТИНА - ?). Здесь X задает вид проверки, отношения (G, GE, NE, E, LE, L), а значения операндов A, B - сравниваемые величины. Если результат проверки – ИСТИНА, то транзакт отправляется в следующий по порядку оператор (узел). В противном случае транзакт останавливается на входе в оператор TEST (становится задержанным, перестает продвигаться по сети и ждет, когда проверяемое условие удовлетворится), либо, если использован операнд С, отправляется в узел с указанной в нем меткой. Во фрагменте ниже сравнивается число каналов, требуемое пришедшему транзакту (содержимое его 7-го параметра), с числом оставшихся свободных каналов в многоканальном узле SKLAD, т.е. анализируется условие [P7] <= R$SKLAD. При наличии необходимого числа каналов происходит их занятие, а в противном случае транзакт уходит к оператору с меткой SKLAD1

```
          Test    'LE'   P7, R$SKLAD, SKLAD1
          Enter          SKLAD, P7
              ...
```

SKLAD1 Enter SKLAD1, P7 .

GATE X A, [B] – определяет маршрут дальнейшего движения транзакта в зависимости от текущего состояния X у указанного объекта (узла) A (т.е. X[A] = ИСТИНА - ?). Поле X задает тип проверяемого состояния (см. таблицу ниже)

ЛОГИЧЕСКИЕ КЛЮЧИ	
LS	ключ включен
LR	ключ выключен
УСТРОЙСТВА	
U	устройство используется (занято)
NU	устройство не используется (свободно)
ПАМЯТЬ	
SF	память заполнена
SNF	память не заполнена
SE	память пуста
SNE	память не пуста

устройства, многоканального устройства, логического ключа, синхронизируемого транзакта, а операнд A задает его адрес. Если результат проверки истина, то транзакт отправляется в следующий по порядку оператор (узел) модели. В противном случае транзакт задерживается на его входе (перестает продвигаться и ждет, когда узел примет необходимое состояние) либо, если использован операнд B, отправляется в узел с указанной там меткой. Во фрагменте ниже оператор GATE проверяет заполненность памяти TTT, т.е. условие SNF[TTT] = ИСТИНА - ? При положительном ответе происходит занятие памяти, в противном случае транзакт переходит к оператору с меткой BL2

```
        Gate   SNF    TTT, BL2
        Enter         TTT
        Seize           1
          ...
BL2     ...  .
```

В следующем примере каждая новая деталь поступает через 20±12 единиц времени (оно распределено по равномерному закону) сначала на склад, а затем на станок к рабочему и обрабатывается им в течение 19±5 единиц времени (время распределено по равномерному закону). Емкость склада – 100 деталей. Требуется промоделировать процесс обработки 1000 деталей и определить также вероятность отказа в обработке деталей из-за переполненности склада.

```
SKLAD   Storage     100
        Generate    20, 12
        Gate  SNF   SKLAD, OTKAZ
        Enter       SKLAD, 1
        Seize       1
        Leave       SKLAD, 1
```

```
          Advance      19, 5
          Release      1
          Terminate    1
OTKAZ     Savevalue    1+, 1
          Terminate    1
          START        1000
```

Во фрагменте ниже предыдущая модель модифицирована: длительность обслуживания задается средним значением, указанным в операторе задержки ADVANCE 10, FN$DELAY и распределения, описанной функцией DELAY FUNCTION RN1, C2 (здесь равномерное в диапазоне 0-6)

```
DELAY     Function     RN1,C2
          0,0/0.999,6
SKLAD     Storage      100
          Generate     20,12
          Gate    SNF  SKLAD, OTKAZ
          Enter        SKLAD, 1
          Seize        1
          Leave        SKLAD,1
          Advance      10, FN$DELAY
          Release      1
          Terminate    1
OTKAZ     Savevalue    1+, 1
          Terminate    1
          START        1000
```

2.12.7. "Семафорные" средства маршрутизации предназначены для имитации управления в модели, организации взаимодействия узлов и транзактов. Они основаны на использовании логических ключей в роли переключаемого "семафоров" и операторов GATE для выбора маршрута в зависимости от их состояния. *Логический ключ* – объект с двумя состояниями: "установлен в 1" (set) или "сброшен в 0" (reset). При инициализации модели по умолчанию все ключи выключаются. Управление ключами производится операторами LOGIC и INITIAL.

LOGIC X A – устанавливает ключ A в состояние X (S – для включения, R – для выключения, I – для инвертирования текущего состояния ключа). Транзакты, проходя через оператор LOGIC, могут менять состояние указанного ключа. Другие транзакты могут проверить его состояние с помощью оператора GATE. Например, в фрагменте ниже оператор GATE проверяет, сброшен ли ключ LOCK, т.е. условие LR [LOCK] = ИСТИНА - ? При положительном ответе происходит занятие устройства с номером 1, а в противном случае транзакт переходит к оператору с меткой BL2

```
          Gate   LR  LOCK, BL2
          Seize        1
            ...
BL2       ... .
```

2.12.8. Операторы управления копиями транзактов. В GPSS можно создавать копии (семейства) транзактов и имитировать их параллельную обра-

ботку. Управляют копиями с помощью операторов SPLIT, ASSEMBLE, GATHER, MATCH.

SPLIT A, [B], [C] – создает семейство (дополнительно А копий) текущего транзакта и отправляет их к оператору с меткой B, а при его отсутствии к следующему по порядку оператору. Операнд С задает номер параметра транзакта для хранения номера копии. Номера присваиваются копиям последовательно.

ASSEMBLE A – объединяет А копий одного семейства (для каждого семейства) в один транзакт, который далее движется вместо копий.

GATHER A – ждет прихода А копий одного семейства (для каждого семейства), которые затем одновременно продолжают движение по сети.

В MATCH A в паре с другим оператором A MATCH B – синхронизирует движение двух транзактов (копий одного семейства) с целью обеспечения одновременного прохождения ими указанных точек модели. Операнд А является меткой сопряженного с ним оператора A MATCH B. Обе копии продолжат движение только после того как и та и другая достигнет своего оператора MATCH.

Пример: новая деталь поступает через 20±12 единиц времени (время распределено по равномерному закону) и обрабатывается параллельно двумя рабочими каждым в течение 19±5 единиц времени (время распределено по равномерному закону)

```
            Generate    20, 12
            Split       1, WORKER_2
WORKER_1    Seize       1
            Advance     19, 5
            Release     1
            Transfer    ,JOIN
WORKER_2    Seize       2
            Advance     19, 5
            Release     2
JOIN        Assemble    2
            Terminate   1
            START       1000
```

Пример: новая деталь поступает через 20±12 единиц времени. Деталь обрабатывается параллельно двумя рабочими в два этапа. Сначала каждым в течение 10±5 единиц времени, а затем, после сверки результатов, еще каждым из них параллельно в течение 5±5 единиц времени. Все времена распределены по равномерному закону.

```
            Generate    20, 12
            Split       1, WORKER_2
WORKER_1    Seize       1
            Advance     10, 5
A           Match       B
            Advance     5, 5
            Release     1
            Transfer    ,JOIN
WORKER_2    Seize       2
            Advance     10, 5
B           Match       A
            Advance     5, 5
```

```
        Release     2
JOIN    Assemble    2
        Terminate   1
        START       1000
```

2.13. Приоритетное обслуживание

Одноканальные узлы можно не только захватывать и освобождать, но и прерывать обслуживание текущего транзакта, если его приоритет оказался ниже, чем у пришедшего транзакта. В последнем случае приоритеты транзактов трактуются как абсолютные. Прерванный транзакт ждет освобождения узла, либо отказывается от попытки возобновить обслуживание и отправляется в другую точку модели, либо, отправляясь в другую точку модели, не отказывается от попытки возобновить обслуживание в этом узле. Для прерванного транзакта, ждущего продолжения обслуживания, интерпретатор вычисляет остаток времени дообслуживания и помещает его в список прерванных транзактов к данному устройству. При возобновлении обслуживания транзакт может быть прерван снова не ограниченное число раз. Нет ограничений и на число устройств, используемых транзактом одновременно, а следовательно и на возможность одновременного прерывания транзакта многократно на разных устройствах. Но нельзя прерывать устройство подряд одним и тем же транзактом.

PREEMPT A, [B], [C], [D], [E] – приостанавливает обслуживание текущего транзакта в устройстве **A** при наличии условий для режима прерываний, заданного операндом **B**. По умолчанию используется "общий" режим прерываний, когда допустимо только одноуровневое прерывание. Это означает, что если текущий транзакт был прерван либо устройство было занято с помощью оператора PREEMPT без прерывания транзакта, то новый транзакт нельзя прервать до конца его обслуживания. Если B=PR, то используется "приоритетный" режим прерываний, допускающий многоуровневые прерывания. Т.к. в GPSS используют приоритеты со значениями от 0 до 127, то возможно 127 уровней прерывания. Операнды C, D, E используются только во втором режиме. Операнд C задает номер альтернативного оператора модели, куда отправляется прерванный транзакт, теряя управление устройством A. В параметре транзакта, указанном в операнде D, хранится время дообслуживания. Значение операнда E=RE задает режим удаления транзакта из списка прерванных и перемещение его по метке, указанной в операнде C.

RETURN A – освобождает устройство A, ранее захваченное с помощью оператора PREEMPT.

Пример: требуется промоделировать неоднородную одноканальную СМО, обслуживающую транзакты пяти классов с одинаковыми параметрами обслуживания, но разными приоритетами, возрастающими с ростом номера класса. Транзакты поступают через 20 ± 12 единиц времени и обрабатывается в течение 10 ± 5 единиц времени. Все времена – случайные величины, подчиняющиеся равномерному распределению. Транзакты первого и второго класса имеют наименьшие относительные приоритеты 1 и 2, обслуживаются с учетом

приоритетов, но не могут вызывать прерываний. Транзакты 3-5 классов имеют более высокие абсолютные приоритеты, т.е. они могут прерывать обслуживание транзактов более низкого приоритета. Причем, транзакты 3 и 4 классов могут быть прерваны только однажды.

```
        Generate    20, 12,,, 1
        Seize       1
        Advance     10, 5
        Release     1
        Transfer    ,JOIN
        Generate    20,12,,, 2
        Seize 1
        Advance     10, 5
        Release     1
        Transfer    ,JOIN
        Generate    20,12,,, 3
        Preempt     1
        Advance     10, 5
        Return      1
        Transfer    ,JOIN
        Generate    20,12,,, 4
        Preempt     1
        Advance     10, 5
        Return      1
        Transfer    ,JOIN
        Generate    20,12,,, 5
        Preempt     1, PR
        Advance     10, 5
        Return      1
JOIN    Terminate   1
        START       1000
```

3. Аналитические методы расчета ССМ и их частей

Как было показано в Главе 2, § 1 точному аналитическому расчету поддаются линейные однородные экспоненциальные сети массового обслуживания (далее просто экспоненциальные сети). В них: - обслуживаются заявки одного класса; - основной обслуживающий узел СМО типа G/M/1 и G/M/K с временем обслуживания заявок в каждом канале распределенным по экспоненциальному закону; - основной маршрутный узел вероятностный; - входной поток заявок в сети простейший, число заявок в потоке распределяется по закону Пуассона, а отрезки времени между соседними заявками потока распределены экспоненциально.

3.1. СМО. Параметры, характеристики, классификация

Параметры СМО. Любую систему, в которой поток заявок встречает ограниченные средства их обслуживания, можно рассматривать как СМО. Расчет характеристик СМО базируется, в частности, на теории марковских цепей.

Основные компоненты СМО (рис.23): 1. Обслуживающие каналы. Задаются количеством (канальностью) K и быстродействием канала V. Все каналы полностью идентичны. 2. Очередь, в которой заявки могут накапливаться, ожидая обслуживания. Размер варьируется от 0 до ∞ (для СМО типа M/M/1 и M/M/K). 3. Дисциплина обслуживания ДО, определяющая правила выбора на обслуживание очередной заявки из очереди (рис.24). 4. Входной поток или потоки заявок. Характеризуются законом распределения f_Θ трудоемкости Θ обслуживания заявок и законом распределения f_τ временных отрезков между соседними заявками потока

$$\tau_k = \tau_{nk} - \tau_{nk-1}$$

Если τ_K постоянна, то поток регулярный (детерминированный). В противном случае поток случайный и характеризуется законом распределения

$$F_k(\tau) = P[\tau_k \le \tau] = F(\tau)$$

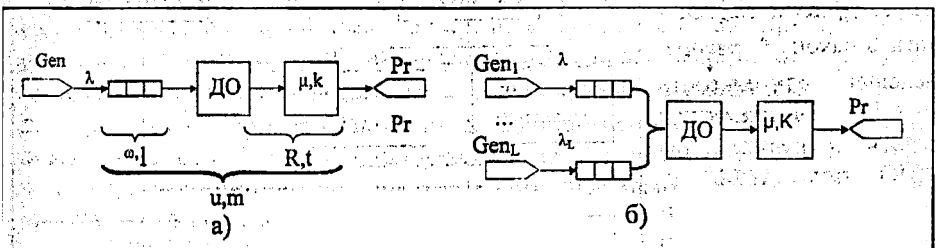


Рис.23. Системы массового обслуживания: а) однородные; б) неоднородные

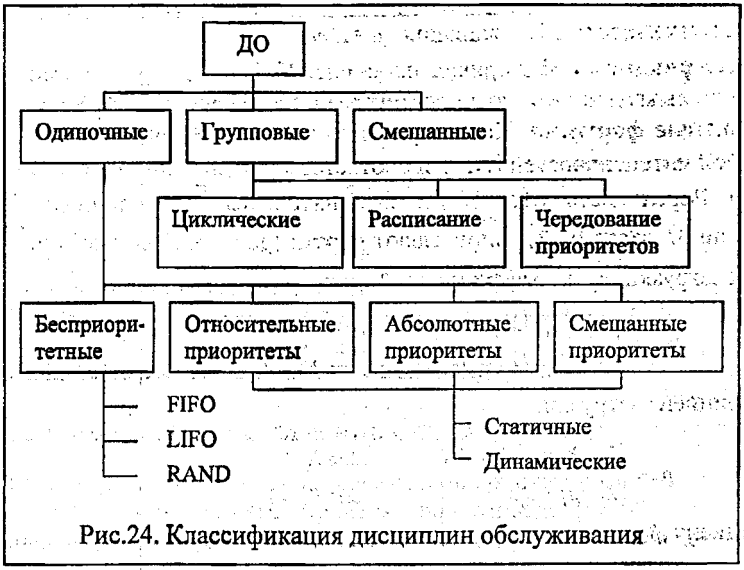


Рис.24. Классификация дисциплин обслуживания

Для СМО M/M/1 и M/M/K входной поток и поток обслуженных заявок является простейшим, значения τ подчиняются экспоненциальному закону

$$f(\tau) = \lambda \cdot e^{-\lambda\tau}$$

, а распределение числа заявок в потоке подчиняется закону Пуассона

$$P(X_{k,\tau}) = (\lambda\tau)^k e^{-\lambda\tau} / k!$$

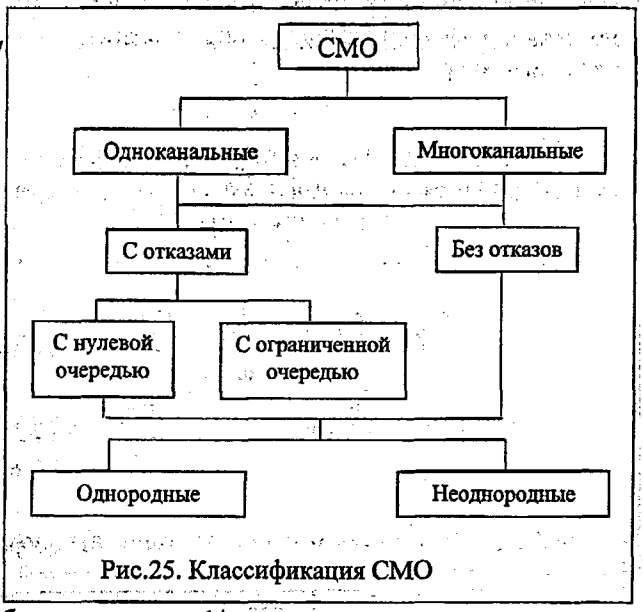


Рис.25. Классификация СМО

т.е. достаточно задать интенсивность поступления заявок $\lambda = 1/m_t$. Закон распределения f_Θ легко пересчитать в закон f_t распределения длительности $t = \Theta/V$ обслуживания заявок в канале. Для СМО типа M/M/1 и M/M/K время обслуживания заявок в каждом канале подчиняется экспоненциальному закону и достаточно задать среднюю интенсивность обслуживания $\mu = 1/m_t$.

Классификация СМО приведена на рис. 25. Наиболее удобными для математических выкладок являются простейшие СМО M/M/1 и M/M/K.

Расчетные формулы. Характеристики таких СМО однозначно определяются парой интенсивностей λ и μ и соответственно значением коэффициента загрузки ρ. Вероятность того, что j-я СМО находится в состоянии M_j (обслуживает ровно M_j заявок), P_{M_j} определяют через вероятность ее простоя P_{0_j}, коэффициент загрузки ρ_j и коэффициент β_j как

$$P_{M_j} = \begin{cases} p_{0j}\rho_j^{M_j}\beta_j(M_j) & ,\text{для } K_j \geq 1 \\ p_{0j}\rho_j^{M_j} = (1-\rho_j)\rho_j^{M_j} & ,\text{для } K_j = 1 \end{cases}.$$

где коэффициент загрузки

$$\rho_j = \begin{cases} \lambda_j / K_j\mu_j = \lambda_j v_j / K_j & ,\text{для } K_j \geq 1 \\ \lambda_j / \mu_j = \lambda_j v_j & ,\text{для } K_j = 1 \end{cases},$$

а коэффициент β_j

$$\beta_j(M_j)=\begin{cases}1 & ,для \quad K_j=1\\ K_j^{M_j}/M_j! & ,для \quad K_j>1, M_j\le K_j\\ K_j^{K_j}/K_j! & ,для \quad K_j>1, M_j>K_j\end{cases}$$

Вероятность простоя СМО находят из условия нормировки $\sum_{M_j=0}^{\infty} P_{M_j}=1$:

$\sum_{M=0}^{\infty} P_{M_j}=p_{0j}+\sum_{M=1}^{\infty} p_{0j}\rho_j^{M_j}\beta_j(M_j)=p_{0j}\left[1+\sum_{M=1}^{\infty}\rho_j^{M_j}\beta_j(M_j)\right]=1$ и $p_{0j}=\left[1+\sum_{M=1}^{\infty}\rho_j^{M_j}\beta_j(M_j)\right]^{-1}$, а для одноканальной СМО $p_{0j}=1-\rho_j$.

3.2. Законы Литтла и Клейнрока

В СМО и сетях МО действуют определенные законы.

Л. Клейнрок показал [17-18], что для СМО с одним каналом и любой дисциплиной обслуживания при наличии некоторых ограничений (отсутствие отказов в обслуживании; СМО простаивает лишь при отсутствии заявок на обслуживание; длительность обслуживания не зависит от входных потоков, а при наличии прерываний описывается экспоненциальным распределением; все входные потоки независимые с пуассоновскими распределениями) действует закон сохранения среднего времени ожидания - оно инвариантно относительно дисциплины обслуживания

$$\sum_{i=1}^{L}\rho_i\omega_i=const$$

Здесь ρ_j – загрузка канала, а ω_j – среднее время ожидания в очереди заявок *i*-го типа (*1, ..., L*). Это означает, что при изменении дисциплины обслуживания время ожидания в очередях для заявок одних типов сокращается за счет его увеличения для заявок других типов. Если заявки обслуживаются в порядке их поступления, то константа определяется как

$$\sum_{i=1}^{L}\rho_i\omega_i=const=\omega R=R\sum_{i=1}^{L}\lambda_i a_2(\upsilon_i)/(2(1-R))$$

Литтл показал [5], что в сетях МО действует закон, который иллюстрируется на рис.26. В соответствии с ним для любого фрагмента сети ССМ, в который в единицу времени входит такое же число заявок что и выходит, произведение $\lambda\times U=M$ дает среднее количество заявок, находящихся в этом фрагменте сети на обслуживании. Применяя закон к СМО, получим соотношения $\lambda\times\omega=1$, $\lambda\times u=m$ и $\lambda\times t=\rho$.

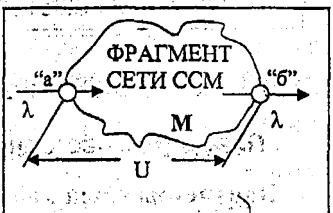


Рис.26. Иллюстрация к закону Литтла.

