МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА

Кафедра «ПРИКЛАДНАЯ МАТЕМАТИКА»

И. В. МАКСИМЕЙ, В. Д. ЛЕВЧУК,
С. П. ЖОГАЛЬ, В. Н. ПОДОБЕДОВ

ЗАДАЧИ И МОДЕЛИ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Часть 3

Технология имитации на ЭВМ и принятие решений

Гомель 1999

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА

Кафедра «ПРИКЛАДНАЯ МАТЕМАТИКА»

И. В. МАКСИМЕЙ, В. Д. ЛЕВЧУК,
С. П. ЖОГАЛЬ, В. Н. ПОДОБЕДОВ

ЗАДАЧИ И МОДЕЛИ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

Часть 3

Технология имитации на ЭВМ и принятие решений

Допущено Министерством образования Республики Беларусь
в качестве учебного пособия
для студентов инженерно-технических специальностей
высших учебных заведений

Под общей редакцией профессора **И. В. Максимея**

Гомель 1999

УДК 519.85 (075.8)
З-153

З-153 **Задачи и модели исследования операций.** Ч. 3. Технология имитации на ЭВМ и принятие решений: Уч. пособие / И. В. Максимей, В. Д. Левчук, С. П. Жогаль, В. Н. Подобедов – Гомель: БелГУТ, 1999. – 150 с.
ISBN 985-6550-10-6

Содержатся технологии организации имитационного эксперимента (ИЭ) на ЭВМ и элементы теории принятия решений. Описываются основные операции и способы автоматизации ИЭ на ЭВМ. Для ускорения моделирования трудоёмких этапов ИЭ предлагается описание системы моделирования MICIC. Для эффективного использования результатов ИЭ приведены типовые методики принятия решений. Приведён ряд типовых примеров построения моделей и методик применения средств автоматизации ИЭ.

Предназначено для студентов и аспирантов инженерно-технических специальностей высших учебных заведений.

Рецензенты: **кафедра «Прикладная математика»** Национальной политехнической академии Украины «Киевский политехнический институт»;
докт. техн. наук профессор Белорусского государственного университета **О. М. Тихоненко**;
проректор БелГУТа докт. техн. наук профессор **В. Я. Негрей**.

ISBN 985-6550-10-6

Учебное издание

Иван Васильевич Максимей
Виктор Дмитриевич Левчук
Сергей Петрович Жогаль
Владимир Николаевич Подобедов

ЗАДАЧИ И МОДЕЛИ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ
Часть 3. Технология имитации на ЭВМ и принятие решений

Учебное пособие

Редактор Н. А. Дашкевич
Технический редактор Т. А. Пугач
Корректор И. И. Эвентов
Компьютерный набор и вёрстка кафедры «Прикладная математика»

Подписано в печать 04.09.99.
Формат бумаги 60 × 84 $^{1}/_{16}$. Бумага писчая № 1. Гарнитура Таймс.
Печать офсетная. Усл. печ. л. 8,84. Уч.-изд. л. 9,68. Тираж 200 экз.
Зак. № 1861. Изд. № 3084

Редакционно-издательский отдел БелГУТа,
246653, г. Гомель, ул. Кирова, 34. Лицензия ЛВ № 57 от 22.10.97.

Ризограф типографии БелГУТа,
246022, г. Гомель, ул. Кирова, 34. Лицензия ЛП № 75 от 12.03.93.

ВВЕДЕНИЕ

В данной части учебного пособия «Задачи и модели исследования операций» изложена технология организации имитационного эксперимента на ЭВМ и элементы теории принятия решений. Последовательно излагаются предпосылки и необходимость построения имитационных моделей (ИМ). Обсуждаются достоинства и недостатки имитационного метода исследования на ЭВМ сложных технических систем (СТС). Вводится оригинальная трактовка квазипараллелизма в ИМ на ЭВМ. Даётся понятие о модельном времени и о наиболее употребительных способах организации квазипараллелизма при имитации процессов в СТС. Даётся перечень основных операций на всех этапах имитационного эксперимента (ИЭ) на ЭВМ. Рассмотрены возможные способы автоматизации трудоёмких этапов ИЭ.

После того как ИМ создана, её необходимо испытать и исследовать свойства модели. Предложены простейшие типовые процедуры верификации алгоритма ИМ. Приведены типовые методики оценки точности, чувствительности и устойчивости ИМ. Изложен простейший алгоритм проверки адекватности ИМ реальному объекту. Поскольку авторами выпущено пособие по основам регрессионного анализа данных и планированию экспериментов на ЭВМ, традиционные вопросы постановки экспериментов с моделями СТС не рассматриваются. Вместо этого авторы сосредоточили основное внимание на таких технологических аспектах ИЭ, как определение требуемого размера выборки при планировании ИЭ; определение интервалов изменения параметров модели и исключение ошибок моделирования; особенности эксплуатации вероятностных моделей СТС.

Для ускорения трудоёмких этапов разработки ИМ для ПЭВМ предлагается использовать систему моделирования (СМ) MICIC. Описаны: состав и структура СМ; операторы языка моделирования; технология диалогового взаимодействия исследователя с СМ MICIC. Изложены типовые возможности и средства интерфейса пользователя с СМ MICIC для представления результатов имитации. В ходе серии ИЭ исследователь получает большой объём информации, которая зачастую противоречива и разнопланова. Чтобы оперативно и эффективно использовать результаты ИЭ, предлагается ряд методик принятия решений. Эти методики пригодны и при аналитическом моделировании СТС с помощью моделей задач ИСО, изложенных в первых

двух частях данного пособия. Поэтому овладение студентами методиками анализа ситуаций выбора решений и знакомство с типовыми алгоритмами векторной оптимизации является основным условием успешного овладения материала курса «Исследование операций».

Завершается пособие рассмотрением типовых примеров построения и использования предложенных средств автоматизации ИЭ и методик их применения. Вначале студентам предлагается пример реализации управляющей программы имитации на языке С. Затем иллюстрируется технология разработки ИМ СТС на примере моделирования процессов обслуживания пассажиров в метрополитене, состоящем из двух линий.

После завершения программирования ИМ метрополитена предлагается ряд заданий для испытания и исследования свойств ИМ. Затем студентам предлагается, используя классические методики, построить типовые планы эксперимента на ЭВМ для нахождения регрессионной зависимости между откликом и параметрами модели. Вторым важным аспектом использования ИМ метрополитена является пример принятия решений по результатам ИЭ. Предложен ряд заданий, которые можно использовать при выполнении лабораторных работ курса ИСО из раздела «Задачи и методы принятия решений».

1. ИМИТАЦИОННЫЕ МОДЕЛИ СЛОЖНЫХТЕХНИЧЕСКИХ СИСТЕМ

1.1. Когда и в каких случаях переходят к имитации процессов в СТС

Реальные сложные технические **системы** (СТС) можно исследовать с помощью двух типов моделей: аналитических и имитационных. В предыдущих частях данного пособия были рассмотрены наиболее типовые классы аналитических моделей СТС, когда поведение СТС записывается в виде некоторых функциональных соотношений или логических условий. Приведённые ранее классы детерминированных и вероятностных аналитических моделей позволяют оценить возможности аналитического подхода к исследованию СТС. Рассмотренный нами математический аппарат позволяет оперативным образом исследовать процессы в СТС.

Когда явления в СТС настолько сложны и многообразны, что аналитические модели становятся слишком грубым приближением к действительности, то исследователь вынужден использовать имитацию этих явлений. В любой имитационной модели (ИМ) поведение компонентов СТС описывается набором алгоритмов, которые затем реализуют ситуации, возникающие в реальной СТС. Моделирующие алгоритмы позволяют по исходным данным, содержащим сведения о начальном состоянии СТС, и фактическим значениям параметров СТС отобразить реальные процессы и получить сведения о возможном поведении СТС для данной конкретной ситуации. На основании этой информации можно принять соответствующие решения. Как известно, предсказательные возможности ИМ значительно меньшие, чем аналитических моделей. На основании опыта имитационного моделирования авторы рекомендуют использовать ИМ при решении задач исследования СТС в следующих случаях.

1. Если не существует законченной постановки задачи исследования и идёт процесс познания СТС, ИМ здесь служит средством изучения явлений в СТС.

2. Если аналитические методы решения подобных задач имеются, но математические процедуры столь сложны и трудоёмки, то ИМ даёт более простой и экономичный способ решения задачи исследования СТС.

3. Когда кроме оценки влияния параметров СТС желательно осуществлять наблюдение за поведением компонентов СТС в течение определённого периода времени.

4. Когда имитационное моделирование оказывается единственным способом исследования СТС из-за невозможности наблюдения явлений в реальной СТС.

5. Когда необходимо контролировать протекание процессов в СТС путём замедления или ускорения явлений в ходе имитации их на ЭВМ.

6. При подготовке специалистов и освоении новой техники, когда на ИМ обеспечивается возможность приобретения новых навыков в эксплуатации этой техники.

7. Когда изучаются новые ситуации в СТС, о которых мало что известно или неизвестно ничего. В этом случае имитация служит для предварительной проверки новых стратегий и правил принятия решений перед проведением экспериментов на реальной СТС.

8. Если особое значение имеет последовательность событий в проектируемой СТС и модель используется для предсказания узких мест в функционировании СТС и других трудностей, появляющихся в поведении СТС при введении в неё новых компонентов.

Однако ИМ наряду с характерными для них достоинствами имеют ряд существенных недостатков. Разработка хорошей ИМ часто обходится дороже создания аналитической модели и требует больших затрат времени. Иногда может показаться, что ИМ точно отражает реальное положение дел в СТС, а в действительности это не так. ИМ в принципе может быть неточна, и невозможно измерить степень этой неточности. Из достоинств имитации можно выделить: возможность описания поведения компонентов СТС на высоком уровне детализации; отсутствие ограничений на вид зависимостей между параметрами ИМ и состоянием внешней среды СТС; возможность комплексных исследований динамики взаимодействия компонентов СТС во времени и пространстве параметров СТС. Указанные достоинства обеспечивают имитационному методу широкую перспективу распространения.

1.2. Понятие о модельном времени и принципы организации квазипараллелизма операций на ЭВМ

Допустим, что СТС состоит из n компонентов ($K_i,\ i=\overline{1,n}$), функционирование которых представляет собой последовательность функциональных действий ($ФД_{ij},\ j=\overline{1,J}$). Будем говорить, что в результате выполнения $ФД_{ij}$ в СТС происходит событие C_{ij}. Каждое из событий в реальной СТС, как правило, связано с соответствующим компонентом K_i. При этом любое $ФД_{ij}$ выполняется на некотором временном интервале τ_{ij}. Для каждой K_i введём понятие локальной временной координаты t_i. В любой СТС все t_i изменяются одновременно. Однако характер этих изменений различен и определяется последовательностью временных интервалов $\{\tau_{ij}\}$.

При построении ИМ СТС все $ФД_{ij}$ аппроксимируются некоторыми упрощёнными функциональными действиями $ФД_{ij}^{*}$. Степень этого упрощения и определяет уровень детализации ИМ. Отличия $ФД_{ij}$ от $ФД_{ij}^{*}$ порождают

ошибки имитации реальной СТС. В ИМ каждое $ФД_{ij}^{*}$ представляется парой ($ФД_{ij}^{*}$, τ_{ij}), которая выполняется на ЭВМ следующим образом. Вначале реализуется $ФД_{ij}^{*}$ при неизменном значении её временной координаты t_i, а затем уже отображается изменение t_i на величину τ_{ij}, инициируя таким образом появление в ИМ СТС события C_{ij}.

На рис. 1.1 представлен пример развития действий компонента K_i в системе координат ($ФД_{ij}^{*}$, t_i). Конечно, подобная система координат носит условный характер, поскольку по оси ординат нельзя отложить «значение» аппроксимированного функционального действия $ФД_{ij}^{*}$. Каждое следующее $ФД_{ij}^{*}$ может иметь различную физическую природу. Поэтому подобное представление используется только для отображения того факта, что с изменением временной координаты t_i некоторая K_i выполняет несколько различных $ФД_{ij}$. Так, согласно рис 1.1 K_i последовательно выполняет $ФД_{i1}$, $ФД_{i2}$, $ФД_{i3}$, а в СТС соответственно происходят события C_{i1}, C_{i2}, C_{i3}. Причём, с появлением каждого нового события C_{ij} происходит изменение $ФД_{ij}$ и увеличение временной координаты t_i соответственно на величины τ_{i1}, τ_{i2}, τ_{i3}.