Дж. Джексон показал [5], что выходной поток в СМО M/M/1 и M/M/K является наряду со входным и потоком обслуживания – простейшим.

Доказано, что простейшие потоки при слиянии в один поток и при разложении на отдельные потоки по вероятностному принципу сохраняют свои свойства и остаются простейшими, что иллюстрируется на рис.27.

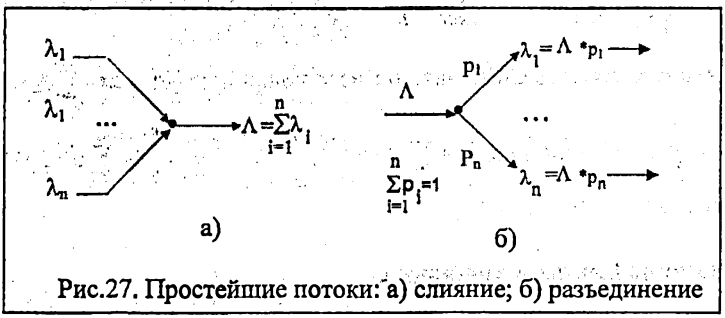


Рис.27. Простейшие потоки: а) слияние; б) разъединение

Тогда, как следует из теоремы Джексона и свойств простейших потоков, в экспоненциальных сетях, состоящих из СМО G/M/1 и G/M/K, потоки заявок в любой точке сети и на входе в любой узел – простейшие. При этом СМО приобретают тип M/M/1 или M/M/K. А так как характеристики таких СМО однозначно определяются парой интенсивностей λ и μ (причем последняя является параметром, который задан), то: 1) сеть может быть декомпозирована на совокупность изолированных СМО (что иллюстрируется рис.28); 2) характеристики СМО рассчитывают независимо друг от друга; 3) системные характеристики определяют на основе узловых характеристик отдельных СМО.

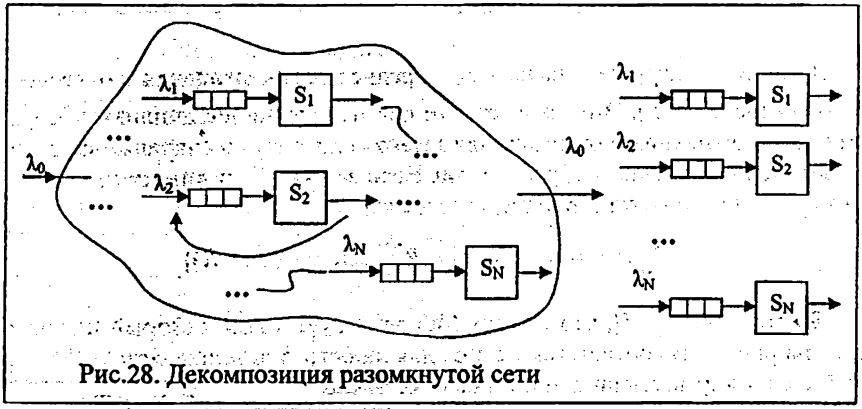


Рис.28. Декомпозиция разомкнутой сети

3.3. Экспоненциальные сети массового обслуживания.

Параметры, состояния и расчет вероятностей состояний

Параметры сетей МО складываются из параметров архитектуры сети и параметров заявок, обслуживаемых в сети. Архитектура сети задается: 1) числом N обслуживающих узлов $\{b_i\}$ (СМО) и числом Np вероятностных узлов; 2)

параметрами $z_i = (K_i, V_i)$ (канальностью и быстродействием) обслуживающих узлов; 3) матрицей связей $C_D = \|d_{ij}\|$ узлов (генераторов, приемников, обслуживающих и маршрутных узлов), где $d_{ij} = \{0;1\}$ в зависимости от наличия связи.

Параметры заявок задаются: 1) матрицей $P = \|p_{ij}\|$ вероятностей переходов заявок из i-го узла сети в j-ый узел либо вектором коэффициентов передач $\vec{\alpha} = (\alpha_1, ..., \alpha_N)$, где α_i – среднее число посещений заявкой i-ой СМО за всё время ее пребывания в сети (значения $P = \|p_{ij}\|$ и $\vec{\alpha}$ легко пересчитываются друг в друга); 2) интенсивностью λ_0 поступления заявок в сеть для разомкнутой сети или постоянным количеством заявок в сети M_0 для замкнутой сети; 3) вектором трудоемкостей $\vec{O} = (O_1, ..., O_N)$, где O_i - средняя трудоемкость (число потребных операций) обслуживания заявки в i-ой СМО за всё время ее пребывания в сети.

На основе этих параметров (рис. 29) можно рассчитать: 1) среднее время $t_i = O_i / V_i$ обслуживания заявки в i-ой СМО за всё время ее пребывания в сети; 2) среднее время $\vartheta_i = t_i / \alpha_i = \vartheta_i = O_i / V\alpha_i$ обслуживания заявки в i-ой СМО при однократном обращении; 3) среднюю трудоемкость $o_i = O_i / \alpha_i$ обслуживания заявки в i-ой СМО при однократном обращении; 4) интенсивность обслуживания $\mu_i = 1/\vartheta_i = \alpha_i / t_i = \alpha_i V_i / O_i$ транзактов, поступающих в i-ую СМО с интенсивностью λ_i и трудоемкостью однократного обращения ϑ_i или o_i; 5) интенсивность обслуживания $\mu_{0i} = 1/t_i = V_i / O_i$ транзактов, поступающих в i-ую СМО с интенсивностью $\lambda_{0i} = \alpha_i \lambda_i$ и трудоемкостью однократного обращения $t_i = \alpha_i \vartheta_i$ или $O_i = \alpha_i o_i$.

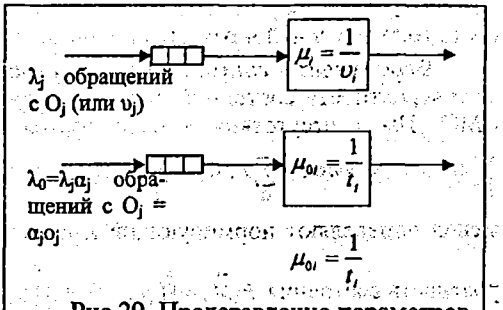


Рис.29. Представление параметров СМО

Состояние сети описывают вектором, задающим характер распределения заявок $\vec{M} = (m_1 = M_1; m_2 = M_2; ...; m_N = M_N) = (M_1; M_2; ...; M_N)$ по системам массового обслуживания. В разомкнутой сети их количество в процессе обработки может меняться от 0 до ∞, а в замкнутой остается постоянным и равным заданному числу M_0. Соответственно в разомкнутой и замкнутой сетях выполняются соотношения $\sum_{i=1}^{N} M_i = M_0 = \overline{0,\infty}$ $\sum_{i=1}^{N} M_i = M_0 = const$. Обозначим Ω множество состояний, допустимых в сети с N узлами. Тогда их число равно $|\Omega|$ и для. разомкнутой сети $|\Omega| = \infty$. В замкнутой сети с N узлами и M_0 заявками из-за условия $\sum_{i=1}^{N} M_i = M_0 = const$ допустимы не все состояния, существующие в аналогичной разомкнутой сети. Здесь $\Omega = \Omega(M_0; N)$, а число возможных состояний $|\Omega| = C^{M}_{M+N+1}$. Например, $\Omega(3,2) = A(3,2) = \{(3,0);(2,1);(1,2);(0,3)\}$ и $\Omega = C^{3}_{4} = 4!/3!1! = 4$.

Поскольку состояния сети могут меняться случайным образом, то для вычисления ее характеристик надо знать с какой вероятностью

$P(\vec{M}) = P(m_1 = M_1, m_2 = M_2, ..., m_N = M_N) = P(M_1, M_2, ..., M_N)$ они случаются. При этом для любой сети справедливо нормирующее условие $\sum_{\Omega} P(\vec{M}) = 1$. Доказано, что для разомкнутых и для замкнутых экспоненциальных сетей решение представимо в виде произведения, т.е. $P(\vec{M}) = P(M_1, M_2, ..., M_N) = p(M_1) \cdot p(M_2) \cdot ... \cdot p(M_N)/G$ $= \prod_{j=1}^{N} P[m_j = M_j]/G = \prod_{j=1}^{N} P_{M_j}/G$, где G – нормирующий коэффициент.

Вероятности состояний разомкнутой сети. Здесь нормирующий коэффициент равен единице, сама сеть в соответствии с теоремой Джексона распадается на ряд изолированных СМО, поэтому вероятности состояний $P(\vec{M}) = P(M_1, M_2, ..., M_N) = p(M_1) \cdot p(M_2) \cdot ... \cdot p(M_N) = \prod_{j=1}^{N} P[m_j = M_j] = \prod_{j=1}^{N} P_{M_j}$ и $\sum_{\Omega} P(\vec{M}) =$ $\sum \prod_{j=1}^{N} P[m_j = M_j] = \sum \prod_{j=1}^{N} P_{M_j} = 1$, где P_{M_j} - вероятность состояния j-ой СМО (вероятность того, что в ней находится ровно M_j заявок).

Вероятности состояний замкнутой сети. Здесь как и для разомкнутой сети вероятности состояний определяются через вероятности составляющих ее СМО. Из-за отсутствия в сети состояний, не удовлетворяющих условию $\sum_{i=1}^{N} M_i = M_0 = const$, $\sum_{\Omega} P(\vec{M}) = \sum_{\Omega} p(M_1, M_2, ..., M_N) = \sum_{\Omega} (\prod_{j=1}^{N} p_{M_j}/G) = 1$. Из этого выражения определяют нормирующий множитель $G = \sum_{\Omega} \prod_{j=1}^{N} p_{M_j}$. Соответственно вероятность состояния $P(\vec{M}) = \prod_{j=1}^{N} p_{M_j}/G = \prod_{i=1}^{N} p_{M_j} / \sum_{\Omega} \prod_{j=1}^{N} p_{M_j}$.

Преобразуем это выражение: 1) подставим в него формулу для вычисления p_{M_j} (см. параграф 3.1); 2) учитывая, что $\lambda_i = \alpha_i \cdot \lambda_0$, заменим в формуле значения коэффициента загрузки на $\rho_j = \lambda_j \cdot \upsilon_j / \kappa_j = \lambda_0 \cdot \alpha_j \cdot \upsilon_j / \kappa_j$; 3) после указанных преобразований сократим одинаковые выражения в числителе и знаменателе формулы, а именно выражение $\prod_{j=1}^{N} \lambda_0^{M_j} = \lambda_0^{M_1} \cdot \lambda_0^{M_2} \cdot ... \cdot \lambda_0^{M_N} = \lambda_0^{\sum_{j=1}^{N} M_j} = \lambda_0^{M_0}$ и выражение $\prod_{j=1}^{N} p_{0j} = \pi$. В результате получим

$$p(M_1, M_2, ..., M_N) = \prod_{j=1}^{N} p_{0j} \cdot \rho_j^{M_j} \cdot \beta_j(M_j) \Big/ \sum_{\Omega} \prod_{j=1}^{N} p_{0j} \cdot \rho_j^{M_j} \cdot \beta_j(M_j) =$$

$$= \prod_{j=1}^{N} p_{0j} \cdot \beta_j(M_j)(\lambda_0 \cdot \alpha_j \cdot \upsilon_j / \kappa_j)^{M_j} \Big/ \sum_{\Omega} \prod_{j=1}^{N} p_{0j} \cdot \beta_j(M_j)(\lambda_0 \cdot \alpha_j \cdot \upsilon_j / \kappa_j)^{M_j} =$$

$$= \prod_{j=1}^{N} \beta_j(M_j)(\alpha_j \cdot \upsilon_j / \kappa_j)^{M_j} \Big/ \sum_{\Omega} \prod_{j=1}^{N} \beta_j(M_j)(\alpha_j \cdot \upsilon_j / \kappa_j)^{M_j}.$$

Если сеть состоит только из одноканальных СМО, то с учетом $\beta_j = 1$ и $K_j = 1$, получим $P(\vec{M}) = P(M_1, M_2, ..., M_N) = \prod_{j=1}^{N} (\alpha_j \cdot \upsilon_j)^{M_j} \Big/ \sum_{\Omega} \prod_{j=1}^{N} (\alpha_j \cdot \upsilon_j)^{M_j}$.

Пример. Пусть задана замкнутая экспоненциальная сеть со значениями $N = 2, M = 2$. Допустимые состояния составляют множество $\Omega = \Omega(M, N)$

$=\Omega(2,2)=\{(0,2),(2,0),(1,1)\}$, их всего $|\Omega|=3$. Вероятность состояний такой сети определится как $p(M_1,M_2)=(\alpha_1\cdot\upsilon_1)^{M_1}\cdot(\alpha_2\cdot\upsilon_2)^{M_2}/[(\alpha_1\cdot\upsilon_1)^0\cdot(\alpha_2\cdot\upsilon_2)^2+(\alpha_1\cdot\upsilon_1)^2\cdot(\alpha_2\cdot\upsilon_2)^0+(\alpha_1\cdot\upsilon_1)\cdot(\alpha_2\cdot\upsilon_2)]=(\alpha_1\cdot\upsilon_1)^{M_1}\cdot(\alpha_2\cdot\upsilon_2)^{M_2}/Z$. Соответственно вероятности состояний $p(0,2)=(\alpha_2\cdot\upsilon_2)^2/Z$, $p(2,0)=(\alpha_1\cdot\upsilon_1)^2/Z$ и $p(1,1)=\alpha_1\cdot\upsilon_1\cdot\alpha_2\cdot\upsilon_2/Z$. В сумме эти вероятности составляют единицу $p(0,2)+p(2,0)+p(1,1)=1$

3.4. Аналитический расчёт характеристик разомкнутых экспоненциальных сетей МО

Включает: расчет коэффициентов передач и интенсивностей потоков заявок в каждую СМО; расчет вероятностей состояний СМО и их узловых характеристик; расчет системных характеристик.

Расчет коэффициентов передач и интенсивностей потоков заявок $\vec{\lambda}=(\lambda_1,...,\lambda_N)$ иллюстрируется рис.30. Входной поток в каждой СМО складывается из суммы выходных потоков всех СМО сети, взвешенных вероятностью перемещения заявок с их выходов на вход рассматриваемой СМО. Соответственно для каждой i-ой СМО получаем уравнение $\lambda_i=\lambda_0\cdot p_{0i}+\lambda_1\cdot p_{1i}+...+\lambda_N\cdot p_{Ni}=\sum_{j=0}^{N}\lambda_j\cdot p_{ji}$. Для сети будет получена система $\lambda_i=\sum_{j=0}^{N}\lambda_j\cdot p_{ji}$ (где $i=\overline{1,N}$) из N линейных уравнений с N неизвестными, решение которой даст искомые значения $\{\lambda_i\}$ $i=\overline{1,N}$. Коэффициенты передачи каждой СМО рассчитывают по формуле $\alpha_i=\lambda_i/\lambda_0$.

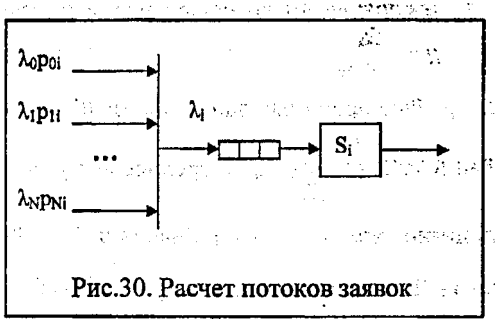


Рис.30. Расчет потоков заявок

Расчет вероятностей состояний СМО и их узловых характеристик производится в следующей последовательности:

- определяют коэффициенты загрузок СМО $\{\rho_j\}$, а также общую загруженность многоканальных СМО $R_i=\rho_i\cdot K_i$;
- определяют коэффициенты $\{\beta_j\}$, вероятности простоя СМО $\{p_{0j}\}$ и вероятности их состояний $\{p_{Mj}\}$;

- определяют средние длины очередей $\{l_j\}$ к каждой СМО сети по формуле $l_j = 1 \cdot p_{K+1} + 2 \cdot p_{K+2} + \ldots = \sum_{i=1}^{\infty} i \cdot p_{K_j+i}$, либо для одноканальных СМО как $l_j = \rho_j^2/(1-\rho_j))$

- среднее число заявок $\{m_j\}$, находящихся в каждой СМО, по формуле $m_j = l_j + R_j$, либо для одноканальных СМО как $m_j = \rho_j/(1-\rho_j)$, где общая загруженность рассчитывается как $R_j = 1 \cdot p_1 + 2 \cdot p_2 + \ldots + K \cdot p_K + (K+1) \cdot p_K + \ldots = \rho_j \cdot K_j$ (см. таблицу ниже)

	p_0	p_1	p_2	....	p_{K-1}	p_k	...			
	0	1	2		... K-1	K	K+1	K+2	...	
	0	0	0	....	0	K	1	2	...	
	0	1	2		... K-1	K	K	K	...	K

f, g, r – случайные величины;
f = g + r – число заявок в многоканальной СМО;
g – количество заявок в очереди;
r - число занятых каналов.

- определяют, используя закон Литтла, средние времена ожидания в очереди $\{\omega_j\}$ и пребывания в СМО $\{u_j\}$ по формулам $\omega_j = l_j/\lambda_j$ и $u_j = m_j/\lambda_j$ соответственно.

Расчет системных характеристик производится на базе узловых:

- определяют среднюю длину очередей в сети $L = \sum_{i=1}^{N} l_i$, среднее число заявок в сети $M_0 = \sum_{i=1}^{N} m_i$ и среднее число обслуживаемых заявок $R = \sum R_i + \sum \rho_j$ (понятно, что $M_0 = L + R$);

- определяют среднее время ожидания в сети $W = \sum_{i=1}^{N} \alpha_i \cdot \omega_i$, среднее время пребывания заявки в сети $U = \sum_{i=1}^{N} \alpha_i \cdot u_i$ и среднее время обслуживания заявки $T = \sum_{i=1}^{N} \alpha_i \cdot \upsilon_i = \sum_{i=1}^{N} t_i$ (понятно, что $U = W + T$). Значения U и W могут быть определены, учитывая закон Литтла, из соотношений $\lambda_0 \cdot U = M_0$ и $\lambda_0 \cdot W = L$.

3.5. Аналитический расчёт характеристик замкнутых экспоненциальных сетей МО

Включает расчет: коэффициентов передач; вероятностей состояний сети; узловых и системных характеристик.

Расчет коэффициентов передач производится также как и интенсивностей потоков заявок в разомкнутой сети. При этом в систему линейных уравнений $\lambda_i = \sum_{j=0}^{N} \lambda_j \cdot p_{ji}$ $(i = \overline{1,N})$ вместо λ_j подставляют их значения из формулы

$\alpha_i = \lambda_i/\lambda_0$. Сокращая λ_0, получают систему линейных уравнений $\alpha_i = \sum_{j=0}^{N} \alpha_j p_{ji}$ $(i = \overline{1,N})$ с N неизвестными, искомыми значениями $\{\alpha_i\}$ i = $\overline{1,N}$.

Расчет вероятностей состояний сети производится по полученной в параграфе 3.3 формуле $\prod_{j=1}^{N} \beta_j(M_j)(\alpha_j \cdot \upsilon_j / \kappa_j)^{M_j} \Big/ \sum_{\Omega} \prod_{j=1}^{N} \beta_j(M_j)(\alpha_j \cdot \upsilon_j / \kappa_j)^{M_j}$.

Расчет узловых характеристик производится в следующей последовательности:

- определяют коэффициенты загрузок отдельных СМО $\{\rho_j\}$ (например, для одноканальных СМО $\rho_j = 1 - \mathrm{P}[\textit{простоя узла}] = 1 - p_{0j} = 1 - \sum_{\Omega, M_j=0} p(M_1, M_2, ..., M_N)$);
- используя выражения для расчета коэффициентов загрузки, определяют интенсивности $\lambda_j = \rho_j K_j / \upsilon_j$ потоков заявок в каждую СМО;
- узловые характеристики СМО определяют как и в разомкнутых сетях.

Расчет системных характеристик производится на базе узловых:

- определяют производительность сети $\lambda_0 = \lambda_i/\alpha_j$ через характеристики любой СМО;
- остальные системные характеристики определяют как в разомкнутых сетях.

3.6. Примеры расчета характеристик экспоненциальных сетей массового обслуживания

Простейшие разомкнутая и замкнутая сети МО представлены на рис.31.