На рис. 1.1 появление в K_i реальной СТС событий C_{ij} при выполнении $ФД_{ij}$ показано штрихпунктирной линией. В ИМ СТС появление событий C_{ij} реализуется ступенчатой линией (0, *a*, C_{i1}, *b*, C_{i2}, *d*, C_{i3}). Это означает, что вначале выполняется $ФД_{ij}^{*}$ при неизменном t_i, а затем уже отображается изменение t_i на величину τ_{ij}, инициируя появление события C_{ij}. Затем реализуется на ЭВМ $ФД_{ij}^{*}$ при неизменном t_i и отображается изменение t_i на величину τ_{i2}, инициируя в K_i свершение события C_{i2}. Далее в два приёма (реализация $ФД_{i3}$ и последующее увеличение t_i на величину τ_{i3}) аналогичным образом инициируется событие C_{i3}. Отметим при этом, что порядок появления событий может быть другим: сначала изменяется t_i, а затем уже выполняется $ФД_{ij}^{*}$.

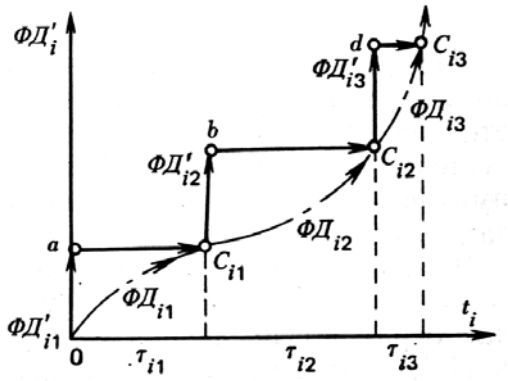

Рис. 1.1. Пример аппроксимации функциональных действий *i*-й компоненты системы K_i имитационной модели

В ИМ СТС каждое $ФД_{ij}^{*}$ описывается в общем случае некоторым алгоритмом $АЛ_{ij}$. В ходе имитации происходят реализация $ФД_{ij}^{*}$ по соответствующим алгоритмам $АЛ_{ij}$ и последующее изменение t_i на величину τ_{ij}. Таким

образом, любая ИМ описывается набором некоторых «молекул», каждая из которых содержит в себе описание алгоритма выполнения $АЛ_{ij}$ соответствующего $ФД_{ij}^{*}$ и оператора Mt_{ij}, осуществляющего изменение временной координаты t_i на величину τ_{ij}. Пару ($АЛ_{ij}$, Mt_{ij}) обычно называют *ij*-й активностью ИМ и обозначают $АК_{ij}$. Любая *ij*-я активность представляет собой запись поведения компонента K_i ИМ СТС. Реализация этой активности в ИМ приводит к появлению в ИМ СТС события C_{ij}.

Если бы на ЭВМ имитировалось поведение только одного компонента системы K_i, то выполнение активностей в ИМ можно было бы осуществить строго последовательно и имитация свелась бы к пересчету временной координаты t_i после очередного выполнения алгоритма $АЛ_{ij}$. В действительности СТС состоит из нескольких компонентов K_i. Все эти компоненты K_i в реальной СТС функционируют одновременно. Это должна отражать ИМ. В большинстве современных ЭВМ в каждый момент времени может реализоваться алгоритм только одной активности одной из компонент ИМ СТС.

Чтобы обеспечить имитацию на ЭВМ параллельных событий реальной СТС, вводят некоторую глобальную переменную t_0, которую называют модельным (системным) временем. С помощью этой переменной организуются синхронизация всех событий C_{ij} в ИМ и выполнение алгоритмов $АЛ_{ij}$ компонентов K_i ИМ СТС. Таким образом, при реализации ИМ используются три представления времени: реальное время СТС t_r, работа которой имитируется данной ИМ, модельное время t_0, по которому организуется синхронизация событий C_{ij} в ИМ СТС; машинное время имитации $t_Э$, отражающее затраты ресурса времени ЭВМ на организацию имитации.

С помощью модельного времени t_0 реализуется квазипараллельная работа компонентов K_i в ИМ. Приставка «квази» в данном случае отражает последовательный характер обслуживания событий в ИМ, одновременно возникающих в разных компонентах реальной СТС. Корректировка временных координат t_i нескольких K_i в ИМ осуществляется с помощью модельного времени t_0 следующим образом. Если значения t_i при выполнении $АЛ_{ij}$ нескольких K_i совпадают (это означает, что в реальной СТС происходит одновременно несколько событий C_{ij}), то последовательно обслуживаются $АЛ_{ij}$, совпадающие по времени выполнения, т. е. имеющие одинаковые значения t_{ij}. Здесь и далее под t_{ij} будем понимать конкретное значение t_i, при котором происходит событие C_{ij}. При этом модельное время t_0 не меняется до окончания выполнения всех совпавших по времени реализации алгоритмов $АЛ_{ij}$. Таким способом последовательно выполняются соответствующие $ФД_{ij}^{*}$ при неизменном значении t_0. После каждой реализации $АЛ_{ij}$, обеспечивающей выполнение в ИМ $ФД_{ij}^{*}$, работает оператор корректировки Mt_{ij} временной координаты t_i. Обычно эта корректировка сводится к вычислению нового значения t_{ij} по формуле $t_{ij} = t_0 + \tau_{ij}$. Это текущее значение временной координаты t_i запоминается и используется в дальнейшем для определения момента

новой активизации компонента модели K_i. Будем понимать под запуском на выполнение следующей его активности $АК_{ij}$ выполнение алгоритма $АЛ_{ij}$ и оператора Mt_{ij} корректировки временной координаты t_i.

Когда имитация одновременно появившихся событий C_{ij} завершена, выполнены соответствующие алгоритмы активностей $АК_{ij}$ и проведены корректировки временных координат t_i, меняется значение глобальной переменной ИМ t_0. Существуют два способа изменения t_o: с помощью фиксированных и переменных интервалов изменения модельного времени. Обычно их называют соответственно способами фиксированного шага и «шагов до следующего события».

Для того чтобы легче было представить оба способа организации изменения t_0, рассмотрим следующий пример (рис. 1.2). Пусть СТС состоит из трех компонентов K_i ($i = \overline{1,3}$). Количество и последовательности переходов K_i из состояния в состояние C_{ij} представлены на рис. 1.2. Компонент K_1 переходит четыре раза из состояния в состояние (C_{11}, C_{12}, C_{13}, C_{14}) в четыре момента изменения t_i (t_{11}, t_{12}, t_{13}, t_{14}). Между этими моментами K_1 выполняет четыре различных функциональных действия ($ФД_{11}$, $ФД_{12}$, $ФД_{13}$, $ФД_{14}$), соответственно в течение четырёх интервалов времени (τ_{11}, τ_{12}, τ_{13}, τ_{14}) аппроксимация $ФД_{1j}$ осуществляется последовательностью ($ФД_{11}^*$, $ФД_{12}^*$, $ФД_{13}^*$, $ФД_{14}^*$). Аналогичным образом $К_2$ последовательно выполняет три функциональных действия ($ФД_{21}^*$, $ФД_{22}^*$, $ФД_{23}^*$) соответственно в течение трёх интервалов времени (τ_{2j}), которые в ИМ аппроксимируются соответственно $ФД_{2j}^*$. Наконец, $К_3$ также последовательно выполняет три функциональных действия ($ФД_{31}^*$, $ФД_{32}^*$, $ФД_{33}^*$) соответственно в течение трёх интервалов времени (τ_{3j}), аппроксимируемых в ИМ СТС соответственно тремя $ФД_{3j}^*$. Отметим, что каждое $ФД_{ij}^*$ описывается соответствующим алгоритмом $АЛ_{ij}$ и реализуется соответствующей активностью $АК_{ij}$. Каждая активность представляет собой пару: алгоритм выполнения активности ($АЛ_{ij}$) и оператор изменения временной координаты Mt_i на величину (τ_{ij}). Алгоритм функционирования K_i в координатах ($ФД_{ij}^*$, t_i) представлен на рис. 1.2. Итак, в ходе выполнения активностей ($АК_{ij}$, τ_{ij}) в ИМ СТС происходят последовательно события C_{ij} в порядке, указанном на рис. 1.2.

Рассмотрим имитацию событий C_{ij} в ИМ СТС при каждом способе изменении t_0. При имитации по способу шагов до следующего события времена (t_0') меняются каждый раз на величину шага Δt (см. рис. 1.2). Тогда в моменты времени 0, Δt_1, $2\Delta t_1$, $3\Delta t_1$ предполагается обслуживание тех событий, которые попадают внутрь очередного интервала времени Δt. Считается, что события происходят в момент изменения шага Δt. Последовательно инициируются все события, попавшие внутрь интервалов Δt, что сводится к выполнению соответствующих активностей ($АЛ_{11}$, τ_{11}), ($АЛ_{31}$, τ_{31}), ($АЛ_{21}$, τ_{21}). Затем в момент $t_0' = 2\Delta t$ инициируется событие S_2. При этом считается, что со-

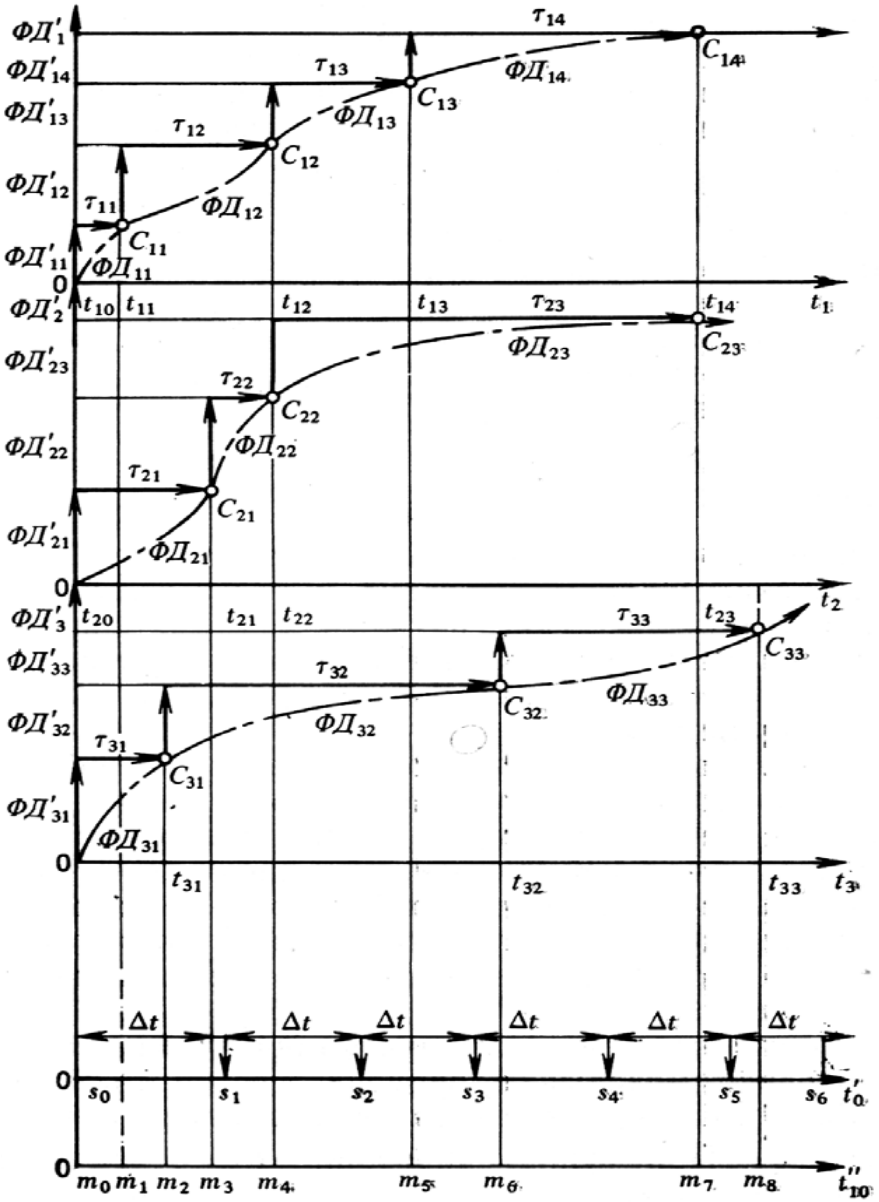

Рис. 1.2. Временная диаграмма имитационного моделирования событий в реальной системе из трех компонент

бытия C_{12} и C_{22} также происходят одновременно. Поэтому последовательно инициируются активности ($АЛ_{12}$, τ_{12}) и ($АЛ_{22}$, τ_{22}). Время t_0 получает приращение Δt. Таким образом, точность моделирования событий C_{ij} определяется шагом Δt изменения модельного времени t_0, и, как правило, все события обслуживаются в точке, соответствующей верхней границе интервала изменения t_0'.

При имитации по способу шагов до последующего события времени t_0'' меняется в моменты m_l, которые соответствуют временам t_{ij} появления событий в реальной системе. Обработка событий C_{ij}, одновременно появляющихся в реальной системе, осуществляется последовательно при неизменном модельном времени t_0. Например, события C_{12} и C_{22} в реальной системе происходят одновременно. В модели вначале реализуется, допустим, алгоритм $АЛ_{12}$ и модифицируется координата t_1 на величину τ_{12}, а затем выполняется алгоритм $АЛ_{22}$ и модифицируется координата t_2 на величину τ_{22}. Таким образом, t_0'' каждый раз сдвигается на величину, равную минимальному значению приращения модельного времени τ_{ij}, которое получают координаты t_i.