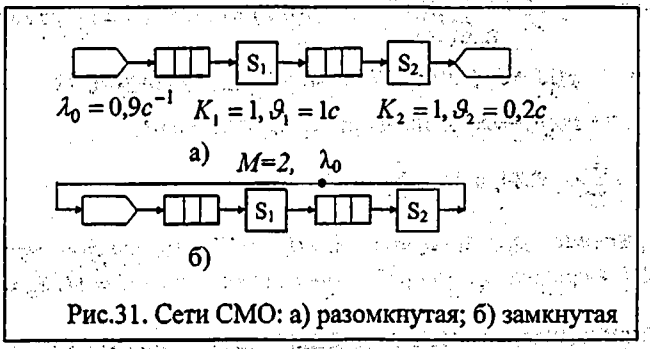


Рис.31. Сети СМО: а) разомкнутая; б) замкнутая

Пример расчета разомкнутой сети. Матрица вероятностей переходов заданной сети, состоящей из двух одноканальных узлов, имеет вид

$$P = \begin{array}{c|ccc} & S_0 & S_1 & S_2 \\ \hline S_0 & 0 & 1 & 0 \\ S_1 & 0 & 0 & 1 \\ S_2 & 1 & 0 & 0 \end{array}$$

Составляем систему линейных уравнений

$$\begin{cases}\lambda_1 = \lambda_0 p_{01} + \lambda_1 p_{11} + \lambda_2 p_{21} \\ \lambda_2 = \lambda_0 p_{02} + \lambda_1 p_{12} + \lambda_2 p_{22}\end{cases}$$

Подставляем в нее заданные параметры и решаем

$$\begin{cases}\lambda_1 = 0{,}9 \cdot 1 + \lambda_1 \cdot 0 + \lambda_2 \cdot 0 \\ \lambda_2 = 0{,}9 \cdot 0 + \lambda_1 \cdot 1 + \lambda_2 \cdot 0\end{cases}$$

получая $\lambda_1 = \lambda_2 = \lambda_0 = 0{,}9$ и коэффициенты передач $\alpha_1 = \alpha_2 = 1$.

Коэффициенты загрузок СМО равны $\rho_1 = \lambda_1 \cdot \upsilon_1 = 0{,}9$ и $\rho_2 = \lambda_2 \cdot \upsilon_2 = 0{,}18$. Вероятности их простоя $p_{01} = 1 - \rho_1 = 0{,}1$ и $p_{02} = 1 - \rho_2 = 0{,}82$.

Основные узловые характеристики первой СМО $l_1 = \rho_1^2/(1-\rho_1) \approx 8{,}1$, $m_1 = \rho_1/(1-\rho_1) \approx 9$, $u_1 = m_1/\lambda_1 \approx 10$. Узловые характеристики второй СМО $l_2 = \rho_2^2/(1-\rho_2) \approx 0{,}04$, $m_2 = \rho_2/(1-\rho_2) \approx 0{,}22$ и $u_2 = m_2/\lambda_2 \approx 0{,}25$.

Основные системные характеристики $U = \alpha_1 \cdot u_1 + \alpha_2 \cdot u_2 \approx 10{,}25$ и $M = m_1 + m_2 \approx 9{,}22$.

Пример расчета замкнутой сети. Сеть описывается той же матрицей вероятностей переходов, поэтому значения коэффициентов передач $\alpha_1 = \alpha_2 = 1$ определяются из системы уравнений аналогичного вида

$$\begin{cases}\alpha_1 = 0{,}9 \cdot 1 + \alpha_1 \cdot 0 + \alpha_2 \cdot 0 \\ \alpha_2 = 0{,}9 \cdot 0 + \alpha_1 \cdot 1 + \alpha_2 \cdot 0\end{cases}$$

Вероятности каждого из трех допустимых состояний определятся по формуле $p(M_1;M_2) = \vartheta_1^{M_1} \cdot \vartheta_2^{M_2}/(\vartheta_1^2 + \vartheta_1 \cdot \vartheta_2 + \vartheta_2^2) = 0{,}2^{M_2}/(1+0{,}2+0{,}04) = 0{,}2^{M_2}\ 1{,}24$ как $p(0;2) = 0{,}03$; $p(2;0) = 0{,}81$ $p(1,1) = 0{,}16$.

Загрузки узлов $\rho_1 = 1 - \sum_{\Omega, M_1 = 0} p(\overline{M}) = 1 - p(0,2) = 0{,}97$ и $\rho_2 = 1 - p(2,0) = 0{,}19$.

Значение $m_1 = 1\ p(1,1) + 2\ p(2;0) = 1{,}77$ и $m_2 = 1\ p(1,1) + 2\ p(0;2) = 0{,}23$.

Значение $l_1 = 1\ p(2;0) = 0{,}81$ и значение $l_2 = 1\ p(0;2) = 0{,}03$.

Значение $L = \sum_{i=1}^{N} l_i = 0{,}84$, а $M_0 = \sum_{i=1}^{N} m_i = 2$.

Из выражения $\rho_1 = \lambda_1 \cdot \upsilon_1 = \alpha_1 \cdot \lambda_0 \cdot \upsilon_1 = 0{,}97$ определяем производительность $\lambda_0 = 0{,}97$ и среднее время пребывания заявок в сети $U = M/\lambda_0 \approx 2$.

3.7. Аппроксимационные подходы к исследованию сетей МО

В тех случаях, когда вероятность состояния сети не сводится к произведению вероятностей состояний отдельных узлов, используются приближенные методы. Например, если в сети используются узлы типа память, если заявки могут распараллеливаться и выполняться одновременно на разных устройствах и т.п. Аппроксимационные подходы базируются на принципе агрегирования. Основная идея - декомпозировать сеть на подсети (агрегаты). В качестве агре-

гатов выделяют части исходной сети со слабой степенью связанности с остальной сетью, которые удобно исследовать автономно. Каждый агрегат (подсеть) анализируется самостоятельно, а системные характеристики получают из характеристик отдельных агрегатов.

На аппроксимационном подходе основан **метод эквивалентных потоков** (введен в рассмотрение Коуртоисом, Чанди, Герцогом, Ву). Его суть в замене подсети единственным агрегатом - эквивалентным узлом обслуживания с сохранением интенсивностей входных и выходных потоков. При этом время пребывания заявок и соответственно интенсивность обслуживания в агрегате зависят от числа заявок в нем. Системные характеристики получают на основе характеристик агрегатов итеративным образом с учетом уравнений баланса, описывающих фундаментальные свойства всей сети. Т.е. после вычисления системных характеристик и оценки их точности состояния агрегатов корректируются, снова вычисляются их характеристики, а затем определяются системные и т.д.

Прямой метод вычисления средних [8] может использоваться как для расчета сетей МО, так и сетей из агрегатов, получаемых методом эквивалентных потоков. Основывается на следующих теоремах. Теорема 1: в замкнутой экспоненциальной сети МО стационарная вероятность состояния любого узла в момент поступления в сеть новой заявки совпадает со стационарной вероятностью того же состояния в этом узле при наличии в сети на единицу меньшего числа заявок. Соответственно для всех узлов $i=1,2,...N$ $m_i(M)=m_i(M-1)$, $m_i(0)=0$, $u_i(M)=\omega_i(M)+\vartheta_i$ и $\omega_i(M)=m_i(M-1)\vartheta_i$, $u_i(M)=\vartheta_i(1+m_i(M-1))$.

Теорема 2: пусть сеть МО, для которой решение представимо в виде произведения $P(\bar{M})=\prod_{j=1}^{N} P_{M_j}/G$, преобразована в два агрегата (один из узла b_i, а второй – эквивалент $b_э$ оставшихся узлов). Тогда распределение длин очередей в узле b_i в преобразованной сети такое же, как и в узле b_i исходной сети, т.е. $p'_i(m)=p_i(m)$. В разомкнутой сети эквивалентный узел заменяется новым пуассоновским источником заявок с интенсивностью λ, равной пропускной способности оставшейся части сети, при условии, что время ожидания во всех узлах сети нулевое.

На основе перечисленных теорем и закона Литтла для каждого узла можно получить соотношения $u_i(M)=\vartheta_i(1+m_i(M-1))$, $\lambda_0(M)=M/U=M/\sum_{i=1}^{N}\alpha_i\cdot u_i$ и $m_i(M)=\lambda_0(M)\alpha_i u_i(M)$.

Применяя их к каждому узлу сети или агрегату и изменяя M=1,2,... (с учетом начальных значений для каждого узла $m_i(0)=0$), можно рассчитать сеть.

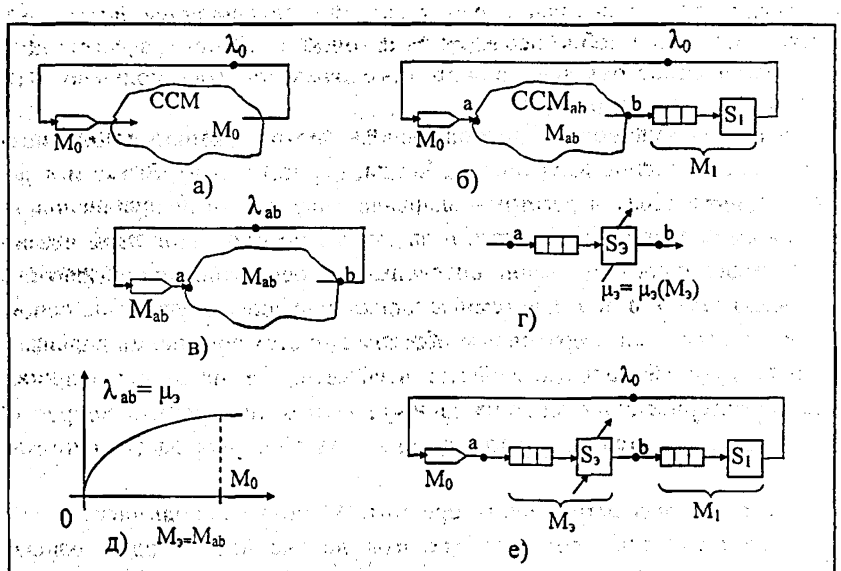


Рис.32. Аппроксимационый подход к расчету СМО: а) исходная сеть; б) исходная сеть в виде двух подсистем; в) модель для расчета первой подсистемы; г) эквивалентная СМО с переменным $\mu_э(M)$; д) график зависимости $\mu_э(M)$; е) итоговое представление исходной сети

Применение метода эквивалентных потоков иллюстрируется на рис.32. Исходная замкнутая сеть декомпозируется на узел S_1 и оставшуюся часть сети между точками a-b. Она заменяется эквивалентным агрегатом (рис.32, г) – СМО с переменной интенсивностью обслуживания $\mu_э = \mu_э(M_э)$, которая зависит от числа заявок $M_э = M_{ab}$, находящихся в агрегате. Параметр агрегата $\mu_э = \lambda_{ab}$ (рис.32, д) вычисляется как производительность соответствующей замкнутой сети (рис.32, в). Вся сеть выглядит, как показано рис.32, е. Расчет системных характеристик производится на ее основе с учетом значений параметра $\mu_э$ (рис.32, д) и балансовых соотношений сети $\lambda_0 = \lambda_э = \lambda_1 = \mu_э$ и $M_1 + M_э = M_0$ итерационно: - задают $M_э$; - рассчитывают интенсивность обслуживания агрегата $\mu_э(M_э)$ (если агрегат просчитан заранее, то значения берут из графика на рис.32, д); - устанавливают $\lambda_1 = \mu_э$ и рассчитывают число заявок m_1 в узле S_1 (например, для одноканальной СМО по формуле $\rho_i/(1-\rho_i)$); - уточняют значение $M_э = M_0 - m_1$ и возвращаются к исходному пункту.

3.8. Асимптотические (алгебраические) модели

Предельные (асимптотические) характеристики объекта можно получить, построив асимптотическую модель. Допустим, что выполнение всех заявок в сети спланировано идеальным образом. Т.е. заменим случайные законы поступления и обслуживания заявок в сети детерминированными с сохранением средних

значений их параметров λ_i, μ_i и ϑ_i. Это приведет к тому, что модель будет характеризоваться максимальной производительностью λ_0 и соответственно минимальным временем пребывания (обработки) заявок в сети U_0. Указанное иллюстрируется рис.33, где жирными линиями показаны асимптотические зависимости $\lambda_0(M)$ и $U_0(M)$, а тонкими линиями реальные зависимости. Значение $U_1 = U_0(1)$ соответствует времени пребывания заявки в сети при условии, что других заявок в сети нет. Соответственно $U_1 = \sum_{i=1}^{N} \alpha_i \cdot \vartheta_i$.

Тогда в сети можно выделить два режима работы: до насыщения и после насыщения. В режиме насыщения в сети появится хотя бы один полностью загруженный узел с коэффициентом загрузки $\rho_s = 1$, т.е. "узкое место" и пусть M^* - число заявок, переводящих сеть в режим насыщения. Рассмотрим каждый режим в отдельности.

В режиме до насыщения при любом числе заявок в сети - M, удовлетворяющем условию $M < M^*$, заявки могут быть "идеально" спланированы и обрабатываются так, как если бы в сети была только одна заявка. Соответственно все $\omega_j = 0$, время пребывания заявки в сети постоянно и равно длительности обработки одной заявки $U_1 = \sum_{i=1}^{N} \alpha_i \cdot \vartheta_i$, а производительность сети определится по закону Литтла

$$\begin{cases} U_0 = U_1 = \sum_{i=1}^{N} \alpha_i \cdot \vartheta_i = \sum_{i=1}^{N} t_i = const \\ \lambda_0 = \text{var} = M / U_0 = M \Big/ \sum_{i=1}^{N} \alpha_i \cdot \upsilon_i = M / U_1 = \lambda_0(M) \end{cases}$$

Если увеличивать число заявок в сети M, то наступит момент, когда хотя бы один из узлов окажется полностью загруженным и станет "узким местом". Сеть войдет в режим насыщения, дальнейшее увеличение числа заявок в сети не будет вести к росту ее производительности, достигнутой в точке $M = M^*$ и будет определяться параметрами (пропускной способностью) "узкого места".

В режиме насыщения ($M > M^*$) производительность сети постоянна и равна $\lambda_0(M) = M / U_0 = M^* / U_0$ (при $M = M^*$), а время пребывания заявки в сети определится по закону Литтла

$$\begin{cases} \lambda_0 = M^* / U_0 = M^* \Big/ \sum_{i=1}^{N} \alpha_i \cdot \vartheta_i = M^* / U_1 = const \\ U_0 = \text{var} = M / \lambda_0 = M \cdot \sum_{i=1}^{N} \alpha_i \cdot \upsilon_i \Big/ M^* = U_0(M) \end{cases}$$

В полученных соотношениях неизвестной величиной является значение M^*. Для его определения выявляется потенциальное узкое место – узел с $\rho_S = \max\{\rho_i\}$, и из уравнения $\rho_s = 1$ определяют M^*:

$$\rho_S = \lambda_S \cdot \vartheta_S / K_S = \lambda_0 \cdot \alpha_S \cdot \vartheta_S \Big/ K_s = M^* \cdot \alpha_S \cdot \vartheta_S \Big/ K_S \cdot \sum_{i=1}^{N} \alpha_i \cdot \vartheta_i = 1 \Rightarrow M^* =$$

$$K_S \cdot \sum_{i=1}^{N} \alpha_i \cdot \vartheta_i \Big/ \alpha_S \cdot \vartheta_S .$$

Глава 3

ЭВМ, КОМПОНЕНТЫ И СИСТЕМЫ ЭВМ КАК ОБЪЕКТЫ МОДЕЛИРОВАНИЯ

1. Особенности моделирования ЭВМ

1.1. ЭВМ как сложные системы

ЭВМ, компоненты и системы ЭВМ относятся к категории сложных систем, что обусловлено: большим количеством элементов и типов элементов, из которых состоят компоненты ЭВМ и большим числом устройств и их типов, из которых состоят ЭВМ, системы и комплексы; сложностью и иерархичностью структуры; сложностью реализуемых функций, многофункциональностью, параллельностью протекающих процессов; наличием целенаправленного управления, сложными алгоритмами управления, реализуемыми ОС; работой в условиях воздействия случайных и человеческого факторов, внешних помех и собственной ненадежности. Моделирование, как правило, сопровождает ЭВМ на всех стадиях, включая проектирование, разработку, модернизацию, настройку и эксплуатацию.

1.2. Уровни детализации представления ЭВМ при моделировании

В зависимости от целей моделирования изменяется необходимый уровень детальности рассмотрения ЭВМ, смысл, вкладываемый в понятие "элемент", а также методы, применяемые для моделирования и возможности отображения в моделях структурных и функциональных особенностей архитектуры ЭВМ (см. табл. ниже).

№	Название уровня	Структурные компоненты	Характер поведения	Цели
1	Сети и комплексы ЭВМ	ЭВМ, средства связи, сети, системы ЭВМ и т. п.	Модели теории массового обслуживания	Определение системных характеристик (производительности, надежности), экономического эффекта
2	Вычислительные системы	Устройства	Модели теории массо-вого обслуживания	Определение системных характеристик (производительности, надежности), экономического эффекта, опреде-ление оптимальных архитектур
3	Процессор, память, коммутатор ("ядро" ЭВМ)	Процессор, память, шины и т.д.	Модели теории массового обслуживания	Определение системных характеристик, исследо-вание конфликтов на шинах и т.п.
4	Система команд	Регистры, счетчики, микропрограммы	Имитационные модели на базе языков описания микропрограмм	Выбор оптимального состава машинных команд
5	Микросхемный (уровень интегральных схем – ИС)	Микропроцессоры, памяти, контроллеры, порты и т.д.	Имитируются на базе моделей типа "черный ящик", поведенческих моделей по спецификациям проектов ЦА	Оценка корректности функционирования БИС, СБИС с учетом временных параметров
6	Регистро-	Регистры, счетчики,	Имитационные	Выбор оптимальной архитектуры

	вый	*АЛУ и т.д.*	*мо-дели на базе языков регистро-вых пере-дач, табли, истин-ности и языков мик-ропрограммирова-ния*	*устройства на регистровом уровне*
7	*Вентиль-ный (логи-ческий)*	*Логические элементы*	*На базе моделей логических схем*	*Оценка корректности функ-ционирования, уточнение временных соотношений и т.п.*
8	*Схемо-принципи-альный*	*Активные и пассив-ные электрические элементы*	*В терминах диф-ференциальных уравнений*	*Исследование нагрузоч-ной способнос-ти, помехо-защищенности и т.п.*
9	*Кремниевый*	*Площадь и геометри-ческие размеры полу-проводниковой пла-стины*	*Не моделируется*	*Оценка процента выхода годных схем, повышение помехозащищенности и т.п.*

Здесь уровни 1-3 относят к системному моделированию, уровни 4-5 – к структурно-алгоритмическому, а уровни 6-7 – к функционально-логическому моделированию. Наиболее полно проработаны, исследованы и обеспечены средствами моделирования уровни 6-9, а наиболее перспективны с точки зрения автоматизации проектирования БИС и СБИС и соответственно интенсивно развиваются средства моделирования 5-го уровня.

2. Уровень системного моделирования

2.1. Коцептуальная модель ЭВМ

На уровне системного моделирования ЭВМ рассматривается как система $<X, H, G>$, где X задает параметры вычислительной нагрузки (потоков обрабатываемых задач, запросов), $H=<A, F>$ задает параметры A архитектуры компьютера (типы и параметры устройств, а также структуру их связей) и параметры F закона функционирования ЭВМ, реализуемого ОС, G описывает параметры внешнего (административного) управления системой. Системные характеристики, определяемые на этом уровне: производительность и надежность.

Надежность ЭВМ оценивается многими показателями и должна учитывать как надежность программ так и апаратуры. Так как проявление ошибок в программах не зависит от отказов аппаратуры, а отказы аппаратуры не зависят от отказов программы, то вероятность безотказной работы системы определяется как произведение вероятностей безотказной работы аппаратуры вместе с операционной системой и программ пользователя. Среди системных показателей можно отметить коэффициент использования ЭВМ $Ku=T0/(T0+TB+Tn)$ и коэффициент готовности $Kr=T0/(T0+TB) \geq Ku$. Здесь $T0$ - время исправной работы ЭВМ, TB – среднее время восстановления ЭВМ после отказа, Tn – среднее время проведения профилактических работ.

Оценка и прогнозирование надежности программ осуществляется на основе математических моделей, прогнозирующих функцию плотности распределения времени до следующего отказа в программе. Это модели: - Б. Литтлвуда и Дж. Л. Верраллома; - модель Джелинского-Моранды (основанная на допущениях: - время до следующего отказа распределено экспоненциально; - интенсивность отказов программы пропорциональна количеству оставшихся в программе ошибок); - мо-

дель Шика-Волвертона, основанная на допущении о том, что интенсивность проявления ошибок программы пропорциональна не только количеству оставшихся в программе ошибок, но и времени, потраченному на отладку.

Производительность ЭВМ может оцениваться как потенциальное быстродействие "ядра" компьютера процессор-память $B_{пр.я}$, вычисляемое по паспортным данным. Например, как скорость выполнения отдельных групп машинных команд (коротких, длинных и т.п.) либо как средняя скорость $\lambda_0 = \sum_{s=1}^{h} k_s \Big/ \sum_{s=1}^{h} k_s t_s$, измеряемая числом реализуемых в единицу времени усредненных с учетом смесей решаемых задач операций. Здесь k_s - вес соответствующей команды или группы команд в смеси, а t_s - среднее время ее выполнения. Реальное быстродействие "ядра" ЭВМ $B_{пр.р} = B_{пр.я}\rho_{пр}$ оценивается с учетом реальной загруженности процессора, учитывающей оптимальность организации системы и управления ею. Более полной оценкой производительности может служить вектор потенциальных быстродействий устройств ЭВМ $\bar{B}_{\text{п}}$ и вектор их реальных быстродействий $\bar{B}_p = (\rho_1 B_1, \ldots, \rho_N B_N)$. Интегральной оценкой производительности системы может служить как среднее количество задач l-го класса, решаемых в единицу времени $\lambda^l = \lim_{T \to \infty} J^l / T,\ l = 1, L$ так и общая производительность системы $\lambda = \sum_{e=1}^{L} \lambda^e$.

В качестве критериев эффективности используют стоимость единицы производительности $I = (S_{ЭВМ} + S_{эксплуат.}) / \lambda_0$ и стоимость единицы производительности без учета стоимости эксплуатации системы $I = S_{ЭВМ} / \lambda_0$.

2.2. Модели вычислительной нагрузки

Вычислительная нагрузка (ВН) - это модель внешней среды, множество исходных данных и программ, описывающих алгоритмы их обработки. Успех моделирования ЭВМ во многом определяется точностью представления и воспроизведения вычислительной нагрузки. ВН может рассматриваться как единая совокупность задач или как совокупность классов задач. Соответственно модели ВН должны отображать состав классов задач и учитывать случайный характер параметров задач. Для каждого класса задач модель ВН задает: законы появления задач в системе; потребности и порядок использования ресурсов. Математической моделью ВН может служить формализованное описание на основе перечисленных параметров.

Трассовые модели ВН, полученные мониторингом процесса обработки задач, относят к наиболее точным. В процессе наблюдения, выполняемого специализированными программными или аппаратными средствами, фиксируются все параметры обработки каждой задачи, существенные для целей моделирования. Например, параметры, описывающие маршрут и время продвижения задачи от одного устройства к другому устройству в процессе обработки с фиксацией объёмов потребляемых при этом ресурсов (занимаемой емкости памятей, размеров буферных областей, числа процессорных операций, количества распечатанных символов и т.п.).

В то же время существует большой класс моделей ВН, основанных на использовании математического аппарата, усреднении параметров задач (параметров

алгоритмов задач), использовании вероятностных оценок. К их числу относят: сетевые модели, марковские модели на уровне алгоритмов, файлов или устройств и др.

Марковские модели ВН на уровне алгоритмов относят к наиболее детальным. С их помощью можно оценить параметры входного потока задач, необходимые для последующего использования при моделировании, например, среднюю трудоемкость процессорных операций, число обращений к файлам и устройствам, среднее число передаваемой при этом информации. Здесь алгоритм представляется графом марковского поглощающего процесса, операторам соответствуют вершины $S = \{S_i \mid i = \overline{1,N}\}$, а направлениям передачи управления - дуги графа (рис.34, а). Дугам ставят в соответствие вероятности переходов по графу $P = [p_{ij}]$.

В качестве операторов $S = S_B \bigcap S_{BB}$ учитываются вычислительные операторы, выполняемые процессором, и операторы ввода-вывода информации, связанные с переносом информации в файлах и выполняемые устройствами ввода-вывода и запоминающими устройствами. Вершинам первого типа соответствует требуемое число процессорных операций Θ_i, а вершинам второго типа среднее количество передаваемой информации $\vec{\Theta} = \{\Theta_i \mid i = \overline{1,N}\}$ по каждой операции ввода-вывода $S_{BB} = S_{BB_1} \bigcap S_{BB_2} \cdots \bigcap S_{BB_n}$. Метод иллюстрируется примером (рис.34, б), где представлен алгоритм, наблюдение за многократным выполнением которого позволило построить граф марковского процесса (рис.34, в). Используя модель, можно рассчитать количество выполнений каждой вершины $\vec{\alpha} = \{\alpha_i \mid i = \overline{1,N}\}$. Для этого составляется система линейных уравнений

$$\alpha_i = \sum_{j=1}^{N-1} \alpha_j p_{ij}, i = \overline{2, N-1}.$$

Средняя суммарная трудоемкость каждой вершины определяется как $\Theta_i = \alpha_i \theta_i$. Суммарное число процессорных операций (трудоёмкость процессорной обработки задачи) подсчитывается как $\Theta_B = \sum_{S_i \in S_B} \Theta_i$, суммарное число обращений к i-ому файлу $A_i = \sum_{S_j \in S_{BBi}} \alpha_j$, общее количество переданной в него информации $\Theta_{BBi} = \sum_{S_j \in S_{BBi}} \Theta_j$. Средний объём передаваемых данных при работе с каждым файлом определяется как $\Theta_{1BBi} = \Theta_{BBi} / A_i$.

Сетевые модели основаны на аппарате сетевого планирования и более просты в использовании. Для перехода к сетевой модели в рассмотренном выше графе определяют среднее количество выполнений каждого циклического участка и все циклические участки заменяют эквивалентными по трудоемкости линейными участками. Для рассмотренного примера это иллюстрируется на рис.34, г. Сетевые модели позволяют учитывать случайный характер параметров ВН и оценивать минимальную, максимальную и среднюю трудоемкость ВН, представленной алгоритмом [5].

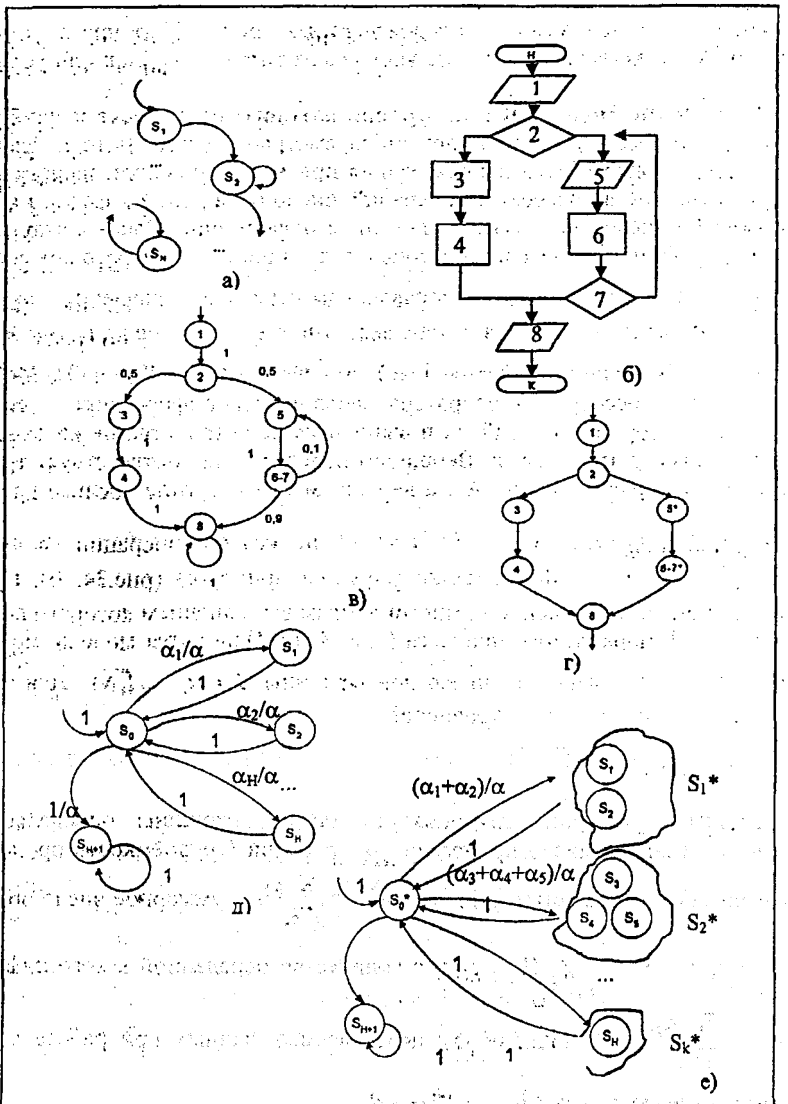


Рис.34. Модели вычислительной нагрузки: а) марковская модель; б) пример алгоритма; в) графовая модель; г) сетевая модель; д) модель на уровне файлов; е) модель на уровне устройств

Марковские модели ВН на уровне файлов. Здесь ВН представляется структурой из процессора и H файлов. Основные параметры модели: число обращений к каждому файлу $\vec{\alpha} = \{\alpha_i \mid \overline{1,H}\}$; количество информации $\vec{\Theta} = \{\Theta_i \mid i = \overline{1,H}\}$, передаваемое при каждом обращении. Суммарная трудоём-

кость процессорной обработки Θ. Соответственно модель представляется марковским процессом с $H+2$ состояниями $\bar{S} = \{S_0, S_1, ..., S_H, S_{H+1}\}$. Состояния соответствуют этапам обработки задачи: S_0 - этапу процессорной обработки; S_{H+1} - этапу завершения обработки; S_i $(i = 1,2,...H)$ – работе с соответствующим файлом. Граф переходов представлен на рис.34, д. Так как общее число обращений к файлам $\alpha_F = \sum_{i=1}^{H} \alpha_i$, к процессору $\alpha = \alpha_F + 1$, то вероятность обращения к i-му файлу $p_{0,i} = \alpha_i / \alpha$, а вероятность завершения обработки $p_{0,1} = 1/\alpha$. Средняя трудоемкость процессорной обработки определится как Θ/α. Соответственно матрица вероятностей передач

$$P = \begin{array}{c|ccccc} & S_0 & S_1 & S_2 & \dots & S_{H+1} \\ \hline S_0 & 0 & \alpha_1/\alpha & \alpha_2/\alpha & \dots & 1/\alpha \\ S_1 & 1 & 0 & 0 & 0 & 0 \\ S_2 & 1 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ S_{H+1} & 0 & 0 & 0 & 0 & 1 \end{array}$$

, а вектор начальных состояний $\bar{p}_0 = (1,0,...,0)$.

Марковские модели ВН на уровне устройств. Здесь ВН отображается на структуре из процессора и K внешних устройств, на которых располагается файловая система. Соответственно модель представляется марковским процессом с $K+2$ состояниями $\bar{S} = \{S_0, S_1, ..., S_K, S_{K+1}\}$. Состояния соответствуют этапам обработки задачи: S_0 - этапу процессорной обработки; S_{K+1} - этапу завершения обработки; состояния S_i $(i = 1,2,...K)$ – работе с соответствующим внешним устройством. Модель легко получается из предыдущей. Для этого все состояния из множества S_i $(i = 1,2,...H)$, соответствующие работе с файлами, размещенными на одном устройстве, заменяют одним эквивалентным состоянием. А число обращений к соответствующему устройству находится суммированием числа обращений ко всем файлам, размещенным на нем. Модель иллюстрируется на рис.86, е, где состояния S_1 и S_2 объединяются в состояние S_1^*, состояния S_3, S_4 и S_5 в состояние S_2^* и т.д. с числом обращений соответственно $(\alpha 1+\alpha 2)/\alpha$, $(\alpha 3+\alpha 4+\alpha 5)/\alpha$ и т.д.

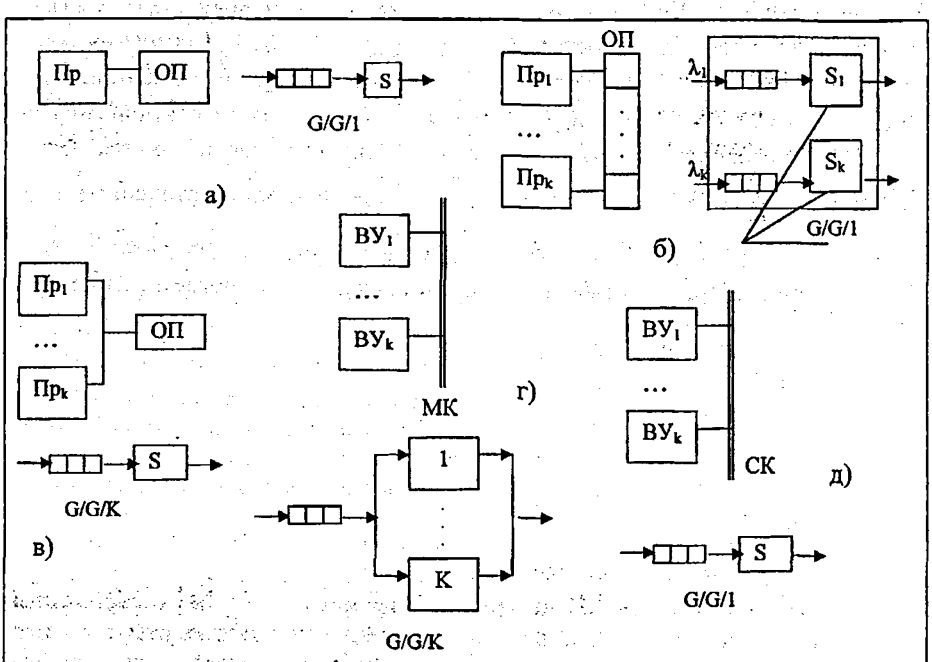


Рис.35. Модели элементов вычислительных систем: а) "ядра" процессор-оперативная память; б) "ядра" многопроцессорной системы с раздельным использованием памяти; в) "ядра" многопроцессорной системы с общей памятью; г) системы внешних устройств с мультиплексным подключением; д) системы внешних устройств с селекторным подключением

2.3. Модели производительности ЭВМ на базе ССМ

Способы отображения типовых компонентов ЭВМ и систем в терминах ССМ иллюстрируются на рис.35. "Ядро" ЭВМ процессор-память отображается одноканальной СМО типа $G/G/1$ с интенсивностью обслуживания μ, определяемой скоростью работы процессора и памяти. Аналогичные компоненты многопроцессорной системы в зависимости от варианта организации отображаются либо набором одноканальных СМО $G/G/1$ либо одной многоканальной СМО $G/G/K$. Компоненты, представляющие собой медленно работающие внешние устройства с мультиплексным подключением, функционируют практически параллельно, а задержки в обслуживании команд ввода-вывода со стороны системы передачи данных (шин, каналов) несущественны по сравнению с длительностью работы самих устройств. Такие компоненты отображаются СМО $G/G/K$ с интенсивностью обслуживания μ, определяемой скоростью работы устройства. Компоненты из быстро работающих внешних устройств требуют монопольного владения системой передачи данных (шинами, каналами) на все время выполнения команды ввода-вывода. Соответственно для них орга-

низуется селекторное подключение. Такие компоненты отображаются СМО типа $G/G/1$.

2.4. Пример построения иерархии моделей системного уровня

Пусть требуется промоделировать работу вычислительной системы (ВС, см. рис.36, а) и оценить ее производительность системы $\lambda 0$, среднее время обслуживания (время ответа) и характеристики использования устройств. Система относится к классу многотерминальных комплексов, включает процессор, оперативную память, два внешних устройства с селекторным и два устройства с мультиплексным подключением. Предназначена для оперативной обработки информации (дисциплина использования процессора RR – выделение постоянного кванта времени) и обслуживает одновременно М пользователей, подключенных к мультиплексному каналу посредством удаленных терминалов. Процесс обработки включает чередующиеся этапы: обдумывание и формулирование пользователем задачи или запроса и отправку его в ВС; выделение ресурсов для выполнения задачи или запрос; обработка, получение и отправка результатов на пользовательский терминал, освобождение ресурсов. В процессе счета задача может покинуть процессор: если произошло обращение к одному из внешних устройств; если задача завершилась; если истек квант времени (задача возвращается в очередь задач, ожидающих освобождения процессора). Соответственно поток задач к процессору включает новые задачи, задачи с истекшим квантом и задачи, завершившие операцию ввода-вывода.

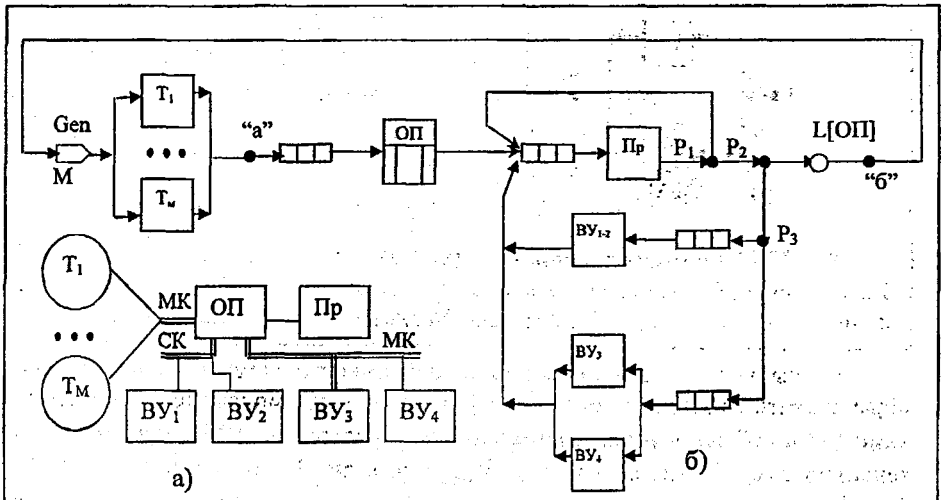


Рис.36. Моделирование вычислительной системы: а) вычислительная система; б) имитационная модель

Модель 1. Имитационная модель ВС выполнена в терминах ССМ и представлена на рис.36, б. При ее построении сделаны необязательные допущения: - все пользователи генерируют задачи одного класса (с одинаковыми маршрутами, законами распределения параметров и времени обдумывания); - каждый пользователь генерирует следующую задачу только по завершении предыдущей. Модель может быть реализована средствами GPSS.

Модель 2. Аналитическая, аппроксимационная модель ВС построенна по методу эквивалентных потоков и представлена на рис.37. При ее построении сделаны дополнительные допущения: - длительности обслуживания в узлах и длительность обдумывания распределены по экспоненциальному закону; - из модели, построенной выше, исключается узел памяти и связанный с ним узел ее освобождения.

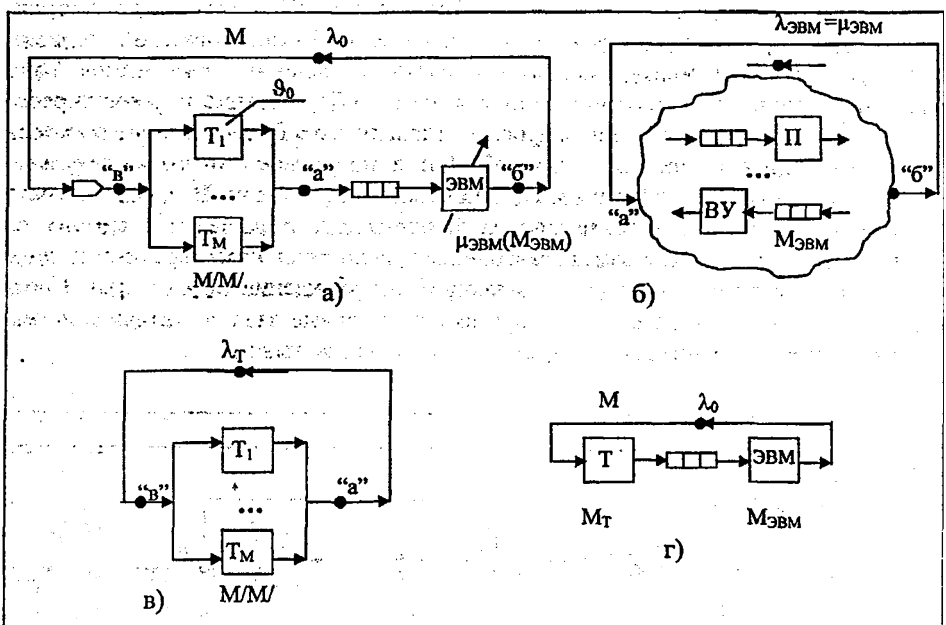


Рис.37. Аппроксимационная модель: а) упрощенная модель объекта без блока памяти; б) эквивалентная модель участка сети "а-б"; в) эквивалентная модель участка сети "а-в"; г) модель объекта из эквивалентных узлов

Чтобы учесть влияние ограниченной емкости памяти V, позволяющей обрабатывать одновременно до $]V/\Delta v[$ задач, участок модели (сети) между точками "а" и "б" заменяется эквивалентным агрегатом - ЭВМ с переменной интенсивностью обслуживания $\mu_{ЭВМ}(M_{ЭВМ})$ (см. рис.37, а). Здесь $M_{ЭВМ} = 1, 2,\ldots,]V/\Delta v[$, а Δv - средняя емкость памяти, требуемая задаче. Модель агрегата в виде замкнутой сети МО представлена на рис.37, б. Она обладает производительностью $\lambda_{ЭВМ}$, соответствующей значению $\mu_{ЭВМ}$. Оставшаяся часть модели

представляет собой терминальный узел (рис.37, в), который также можно заменить эквивалентным агрегатом *Т*. Его производительность λ_T определяется так же как и у предыдущего агрегата замыканием выхода и входа. Таким образом, модель декомпозируется на два агрегата (рис.37, г) - две замкнутые сети, которые при сделанных допущениях рассчитываются отдельно аналитически.