Независимо от способа изменения t_0 механизм регламентации изменения модельного времени обычно предусматривает выполнение следующих действий:

– выбор событий в модели, которые необходимо обслужить при одном и том же модельном времени t_0;

– обслуживание событий (инициализация активностей), которые имеют одинаковое время инициализации;

– по окончании обслуживания всех одновременных (в пределах шага) событий определение очередного значения модельного времени;

– корректировка временной координаты модели t_0;

– проверка условий окончания моделирования либо по времени завершения имитации ($T_{ОК}$), либо по выполнению других событий в системе.

Все эти действия выполняет управляющая программа моделирования (УПМ). Каждое событие C_{ij} в модели системы, являющееся результатом выполнения активности $АК_{ij}$, обслуживается УПМ и требует для реализации некоторого ресурса времени работы ЭВМ. Поэтому чем чаще обслуживаются события УПМ, тем больше возрастает машинное время $t_Э$. С этой точки зрения способ фиксированного шага изменения t_0 является более экономичным, поскольку имеет место групповое обслуживание всех событий C_{ij}, которые попали внутрь очередного шага изменения Δt модельного времени. Обычно предпочтение способу фиксированного шага отдаётся в двух случаях. Во-первых, когда события C_{ij} распределены равномерно на всём интервале моделирования (0, $T_{ОК}$) и исследователь может подобрать интервал изменения временной координаты Δt, обеспечивающий минимальную погрешность имитации. Во-вторых, когда событий очень много и они появляются

группами. Во всех остальных случаях способ задания шага до следующего события более предпочтителен (особенно, когда события C_{ij} распределены неравномерно и появляются они через значительные временные интервалы τ_{ij}). С учётом достоинств и недостатков каждого из способов изменения t_0 наметилась специализация их применения при моделировании СТС. На практике наибольшее распространение получил способ шага до следующего события. Поэтому в дальнейшем будем рассматривать именно этот способ имитации.

1.3. Способы организации квазипараллелизма в имитационных моделях СТС

Любая ИМ СТС представляет собой совокупность набора «молекул», отражающих поведение объектов имитации, и УПМ, организующих взаимодействие этих «молекул» друг с другом. Для задания входных условий, начальных значений параметров и запуска ИМ в состав модели включается подпрограмма «НАЧАЛО». ИМ создаётся для изучения на основе статистики процессов, протекающих в СТС. Поэтому для сбора статистики моделирования в ИМ вводится подпрограмма « СТАТИСТИКА», которая присутствует в ИМ либо в явном виде, либо она рассредоточена по всем компонентам K_i в ИМ. Наконец, по достижении условий окончания имитации УПМ передаёт управление на подпрограмму окончание имитации (**«окончание»**).

На рис. 1.3 представлена схема перевода объекта моделирования в его имитационную модель. Из рисунка видно, что любое $ФД_{ij}$ в ИМ описывается соответствующей активностью $АК_{ij}$. Каждая $АК_{ij}$ представляет собой попарное сочетание описания алгоритма $АЛ_{ij}$ и оператора модификации временной координаты модели Mt_{ij}. УПМ передаёт управление на начало программы, реализующей алгоритмы $АЛ_{ij}$. Возврат на УПМ имеет место при выполнении оператора Mt_{ij}. УПМ и служебные подпрограммы **«начало»**, **«статистика»**, **«окончание»** обычно являются универсальными и не меняются при переходе от одной модели к другой. Однако принципы их построения и способ управления выполнением активности зависят от способа формализации СТС. Наибольшее распространение получили пять способов описания ИМ: непосредственно активностями, аппаратом событий, транзактами, агрегатами, процессами. Каждому способу формализации СТС соответствует свой способ организации квазипараллелизма обслуживания УПМ активностей, из которых составлена ИМ.

Поэтому различают соответственно следующие способы организации квазипараллелизма в ИМ: просмотр активностей, составление расписания событий, управление обслуживанием транзактов, управление агрегатами, синхронизация процессов. Одну и ту же СТС принципиально можно представить любым из указанных способов формализации. Однако построенные

на их основе ИМ будут отличаться размерами и количеством ресурсов, затраченных на их создание, испытание и использование.

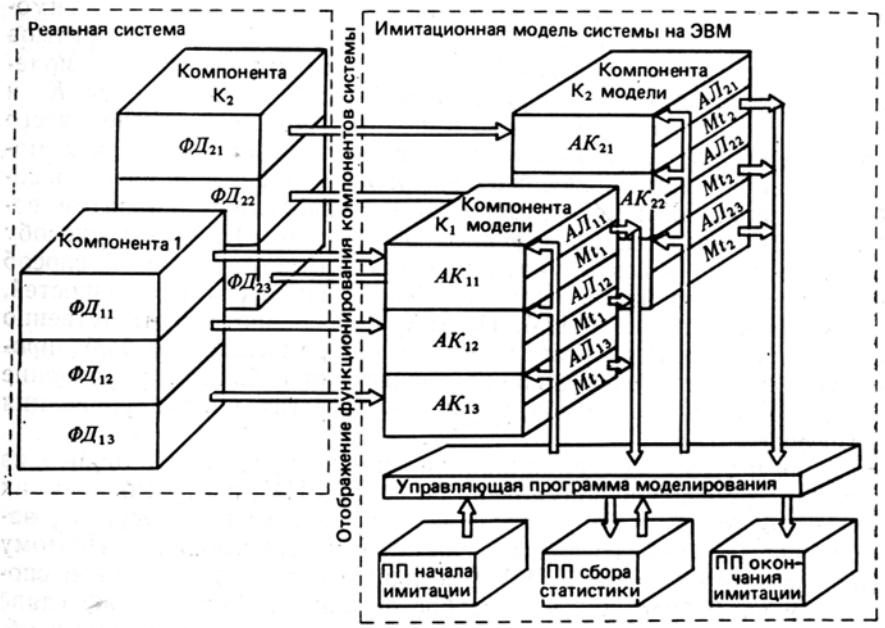

Рис. 1.3. Схема перевода объекта моделирования в его имитационную модель

Организация квазипараллелизма способом просмотра активностей используется при моделировании СТС, характеризующихся следующим. Все $ФД_{ij}$ компонентов K_i реальной системы различны. Причём для выполнения каждого из них требуется выполнение своих условий. Эти условия конкретны, известны заранее исследователю и могут быть представлены алгоритмически. В результате выполнения $ФД_{ij}$ в СТС происходят различные события C_{ij}. Связи между $ФД_{ij}$ отсутствуют. Все $ФД_{ij}$ функционируют независимо друг от друга. В таких случаях исследователь описывает ИМ в виде двух частей: множества активностей $\{АК_{ij}\}$ и набора процедур проверки выполнимости условий инициализации активностей. Под инициализацией $АК_{ij}$ понимают передачу управления от УПМ на выполнение $АЛ_{ij}$ данной активности. Завершается обслуживание $АК_{ij}$ выполнением оператора модификации Mt_{ij}. По этому оператору управление возвращается УПМ. Обычно проверка выполнимости условий инициализации $АК_{ij}$ состоит либо в определении значений параметров ИМ, либо в вычислении моментов t_{ij}, в которые должно начаться выполнение соответствующего $ФД_{ij}$, либо в проверке значений переменных модели. Алгоритмически реализуется своя последовательность проверок выполнимости условий инициализации $АК_{ij}$. Отметим, что пользователь описывает не только алгоритмы активностей $АЛ_{ij}$ и операторы Mt_{ij}, но и алгоритмы проверки условий инициализации $АК_{ij}$.

На рис. 1.4 представлена схема модели объекта, состоящей из активностей. Для простоты предполагаем, что каждая $АК_{ij}$ имеет только одну проверку условия инициализации.

Поскольку выполнение алгоритмов одних активностей $АЛ_{ij}$ может привести к инициализации других $АК_{ij}$, то УПМ возможны повторные циклы проверки выполнимости условий инициализации $АК_{ij}$. При этом события C_{ij} в ИМ не регламентированы, а лишь указываются условия, при которых они могут произойти. Инициализация $АК_{ij}$ и сдвиг координат t_i операторами Mt_{ij} разрешаются только тогда, когда выполняются все условия начала инициализации. Если хотя бы одно из этих условий остаётся невыполненным, то соответствующая $АК_{ij}$ не попадает в список инициализируемых активностей. Иногда удельный вес безуспешных поверок очень велик в работе УПМ. Поэтому данный способ организации квазипараллелизма $АК_{ij}$ выгоден только при наличии достаточно простых алгоритмов проверки выполнимости условий инициализации активностей.

Транзактный способ организации квазипараллелизма используется в тех случаях когда $ФД_{ij}$ реальной СТС одинаковы, а общее число функциональных действий ограничено. Каждое $ФД_{ij}$ представляет собой набор простейших операций и его можно аппроксимировать активностями, алгоритмы выполнения которых лишь корректируют значения t_i компонентов системы K_i. При этом существует такая зависимость выполнения $ФД_{ij}$ друг от друга, которую удобно представить в виде блок-схемы. Взаимодействие такого рода активностей аналогично работе систем массового обслуживания (СМО). Однотипные $АК_{ij}$ пользователем объединяются и называются приборами СМО. Инициаторами появления событий C_{ij} в ИМ становятся требования (транзакты) на обслуживание этими приборами ($ПР_i$). Связь между $ПР_i$ устанавливается с помощью системы очередей ($ОЧ_l$) и выбранных дисциплин обслуживания очередей ($DISC_l$).

Пусть $ФД_{12}$, $ФД_{22}$, $ФД_{23}$ в рассмотренном ранее примере совпадают. Кроме того, в любой данный момент выполняется только одно из них, и исследователя интересует влияние этих $ФД_{ij}$ на поведение всей системы. Взаимосвязь между $ФД_{ij}$ представляется в виде блок-схемы. ИМ представляется в виде схемы, отображающей рождение транзактов, их пространственное перемещение по схеме и, наконец, уничтожение уже обслуженных транзактов. Для описания ИМ создаётся достаточно широкий, но фиксированный набор стандартных блоков обслуживания транзактов. С их помощью представляются действия по созданию и уничтожению транзактов, управлению движением транзактов, занятию и освобождению различных типов ресурсов СТС,

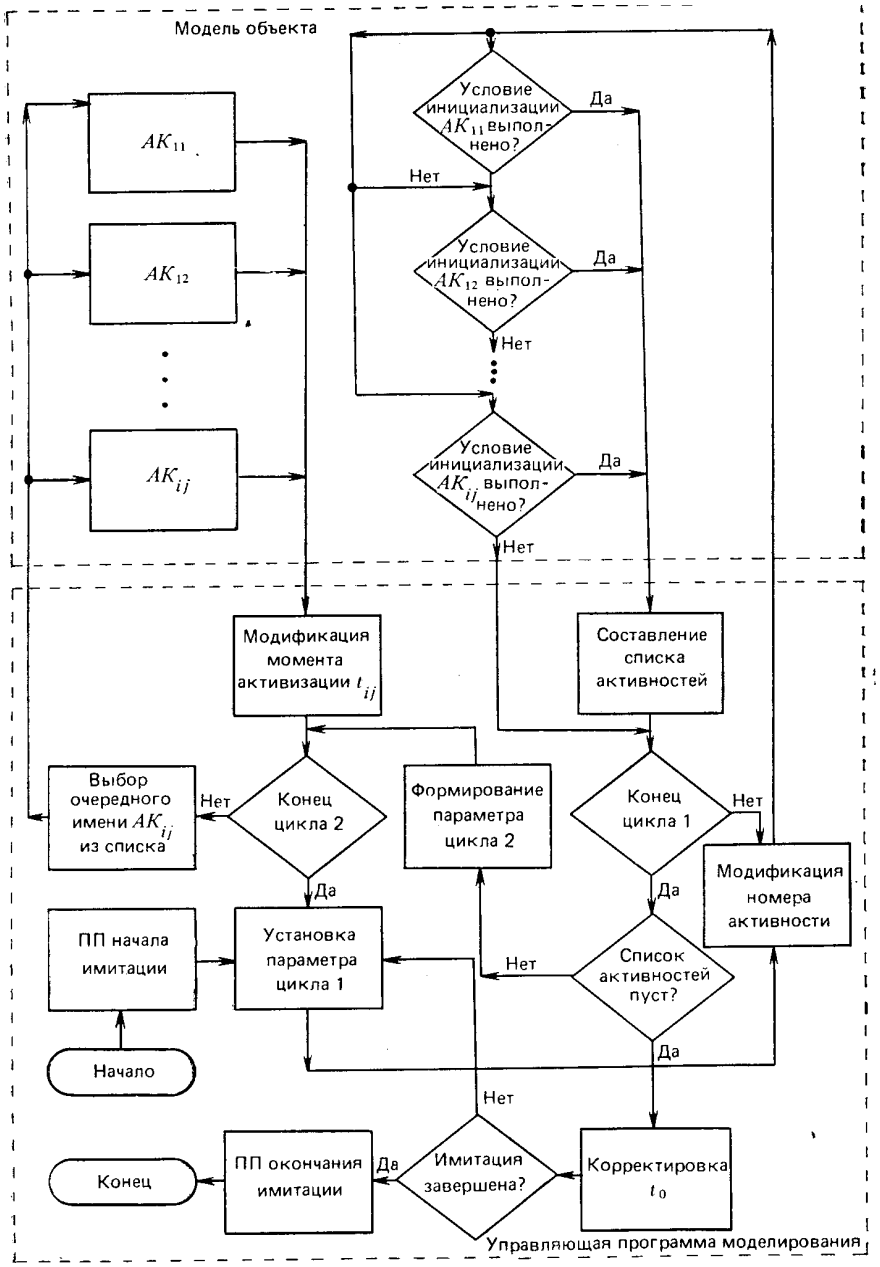

Рис. 1.4. Схема модели объекта, состоящей из активностей

имитации задержек и продвижения транзактов. Для этого используется несколько десятков графически различимых блоков, позволяющих в наглядной форме представлять алгоритмы функционирования СТС.