Тогда расчет модели производится в следующей последовательности: 1. Определяется производительность $\lambda_{ЭВМ}$ агрегата ЭВМ. Она рассчитывается аналитически как производительность замкнутой сети МО (рис.37, б) при значениях $M_{ЭВМ} = 1, 2, \ldots,]V/\Delta v[$. Тем самым определяется зависимость (рис.38, а) производительности агрегата от числа обрабатываемых заявок $\lambda_{ЭВМ} = \mu_{ЭВМ}(M_{ЭВМ})$. 2. Рассчитывается производительность терминального агрегата на основе закона Литтла. Так из соотношения $\lambda_T U_T = \lambda_T(\vartheta_0 + \omega_0) = M_T$ с учетом отсутствия очередей (так как максимальное число заявок в этом агрегате равно числу терминалов) следует, что $\lambda_T \vartheta_0 = M_T$ и $\lambda_T(M_T) = M_T/\vartheta_0$, т.е. его производительность (рис.38, б) зависит от числа обрабатываемых заявок линейно. 3. Рассчитывается производительность λ_0 всей сети (рис.38, г) с учетом балансовых соотношений $M_T + M_{ЭВМ} = M$ и $\lambda_0 = \lambda_{ЭВМ} = \lambda_T$. Подставляя в последнее выражение значение M_T из балансового соотношения и значение λ_T из п.2, получим уравнение $(M - M_{ЭВМ})/\vartheta_0 = \mu_{ЭВМ}(M_{ЭВМ})$ для нахождения среднего числа заявок, обрабатываемых первым агрегатом $M_{ЭВМ} = M^*_{ЭВМ}$. Тогда производительность сети $\lambda_0 = \lambda_0(M^*) = \lambda^*_0 = (M - M^*_{ЭВМ})/\vartheta_0$ (описанные действия иллюстрируются рис.39). 4. Определяется среднее время ответа $U_{ОТВЕТА} = (M - \lambda^*_0 \vartheta_0)/\lambda^*_0$ из соотношения $\lambda_0(U_{ЭВМ} + \vartheta_0) = M$ и другие характеристики системы.

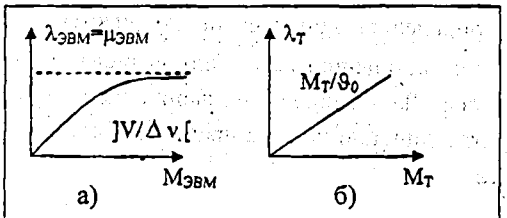


Рис.38. Зависимость производительности: а) эквивалентного узла "а-б" ; б) эквивалентного узла "а-в"

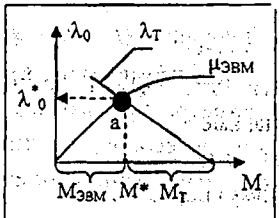


Рис.39. Графическое решение сети

Модель 3. Классическая аналитическая модель ВС, построенная в виде замкнутой однородной экспоненциальной сети МО с центральным обслуживающим узлом. При ее построении сделано дополнительное допущение о том,

что ограниченная емкость памяти не оказывает существенного влияния на характеристики модели.

Соответственно из модели, построенной выше, не только исключается узел памяти и связанный с ним узел освобождения памяти, но и полностью игнорируется влияние ограниченной емкости памяти на характеристики обслуживания заявок.

Модель 4. Аналитическая модель ВС, построенная в виде замкнутой однородной экспоненциальной сети МО, состоящей из двух узлов - терминального в виде многоканальной СМО и обрабатывающего в виде одноканальной СМО. Т.е. участок модели (сети) между точками "а" и "б" заменяется эквивалентным агрегатом - ЭВМ, но с постоянной средней интенсивностью обслуживания $\mu_{ЭВМ}(M_{ЭВМ}) = \mu_{ЭВМ} = \text{const}$.

Модель 5. Асимптотическая модель ВС без учета параметров узла памяти сводится к асимптотической модели замкнутой сети МО, рассмотренной в Главе 2, § 3.8.

Применение полученных там соотношений даст следующие выражения для расчета значений характеристик для режима работы сети до насыщения, когда $M < M^*$

$$U_{ОТВЕТА} = \sum_{i=1}^{N} \alpha_i \cdot \vartheta_i = U_1 = const \text{ и}$$

$$\lambda_0 = M/(\vartheta_0 + U_1) = \lambda_0(M) = \text{var}$$

и после наступления насыщения, при $M > M^*$

$$\lambda_0 = M^*/(\vartheta_0 + U_1) = const \text{ и}$$

$$U_{ОТВЕТА} = M(\vartheta_0 + U_1)/M^* - \vartheta_0$$

Значение $M^* = (U_1 + \vartheta_0)/\alpha_s \cdot \vartheta_s$ определяет критическое количество подключенных терминалов, приводящее к насыщению сети. Оно вычисляется через параметры "узкого места" - s-го узла. Дальнейшее увеличение числа терминалов не целесообразно, т.к. приводит к линейному (а на практике экспоненциальному) нарастанию времени ответа.

$$\lambda_0 U_{ТЕРМ} = M_{ТЕРМ}; \quad \lambda_0 \vartheta_0 = M_{ТЕРМ}; \quad \lambda_0 = \frac{M_{ТЕРМ}}{\vartheta_0} = \frac{M - M_{ЭВМ}}{\vartheta_0}$$

3. Уровень структурно-алгоритмического моделирования

3.1. Особенности, цели и средства структурно-алгоритмического моделирования

В вычислительной технике это уровень интегральных схем (БИС и СБИС), функционирование которых задается спецификациями, описаниями законов функционирования на алгоритмическом уровне. Здесь основные задачи моделирования: - составление спецификаций, документирование проектов цифровых устройств (ЦУ); - моделирование ЦУ с целью разработки и отладки алгоритмов их функционирования; - разработка и отладка регистровой структуры ЦУ; - отладка ЦУ с использованием моделей окружающей среды; - генерация тестов; - отладка на моделях ЦУ программного и микропрограммного обеспечения; - подготовка исходных данных для кремниевой компиляции.

Широко используются программные средства: языки и системы типа ОККАМ, АДА, VHDL, HHDL, MG3, DDL, ФОРОС, МОДИС-ВЕС, ОСС2 и др. Их характерные черты: специфичные программные компоненты (например, архитектура, интерфейс, порт и т.п.); - специфичные объекты обработки (например, данные типа бит, битовый вектор, данные с размерностью; особые классы объектов – сигналы, многозначные сигналы и т.п.); - специфичные операторы (например, для управления параллельными процессами и т.п.); - средства ведения иерархических проектов, библиотек проектов и примитивов.

3.2. Язык VHDL. Компоненты, объекты, операторы

Язык VHDL (very high speed integrated circuits Hardware Description Language) – язык описания аппаратуры, построенной на базе сверхскоростных ИС. Это входной язык многих САПР, например, фирм Cadence, Synopsys, САПР Xilinx Foundation Series 2.1i, MAX+PLUSII, САПР заказных СБИС фирмы Mentor Graphics, системы моделирования ModelSim. С 1983 г. спонсором в его развитии выступало МО США, с 1987 г. общество радиоинженеров IEEE. Язык является своеобразным стандартом в области спецификации ЦУ, с 1987 г. принят в качестве стандарта ANSI/IEEE Std 1076-1987; в 1993 г. принят новый стандарт Std 1076-1993 (стандарт VHDL'93). В 1999 г. утвержден стандарт Std 1076.1-1999 (VHDL-AMS), позволяющий описывать модели аналоговых и смешанных (цифро-аналоговых) схем. Обеспечивает три уровня (стиля) описания архитектуры. Первый - поведенческий, алгоритмический (behavioral) для моделирования ЦУ на уровне ИС. Второй - потоковый (data-flow) для описания ЦУ в виде множества операций регистрового уровня. Третий - структурный (structural) для описания ЦУ в виде иерархически связанных компонентов при моделировании на регистровом и вентильном уровне. Общая структура языка, области его преимущественного применения и составляющие эффективности моделирования представлены на рис.40-42.

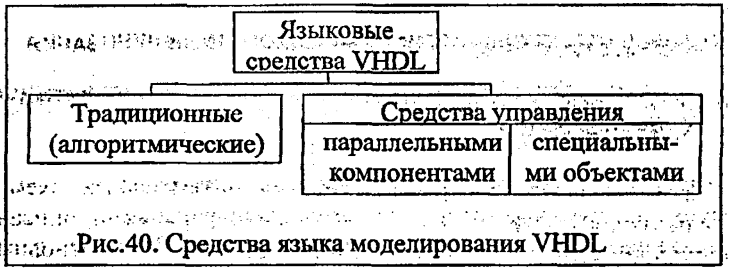


Рис.40. Средства языка моделирования VHDL

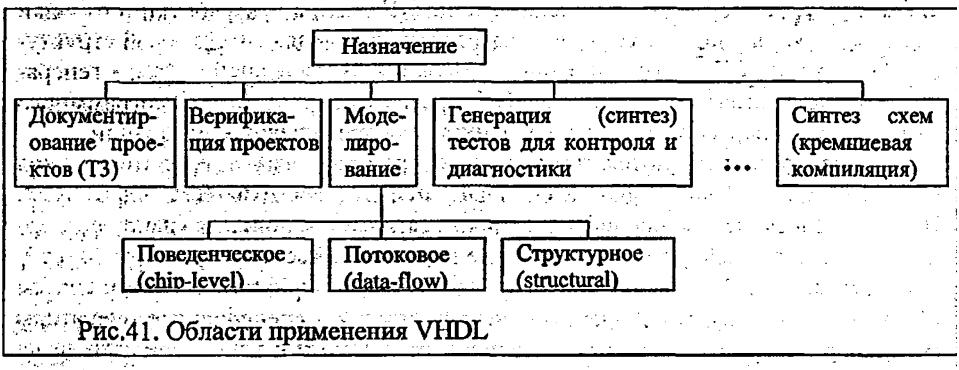


Рис.41. Области применения VHDL

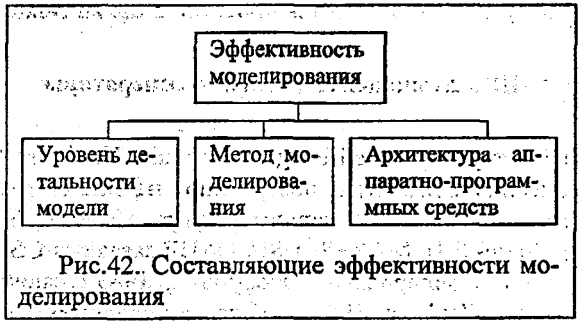


Рис.42. Составляющие эффективности моделирования

Компоненты. Структура средств языка представлена на рис.43. Цифровое устройство, как правило, состоит из блоков, имеющих входы и выходы, им соответствуют параллельный оператор VHDL - block (блок). В блоках происходят параллельные, последовательные, параллельно-последовательные, синхронные и асинхронные процессы обработки сигналов. Реальным процессам соответствует параллельный оператор VHDL - process (процесс), а сигналам соответствует класс объектов VHDL - signal (сигнал). ЦУ взаимодействует с внешней средой, передавая и принимая сигналы через систему входов-выходов, им соответствуют компоненты VHDL - interface (интерфейс) и port (порт). Блоки и процессы VHDL образуют такие компоненты как architecture (архитектура) и configuration (конфигурация), а их объединение с компонентом интерфейс полностью описывает ЦУ и образует объект моделирования - entity (объект проекта).

Во всех компонентах и параллельных операторах VHDL обрабатываются сигналы и только в операторе process, а также в вызываемых им подпрограммах допустимо использование традиционных переменных. Тем самым процессы обеспечивают поведенческий стиль моделирования с возможностью алгоритмического описания функционирования ЦУ в терминах переменных и сигналов на базе последовательных операторов как стандартного так и специфического набора. В блоках реализуется структурный и потоковый стиль описания и используются только сигналы и параллельные операторы.

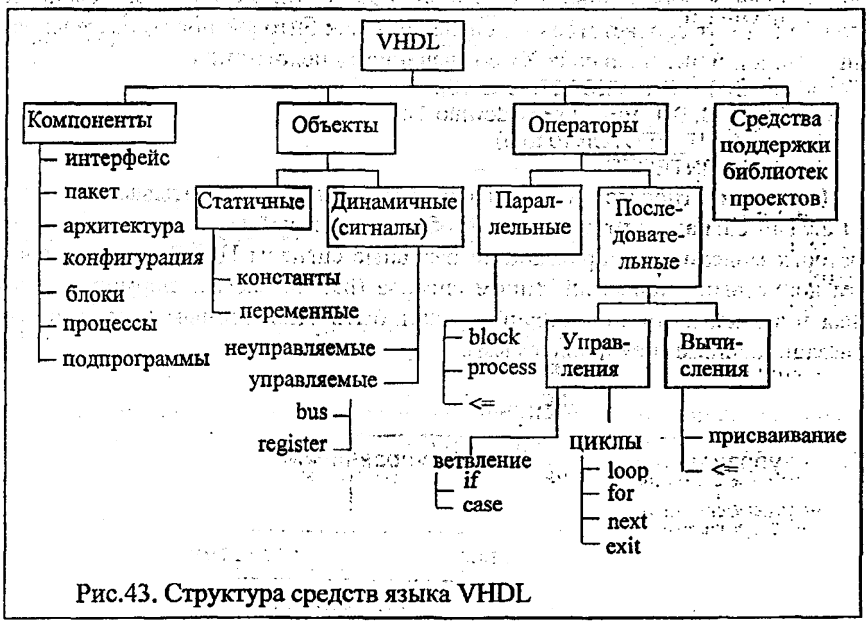


Рис.43. Структура средств языка VHDL

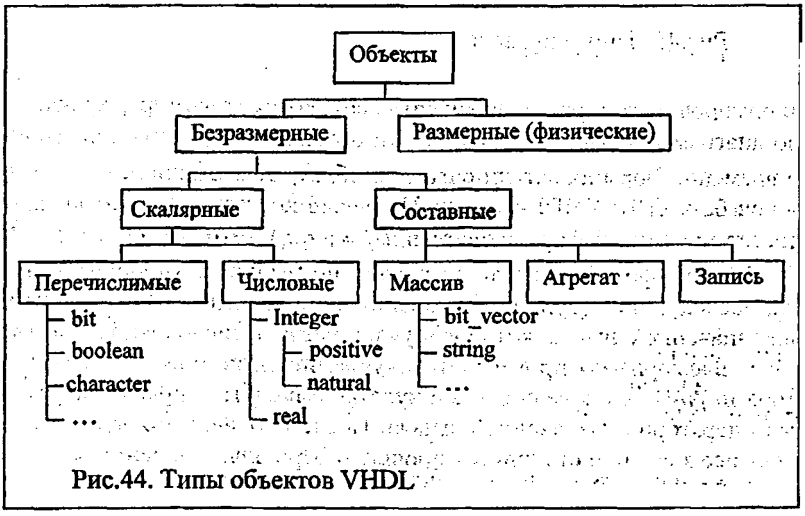


Рис.44. Типы объектов VHDL

Объекты VHDL и их типы классифицированы на рис.43, 44. Они включают класс традиционных статичных объектов-данных (это переменные и константы) и класс динамичных объектов-сигналов. При этом объекты-сигналы могут принимать значения любого типа, допустимого для объектов-данных. Основные типы VHDL приведены на рис.44. Среди них следует особо отметить размерные типы, тип bit и тип bit_vector

type BIT_VECTOR is array (NATURAL range <>) of BIT.

Используя типы, можно описать переменные и сигналы. Ниже приведен пример описания переменных DATA и BUS как битовых последовательностей длиной 16 и 7 бит соответственно, сигнала X как битовой последовательности (шины) длиной 6 бит и сигнала X1 со значениями целого типа

```
variable DATA: BIT_VECTOR (0 to 15);
variable BUS: BIT_VECTOR (7 downto 1);
signal   X: BIT_VECTOR (0 to 5);
signal   X1: INTEGER;
```

Сигналы относятся к специфическому классу обрабатываемых объектов. Каждый сигнал - это глобальный объект, видимый во всех компонентах и операторах модели и отображающий реальные сигналы ЦУ. Задается именем, типом допустимых значений, типом сигнала (bus, register) и формирователем. Сигнал в зависимости от описания может быть защищенным (управляемым) или незащищенным (неуправляемым).

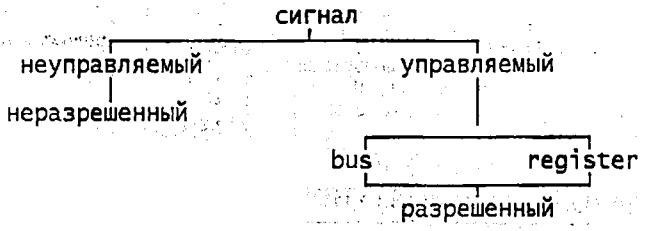


Рис.45. Типы сигналов

Формирователь (трасса, драйвер) – списковая структура, хранящая временную диаграмму сигнала $D_x = \{(x_i,t_i)\}$ и отображающая характер его изменения во времени. Формирователи сигналов образуют временные списки информационной базы (ИБ) VHDL-модели. На рис.46 показана диаграмма сигнала X и соответствующий ей формирователь $D_x = \{(x_i,t_i)\} = \{(0; 0), (1; 2), (0; 4), \ldots (1, 10), \ldots\}$. Каждый формирователь включает предыдущие (прошлые) значения сигнала, последующие (будущие) значения, а для текущего модельного времени задает значение сигнала, которое и участвует в вычислениях как переменная (например, в алгоритмах процессов). Текущее значение может анализироваться операторами WAIT, а также списками чувствительности в правых частях параллельных операторов назначения сигнала. После продвижения модельного времени текущее значение становится прошлым. Прошлые значения сигнала в хо-

де моделирования не меняются, однако могут использоваться для анализа работы модели, например, с помощью атрибутивных функций и для управления ею. Например, функция S'LAST определяет в формирователе значение сигнала S, предшествующее текущему. Функция S'STABLE(T) возвращает значение ИСТИНА, если в течение отрезка времени T, предшествующего текущему модельному времени, сигнал S не менялся. Функция S'DELAYED(T) возвращает значение, которое сигнал S имел T времени назад. Будущие значения являются прогнозируемыми, они могут неоднократно меняться в результате выполнения операторов назначения сигналов

<имя_сигнала> <= [transport] <трасса_будущих_значений>,
<трасса_будущих_значений>::= <выражение> after <задержка>.

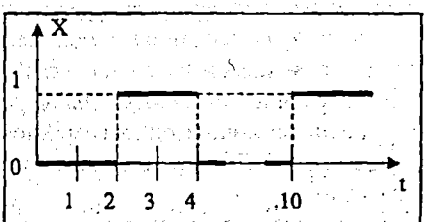


Рис.46. Драйвер сигнала X

Выполнение оператора назначения определяется комбинацией типа сигнала и режима работы. Режим работы оператора назначения [guarded], [transport] определяет специфику взаимодействия оператора с оператором блока и формирователем сигнала.

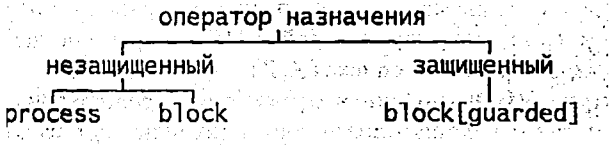


Рис.47. Типы операторов назначения

Параллельный оператор назначения может работать в режимах защищенного [guarded] и не защищенного назначения сигнала. Операторы назначения сигналов выполняются параллельно и однократно, причем на каждый сигнал в процессе может быть только один формирователь, а взаимодействие между одноименными сигналами разных процессов управляется функцией разрешения. Так каждый сигнал, производимый множеством формирователей, получает ведущие значения (driven), которые с помощью функции разрешения или мультиплексоров сигналов превращаются в разрешенные значения (resolved).