Далее осуществляется символическое кодирование схемы, при котором каждому блоку ставится в соответствие определённый оператор соответствующего языка моделирования. Таким способом формируется текст программы-модели, которая затем реализуется в режиме интерпретации. На рис. 1.5 изображена схема взаимодействия УПМ с моделью объекта при транзактом способе имитации реальной СТС, представленной на рис. 1.3. В источниках ($ИСТ_i$) транзакты создаются, а в блоке поглотителя все добравшиеся до него транзакты уничтожаются. УПМ просматривает списки $ИСТ_i$ и создаёт новые транзакты с заданной интенсивностью их поступления в систему. Таким образом, все поступающие в систему транзакты имитируют собой внешнее окружение СТС. Событием в такой ИМ является момент инициализации любого транзакта. Поэтому транзакт выступает в роли активности. УПМ назначает транзактам новые моменты инициализации t_{ij} и до наступления этих моментов закрепляет за транзактами соответствующие блоки обслуживания ($БЛ_{ij}$). Программы сканирования $ИСТ_j$ просматривают условия создания новых транзактов каждым из источников и передаёт управление подпрограмме создания транзактов при совпадении этих условий. Она

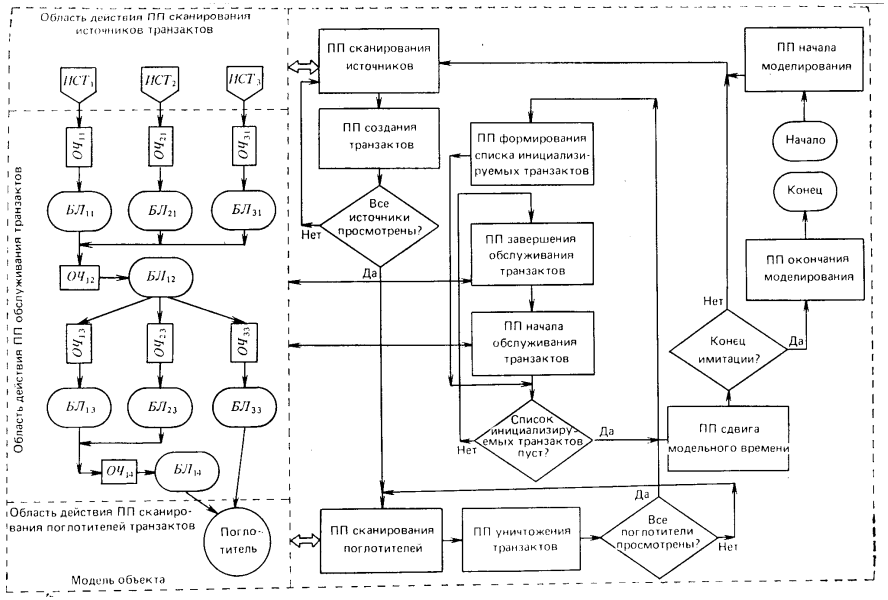

Рис. 1.5. Схема взаимодействия управляющей программы с моделью объекта при транзактном способе имитации

формирует новые транзакты и помещает их в очередь к блокам (в нашем примере к $БЛ_{11}$, $БЛ_{21}$, $БЛ_{31}$). После создания всех транзактов, для которых выполнялись условия их создания, УПМ приступает к просмотру поглотителей. При каждом уничтожении транзакта формируется статистика пребывания транзакта в ИМ. Когда просмотрены все поглотители ИМ, управление передаётся подпрограмме формирования списка инициализируемых транзактов.

Под инициализацией транзакта понимают завершение пребывания транзакта в каком-либо блоке модели или поступление транзакта в соответствующие очереди к блокам модели системы. Подпрограмма формирования списка инициализируемых транзактов просматривает списки транзактов (очереди), поступивших на входы к блокам $БЛ_{ij}$, и выбирает из них те, у которых время инициализации совпадает с t_0, образуя список активизируемых транзактов. По окончании просмотра всех блоков УПМ проверяет условие «список инициализации транзактов пустой». Если в этом списке имеется хотя бы один транзакт, управление передаётся на подпрограмму завершения обслуживания транзактов. Как только список активизируемых транзактов оказывается пустым, подпрограмма сдвига модельного времени выбирает из списка моментов инициализации $\{t_{ij}\}$ минимальное значение, которое становится новым модельным временем t_0. Просматриваются все те транзакты в этом списке, у которых моменты инициализации совпадают, формируется начальный список инициализируемых транзактов. Затем проверяется выполнение условий окончания имитации. Если условия не выполнены, то управление передаётся на подпрограмму сканирования источников. Таким образом, УПМ циклически сканирует различные списки транзактов с целью инициализации и организации обслуживания транзактов соответствующими блоками, сдвигает модельное время и проверяет выполнимость условий окончания имитации. Итак, за внешнюю простоту описания ИМ приходится платить достаточно большими накладными расходами на организацию имитации.

Процессный способ организации квазипараллелизма в ИМ используется при моделировании СТС, характеризующихся следующим.

Когда все $ФД_{ij}$ реальной СТС различны, условия появления событий C_{ij} индивидуальны, у каждой компоненты K_i существует определённая последовательность выполнения $ФД_{ij}$, в любой момент времени в данной компоненте может выполняться только одно $ФД_{ij}$, то перечисленные ограничения определяют выбор исследователем процессного способа организации квазипараллелизма в ИМ. Процессный способ имитации особенно эффективен в тех случаях, когда требуется высокий уровень детализации выполнения $ФД_{ij}$ при их аппроксимации с помощью $АЛ_{ij}$ и сама ИМ используется для поиска узких мест в СТС. В таких случаях очень важно соблюдение сходства структуры ИМ и СТС, что обеспечивается процессным способом имитации. В

нашем примере (см. рис. 1.3) СТС удобнее всего имитировать процессным способом, когда в один и тот же момент времени не может реализоваться более одного $ФД_{ij}$ данной i-й компоненты СТС. В этих случаях удобнее рассматривать функционирование K_i как единое целое. Всю ИМ можно представить в виде набора описаний процессов, каждое из которых раскрывает поведение одного класса процессов, например, компоненты K_i для нашего случая. При этом могут иметь место информационные и управляющие связи не только между K_i, но даже и между отдельными алгоритмами $АЛ_{ij}$ их функционирования.

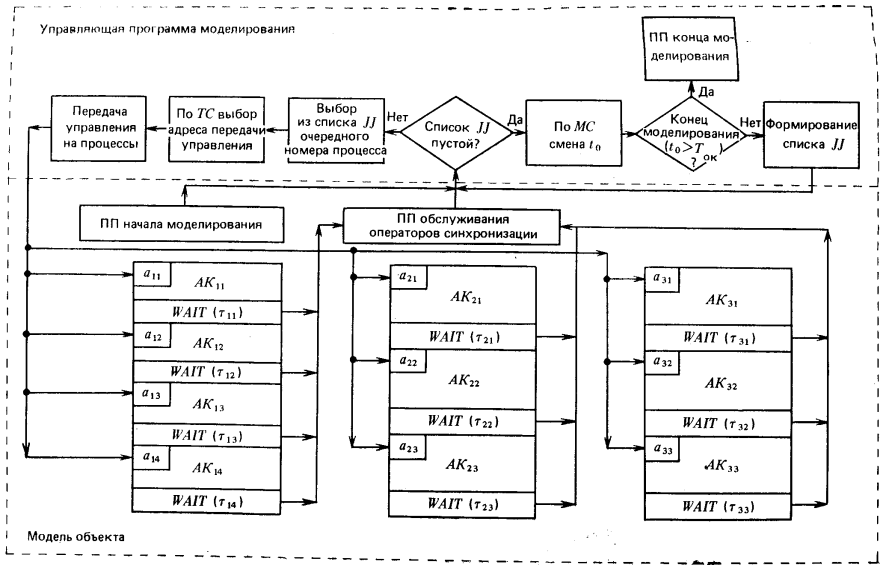

Рис. 1.6. Схема взаимодействия УПМ с моделью объекта, состоящей из описаний процессов

На рис. 1.6 представлена схема взаимодействия УПМ с моделью объекта, состоящей из описаний процессов. Алгоритм функционирования ИМ представляется последовательным взаимодействием процессов и УПМ. Причём в процессы объединяются связанные между собой активности, которые определяют функционирование одной и той же компоненты модели K_i. Таким образом, имеет место полное соответствие компонент реальной СТС и её ИМ. Каждой K_i СТС соответствует свой процесс. Переход от выполнения одной активности к другой активности того же процесса считают изменением его состояния и называют активизацией процесса. Обычно под состоянием процесса понимают номер j-й активности, которая входит в состав i-го процесса и на которую УПМ передаёт управление при совершении события C_{ij} в K_i. Следовательно, изменениям состояний реальной СТС соответствуют

изменения определённых для них состояний процессов ИМ, что приводит к появлению событий C_{ji}.

Вся ИМ представляет собой набор процессов, реализованных на соответствующем языке моделирования. Процессы связаны с УПМ с помощью некоторых операторов этого языка. Эти операторы обеспечивают обращение к УПМ при завершении активности данного процесса. Таким способом происходит переход процесса в другое состояние. Отметим, что проверка выполнимости условий активизации процесса и появления событий C_{ij} осуществляется самим процессом. Будучи активизированным, выполнение процесса (некоторой его активности $АК_{ij}$) может начаться немедленно либо задержаться до появления определённых условий или до изменения состояний других процессов. Процессы могут переходить в новые состояния как по своей инициативе, так и в результате действий, выполняемых активностями других процессов.

Будем считать, что взаимодействия между активностями процессов в нашем примере (см. рис. 1.3) нет и все $АК_{ij}$ обращаются к УПМ с помощью операторов синхронизации процессов $WAIT(\tau_{ij})$. Каждый такой оператор означает, что i-му процессу по окончании выполнения алгоритма $АЛ_{ij}$ активности $АК_{ij}$ назначается момент следующей активизации процесса (t_{ij}) по окончании ожидания процесса в модельном времени t_0 длительностью τ_{ij}. Допустим также, что началу выполнения $АЛ_{ij}$ процессов соответствуют адреса a_{ij} подпрограмм, реализующих выполнение тех $АК_{ij}$, которые объединены в процессы. УПМ работает с массивом состояний процессов (МС) и таблицей состояний процессов (ТС). В МС каждый элемент представляет собой пару значений (i, t_{ij}), где i – номер процесса, t_{ij} – момент активизации процесса в будущем и появления события перехода в состояние C_{ij}. Для выбора процессов, которые необходимо активизировать в момент t_0, УПМ использует ТС. Строками этой таблицы являются списки параметров процессов. Так, i-му процессу соответствует строка, в которой указаны следующие параметры процесса: a_{ij} – адрес передачи управления на выполнение j-й активности в j-м процессе, $t_{НДi}$ – время, которое осталось i-му процессу находиться в состоянии ожидания, когда он останавливается другим процессом; π_j – приоритет процесса, согласно которому осуществляется последовательное обслуживание УПМ двух одновременно активизируемых процессов. Все элементы МС упорядочены по возрастанию значений t_{ij}. Если какой-либо i-й процесс остановлен, то время активизации в МС бесконечно велико ($t_{ij}=\infty$), а соответствующий ему элемент массива находится в конце МС.

Определено также понятие «конфликтная ситуация в ИМ», когда несколько событий C_{ij} в различных процессах происходят одновременно и требуют немедленного обслуживания со стороны УПМ. Такая ситуация имеет место, например, при наличии в МС нескольких элементов с одним и тем же значением t_{ij}. Учёт конфликтных ситуаций УПМ производится с помощью

списка одновременно активизируемых процессов (список JJ). В списке JJ находятся номера тех процессов, которые необходимо активизировать в одно и то же модельное время t_0. Запись номеров процессов i в список JJ и выбор УПМ из этого списка процессов осуществляется согласно приоритетам процессов π_i. Обслуживание УПМ очередного элемента списка JJ состоит в передаче управления на выполнение алгоритма $АЛ_{ij}$ i-го процесса. После передачи управления по адресу a_{ij} выполнение этого алгоритма продолжается до появления в нём очередного оператора синхронизации процесса $WAIT(\tau_{ij})$. Таким образом, с помощью операторов ожидания типа $WAIT(\tau_{ij})$ в нашем примере имитируется изменение временных координат активностей, входящих в состав i-го процесса.

В нашем примере (см. рис. 1.3) реальный объект состоит из трёх K_i, и поэтому его ИМ состоит также из трёх процессов ($i=1, 2, 3$). Пусть задано время окончания моделирования ($T_{ОК}$). УПМ обслуживает процессы согласно алгоритму, представленному на рис. 1.6.

Пусть в начальный момент моделирования ($t_0=m_0$, см. рис. 1.6) список JJ пуст и процессам назначены соответственно моменты активизации t_{11}, t_{21}, t_{31} в МС. Реализуется алгоритм имитации в 5 шагов.

Шаг 1. Поскольку список JJ пуст, то осуществляется процедура смены t_0, представляющая собой выбор минимального значения из МС ($t_0=\min\{t_{ij}\}$).

Шаг 2. Формируется список JJ следующим образом. Все процессы, у которых $t_{ij}\le t_0$ выбираются из МС и заносятся в список JJ согласно их приоритетам π_i, которые определяются из таблицы ТС.

Шаг 3. Выбирается первый элемент списка JJ, который необходимо в момент $t_{ij}=t_0$ активизировать. По ТС определяется адрес $a_{ij,}$ и управление от УПМ передаётся на выполнение алгоритма соответствующей активности $АК_{ij}$. По окончании выполнения алгоритма $АЛ_{ij}$ появляется оператор ожидания i-го процесса до следующей активизации $WAIT(\tau_{ij})$. В ходе реализации $АЛ_{ij}$ происходит вычисление или задание значения τ_{ij} для операторов синхронизации процессов. Появление в алгоритме i-го процесса операторов синхронизации $WAIT(\tau_{ij})$ возвращает управление УПМ, которая выполняет в дальнейшем следующие действия:

– формирование нового элемента МС и занесения его в этот массив согласно значению $t_{ij}=t_0+\tau_{ij}$;

– модификация текущего состояния процесса и изменение при этом адреса последующей передачи управления a_{ij} для очередной активизации процесса i в ТС.

Шаг 4. Проверяется, исчерпан ли список JJ. Если в списке JJ есть ещё элементы, то УПМ переходит к шагу 2. Иначе выполняется шаг 5.

Шаг 5. Проверяется момент окончания моделирования ($t_0\ge T_{ОК}$). Если это неравенство не выполняется, то это означает необходимость продолжения

имитации, начиная с шага 1. В противном случае управление передаётся программе окончания моделирования.

Продолжим рассмотрение приведённого примера, используя временную диаграмму перехода K_i (см. рис. 1.2) из состояния в состояние и схему взаимодействия УПМ с процессами (см.рис. 1.6). Итак, первоначально в момент t_0=m_0 в списке *JJ* находятся все три процесса, которые расположены соответственно их приоритетам (например, $\pi_1 > \pi_2 > \pi_3$). Допустим, что время окончания моделирования $T_{ОК}$ равно моменту m_8 на оси изменения модельного времени t_0 (см. рис. 1.2).

УПМ вначале передаёт управление по адресу a_{11}. Выполнение алгоритма активности $АК_{11}$ завершается появлением оператора *WAIT*(τ_{11}), и управление передаётся подпрограмме обслуживания операторов синхронизации, которая выполняет следующие действия:

– формирует новый адрес продолжения процесса 1 в ТС (a_{12});

– назначает момент будущей активизации процесса 1 ($t_{11} = m_0 + \tau_{11}$);

– формирует новую строку *MC* (1, t_{11}) и заносит её в этот массив.

Далее, поскольку *JJ* не пустой, происходит выбор процесса 2, и управление передаётся по адресу a_{21} для выполнения $АЛ_{12}$ до появления оператора *WAIT*(τ_{21}) и возврата управления на подпрограмму обслуживания операторов синхронизации. Формируется новый момент активизации процесса 2 ($t_{21} = m_0 + \tau_{22}$). В ТС модифицируется адрес продолжения выполнения процесса 2 (a_{22}) и формируется новая строка МС (2, t_{21}), которая заносится в соответствии со значением t_{21} (после строки для процесса 1) в этот массив. Аналогично обслуживается процесс 3, и управление передаётся по адресу a_{31} для выполнения $АЛ_{31}$ до появления оператора *WAIT*(τ_{31}) и возврата на УПМ. Далее формируется новый момент активизации процесса 3 ($t_{31} = m_0 + \tau_{31}$). В ТС модифицируется адрес продолжения выполнения процесса (a_{32}) и формируется новая строка МС (3, t_{31}), которая в соответствии со значением t_{31} заносится в этот массив (между строками первого и второго процессов, см. рис. 1.2).

Когда список *JJ* исчерпан, происходит смена модельного времени t_0. Новым значением t_0 становится минимальное значение: $t_0 = \min(t_{11},\ t_{31},\ t_{21}) = t_{11} = m_1$. При этом $i = 1$ заносится в список *JJ* и поскольку $t_{11} < m_8$, то управление передаётся на формирование списка *JJ*. Так как в момент $t_0 = m_1$ активизируется только один процесс, то в списке *JJ* будет находиться всего один процесс. Управление передаётся по адресу a_{12} на выполнение $АЛ_{12}$ до появления оператора *WAIT*(τ_{12}). Процессу 1 назначается новый момент активизации $t_{12} = t_0 + \tau_{12} = m_1 + \tau_{12}$, и в ТС модифицируется адрес последующей активизации процесса 1 (a_{13}). В соответствии со значением t_{12} строка (1, t_{12}) заносится в МС после строк для процессов 2 и 3. Далее аналогичным образом продолжаются обслуживанием УПМ процессов и смена t_0 согласно временной диаграмме (см. рис. 1.2) до момента, когда модельное время $t_0 = m_8$. В этот момент по окончании обслуживания процессов 1 и 2 выполняется условие $t_0 \geq T_{ОК}$, и процесс моделирования завершается, а управление передаётся на программу окончания моделирования.

Если в СТС имеет место тесное взаимодействие между $ФД_{ij}$, то K_i обмениваются между собой сигналами. Причём каждый выходной сигнал от одной компоненты K_i является входным сигналом для другой компоненты. Если при этом сами $ФД_{ij}$ аппроксимируются явно задаваемыми математическими зависимостями, позволяющими определить момент появления выходных сигналов K_i при наличии входных сигналов, поступающих от других компонентов, то создаются подходящие условия для построения ИМ по модульному принципу. В этом случае каждый из модулей ИМ строится по унифицированной структуре и называется ***агрегатом***. Агрегат является математической схемой, с помощью которой возможно описание большого круга реальных процессов в СТС. В любой момент t_{ij} агрегат может находиться в одном из возможных состояний. Состояния агрегата Z являются функциями времени. Зависимости $Z(t)$ называют фазовыми траекториями. Переход агрегата Z из состояния в состояние описывается с помощью некоторого оператора перехода H, который позволяет по предыдущему состоянию определить очередное его состояние. Агрегат имеет входы, куда поступают входные сигналы $X_l(t)$ от других агрегатов, и выходы, на которых формируются выходные сигналы $Y_r(t)$. Здесь l и r означают номера соответственно входных и выходных сигналов. Кроме того, у агрегата имеются дополнительные входы, на которые поступают управляющие сигналы $g(t)$. Выходные сигналы $Y_r(t)$ формируются из входных $X_l(t)$ и управляющих $g(t)$ сигналов оператором выхода G в результате его взаимодействия с оператором M. Значения операторов G и H задаются исследователем при аппроксимации агрегатами и выполняются $ФД_{ij}$ реальной СТС. Квазипараллельная работа агрегатов может быть реализована различными способами (активностями, транзактами, процессами).

Организация квазипараллелизма способом составления расписания событий используется в следующих случаях. Различные компоненты K_i выполняют одни и те же $ФД_{ij}$. Начало этих $ФД_{ij}$ определяется одними и теми же условиями, которые также заранее известны исследователю и могут быть предоставлены алгоритмически. В результате выполнения одних и тех же $ФД_{ij}$ в системе происходят одинаковые события C_{ij} независимо друг от друга. Связи между различными $ФД_{ij}$ отсутствуют, и каждое $ФД_{ij}$ выполняется независимо. Например, пусть у компонентов системы (см. рис. 1.3) совпадают следующие $ФД_{ij}$, приводя к одним и тем же событиям:

$ФД_{11}$, $ФД_{12}$, $ФД_{11}$ – к событию C_{11};

$ФД_{21}$, $ФД_{22}$, $ФД_{23}$ – к событию C_{12};

$ФД_{31}$, $ФД_{32}$, $ФД_{33}$ – к событию C_{13};

$ФД_{14}$ – к событию C_{14}.

В таких случаях исследователь описывает ИМ в виде двух частей: множества активностей $\{АК_{ij}\}$ и набора процедур проверки появления событий C_{ij} и инициализации соответствующих активностей. При этом каждая $АК_{ij}$ имитирует выполнение группы совпавших $ФД_{ij}$ у различных компонентов K_i системы. В нашем случае $АК_{11}$ инициируется при появлении события C_{11} и имитирует выполнение $ФД_{11}$, $ФД_{12}$ и $ФД_{13}$. Аналогичным образом инициируются: $АК_{12}$ – при появлении события C_{12}; $АК_{13}$ – при появлении события C_{13}; $АК_{14}$ – при появлении события C_{14}. Инициализация $АК_{ij}$ имеет тот же смысл, что и для предыдущих способов организации квазипараллелизма. В процедурах проверки появления событий C_{ij} реализуется зависимость выполнения $ФД_{ij}$ от конкретной ситуации, имеющей место в реальных СТС. Выполнение $АЛ_{ij}$ также называется обслуживанием $АК_{ij}$, которое завершается оператором Mt_{ij} активности, обслуживающей группу одинаковых $ФД_{ij}$ у разных K_i реальной СТС. Часто такие групповые активности называют процедурами обслуживания событий C_{ij}. В результате ИМ состоит из двух типов процедур: проверки выполнимости событий C_{ij} в модели системы и обслуживания событий C_{ij}. Выполнение этих процедур синхронизируется в t_o списковым механизмом планирования УПМ. Каждый элемент этого списка определяет момент t_{ij} свершения события C_{ij}, а также имя или номер той процедуры обслуживания событий, которая должна выполняться после завершения этого события.

Более подробно с организацией имитации способами агрегатным и составлением расписаний событий можно ознакомиться в монографии [1].

1.4. Технологические этапы имитационного моделирования СТС

Независимо от назначения моделирования можно выделить следующие этапы создания и использования имитационных моделей:

– определение объекта имитации, установление границ и ограничений моделирования, выбор показателей для сравнения эффективности вариантов СТС (***составление содержательного описания*** СТС);

– формулировка замысла модели, переход от реальной СТС к логической схеме её функционирования (***составление концептуальной модели***);

– реализация описания объекта в терминах математических понятий и алгоритмизация функционирования его компонентов (***составление формального описания объекта***);

– преобразование формального описания объекта в описание ИМ (***составление описания модели***);

– программирование и отладка модели (***программирование модели***);

– проверка модели, оценка её свойств и затрат ресурсов на имитацию

(***испытание*** ИМ);

– организация модельного эксперимента на ЭВМ (***эксплуатация*** ИМ);

– интерпретация результатов моделирования и их использование при анализе поведения СТС (***анализ результатов***).

На рис. 1.7 представлена схема взаимосвязи технологических этапов имитационного моделирования СТС. Рассмотрим порядок действий исследователя на каждом из этапов имитационного эксперимента (ИЭ) на ЭВМ.

Составление содержательного описания объекта моделирования представляет собой выполнение следующих действий. Вначале определяется объект имитации, достаточный для изучения тех сторон его функционирования, которые представляют интерес для исследователя. Устанавливаются границы изучения функционирования объекта. Составляется возможный список ограничений ИМ, которые допустимы при организации имитации или при наличии которых ещё имеет смысл имитация функционирования СТС. Перед разработчиками ИМ ставятся вполне конкретные цели моделирования и формулируются основные критерии эффективности, по которым предполагается проводить сравнение на модели вариантов организации СТС. Результатом работ на данном этапе является содержательное описание объекта моделирования с указанием целей имитации и аспектов функционирования объекта моделирования, которые необходимо изучить на ИМ. Обычно оно представляет собой техническое описание объекта моделирования, описание внешней среды, с которой он взаимодействует, и временную диаграмму этого взаимодействия.