Например, выполнение оператора X<= (Z+S) after 3 ns означает, что сигнал X через 3 ns станет равным сумме значений Z+S и это прогнозируемое значение

как отдельная запись будет добавлено в список будущих событий формирователя сигнала.

В VHDL моделируется два типа задержек. Транспортная задержка означает, что сигнал безусловно принимает вычисленное в операторе значение по истечении указанной задержки. Т.е. такая запись будущего значения сигнала в формирователе не может быть отменена и изменение сигнала в правой части оператора приводит к его изменению на выходе независимо от длительности сигнала на входе. Инерционная задержка предполагает, что сигнал появится в будущем только в том случае, если сигналы в правой части оператора будут сохранять свои значения в течение всего времени задержки. В противном случае, это прогнозируемое значение будет отменено следующим, сформировавшимся этим или другим оператором, пока время задержки предыдущего значения не было исчерпано. Соответствующая запись будет исключена из списка будущих значений формирователя сигнала и заменена новой записью.

Операторы VHDL разделяются на последовательные, выполняемые один за другим, и параллельные, предполагающие одновременное выполнение. Архитектура, конфигурация проекта в VHDL строится из параллельных операторов, а процессы из последовательных.

Среди последовательных операторов особую роль выполняют операторы ЖДАТЬ следующих типов:

wait on <список_сигналов> ,
wait until < условие_ожидания > ,
wait for <длительность_ожидания >.

Например, при выполнении операторов

```
wait until (R=0),
wait for 100 ns,
wait on X,Z
```

Выполнение процесса может быть приостановлено, пока значение R не станет равным нулю, либо пока не пройдет 100 ns времени, либо пока не изменится хотя бы один сигнал из списка (X, Z).

Структура VHDL-модели в общем виде представлена на рис.48. Она включает: - интерфейс с описаниями портов (входных in и выходных out сигналов); - архитектуру из последовательных и параллельных операторов, описывающую объект; - ИБ, основной частью которой являются формирователи сигналов. Модель взаимодействует с внешней средой посредством сигналов, передаваемых через порты интерфейса. При этом входные сигналы попадают в со формирователи ИБ входных сигналов, а значения выходных сигналов берутся из формирователей выходных сигналов.

Архитектура состоит хотя бы из одного оператора block, который в свою очередь также может состоять из параллельных операторов block, process, операторов назначения сигналов. Параллельные операторы чувствительны к изменению сигналов, от которых они зависят. При их изменении операторы активизируются и делается попытка их выполнить. Предполагается, что исполнение параллельных операторов может относиться к одному моменту модельного времени. При их выполнении модель оказывает влияние на формирователи сигналов, внося новые, отменяя или изменяя ранее сформированные записи бу-

дущих значений формирователей. Таким образом параллельные операторы, взаимодействуют друг с другом через ИБ.

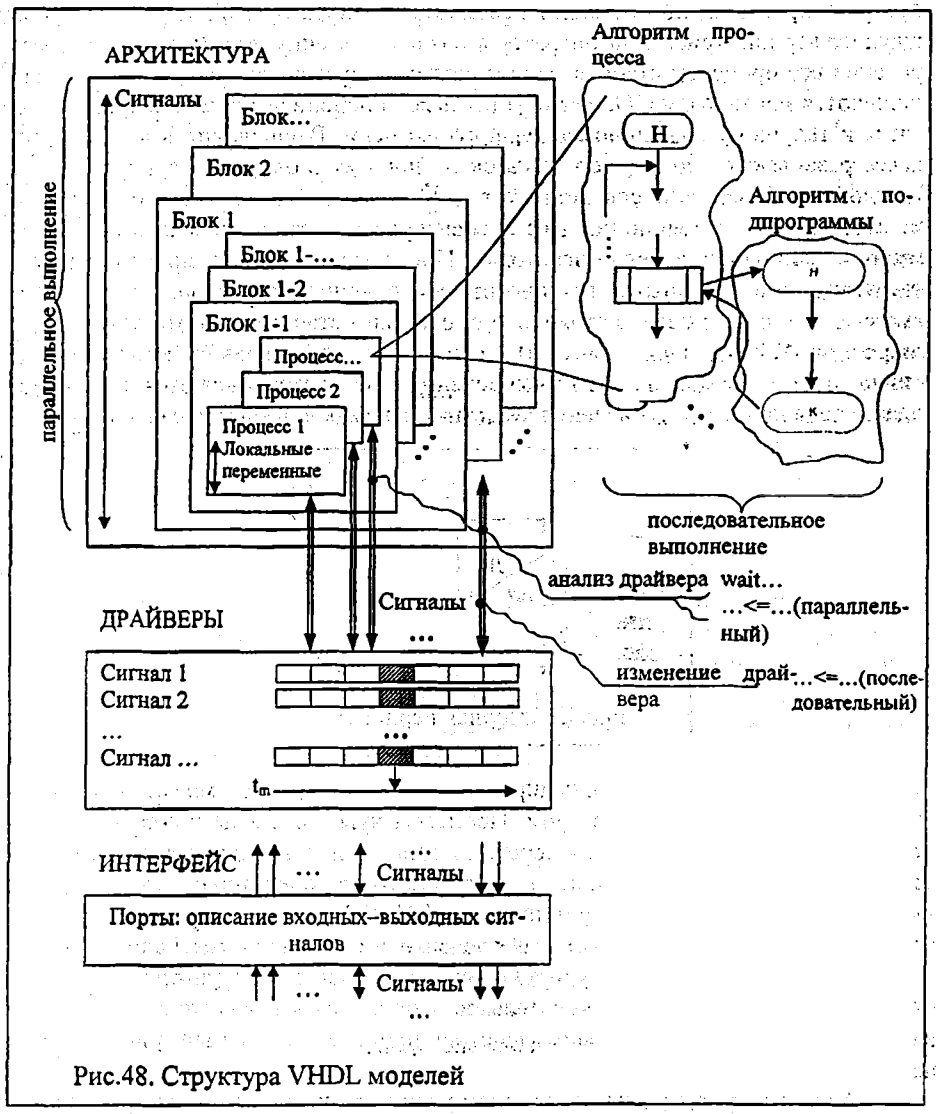


Рис.48. Структура VHDL моделей

3.3. Процессы. Особенности описания и взаимодействия

Если модель является поведенческой, то ее архитектура описывается набором операторов process. Особенность использования и выполнения парал-

лельного оператора process – алгоритмическое описание закона функционирования объекта с применением как переменных так и сигналов. Описание процесса сходно с описанием модулей на универсальных алгоритмических языках программирования, но позволяет использовать операторы Ждать для создания ждущих вершин (рис.49) и операторы назначения сигналов. В начале моделирования все процессы становятся активными, запускаются одновременно и выполняются параллельно. Операторы процессов исполняются мгновенно и относятся к текущему значению модельного времени. Вычисления в каждом процессе развиваются до тех пор, пока он не попадет в одну из ждущих вершин. Каждой такой вершине соответствует одно состояние процесса. При переходе из состояния в состояние процесс активизируется, что сопровождается "мгновенным" выполнением его операторов. При каждом продвижения модельного времени делается попытка продолжить выполнение процессов, для чего они запускаются с текущего состояния, т.е. с точки останова - соответствующего оператора Wait (ждущей вершины). Если условия для перевода процесса в активное состояние появились, то вычисления в нем возобновляются, а по завершении тела процесса вычисления автоматически продолжаются с его начала.

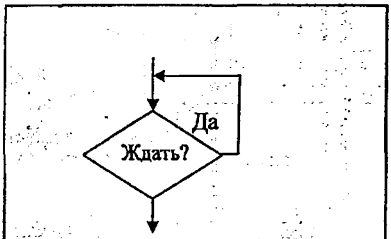


Рис.49. Ждущая вершина

Таким образом, при моделировании каждый процесс может быть представлен соответствующим графом. Процессы чувствительны к событиям - условиям их активизации. Это: - переход процесса из пассивного в активное состояние по истечении времени, указанного в операторах типа wait for <длительность_ожидания >; - установка в ИСТИНУ значения проверяемого выражения в операторах типа wait until < условие_ожидания >; - изменение значения (значений) сигналов, фигурирующих в списках операторов wait on <список_сигналов> и wait for <длительность_ожидания >. Последние два варианта и позволяют организовать взаимодействие процессов по "семафорному" принципу через общую ИБ.

Алгоритм управления процессной моделью включает следующие шаги: - анализируется ИБ, просматриваются формирователи и ищется ближайшее событие (например, изменение значения какого-либо сигнала либо разблокировка процесса); - модельное время продвигается к времени ближайшего события; - последовательно запускаются все процессы, которые чувствительны к наступившему событию, и выполняются до тех пор, пока процесс не переходит в

следующее состояние ожидания. Указанные шаги повторяются до тех пор, пока не истечет модельное время.

3.4. Примеры описания VHDL-моделей

Пример 1. Представляет VHDL-модель объекта с законом функционирования, реализующим логическую функцию Y=X1*X2+X3. Объект проекта можно считать будущим цифровым устройством с заданным в виде логической функции законом функционирования. Модель выполнена в стиле поведенческого описания.

```
entity EX1 is
    port (X1, X2, X3: in BIT; Y: out BIT)
end EX1;
architecture AR_EX1_1 of EX2 is
begin
    process
    begin
          wait on X1, X2, X3;
          Y<=X1 and X2 or X5 after 15 ns;
    end process;
end AR_EX1_1
```

Здесь и входные сигналы X1, X2, X3 и выходной сигнал Y представляют собой битовые значения (нули или единицы). Архитектура содержит один блок, по умолчанию состоящий из одного параллельного оператора процесса. В процессе есть одна ждущая вершина, поэтому при запуске процесс ждет, пока не изменится хотя бы один входной сигнал. Как только это произойдет процесс становится активным, производится вычисление и назначение значения сигнала Y и переход в начало алгоритма на ждущую вершину. Здесь процесс может стать пассивным до нового изменения входных сигналов и так до тех пор, пока не завершится моделирование.

Пример 2. Представляет VHDL-модель объекта с тем же законом функционирования. Модель выполнена в стиле поведенческого описания. В отличие от предыдущей здесь использованы переменные T1 и T2, на базе которых и реализуется алгоритм, а вычисленное в алгоритме значение назначается выходному сигналу Y с соответствующей задержкой

```
entity EX1 is
    port (X1, X2, X3: in BIT; Y: out BIT)
end EX1;
architecture AR_EX1_2 of EX2 is
begin
    process
        variable T1,T2 : BIT;
    begin
        wait on X1,X2,X3
```

```
    T1:=X1 and X2;
    T2:=T1 or X3;
    Y<=T2 after 15ns;
  end process;
end AR_EX1_2
```

Пример 3. Представляет VHDL-модель объекта проекта с двумя входными и одним выходным сигналом. Объект реализует закон функционирования, описанный алгоритмически на рис. 50, а. Модель выполнена в стиле поведенческого описания.

```
entity EX2 is
    port (A,B: in BIT_VECTOR (0 to 10);
          Z: out BIT_VECTOR (0 to 10));
end EX2;
architecture AR_EX2 of EX2 is
begin
    process
        variable Y: BIT_VECTOR(0 to 10);
    begin
        wait on A,B;
        Y:=A+B;
        if Y<3 then
            wait for 10 ns
        else
            Y:=sqrt(Y)
        end if;
        wait for 5 ns;
        Z<= Y after 5 ns;
    end process;
end AR_EX2
```

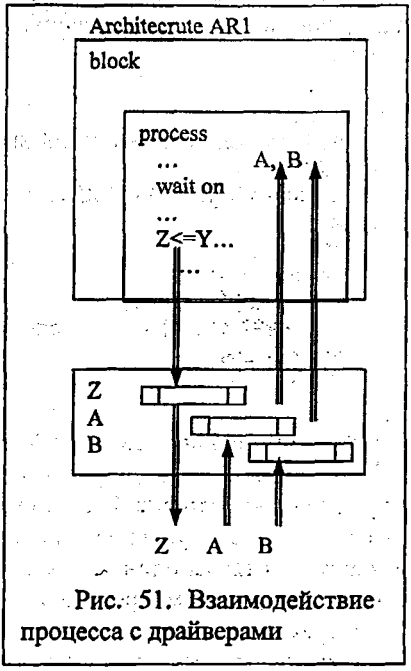


Рис. 51. Взаимодействие процесса с драйверами

Здесь входные сигналы A и B и выходной сигнал Z - битовые последовательности из 11 нулей и единиц. Характер взаимодействия сигналов и процессов с соответствующими формирователями ИБ показан схематично на рис.51. Архитектура модели включает по умолчанию один блок, состоящий из одного параллельного оператора процесса. В процессе кроме указанных сигналов используется переменная У. Процесс может находиться в одном из трех состояний.

После запуска процесс попадает в состояние S1 и ждет, пока не изменится хотя бы один входной сигнал. Как только это произойдет, процесс активизируется и его операторы выполняются в соответствии с алгоритмом (в текущее модельное время) до тех пор пока, пока он не перейдет в одно из возможных состояний. Здесь это либо состояние S2 (задержка на 10 нс) либо S3 (задержка на 5 нс). При последующей активизации процесса выходному сигналу Z будет назначено ранее вычисленное значение (сигнал примет его в будущем через 5 нс, о чем в его формирователе сигнала будет сформирована соответствующая запись), а вычисления продолжатся с начала, т.е. с анализа изменения хотя бы

одного входного сигнала. Графовое представление модели представлено на рис.52, б.

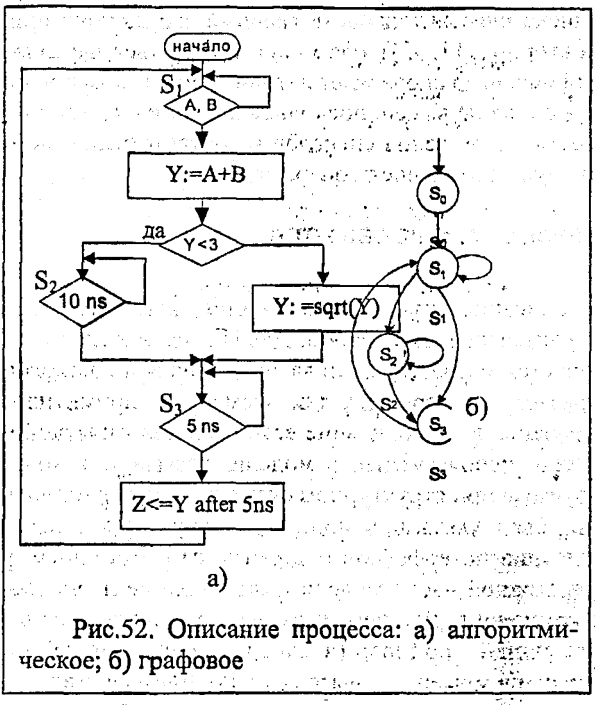


Рис.52. Описание процесса: а) алгоритмическое; б) графовое

Пример 4. Представляет VHDL-модель объекта проекта, изображенного на рис.53, с законом функционирования, реализующим логическую функцию Y=X1* X2+ X3. Можно считать объект цифровым устройством с заданной функцией и определенной структурой. Модель выполнена в стиле потокового описания (уровень регистровых передач), каждому элементу объекта соответствует свой параллельный оператор назначения сигнала.

```
entity EX1 is
    port (X1, X2, X3: in BIT; Y: out BIT)
end EX1;
architecture AR_EX1_3 of EX1 is
begin
   block
      signal T1: BIT;
   begin
       T1<=X1 and X2 after 5ns;
       Y<=T1 or X3 after 10ns;
   end block;
end AR_EX1_3
```

Здесь архитектура включает один блок, состоящий из двух параллельных операторов назначения сигналов. Каждый из них чувствителен к изменению сигналов своего списка чувствительности (первый запускается при изменении сигналов X1, X2, а второй - T1, X3), оба могут запускаться параллельно. После выполнения каждый оператор снова ждет активизации. Таким образом, порядок записи операторов в блоке не важен, поскольку порядок их запуска определяется последовательностью изменения сигналов и соответственно порядком "срабатывания" списков чувствительности операторов.

3.5. Структурное описание объектов

Структурное описание объекта выполняется, когда известны элементы структуры и связи элементов между собой. Для ЦУ это может означать, что его структура проектируется с учетом состава имеющейся технологической библиотеки, т.е. описывается в терминах тех элементов (примитивов), которые созданы, спроектированы, и чье описание есть в технологической библиотеке. Элементы библиотеки, используемые в модели, называются компонентами - component. Используемые при структурном описании типы компонентов (порты и названия) должны быть указаны в модели до операторной части (в манере, напоминающей описание интерфейсов подпрограмм с указанием формальных параметров). В операторной части модели каждый элемент описывается через соответствующий компонент. Для него указывается уникальное название, тип компонента и карта связей - port map (в манере, напоминающей описание интерфейсов подпрограмм в момент их вызова с указанием фактических параметров), описывающая его входные и выходные сигналы. Каждый компонент в модели рассматривается как параллельный оператор, чувствительный к изменению сигналов, являющихся для него входными.

Пример 5. Представляет VHDL-модель объекта проекта, изображенного на рис.53. Его структура в терминах библиотеки компонентов (примитивов) изображена в верхней части рис.50 и включает два типа компонентов. Предполагается, что каждый компонент был предварительно разработан и описан на языке VHDL. Это компонент AND2 port(I1,I2:in BIT;O:out BIT), представляющий двухвходовой логический элемент И с задержкой в 5ns

```
entity AND2 is
   port(I1,I2:in BIT;O:out BIT);
end AND2;
architecture AR_AND2 of AND2 is
begin
   O<=I1 and I2 after 5ns;
end AR_AND2
```

и компонент OR2 port(I1,I2:in BIT;O:out BIT), представляющий двухвходовой логический элемент ИЛИ с задержкой в 5ns

```
entity OR2 is
   port(I1,I2:in BIT;O:out BIT);
end OR2;
architecture AR_OR2 of OR2 is
```

```
begin
   O<=I1 or I2 after 10ns;
end AR_OR2.
```

VHDL-модель объекта проекта выполнена в стиле структурного описания

```
entity EX1 is
    port (X1, X2, X3: in BIT; Y: out BIT)
end EX1;
architecture AR_EX1_4 of EX1 is
   component AND2  port(I1,I2:in BIT;O:out BIT);
   component OR2   port(I1,I2:in BIT;O:out BIT);
   signal X4:BIT;
begin
   ELEM_1:AND2   port map (X1,X2,X4);
   ELEM_2:OR2    port map (X3,X4,Y);
end AR_EX1_4
```

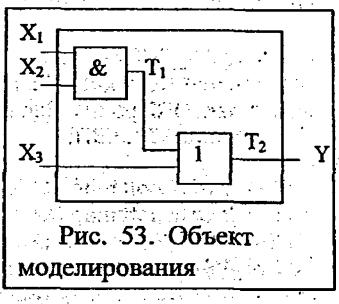


Рис. 53. Объект моделирования

Пример 6. Представляет VHDL-модель объекта проекта EX3 с 4 входными сигналами X1-X4 и двумя выходным сигналами Y1 и Y2. Объект реализует закон функционирования, описанный структурно в нижней части рис. 50. Структура объекта включает 6 элементов ELEM_1-ELEM_6 и использует компоненты двух типов AND2 и OR2 из технологической библиотеки, описанной в примере 5. В модели используются внутренние сигналы X5-X8. Все сигналы типа BIT. Модель выполнена в стиле структурного описания