Составление концептуальной модели производится в такой последовательности. На основе анализа поставленной задачи определяется общий замысел модели. Выдвигаются гипотезы и фиксируются все допущения, необходимые для построения ИМ. На основании содержательного описания уточняется задача моделирования, определяется процедура и график её решения. Уточняется методика всего ИЭ в зависимости от наличных ресурсов, выделенных для имитации. Общая задача моделирования разбивается на ряд частных задач. Устанавливаются приоритеты решения этих задач. Обосновываются требования в ресурсах ЭВМ. Выполняются такие работы, как выбор параметров и переменных СТС, представляющих интерес для моделирования; уточнение критериев эффективности вариантов СТС; выбор типов аппроксимации отдельных компонентов модели. Проводятся также предварительный анализ требований к модели СТС; определение необходимых математических уравнений, описывающих реальные процессы; поиск возможных методов проверки правильности функционирования модели. Одновременно с этим исследователь должен выбрать язык будущей формализации процессов в объекте моделирования. Результатом выполнения работ являются концептуальная модель, выбранный язык формализации и способ

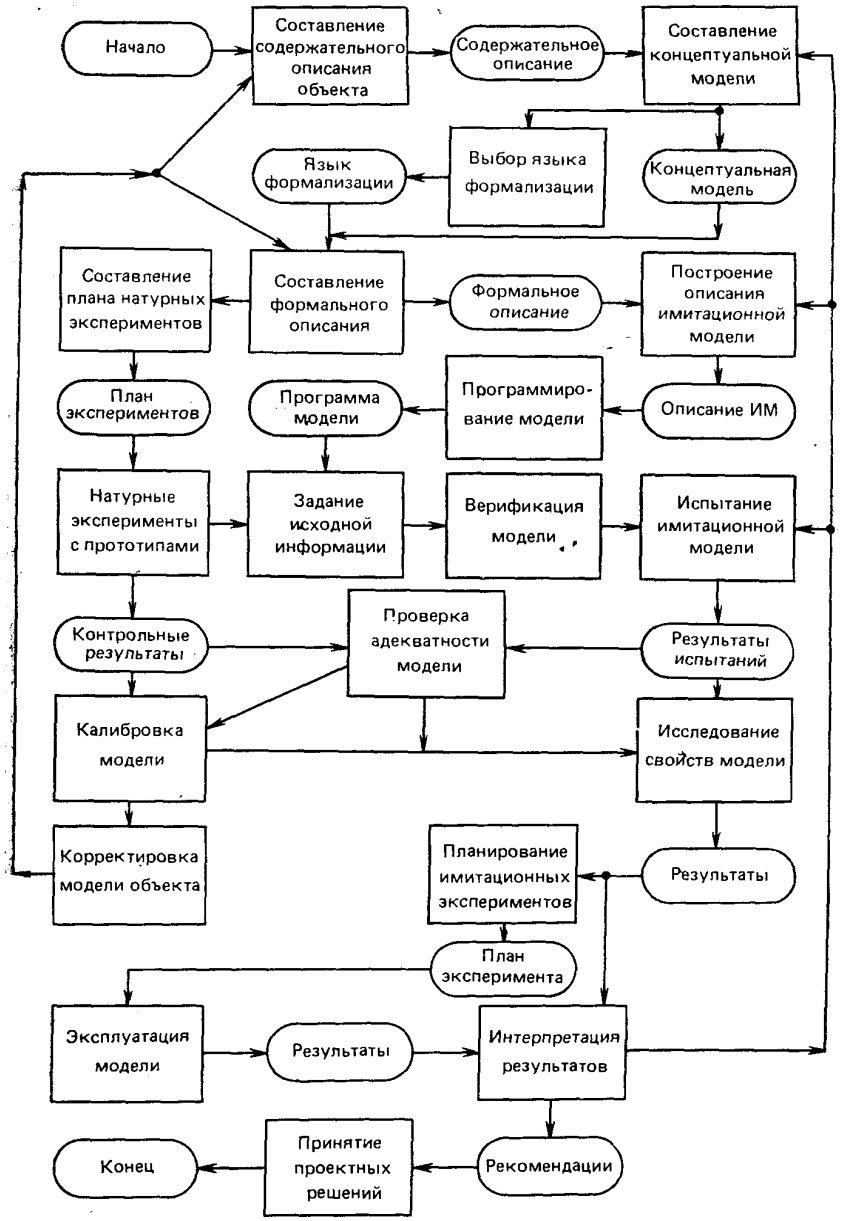

Рис. 1.7. Схема взаимосвязи технологических этапов имитационного моделирования СТС

организации имитации. В состав концептуальной модели входят: уточнённое содержательное описание, свободное от всего того, что не представляет интереса для имитации поведения СТС, список параметров и переменных моделирования; критерии эффективности функционирования вариантов системы; список используемых методов обработки результатов имитации и перечисление способов представления результатов моделирования. При создании небольших ИМ данный этап работ совмещается с этапом составления содержательного описания моделируемой системы. Только с усложнением объекта моделирования и задач имитации появляется необходимость определения способа формализации, который подходит для решения конкретной задачи исследования СТС.

Составление формального описания объекта моделирования на выбранном языке формализации. В зависимости от сложности СТС могут использоваться три вида формализации: аппроксимация явлений функциональными зависимостями, алгоритмическое описание процессов в СТС, смешанное представление в виде последовательности формул и алгоритмических записей. В зависимости от принятого способа имитации используются свои способы формализации (активностями, транзактами, процессами, агрегатами и событиями). При составлении формального описания СТС исследователю рекомендуется такая последовательность действий: уточнение декомпозиции системы, алгоритмизация компонентов модели; информационные стыковки алгоритмов модели; уточнение взаимодействия с УПМ, документация этапа. Подробно технология составления формального описания СТС для каждого из способов имитации изложена в работе [1]. После составления формального описания СТС приступают к его проверке. Это первая главная проверка достоверности будущей ИМ СТС. Для обеспечения контроля правильности функционирования ИМ вводят классические модели, достоверность которых доказана. Они фигурируют в ИМ в виде составных частей. На вход таких моделей поступают данные, вычисляемые в других частях ИМ, достоверность которых проверяется. Если результат работы классической модели окажется недостоверным, то считают, что предшествующая часть формального описания СТС также недостоверна. В процессе проверки достоверности ИМ необходимо ответить на следующие вопросы: позволяет ли модель решить поставленные задачи моделирования, насколько полна предложенная схема ИМ и отражает ли она фактическую последовательность развития процессов в реальной СТС. Необходимо провести анализ каждой функции ИМ и убедиться, что она нашла своё отражение в формальном описании СТС. В том случае, когда уравнения получены на основании анализа опытных данных, необходимо провести выборочную проверку согласия уравнений с исходной информацией, по которой они получены. Для уравнений, полученных теоретическим путем, необходимо провести вычисления в нескольких контрольных точках с целью определения прием-

лемости результатов. Для дополнительной проверки уравнений желательно провести анализ размерностей и масштабов переменных системы. Результатом этапа является проверенное формальное описание исследуемой системы на выбранном языке формализации.

Составление текста имитационной модели является переходным этапом от формализации к этапу программирования. Когда формальное описание СТС составлено в виде набора агрегатов или в виде СМО, данный этап является необязательным. Зачастую исследователя не удовлетворяет состав стандартной статистики, формируемый выбранными средствами автоматизации моделирования. Кроме того, многие СТС ставят перед исследователем множество дополнительных вопросов в ходе формализации. Сюда входят следующие вопросы реализации ИМ: формирование элементов модели и отработка синхронизации взаимодействия частей компонентов модели друг с другом в модельном времени; организация сбора статистики в динамике имитации процессов в СТС; задание начальных условий моделирования; планирование процесса имитации отдельных вариантов системы; проверка окончания моделирования; обработка результатов имитации. Все эти действия весьма трудоёмки. Только после их решения исследователь получает ИМ на бумаге (описание модели процессов).

На этапе ***программирования ИМ*** выполняются следующие действия. Во-первых, составляется план создания и использования программной модели. В плане указывается тип ЭВМ, средство автоматизации моделирования, примерные затраты памяти и времени на создание ИМ. Во-вторых, приступают к программированию алгоритмов ИМ. Не существует существенных отличий создания программы ИМ от обычной автономной отладки программных модулей большой программы или пакета программ. В соответствии с текстом производится деление модели на блоки и подблоки. Объём отладочных работ существенно возрастает из-за того, что для каждого модуля необходимо создать и отладить ещё имитатор внешнего окружения. Весьма существенно выверить реализацию функций модуля в модельном времени. Завершаются работы при автономной отладке компонент модели подготовкой форм представления вводных и выходных данных моделирования. Далее переходят ко второй проверке достоверности ИМ, в ходе которой устанавливается соответствие операций в программе и описании модели. Для этого производится обратный перевод программы ИМ в схему модели, который позволяет найти грубые ошибки статики модели СТС. После исключения грубых ошибок ряд блоков программы ИМ объединяется и начинается комплексная отладка модели с использованием тестов. В этот процесс постепенно вовлекается всё большее число блоков модели. Комплексная отладка ИМ намного сложнее отладки пакетов программ, поскольку ошибки имитации динамики моделирования найти значительно труднее вследствие квазипараллельной работы различных компонентов ИМ. Следующим действием является составление технической документацуии на ИМ СТС. Результатом этапа к моменту окончания комплексной отладки программы ИМ являются следующие документы: описание ИМ, описание программы модели с указанием средств программирования, полная схема программы модели; полная запись программы ИМ на языке моделирования; результаты отладки (доказательство достоверности программы модели); описание входных и выходных величин; инструкция по работе с программой модели.

Для ***проверки адекватности ИМ*** объекту имитации после составления формального описания СТС составляются планы проведения натурных экспериментов с прототипами СТС. Если прототип СТС отсутствует, то можно использовать систему вложенных ИМ, отличающихся друг от друга степенью детализации имитации одних и тех же явлений. Тогда более детальная ИМ служит в качестве прототипа для обобщённой ИМ. Параллельно с отладкой ИМ осуществляется серия натурных экспериментов на реальной СТС, в ходе которых накапливаются контрольные результаты. Имея в своём распоряжении контрольные результаты и результаты отладки ИМ, исследователь проверяет адекватность ИМ объекту моделирования. Как видно из рис. 1.7, при обнаружении ошибок на этапе отладки, устранимых только на предыдущих этапах, может иметь место возврат на предыдущий этап.

Испытание модели является важным этапом создания ИМ. Необходимо убедиться в правильности динамики развития алгоритма моделирования СТС в ходе имитации его функционирования (т. е. провести верификацию модели). Затем необходимо определить степень адекватности ИМ и объекта исследования. Под адекватностью программы ИМ реальной СТС понимают совпадение с заданной точностью векторов характеристик поведения объекта и модели. При отсутствии адекватности проводят калибровку ИМ («подправляют» характеристики алгоритмов компонентов модели). Наличие ошибок во взаимодействии компонентов ИМ возвращает исследователя к этапу создания ИМ на бумаге. Возможно, что в ходе формализации исследователь слишком упростил процессы и исключил из рассмотрения ряд важных сторон функционирования СТС, что привело к неадекватности ИМ. В этом случае исследователь должен вернуться к этапу формализации СТС (см. рис. 1.7). В тех случаях, когда выбор способа формализации оказался неудачным, исследователю необходимо повторить этап составления концептуальной модели с учётом новой информации и появившегося опыта. Наконец, когда у исследователя оказалось недостаточно информации об объекте, он должен вернуться к этапу составления содержательного описания СТС и уточнить его с учётом результатов испытания предыдущей ИМ СТС.

На этапе ***исследования свойств*** ИМ оцениваются точность имитации явлений, устойчивость результатов моделирования, чувствительность критериев качества к изменению параметров ИМ. Получить эти оценки в ряде случаев бывает весьма сложно, однако без успешных результатов этой работы

доверия к ИМ не будет. Точность имитации явлений обычно представляет собой оценку влияния стохастических элементов на функционирование ИМ СТС. Устойчивость результатов моделирования характеризуется сходимостью контролируемого параметра моделирования к определённой величине при увеличении времени моделирования варианта СТС. Стационарность режима моделирования характеризует собой некоторое установившееся равновесие процессов в модели СТС, когда дальнейшая имитация бессмысленна, поскольку новой информации из ИМ исследователь не получит и продолжение имитации приведёт к увеличению затрат машинного времени. Поэтому необходимо разработать процедуру проверки момента достижения стационарного режима имитации. Чувствительность ИМ представляется величиной минимального приращения выбранного критерия качества, вычисляемого по статистикам моделирования, при последовательном варьировании параметров моделирования на всём диапазоне их изменения.

Этап ***эксплуатации ИМ*** начинается с составления плана эксперимента, позволяющего исследователю получить максимум информации при минимальных усилиях на вычисление. Составляется статистическое обоснование плана эксперимента. Планирование эксперимента представляет собой процедуру выбора числа и условий проведения опытов, необходимых и достаточных для решения поставленной задачи с требуемой точностью. Стремятся минимизировать общее число опытов на ИМ с одновременным варьированием всеми переменными. Выбирают такую стратегию ИЭ, которая позволяет принимать обоснованную стратегию с помощью процедур принятия решений после каждой серии экспериментов на ИМ.

Этап ***анализа результатов моделирования*** завершает технологическую цепочку этапов создания и использования ИМ. Получив результаты моделирования, исследователь приступает к интерпретации результатов. Здесь возможны следующие циклы ИЭ. В первом цикле ИЭ в ИМ заранее предусмотрен выбор вариантов СТС путём задания начальных условий имитации для машинной программы ИМ. Во втором цикле ИЭ модель модифицируется на языке моделирования, и поэтому требуется повторная трансляция и редактирование программы ИМ. Возможно, что в ходе интерпретации результатов исследователь установил наличие ошибок либо при создании модели, либо при формализации объекта моделирования. В этих случаях осуществляется возврат на этап построения описания ИМ или на составление концептуальной модели СТС соответственно (третий цикл ИЭ). Результатом этапа интерпретации данных ИЭ являются рекомендации по проектированию или модификации СТС. На их основе исследователи приступают к принятию простых решений. На интерпретацию результатов ИЭ оказывают существенное влияние изобразительные возможности средств моделирования на ЭВМ. Можно рекомендовать использовать графики как наиболее удобное изобразительное средство. В ряде случаев применяются дисплеи и средства

для создания кинофильмов о моделируемом явлении. С их помощью можно помочь исследователю наблюдать и изучать явление в замедленном или убыстренном темпе по сравнению с реальной скоростью протекания исследуемых процессов.

В конечном итоге после выполнения всех перечисленных выше итерационных этапов имитации исследователь либо окажется удовлетворённым результатами моделирования и будет их учитывать при проектировании СТС, либо забракует проектируемую систему и сформулирует техническое задание на разработку новой архитектуры СТС.

1.5. Автоматизация этапов создания имитационных моделей СТС

Для того чтобы оперативным образом реализовать ИМ СТС на ЭВМ, нужен аппарат моделирования, который в большинстве существующих универсальных алгоритмических средств программирования отсутствует или недостаточно развит. В нём должны быть предусмотрены:

– способы организации данных, обеспечивающие простое и эффективное моделирование;

– удобные средства формализации и воспроизведения динамических свойств моделируемой СТС;

– возможности имитации стохастических систем, включающие в себя процедуры моделирования и анализа случайных процессов.

Такой аппарат должен иметь специализированные средства имитации, в состав которых входят: язык описания объекта моделирования; средства обработки на ЭВМ языковых конструкций (компилятор, транслятор или интерпретатор); УПМ, осуществляющую имитацию во времени; набор стандартных программных средств, реализующих дополнительные возможности по организации ИЭ.

Применение специализированных средств имитации имеет следующие преимущества перед использованием универсальных средств программирования: меньшие затраты времени на программирование; более эффективные методы выявления ошибок имитации; краткость, точность выражения понятий, характеризующих имитационные процессы; возможность для некоторой предметной области исследований заранее построить стандартные компоненты, которые могут использоваться при построении ИМ; автоматическое формирование типов данных, соответствующих принятому способу имитации и необходимых в ходе ИЭ; удобство накопления и представления выходных данных.

Специализированные языки имитации делятся на две группы, соответствующие двум видам имитации: для непрерывных и дискретных процессов. Группа языков непрерывного ИЭ делится, в свою очередь, на языки аналого-

вого моделирования и языки, которые применяются для решения систем детерминированных замкнутых непрерывных систем. В языках аналогового моделирования используется покомпонентная эмуляция аналоговых и гибридных ЭВМ. Например, сумматор ЭВМ заменяется кодом операции суммирования, интегратор – кодом операции интегрирования и т. д. Взаимодействие между этими дополнениями функциональных компонентов описывается с помощью блочно-ориентированного языка аналогично тому, как коммутационная панель аналоговой ЭВМ связывает компоненты аналоговой вычислительной программы. При этом эмулируются и структура и элементы аналоговой или гибридной ЭВМ.

Для моделирования явлений, представляемых переменными, непрерывными по диапазону своих значений, но дискретными по времени свершения событий, удобен язык ДИНАМО [1].На языке ДИНАМО непрерывные процессы описываются соответствующими операторами, позволяющими представить зависимости в СТС в виде дифференциальных уравнений первого порядка. Вводятся два типа переменных: состояний и скорости. Переменные состояний используются для описания состояний объекта моделирования и условий перехода СТС в эти состояния. Переменные скорости позволяют исследователю описывать, как меняются эти состояния за некоторый отрезок времени. Язык ДИНАМО нашёл применение в тех случаях, когда исследователя интересуют информационные системы с обратной связью и он апробирует на ИМ методы решения задач в области управления и автоматического регулирования работы СТС.

В ряде случаев исследователь изучает причинно-следственные связи в СТС, и ему при этом необходимо использовать такие средства, как интегрирование, параметрический анализ непрерывной системы или графическая оптимизация её структуры. Для этой цели он строит имитатор непрерывной СТС на ЭВМ. Модель представляется состоящей из взаимосвязанных функциональных элементов и эмулируется на цифровой ЭВМ. Наиболее удобным средством для этих целей является язык 360\ sistem CSMP [2]. С помощью этого языка специалисты, которые ранее применяли аналоговую секцию, легко могут перейти к работе на цифровой ЭВМ. Описание модели СТС на этом языке представляет собой совокупность фраз, задающих правила коммутации общепринятых элементов аналогового моделирующего устройства: интеграторов, сумматоров, делителей, инверторов и т. д. Поведение этих элементов реализуется на ЭВМ соответствующими стандартными функциями. Для моделирования типовых элементов аналогового устройства у исследователя имеется возможность самому строить набор специальных функций применительно к его конкретной задаче. Допускаются также вставки в текст ИМ на алгоритмическом языке для вычисления начальных условий, анализа результатов, выбора нового варианта значений параметров моделирования.

В мире разработано несколько сотен языков моделирования дискретных

систем. В нашей стране получили распространение следующие группы языков моделирования (в зависимости от способа организации квазипараллелизма):

– ориентация на просмотр активностей реализуется в языке SMPL;

– составление расписания событий организуется в языках GASP, СИМСКРИПТ, CMPL, ДИСМ, МОДИС-BZC;

– процессный способ имитации реализован в языках SOL, ASPOL, SIMULA, ДИС, СЛЕНГ, АЛСИМ, СИМУЛА-67, МПЛ/1, СКИФ, МК PLSIM;

– организация взаимодействия транзактов обеспечивается языками GPSS, ИМСС, CSS, АСИМ;

– агрегативный метод имитации обеспечивается языками систем моделирования АИС И САПАС.

В языках моделирования систем моделирования НЕДИС и DISLIN реализуется описание непрерывно-дискретных процессов.

К классификации языков моделирования можно подходить по-разному. Иногда к языкам предъявляют требования, определяемые предметной областью их использования. В работе [1] содержится достаточно подробная классификация языков моделирования по описательным возможностям. Можно выделить соответствие языков моделирования рассмотренным ранее способом формализации реальных объектов. Поскольку универсальные языки моделирования определяют изобразительные возможности представления алгоритмов, то естественна классификация по способам реализации систем моделирования СМ. При этом классификация на такой основе фактически представляет собой сравнение алгоритмических возможностей языков моделирования.

По способу реализации различаем СМ, использующие универсальную базовую систему программирования, и специально построенные СМ на моделирование определённого класса СТС. В первом случае СМ представляет собой расширение возможностей универсальной системы программирования. Обычно в качестве базового языка программирования используется ФОРТРАН, АЛГОЛ, PL/1 или С. Так, алголоподобную базу реализации имеют языки в таких СМ, как ДИС, СЛЕНГ, СИМУЛА, АЛСИМ, ASPOL, SOL, MICIC. На базе Фортрана реализованы языки в СМ DISLIN, ДИНАМО, ИМСС, СКИФ, СИМСКРИПТ, GASP. Язык PL/1 в качестве базового используется в СМ ДИСМ, SMPL, МПЛ/1, MKPLSIM. В ряде случаев для моделирования разработаны СМ с собственными языками моделирования: НЕДИС, CSS, GPSS, АИС, АСИМ. Практически у всех перечисленных ранее СМ название языка моделирования совпадает с названием СМ. На ранних этапах разработки СМ считалось более эффективным использовать базовый язык системы программирования, расширив его до уровня языка моделирования. Однако с развитием моделирования в ряде предметных областей исследования оказалось, что эффективнее разрабатывать языки моделирования

со своими собственными системами компилирования. Этот способ более перспективен из-за отсутствия ограничений, налагаемых на описание модели, полученных с помощью универсальных языков программирования.

При выборе средств автоматизации моделирования исследователя обычно интересуют две группы свойств: изобразительные возможности представления алгоритма и удобство работы при использовании программного продукта. Из изобразительных возможностей языка моделирования исследователя интересуют три аспекта: состав операторов моделирования, наличие доступа к модельным характеристикам, возможности универсального базового языка программирования для реализации алгоритмов. Удобство работы СМ определяется технологией её использования.

Инструментальные возможности средств автоматизации моделирования обычно обусловлены:

-описательными возможностями языка моделирования в части отражения особенностей способа имитации;

– возможностью (или отсутствием) доступа к модельным характеристикам;

– возможностью рекурсии в описании модели;

– использованием универсальных средств программирования, особенно при реализации арифметических операций.

Для более полного ознакомления с описательными возможностями существующих языков моделирования можно рекомендовать работу [2].

Технологические возможности СМ определяются:

– способом компилирования программ модели;

– способом организации ввода-вывода информации в программе модели;

– возможностью выдачи сообщения пользователю в ходе имитации;

– наличием средств организации отладки и диагностики программы ИМ;

– наличием средств организации сбора статистики и возможностью управления ими;

– составом средств обработки результатов моделирования;

– наличием специальных средств испытания ИМ;

– способностью управления имитационным экспериментом;

– наличием средств автоматизации представления и интерпретации результатов имитации;

– типом ЭВМ и характеристиками организации вычислительного процесса.

Сравнивать между собой можно только СМ, реализующие один и тот же способ имитации. Иначе достоинства способа формализации на конкретной ИМ можно отнести за счёт языка моделирования СМ.

2. ИСПЫТАНИЕ И ИССЛЕДОВАНИЕ СВОЙСТВ МАТЕМАТИЧЕСКИХ МОДЕЛЕЙ СТС

2.1. Процедуры верификации математических моделей СТС

Верификация модели обычно состоит в проверке её соответствия тому смыслу, который был заложен при её разработке. Исследователь должен получить гарантию того, что созданная ИМ во всех ситуациях будет правильно отражать поведение реальной системы. С помощью верификации устанавливается верность логической структуры модели. Обычно верификация выполняется в ходе комплексной отладки программы ИМ на реальном потоке данных. Она состоит в доказательстве утверждений соответствия алгоритма её функционирования замыслу моделирования путём формальных и неформальных исследований реализованной программы ИМ.

Формальные методы верификации программы ИМ включают в себя: использование специальных препроцессоров и постпроцессоров в качестве «читателей» программ; замену некоторых стохастических элементов ИМ детерминированными и проверку на «ожидаемость» результатов моделирования; использование теста на «непрерывность» процесса имитации. Препроцессоры и постпроцессоры анализируют текст программы ИМ путём доказательства логических утверждений и выдают исследователю сообщение о возможном наличии логических ошибок в тексте ИМ. Если при замене стохастических элементов модели детерминированными получаются далёкие от ожидаемых ответы программы ИМ, то это означает, что программная реализация ИМ крайне неудачна. В ходе тестирования на «непрерывность» имитации проверяется соответствие выходных характеристик воздействиям на входе ИМ по всему диапазону значений параметров модели.

Неформальные исследования ИМ представляют собой чаще всего комплексную отладку программы ИМ. Важна возможность использования дополнительных средств отладки в виде набора служебных подпрограмм, входящих в состав стандартного набора используемой СМ. В него входят: различные загрузчики рабочих программ, библиотечные и поддерживающие программы, генераторы данных, программы управления отладкой, средства редактирования данных и программных модулей. Эффективным средством комплексной отладки ИМ является трассировка компонентов ИМ. Возможны три вида организации трассировки: покомандная, фиксация переходов в алгоритмах компонента, регистрация выполнения только макрокоманд имитации. По окончании проверки соответствия информации алгоритму поведения компонента ИМ операторы отладки исключаются из текста ИМ. Существенно помогает при этом наличие библиотеки процедур у СМ и средств оперативной компоновки программы модели из библиотеки процедур. Мо-

дель наращивается по методу «этап за этапом», начиная с отладки двух компонентов ИМ и кончая отладкой взаимодействий всех остальных компонент модели. При каждом добавлении нового компонента ИМ необходимо проверить взаимодействие его с внешней средой. Таким образом, разработчик ИМ фактически создает последовательность вложенных программных моделей, и опыт отладки одной модели используется для улучшения последующей модели.