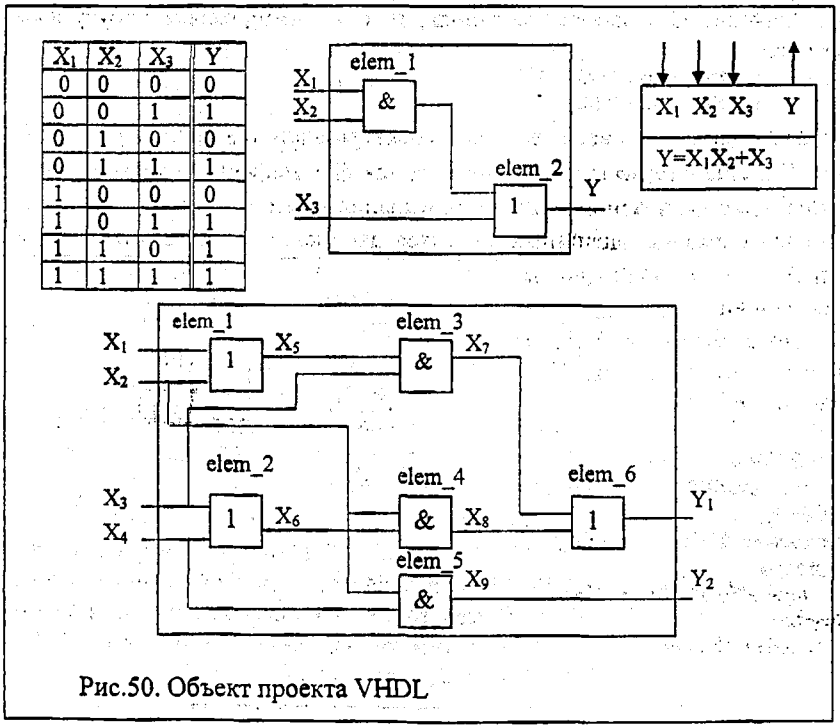


Рис.50. Объект проекта VHDL

```
entity EX3 is
   port(X1,X2,X3,X4:in BIT;Y1,Y2:out BIT);
end EX3;
architecture AR_EX3 of EX3 is
   component AND2 port(I1,I2:in BIT;O:out BIT);
   component OR2 port(I1,I2:in BIT;O:out BIT);
   signal X5,X6,X7,X8:BIT;
begin
   ELEM_1:OR2 port map (X1,X2,X5);
   ELEM_2:OR2 port map (X3,X4,X6);
   ELEM_3:AND2 port map (X5,X3,X7);
   ELEM_4:AND2 port map (X2,X6,X8);
   ELEM_5:AND2 port map (X2,X4,Y2);
   ELEM_6:OR2 port map (X7,X8,Y1);
end AR_EX3
```

3.6. Особенности поведенческого описания последовательностных схем

Пример 7. Представляет VHDL-модель неуправляемого генератора прямоугольных импульсов EX4 (рис.54, а). При включении он генерирует серию из N прямоугольных импульсов с периодом T (рис.54, г). Соответственно у объекта проекта нет входных сигналов и есть один выходной сигнал C (out BIT), который представляет сгенерированные импульсы. Если до начала работы генератора значение C установлено в ноль, то один импульс имитируется парой операторов,

```
C<=transport 1 after T/2;
C<=transport 0 after T,
```

создающих в драйвере сигнала C соответственно две записи (*1; t_M +T/2*) и (*0; t_M + T*). Оператор wait for T позволяет как бы сместить модельное время на значение T, соответственно при втором выполнении той же пары операторов назначения сигналов в драйвере появятся две аналогичные записи (*1; T+t_M +T/2*) и (*0; T+ t_M + T*), описывающие второй импульс и т.д. Модель, выполненная в стиле поведенческого описания, представлена ниже

```
entity EX4 is
   port(C:out BIT);
end EX4;
architecture AR_EX4 of EX4 is
   process
      variable I,N,T:INTEGER;
   begin
      for I in 1 to N
```

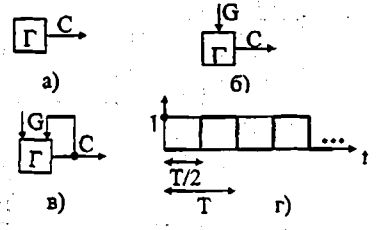


Рис.54. Генераторы импульсов: а) неуправляемый генератор; б) управляемый генератор одиночных импульсов; в) управляемый универсальный генератор; г) диаграмма импульсов

```
            C<=transport 1 after T/2;
            C<=transport 0 after T;
            wait for T;
        end loop;
    end process;
end AR_EX4
```

Пример 8. Представляет VHDL-модель генератора прямоугольных импульсов EX5 (рис.54, б). Его отличие состоит в использовании специального входного сигнала G (in BIT). При установке сигнала G в единицу генератор воспроизводит серию N прямоугольных импульсов с периодом T (рис.54, г). Следующий запуск генератора может произойти не ранее окончания генерации текущей серии импульсов. Таким образом, при включении генератор ждет единичного значения управляющего сигнала, что обеспечивается анализом условия (not G'Stable() and G=1), которое означает, что сигнал G изменился и стал равным единице. Модель, выполненная в стиле поведенческого описания, представлена ниже

```
entity EX5 is
    port(G:in BIT;C:out BIT);
end EX5;
architecture AR_EX5 of EX5 is
    process
        variable I,N,T:INTEGER;
    begin
        wait on G;
        if (not G'Stable() and G=1) then
            for I in 1 to N
                C<=transport 1 after T/2;
                C<=transport 0 after T;
                wait for T;
            end loop;
        endif
    end process;
end AR_EX5
```

Пример 9. Представляет VHDL-модель управляемого генератора прямоугольных импульсов EX6 (рис.54, в). Он отличается большей универсальностью, т.к. полностью управляется из вне сигналом (G:in BIT) и генерирует произвольное число импульсов. Генератор начинает работать при установке сигнала G в единицу – условие (not G'Stable() and G=1), прекращает работу при сбросе сигнала G в ноль – условие (not G'Stable() and G=0). Чтобы заставить генератор по окончании одного импульса генерировать следующий, в модели создается "технологическая" обратная связь с выхода генератора на вход. Соответственно сброс выходного сигнала C в ноль – условие (not C'Stable and C=0 and G=1) используется для очередного запуска генератора и имитации следующего импульса. Модель, выполненная в стиле поведенческого описания, представлена ниже

```
entity EX6 is
   port(G:in BIT;C:out BIT);
end EX6;
architecture AR_EX6 of EX6 is
   process
      variable I,N,T:INTEGER;
   begin
      wait on G,C;
      if (not G'Stable() and G=1) then
         C<=transport 1 after T/2;
         C<=transport 0 after T;
      else if (not G'Stable() and G=0) then
         C<=transport 0
      else if (not C'Stable and C=0 and G=0) then
         C<=transport 1 after T/2;
         C<=transport 0 after T;
      endif
   end process;
end AR_EX6
```

3.7. Пример построения системы моделирования VHDL-описаний

Двумя полярными подходами являются - построение системы интерпретации либо компилирующей системы. Необходимая эффективность моделирующего кода при относительной простоте системы моделирования может быть достигнута: - отказом как от интерпретации спецификации проекта так и прямой генерации загрузочного кода и использованием метода промежуточной трансляции описания закона функционирования устройства в адекватный в функциональном отношении текст на некотором внутреннем языке с последующим получением загрузочного модуля; - выбором в качестве средства внутреннего представления языка высокого уровня с развитыми вычислительными средствами и эффективным транслятором, как, например Си, Паскаль, Ада. Функционально такая система должна содержать: - входной препроцессор для трансляции исходного описания; - генератор промежуточного текста на языке высокого уровня; - монитор моделирования, составляющий ядро системы моделирования. Состав системы моделирования VHDL-описаний представлен ниже.

Моделирование выполняется в два этапа. На первом ТРАНСЛЯТОР и КОНВЕРТОР выполняют лексический, семантический и синтаксический анализ проекта и формируют его внутреннее представление в виде информационных таблиц. На их основе ГЕНЕРАТОР создает адекватный проекту текст (например, на языке СИ) готовый для исполнения. Для этого произвольное VHDL-описание, представленное композицией параллельных операторов, переводится в описание в терминах операторов process (процессную модель) и генерируются модел процессов. К полученному СИ-тексту подключается пакет ДВОИЧНАЯ АРИФМЕТИКА и МОНИТОР управления моделированием на уровне описанных в блоке процессов. На втором этапе, выполняемом много-

кратно, модулями МОДЕЛИРОВАНИЕ и ТЕСТОВЫЕ НАБОРЫ производится имитационное событийное моделирование описания проекта. Последний модуль обеспечивает подготовку входных воздействий в модель, накопление и анализ реакций проекта. Технологическая схема моделирования приведена ниже.

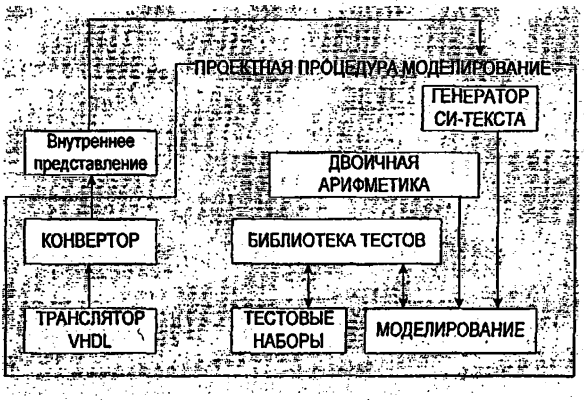


Рис. 55. Состав системы моделирования

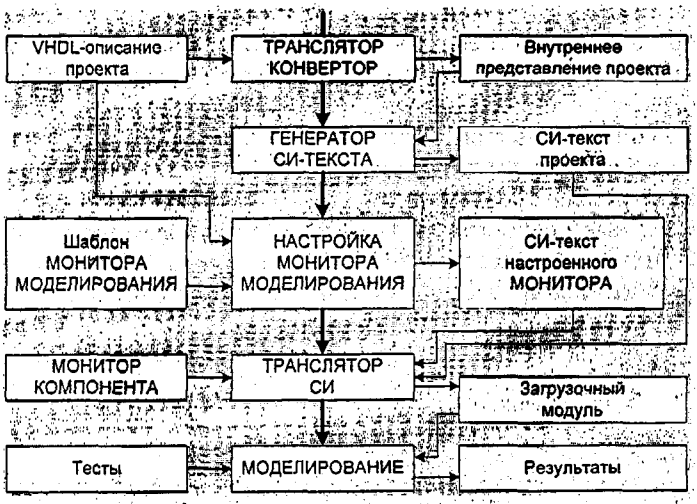


Рис. 56. Технологическая схема моделирования

Основу модуля МОДЕЛИРОВАНИЕ составляет процедура МОДЕЛИРОВАТЬ-ПРОЕКТ, включающая мониторы моделирования отдельных программных компонентов и монитор моделирования, определяющий порядок инициирования процессов.

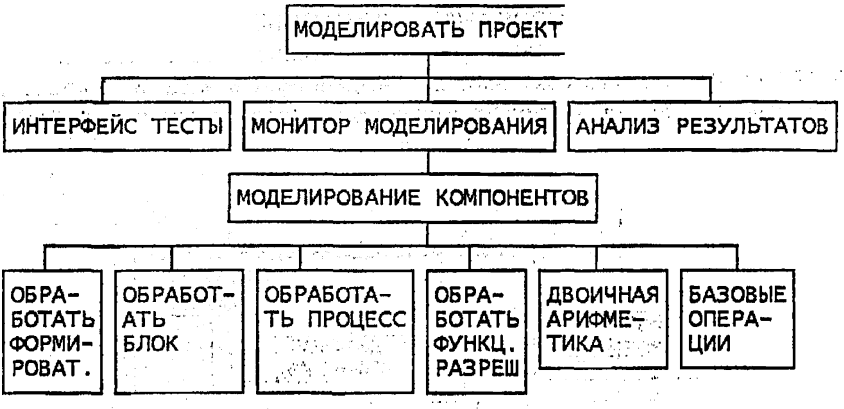


Рис. 57. Структура модуля МОДЕЛИРОВАНИЕ

Моделирование процессного описания основывается на анализе в каждом такте моделирования списков сигналов и заблокированных процессов. При этом фиксируются события, ведущие к срабатыванию процессов либо их переходу в активное состояние. Соответствующие процессы активизируются. В качестве событий рассматриваются: - событие-сигнал, когда произошло изменение входных или входных-выходных сигналов; - событие-разблокировка, когда процесс переходит из пассивного в активное состояние. Соответственно такт интерпретации описания проекта на поведенческом уровне включает шаги: 1) продвинуть модельное время к ближайшему изменению сигналов или статуса процессов; 2) определить список сигналов Si, на которых происходит событие, или список разблокируемых процессов (если событий нет, то перейти на п.1); 3) выполнить все разблокированные процессы и другие программные компоненты, чувствительные к сигналам Si; 4) вычислить все ведущие и разрешенные значения сигналов; 5) обновить список событий (удалить несостоявшиеся события и добавить новые, заблокированные процессы внести в соответствующий список) и, если список не пуст, то перейти на п.1; 6) конец.

Ниже приведен укрупненный алгоритм монитора моделирования:

```
Модельное время установить в 0
Инициализировать программу моделирования
Создать список событий (сигналов)
ЦИКЛ-ПОКА моделирование не завершилось (есть события в списке)
    Анализировать временной список и найти ближайшее событие
    Продвинуть модельное время
    Определить список сигналов, изменивших свои значения, или список
                активизирующихся процессов
    Определить список активных программных компонентов: формирователей,
                блоков с истинным значением охранного выражения;
                процессов, чувствительных к изменившимся сигналам;
                ждущих процессов
    ЦИКЛ-ПОКА не обработаны все активные компоненты
```

```
        Активизировать текущий активный компонент
        ЕСЛИ активизирован формирователь ТО
            ОБРАБОТАТЬ-ФОРМИРОВАТЕЛЬ
        ИНАЧЕ ЕСЛИ активизирован процесс ТО
            ОБРАБОТАТЬ-ПРОЦЕСС
        ИНАЧЕ ЕСЛИ активизирован блок ТО
            ОБРАБОТАТЬ-БЛОК
        ВСЕ-ЕСЛИ
                Вычислить ведущие и разрешенные значения сигналов
                Анализировать появления события на этих сигналах
                Обновить временную цепь
                Перейти к следующему активному компоненту
    ВСЕ-ЦИКЛ
ВСЕ-ЦИКЛ
Обработать результаты моделирования.
```

Основу мониторов моделирования компонентов составляет модуль

```
ОБРАБОТАТЬ-ФОРМИРОВАТЕЛЬ
    ЦИКЛ-ПОКА не обработаны все транзакции сигнала  (элементы форм)
        Обработать текущую транзакцию: вычислить значение и время сигнала
        При необходимости обновить соответствующий формирователь (удалить старые значения и занести новую транзакцию в формирователь)
        Перейти к следующей транзакции
    ВСЕ-ЦИКЛ.
```

4. Регистровое и вентильное (логическое) моделирование

4.1. Особенности моделирования на регистровом уровне

Как правило, на регистровом уровне используют данные битового типа, битовые векторы, а также данные в многозначных алфавитах, что позволяет более точно отображать и моделировать переходные процессы. Основные цели моделирования: проверка корректности описания закона функционирования устройства (входной контроль), оценка точностных, временных и других характеристик ЦУ, генерация тестов.

Основной программный компонент языковых средств этого уровня - блок, соответствующий функционально-законченному описанию устройства или его части. Он представляется интерфейсом и множеством параллельно-последовательных процессов, имитирующих преобразование информации в цифровом устройстве. Процессы представляют комбинацию операторов (присваивания, инициирования процессов, вызова блоков, ожидания, условных операторов), выполняемых параллельно-последовательно (что определяется операцией следования), и описывают действия, выполняемые над переменными. Ниже описаны типовые конструкции языка в нотации Бэкуса-Наура

<Описание_блока>::=

```
<Заголовок_блока><Декларации_блока><Процессы_блока>
<Заголовок_блока>::=ИДЕНТИФИКАТОР_блока (<Интерфейс_блока>)
<Интерфейс_блока>::=<Список_формальных_параметров>
<Список_формальных_параметров>::=
    <Тип_параметра>ИДЕНТИФИКАТОР_параметра{,<Тип_параметра>
    ИДЕНТИФИКАТОР_параметра}
<Тип_параметра>::=ВХОДНОЙ!ВЫХОДНОЙ!ВХОДНОЙ-ВЫХОДНОЙ
<Декларации_блока>::={<Строка_описания>}
<Строка_описания>::=
    <Описание_массива>!<Описание_константы>! <Описание_переменной>
<Описание_массива>::=
    {ИДЕНТИФИКАТОР_массива [ЦЕЛОЕ_индекс_1: ЦЕЛОЕ_индекс_2]}
<Процессы_блока>::={<Описание_процесса>}
<Описание_процесса>::=
    ИДЕНТИФИКАТОР_процесса <Операторная_часть_процесса>
<Операторная_часть_процесса>::=
    <Оператор> !<Оператор> {<Операция_следования> <Оператор>}
<Операция_следования>::=ПОСЛЕДОВАТЕЛЬНО!ПАРАЛЛЕЛЬНО
<Оператор>::=<Оператор_присваивания>!<Условный_оператор>!
    <Оператор_ожидания>!<Оператор_инициирования_процесса>!
    <Оператор_вызова_блока>
<Оператор_присваивания>::=ИДЕНТИФИКАТОР:=<Выражение>
<Выражение> ::= <Терм> {ОПЕРАЦИЯ <Терм>}
<Оператор_ожидания> ::= ЖДАТЬ(<Параметры_ожидания>)
<Параметры_ожидания>::=
    ЦЕЛОЕ_число_тактов !<Список_процессов>!
    <Условие_ожидания>
<Список_процессов>::=
    ИДЕНТИФИКАТОР_процесса[{,ИДЕНТИФИКАТОР_процесса}]
<Условие_ожидания> ::= Булево_выражение
<Оператор_инициирования_процесса> ::= ИДЕНТИФИКАТОР_процесса.
```

Примеры описания моделей рассмотрены ниже на базе языка иерархического функционально-логического моделирования СБИС MG3 (разработка НПО Интеграл, г.Минск). Основные типы данных – битовый, одномерные и двумерные битовые векторы, отображающие регистры. Объект проекта либо его часть описывается программным компонентом block. Блоки могут быть вложенными. Исполнительная часть блоков описывается множеством процессов. Каждый процесс есть множество тактируемых операторов, выполняемых последовательно (операция →) такт за тактом или параллельно (операция ") в одном такте. Основные операторы: - присваивание (например, y:=b или y[0:3]:=b[10:13]); - вызов i-го процесса Fi ; - условный оператор <условие> (прямая ветвь) <> (альтернативная ветвь) (например, <y[5]=1> (...) < > (...)); - оператор ожидания в течение заданного числа тактов wait (<число_тактов>), ожидания завершения заданного списка процессов wait (<список_процессов>) и наступления заданного условия wait (<условие>).

Пример 1. Моделирование блока с входными сигналами X, Y и выходным сигналом Z (рис.58, а), включающего четыре процесса, протекающих параллельно. В первом такте инициируется процесс F0, запускающий параллель-

но процессы F1-F3. Во втором такте в первом процессе параллельно изменяются разряды 0:3 и 4:6 сигнала Y, во втором процессе анализируется значение нулевого разряда входного сигнала X, а третий процесс переходит в пассивное состояние в течение трех последующих тактов и т.д.

```
block (X,Y, .Z)
a[0:10];
F0: F1 " F2 " F3;
F1: Y[0:3]:=X[7:10] " Y[4:6]:=X[4:6] → wait (5);
F2: <X[0]=1> (X[0]:=0 → X[1]=0) < > (X[1]=1);
F3: wait (3) → Z:=X+Y;
```

Пример 2. Моделирование блока (рис.58, б), включающего четыре процесса. Корневой процесс F0 запускает параллельно процессы F1 и F2, а после их завершения инициируется процесс F3

```
block (...)
F0: a:=0 "b:=0 → F1" F2 → wait (F1,F2) → F3;
F1: ...;
F2: ...;
F3: ...;
...
```

Пример 3. Моделирование блока, включающего 5 процессов, с характером соподчинения, представленным на рис.58, в

```
...
block (...)
F0: F1 " F4" F5;
F1: ... F2 " F3...;
F2: ...;
F3: ...;
F4: wait (<M=1>) ...;
F5: ... → wait (F3,F4) → ...;
```

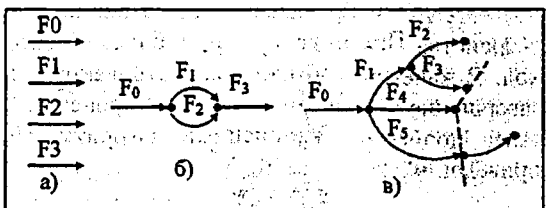


Рис.58. Варианты развития процессов: а) параллельный; б-в) последовательно-параллельный

Пример 4. Здесь представлены модели ЦУ, реализующего функцию синуса на отрезке 0-3,14 через разложение в ряд Маклорена (SIN1 и SIN3) и через произведение (SIN2). У аргумента синуса (переменной X[0-14]) нулевой разряд - знаковый, разряды 1-2 соответствуют целой части аргумента. У значений синуса (переменной Y) нулевой разряд знаковый, разряды 1-4 описывают целую часть результата и используются в промежуточных вычислениях.

```
SIN1(X,.Y)
 (I, Fac1, Fac2, Fac) [0:30], (Y, X2, DeltaY, DeltaX) [0:28], X[0:14]
F1: DeltaY:=0'B → X2:=X*X " I:=01'B " DeltaY[0:16):=X → Y:=DeltaY→F2;
F2: <Del taY ^= 0'B> (F3 → F4 → F5 → F2);
F3: Fac1 :=2*I → Fac2:=Fac1+1 → Fac:=Fac1*Fac2;
F4: DeltaX:=X2//Fac;
F5: DeltaY:= DeltaX[0:16]*DeltaY[0:16] " I:=I+01'B →
```

```
DeltaY:=^DeltaY → DeltaY:=DeltaY+01'B → Y:=Y+DeltcaY;

SIN2(X, .S)
X[0:14], S[0:30], (P, R)[0:28], K[0:28]
F1: S:=0 → S[1:4]:=X[1:2] " S[5:16]:=X(3:14] →
    P:=(X*X)//0100111011110101'B"K:=1 →
    R:=010000000000000' B-P → F2;
F2: S:=S[1:17]*R → K:=K+1 → R:=010000000000000'B-P///(K*K) →
      <R ^= 010000000000000'B>(F2);

SIN3(X, .Y, .OK)
(I, Fac1, Fac2)[0:7], Fac[0:15], (X2, DeltaX, DeltaY) [0:28], (Y, X)[0:14], OK[0]
F1: X2:=X*X " I:=01'B " DeltaY[0:16]:=X " DeltaY[17:28]:=0'B →
    Y:=DeltaY[0:14] → F2;
F2: <DeltaY ^= 0'B> (F3 → F4 → F5 → F2)<>(OK:=01'B);
F3: Fac1 :=2*I → Fac2:=Fac1+1 → Fac:=Fac1*Fac2;
F4: DeltaX:=X2//Fac;
F5: DeltaY:=DeltaX[0:16]*DeltaÓ[0:16] " I:=I+01'B →
    DeltaY:=^DeltaY → DeltaY:= DeltaY+01'B → Y:=Y+DeltaY[0:14];
```

4.2. Пример построения системы моделирования MG3-описаний

Система организована аналогично системе, описанной в Главе 3, § 3.7. Моделирование производится с использованием функциональных тестов и сводится к порождению порядка инициирования блоков и процессов в блоках в соответствии с признаками их активности и алгоритмом функционирования устройства. При моделировании блока (блоков) используется событийный подход. В качестве событий рассматриваются: инициирование нового процесса; завершение обработки активного процесса; завершение обработки текущего такта процесса. Укрупненный алгоритм МОНИТОРА МОДЕЛИРОВАНИЯ приведен ниже

```
Идентифицировать Параметры_блока
ЦИКЛ-ПОКА не исчерпан тестовый набор
    Выделить тестовый вектор
    Задать значения параметров блока
    МОДЕЛИРОВАТЬ-БЛОК(Имя_блока) на тестовом векторе
    Оценить точность моделирования
    Вывести результаты моделирования на тестовом векторе
ВСЕ-ЦИКЛ
Анализировать результаты моделирования
Вывести результаты моделирования на тестовом наборе.
```

Активные процессы выполняются такт за тактом до полного завершения модулем МОДЕЛИРОВАТЬ-БЛОК

```
Активизировать корневой процесс
Поместить процесс в очередь
ЦИКЛ-ПОКА есть активные процессы (моделировать текущий такт блока)
    ЧИТАТЬ-ТРАССУ
```

```
ЦИКЛ-ПОКА в текущем такте есть активные процессы
          или не обработанные вызовы новых процессов
   Начать просмотр очереди с конца
   ЦИКЛ-ПОКА не просмотрена очередь
      Анализировать текущий процесс
      ЕСЛИ процесс активен в текущем такте ТО
         ОБРАБОТАТЬ-ПРОЦЕСС
      ВСЕ-ЕСЛИ
      Перейти к следующему процессу в очереди
      ВСЕ-ЦИКЛ
   ВСЕ-ЦИКЛ
   Востановить признаки активности незавершенных процессов
ВСЕ-ЦИКЛ ,
```

где ОБРАБОТАТЬ-ПРОЦЕСС

```
Сбросить признак активности процесса
Выполнить операторы текущего такта процесса
Анализировать признаки завершения такта
ЕСЛИ процесс завершен ТО
   Удалить процесс из очереди
ИНАЧЕ ЕСЛИ рекурсивный вызов процесса ТО
   Активизировать текущий процесс
ИНАЧЕ ЕСЛИ нерекурсивный вызов процесса ТО
   Активизировать новый процесс
   Поместить его в очередь активных процессов
ВСЕ-ЕСЛИ.
```

4.3. Особенности моделирования вентильного (логического) уровня

Здесь для представления значений сигналов используются многозначные алфавиты, причем в системах моделирования реальных ЦУ применяются десятки градаций сигналов, учитывающих помехи, переходные процессы, длительности фронтов и т.п. Простейшие алфавиты показаны на рис.59. Это булевый или логический, двоичный алфавит, который оперирует двумя значениями сигнала {0,1}. Троичный алфавит Эйхельберга $\{0,1,X\}$, позволяющий учитывать неопределенное значение сигнала X в переходных периодах от 0 к 1 и от 1 к 0. В пятизначном алфавите $\{0,1,E,\bar{E},X\}$ различают стабильные значения 0 и 1, значения E и $\bar{E}$, соответствующие переходу сигнала от 0 к 1 и от 1 к 0. Значение помехи X здесь соответствует последовательности изменений сигнала $\{E,0\}$ или $\{\bar{E},1\}$. Примеры таблиц истинности для функции X1&X2 в троичном и в пятеричном алфавите приведены на рис.60. Из существующих программных систем следует отметить PCAD, MicroLogic, LogicPro и др.

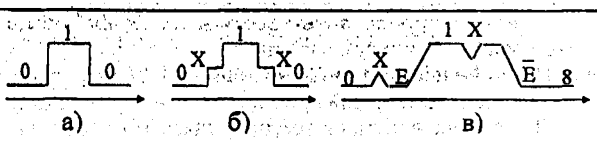


Рис.59. Сигналы: а) двоичного алфавита; б) троичного алфавита; в) пятеричного алфавита

Основные цели моделирования на вентильном уровне: анализ логики работы логической схемы; оценка ее выходных реакций; выявление и исследование состязаний сигналов, начальных неустановок, пиков, возможностей самогенерации схем и т.д.; - анализ помехозащищенности, нагрузочной способности, временных характеристик схемы и т.д. Причем задачи 1-2 решаются на базе булевых алфавитов и синхронного моделирования, а другие задачи требуют моделей с многозначными алфавитами, учитывающими реальные задержки и возможность описания переходных процессов.

X_1 \ X_2	0	1	X
0	0	0	0
1	0	1	X
X	0	X	X

а)

X_1 \ X_2	0	1	E	$\bar{E}$	X
0	0	0	0	0	0
1	0	1	E	$\bar{E}$	X
E	0	E	E	X	X
$\bar{E}$	0	$\bar{E}$	X	$\bar{E}$	X
X	0	X	X	X	X

б)

Рис.60. Таблицы истинности функции $x_1 \& x_2$: а) в троичном; б) в пятеричном алфавите

Закон функционирования объекта – схемы задается таблицами истинности, системами логических уравнений. Адекватность модели на этом уровне определяется: точностью описания сигналов - принятым алфавитом; точностью описания логических элементов, примитивов - их функций, задержек и т.п.; способом отображения характера взаимодействия элементов, учета временных соотношений между сигналами.

Классификация методов и соответственно моделей логического моделирования представлена на рис.61. Первый уровень классификации определяется используемым алфавитом для отображения сигнала. Далее в зависимости от способа учета задержек модели делятся на синхронные (с жестким тактированием), где задержки на всех элементах считаются одинаковыми и асинхронные, где учитываются реальные задержки элементов. В зависимости от способа решения булевы модели разделяются на сквозные (решаются на базе методов простых итераций и метода Зейделя и их модификаций) и событийные.

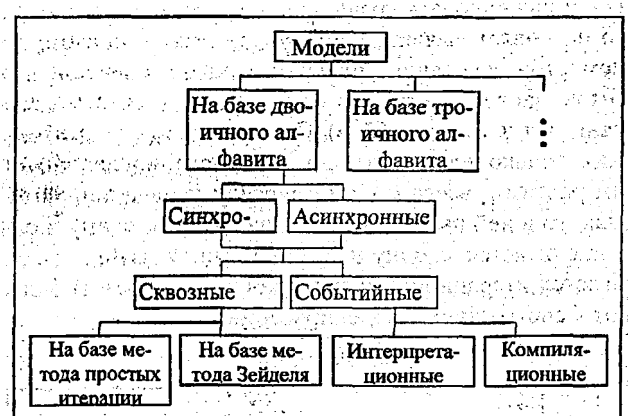
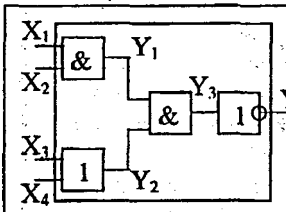


Рис.61. Классификация методов логического моделирования

Сквозное итерационное моделирование и моделирование методом Зейделя иллюстрируются рис.62. При сквозном моделировании для каждого логического элемента ЦУ (рис.62, а) в произвольном порядке выписывается его логическая функция. Их совокупность составляет систему логических уравнений (рис.62, б). Определяется ранг ЦУ *n*, как число элементов наиболее длинного среди всех путей, ведущих от входа схемы ЦУ к ее выходу. Система логических уравнений рассчитывается *n+1* раз, причем лишний просчет делается, чтобы убедиться, что полученные выходные значения более не меняются и в схеме нет самопроизвольной генерации. Метод прост в реализации, но трудоемок для сложных ЦУ с большим числом элементов и малоэффективен (большие накладные расходы) при срабатывании в каждом такте незначительного числа элементов. Так при *k* элементах и *t* тактах моделирования потребуется решить *k x t x (n+1)* логических уравнений.



а)

$$\begin{cases} y_3 = \overline{y_1} \cdot y_2 \\ y_4 = \overline{y_3} \\ y_1 = x_1 \cdot x_2 \\ y_2 = x_3 + x_4 \end{cases}$$

б)

$$\begin{cases} y_1 = \overline{x_1 \cdot x_2} \\ y_2 = x_3 + x_4 \end{cases} \text{ 1 ранг}$$

$$\begin{cases} y_3 = \overline{y_1} \cdot y_2 \end{cases} \text{ 2 ранг}$$

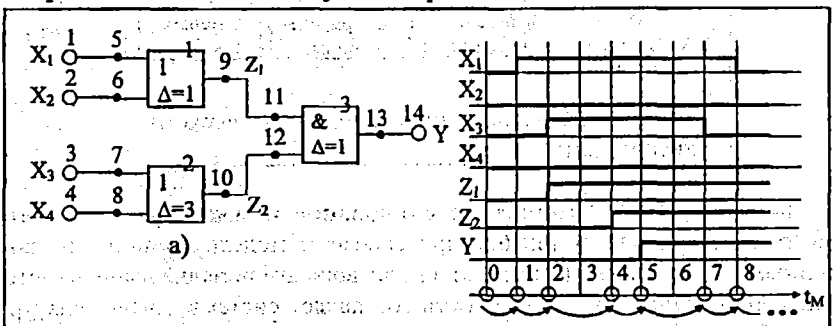
$$\begin{cases} y_4 = \overline{y_3} \end{cases} \text{ 3 ранг}$$

в)

Рис.62. Сквозное моделирование: а) моделируемая схема; б) система уравнений метода простых итераций; в) система уравнений для метода Зейделя

Лишние вычисления можно исключить, если предварительно про-

зировать схему ЦУ и ранжировать уравнения в системе, как это делается в методе Зейделя. Это позволяет выявить ранги уравнений. К первому рангу относят уравнения элементов, зависящие только от входных сигналов, второй ранг - уравнения элементов, зависящие только от входных сигналов и выходов элементов первого ранга и т.д. (см. рис.62, в). Соответственно потребуется просчет каждого уравнения только единожды, но в строгой последовательности от нижних к высшим рангам – всего $k \times t$ логических уравнений. Если схема последовательностная, то в ней выделяют комбинационную схему и память. Комбинационная схема считается за одну итерацию вышеуказанными методами. А память просчитывается итерационно нужное количество раз. В более сложных случаях прибегают к событийному моделированию.



t_M	Сигнал	Полюс	Новое значение сигнала	Время изменения сигнала	Новое событие
0	X_1	1	1	1	V
	X_3	3	1	2	V
	X_3	3	0	7	V
	X_1	1	0	8	V
1	X_3	3	1	2	
	Z_1	9	1	2	V
	X_3	3	0	7	
	X_1	1	0	8	
2	Z_2	10	1	4	V
	X_3	3	0	7	
	X_1	1	0	8	
4			•••		

в)

Рис.63. Моделирование на логическом уровне: а) моделируемая схема; б) диаграмма изменения сигналов; в) последовательность изменения списков событий

Особенности событийного имитационного моделирования на вентильном уровне. Событийное моделирование наиболее универсально и его можно с успехом применять в произвольных алфавитах, когда нужно учесть

индивидуальные задержки и другие особенности элементов. К тому же, как показывает статистика, на каждом такте работы БИС или СБИС срабатывает всего лишь до 1-1,5 % элементов схемы, а остальные элементы свои состояния не меняют и соответственно не должны обрабатываться в ходе моделирования, как это происходит при сквозном подходе. Пример событийной модели приведен на рис.63. Здесь моделируемая схема (рис.63, а) представлена элементами, входными и выходными контактами. Элементы задаются параметрами: индивидуальными номерами, функциями, задержками, номерами входных и выходных контактов. Порядок соединения контактов задается матрицей связей. В модели можно выделить входные сигналы X1-X4, промежуточные (внутренние) сигналы Z1-Z2 и выходной сигнал Y.

В качестве события здесь может рассматриваться изменение значения сигнала на каком-либо из контактов (выходном контакте элемента или входном контакте схемы). Значения сигналов образуют списки событий и продвижение модельного времени в каждом такте к ближайшему событию выполняется путем их анализа.

На каждом такте: - выявляют события (изменившиеся сигналы); - по матрице связей и описанию элементов (их входных и выходных контактов) определяются активные элементы; - имитируют работу - логические функции всех активных элементов и определяют будущие значения сигналов на их выходах; - прогнозируемые значения заносят в списки событий соответствующих сигналов; продвигают модельное время и т.д.

В примере на рис.63 в начальном состоянии полагается: - все входные сигналы равны нулю и в соответствии с логикой работы схемы равны нулю и все производные сигналы, включая выходной; - значения входных сигналов в ближайшие 8 тактов модельного времени заданы (см. диаграммы сигналов X1-X4 на рис.63, б).

Последовательность изменения списков будущих событий показана на рис.63, в. Здесь для каждого момента модельного времени списки упорядочены в соответствии с временем их наступления. В исходный момент модельного времени *tm =0* в списках есть только четыре будущих события (два для сигнала X1 и два для X2). Самое первое событие произойдет в момент модельного времени tM =1 – это изменение сигнала X1. При этом сработает элемент №1, в соответствии с его параметрами будет спрогнозировано изменение сигнала Z1 с нуля на единицу через единичную задержку. Соответственно в списках будущих событий будет убрано текущее и добавлено событие для сигнала Z1. В момент модельного времени *tm* =2 произойдет два события: - изменение сигнала X3 и сигнала Z1. При этом сработают элементы №2 и 3. В соответствии с их параметрами будет спрогнозировано изменение сигнала Z2 с нуля на единицу через три единицы задержку. А изменение сигнала Z1 состояние элемента №3 и соответственно его выходной сигнал Y не изменит. Поэтому из списков будущих событий будут исключены текущие события и добавлено новое событие для сигнала Z2.

ЛИТЕРАТУРА

1. Советов Б.Я., Яковлев С.А. Моделирование систем. - М.: Высшая школа, 2001. – 343 с.

2. В.В.Апаносович, Тихоненко О.М. Цифровое моделирование стохастических систем.– Минск: изд. "Университетское", 1986.

3. Шрайбер Т. Моделирование на GPSS. - М.: Машиностроение, 1980.

4. Советов Б.Я., Яковлев С.А. Моделирование систем: курсовое проектирование. - М.: Высшая школа, 1988.

5. Основы теории вычислительных систем. Учебное пособие /Под ред. Майорова С.А. - М.: Высшая школа, 1978. - 408 с.

6. Альянах М.Н. Моделирование вычислительных систем. – М.: Машиностроение, 1988. – 225 с.

7. Максимей И.В. Имитационное моделирование на ЭВМ. - М.: Радио и связь, 1988.

8. Авен О.И., Гурин Н.Н., Коган Я.А. Оценка качества и оптимизация вычислительных систем. - М.: Наука, 1982.

9. Бусленко Н.П. Моделирование сложных систем. М.: Наука, 1978.

10. Клейнен Д. Статистические методы в имитационном моделировании. Вып. 1 и 2. М.: Статистика, 1978.

11. Шеннон Р. Имитационное моделирование систем - искусство и наука. М.: Мир, 1978.

12. Голованов О.В., Дуванов С.Г., Смирнов В.Н. Моделирование сложных дискретных систем на ЭВМ третьего поколения (опыт применения GPSS). М.: Энергия, 1978.

13. Советов Б.Я. Моделирование систем - лабораторный практикум. - М.: Высшая школа, 1989.

14. Киндлер Е. Языки моделирования. М.: Энергоатомиздат, 1985.

15. Феррари Д. Оценка производительности вычислительных систем. - М.: Мир, 1981. - 576 с.

16. Жожикашвили В.А., Вишневский В.М. Сети массового обслуживания. Теория и применения к сетям ЭВМ . - М.: Радио и связь, 1988. - 192 с.

17. Клейнрок Л. Вычислительные системы с очередями/ - М.: Мир, 1979. - 600 с.

18. Клейнрок Л. Теория массового обслуживания. М.: Машиностроение, 1979.

19. Каган Б.М. Электронные вычислительные машины и системы. Учебное пособие - М.: Энергоатомиздат, 1985. - 552 с.

20. Соболь И.М. Метод Монте-Карло. М.: Наука, 1985.

21. Вероятностные методы в вычислительной технике: Учебн.пособие для вузов /Под ред. А.М.Лебедева и Е.А.Чернявского. - М.: Высшая школа, 1986.

22. Курицкий Б.Я. Поиск оптимальных решений средствами Excel 7.0. - СПб.: BHV- Санкт-Петербург, 1997.

23. Джоунз Г. Программирование на языке Оккам. - М.: Мир, 1989.

24. Бар Р. Язык Ада в проектировании систем. - М.: Мир, 1988.

25. Джонсон Н., Лион Ф. Статистика и планирование эксперимента в технике и науке. Методы обработки данных. - М.: Финансы и статистика, 1980.

26. Снапелов Ю.М. и Старосельский В.А. Моделирование и управление в сложных системах. М: Сов. радио, 1974.

27. Иыуду К.А. Надежность, контроль и диагностика вычислительных машин и систем. - М.: Высш. шк., 1989.

28. Питерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 1984.

29. Рыжиков Ю.И. Решение научно-технических задач на персональном компьютере. – СПб.: КОРОНА принт, 2000. – 272 с.

30. Бибило П.Н. Основы языка VHDL. – М.: Солон-Р, 2000. – 210 с.

31. VHDL для моделирования, синтеза и формальной верификации аппаратуры. М.: Радио и связь, 1995. – 256 с.

32. Армстронг Дж. Р. Моделирование цифровых систем на языке VHDL. М.: Мир, 1992. – 175 с.

33. Гультяев А.К. MATLAB 5.2. Имитационное моделирование в среде Windows: Практическое пособие. – СПб.: КОРОНА принт, 1999. – 288 с.

34. Основы современных компьютерных технологий: Учебное пособие.Под ред. Хомоненко А.Д. – СПб.: КОРОНА принт, 1998. – 448 с.

СОДЕРЖАНИЕ

Учебное издание

Муравьев Геннадий Леонидович

МОДЕЛИРОВАНИЕ СИСТЕМ

**Курс лекций по дисциплине «Моделирование систем»
для студентов специальностей
«Автоматизированные системы обработки информации»
и «Вычислительные машины, системы и сети»**

Ответственный за выпуск: *Г.Л. Муравьёв*
Редактор *Т.В. Строкач*
Технический редактор *А.Д. Никитчик*
**Компьютерный набор и
вёрстка:** *Л.П. Махнист*

9 789856 584643

Издательская лицензия ЛВ № 382 от 1.09.2000 г.
Подписано в печать 18.06.2003. Формат 60x84 1/16.
Бумага писч. Усл. п. л. 9,5. Уч. изд. л. 10,25. Тираж 100 экз.
Заказ № 797
Отпечатано на ризографе УО «Брестский государственный технический университет».
224017, Брест, ул. Московская, 267.
Полиграфическая лицензия ЛП № 178 от 14.01.2003 г.