2.2. Оценка точности, чувствительности и устойчивости моделирования ИМ

После того как завершена верификация ММ СТС и исследователь получил некоторую уверенность в правильности алгоритма ИМ, необходимо исследовать точностные характеристики модели. Наиболее существенными, с нашей точки зрения, являются процедуры оценки: погрешности имитации, обусловленной наличием в ИМ генераторов псевдослучайных чисел; длины переходного процесса в модели, устойчивости результатов моделирования; чувствительности откликов ИМ к изменениям входных параметров модели.

В любой вероятностной ИМ СТС применяются генераторы моделирования псевдослучайных чисел. Любая СМ предоставляет несколько различных генераторов псевдослучайных чисел, каждый из которых использует базовый генератор моделирования равномерно распределённых чисел на интервале [0, 1]. Эти генераторы и сам вероятностный характер ИМ являются источником погрешности имитации. Для проверки качества этих генераторов применяются различные тесты. Большинство исследователей обычно применяют следующую процедуру проверки погрешности имитации с помощью базовых генераторов. В серединной точке области изменения управляющих параметров X организуется несколько прогонов ($k \geq 10$) ИМ с одними и теми же значениями X и задаваемых параметров G, но с разными начальными значениями базового генератора ξ_{Ok} (k – номер прогона). Для каждого k-го прогона ИМ вычисляются значения n-й компоненты вектора откликов Y_{nk}. В результате получают выборки значений откликов $\{Y_{nk}\}$, $k = 1, .., 10$. По этим выборкам определяют оценки математического ожидания и дисперсии ($\overline{Y}_n$, D_n). Определяется доверительный интервал нахождения истинного значения математического ожидания n-й компоненты отклика $Y_{n\text{и}}$. При этом допускают нормальность распределения отклонений $Y_{n\text{к}}$ от $Y_{n\text{и}}$. Поскольку объёмы выбора малы ($k < 30$), то для нахождения доверительных интервалов используется t-статистика:

$$t = (\overline{Y}_n - Y_{nH})\sqrt{(K-1)/D_n}\,, \qquad (2.1)$$

имеющая распределение Стьюдента. Задавшись уровнем значимости

$\alpha = 0{,}05$, мы с вероятностью 0,95 можем утверждать, что $Y_{n\text{и}}$ лежит в пределах

$$\overline{Y}_n - t_{0,05}\sqrt{D_n/(K-1)} \leq Y_{n\text{и}} < \overline{Y}_n + t_{0,05}\sqrt{D_n/(K-1)}\,, \qquad (2.2)$$

где $t_{0,05}$ – значение t-статистики при $(k-1)$ степенях свободы и уровне значимости $\alpha = 0{,}05$. Доверительный интервал для среднего значения n-й компонента вектора отклика (при $k = 10$ и $\alpha = 0{,}05$) можно записать в виде

$$\overline{Y}_n \pm dn = \overline{Y}_n \pm 0{,}753\ S_{1n}. \qquad (2.3)$$

Значение dn и определяет погрешность n-го компонента отклика модели. Затем из всех dn находят максимальное значение, которое в данном случае и будет верхней границей погрешности, связанной с вероятностным характером ИМ и использованием генераторов псевдослучайных чисел. Если эта погрешность окажется достаточно высокой, то исследователь обязан пересмотреть состав генераторов и процедуру их использования в ИМ СТС. В таких случаях можно рекомендовать выбрать большее число базовых генераторов с обязательным хранением для каждого из них своего очередного начального значения ξ_{Oi} для каждой ij-й активности ИМ. Для увеличения периода апериодичности генераторов рекомендуется использовать комбинированные алгоритмы, рассмотренные ранее в части 2.

В большинстве стохастических ИМ требуется некоторое время T_{O} для достижения ИМ необходимого установившегося состояния. Поэтому исследователь должен оценить T_{O} и позаботиться о его уменьшении или исключить из рассмотрения статистику, полученную до истечения этого времени. Наиболее часто используются следующие способы уменьшения влияния начального периода на динамику имитации процессов в СТС: использование длинных прогонов модели; исключение из рассмотрения начального периода прогона; выбор такого начального условия, которое ближе всего к типичному, и тем самым существенное уменьшение длительности переходного режима имитации.

Первый способ можно применять только в случае, если прогон ИМ не требует много машинного времени. При втором способе часть времени ЭВМ тратится бесполезно, и из-за сокращения объёма выборки увеличивается дисперсия отклика модели Y. При этом трудно установить момент окончания переходного режима. Для отделения переходного режима от стационарного у исследователя должна быть возможность наблюдения за моментом входа контролируемого параметра моделирования в стационарный режим. Такой контроль может быть выполнен на любом компоненте ИМ, который позже всего входит в стационарный режим. Необходимо задать временной интервал, по окончании которого происходит вычисление контролируемого параметра (например, длина очереди требований к этому компоненту). Проверку

можно выполнить по критерию Вилкоксона, который требует задания уровня значимости. С интервалом времени, равным Δt, проводится сравнение средних значений длин очереди l_k и l'_k ($k=\overline{1,m}$), где m – объём выборки; k – номера замеров длины очереди. С помощью критерия Вилкоксона проверяется гипотеза H_O об однородности выборок $\{l_k\}$ и $\{l'_k\}$. Предполагается, что обе выборки взаимно независимы и извлечены из одной и той же генеральной совокупности, и, следовательно, функции распределения величин l_k и l'_k одинаковы. Производится объединение случайных величин l_k и l'_k в один вариационный ряд в порядке возрастания их значений. В результате формируется последовательность типа

A	*A*	*B*	*A*	*A*	*A*	*B*	*B*	*B*	*B*	...	*A*	*B*	*B*
1	2	3	4	5	6	7	8	9	10		$N-2$	$N-1$	N

где буквами A и B обозначены члены вариационного рада, принадлежащие выборкам $\{l_k\}$ и $\{l'_k\}$ соответственно. Каждому члену объединённой выборки приписываются ранги r_q, обозначающие порядковые номера в объединённой выборке, $q=\overline{1,N}$; $N=2m$. На основании установленных рангов вычисляется значение статистики Вилкоксона:

$$W_l=\sum_{q=1}^{N} r_q \ . \qquad (2.4)$$

Гипотеза H_O об однородности выборок принимается при выполнении неравенства $\varpi_1(\alpha)\le W_l\le\varpi_2(\alpha)$. Здесь $\varpi_1(\alpha)$ и $\varpi_2(\alpha)$ – верхняя и нижняя границы статистики Вилкоксона при уровне значимости α $(0<\alpha<1)$, которые вычисляются из соотношений

$$\varpi_1(\alpha)=\frac{m(2m+1)-1}{2}-\psi\sqrt{\frac{m^2(2m+1)}{12}} \qquad (2.5)$$
$$\varpi_2(\alpha)=m(2m\text{-}1)-\varpi_1(\alpha),$$

где $\psi=arg\,\Phi(1+\alpha/2)$; $arg\ \Phi(x)$ означает функцию, обратную от функции нормального закона распределения. В ряде случаев исследователи строят графики изменения значений контролируемого параметра в модельном времени t_0. По виду этого графика определяют момент окончания переходного режима имитации.

Не всегда целью моделирования является исследование поведения СТС в стационарном режиме. Бывают случаи, когда предметом имитации является переходный режим реальной СТС. Может оказаться такая ситуация, когда не удаётся достичь стационарного режима, хотя результаты моделирования

имеют смысл только при достижении стационарности в СТС. Несмотря на то, что в такой ситуации трудно доверять результатам имитации, исследователь вынужден ими пользоваться, поскольку зачастую другого средства для изучения явления у него нет.

Под ***устойчивостью результатов имитации*** обычно понимают степень нечувствительности ИМ к изменению входных условий. Для этой цели часто используется апостериорная проверка ИМ. Она состоит в сравнении предсказаний модели при увеличении периода моделирования. Чем ближе структура ИМ к структуре СТС и чем выше степень её детальности, тем обширнее область пригодности ИМ. Однако может оказаться, что из-за чрезмерной подстройки к частным условиям устойчивость результатов имитации снижается. Обычно устойчивость результатов имитации оценивают дисперсией отклика. Если эта дисперсия при увеличении времени моделирования $T_м$ не увеличивается, значит, результаты моделирования устойчивы. Процедура такой оценки следующая. В модельном времени t_0 задаются шаг Δt контролирования показателя качества (отклика Y), число шагов h для контроля и экспертное значение изменения контролируемого параметра $\Delta Y_Э$. По достижении стационарного состояния оценивается амплитуда изменений параметра Y и вычисляется модуль изменения амплитуды ΔY как функции от h. Каждый раз при этом проверяется выполнимость неравенства $|\Delta Y|\le\Delta Y_Э$. Если на всём интервале исследования Y окажется в заданных пределах, то считают, что ИМ находится в устойчивом состоянии. Рост разброса контролируемого параметра ΔY и зависимость его значения от изменения h указывают на неустойчивый характер имитации исследуемого процесса.

Для ***исследования чувствительности*** ИМ исследователь, во-первых, должен установить диапазон изменения отклика модели Y при изменении каждого компонента вектора параметров X. В зависимости от диапазона изменения откликов Y определяется стратегия планирования экспериментов (ПЭ) на ИМ. Если при значительной амплитуде изменений некоторого компонента вектора параметров модели X отклик Y меняется незначительно, то это означает, что точность представления этого компонента в ИМ не играет существенной роли. Кроме того, в планировании ИЭ этот компонент не будет использоваться как основной. Если же отклик модели Y окажется высокочувствительным к изменению некоторого компонента вектора X, то это служит прямым указанием на необходимость представления его в модели с максимально возможной точностью.

Во-вторых, исследователь должен проверить зависимость отклика модели Y от изменения параметров внешней среды (G). Меняя характеристики вектора G в обе стороны на некоторую величину ΔG, исследователь оценивает диапазон изменений вектора откликов δY. Если δY незначителен, то требования к точности задания модели внешней среды могут быть несуще-

ственными. В противном случае измерение характеристики G и сами способы стабилизации значения G в ИМ должны быть высокоточными. Анализ поведения δY при колебаниях X и G иногда позволяет скорректировать алгоритмы ИМ в сторону их упрощения.

Обычно пространство значений параметра X задано и определяется целями моделирования и степенью осведомлённости исследователя о компонентах объекта моделирования. Определение чувствительности ИМ к изменениям параметров ИМ легче всего проводить в центральной точке пространства значений параметра. Выбор центральной точки осуществляется, как правило, на основании априорных суждений и носит неформальный характер. Для вычисления отклика применяются процедуры, описываемые самим пользователем на языке реализации ИМ, результаты работы которых затем уже используются следующей стандартной процедурой. Каждый q-й компонент вектора X отклоняется от значения его в центральной точке в обе стороны на длину выбранного интервала его изменения ($\min X_q$, $\max X_q$). Остальные компоненты вектора параметров X остаются неизменными и соответствуют центральной точке. Проводится пара модельных экспериментов и вычисляются отклики ИМ по n-му компоненту Y_{1n} и Y_{2n}, которые означают значения откликов при граничных значениях q-го компонента вектора параметров ИМ. Далее вычисляются приращения q-го компонента вектора параметров (δX_q) и n-го компонента вектора откликов (δY_n) соответственно по формулам

$$\delta X_q = \frac{(\max X_q - \min X_q)\cdot 2}{(\max X_q + \min X_q)}\cdot 100\ \%;$$

$$\delta Y_n = \frac{(Y_{2n} - Y_{1n})\cdot 2}{(Y_{2n} + Y_{1n})}\cdot 100\ \%. \qquad (2.6)$$

Изменение вектора откликов Y можно определять либо модулем вектора приращений (δY_n, $n = \overline{1, L}$, L – размерность вектора откликов), либо максимальным значением из δY_n. Чаще всего исследователи останавливаются на втором способе. Итак, чувствительность ИМ q-го компонента вектора параметров X определяется парой значений (δX_q, $\delta Y_q^{\circ} = \max\limits_n \delta Y_n$). Эта пара чисел показывает, на сколько процентов может измениться отклик ИМ при увеличении q-й компоненты вектора параметров на δX_q процентов. Затем подобным же образом поступают с остальными компонентами вектора параметров X. В результате получают множество пар значений $\{\delta X_q,\ \delta Y_q^{\circ}\}$; $q = \overline{1, h}$; где δX_q – разность вектора параметров X. Этой информации, как правило, бывает

достаточно для ранжирования компонент вектора параметров модели X по значению чувствительности вектора отклика ИМ. Если ИМ окажется малочувствительной по какой-либо S-й компоненте вектора параметров X_S, то зачастую исследователь не включает в план ИЭ изменение X_S, чем достигается экономия ресурса времени моделирования.

2.3. Проверка адекватности модели реальному объекту исследований

Проверка адекватности модели реальной СТС проводится чаще всего для случая, когда можно определить значение отклика системы в ходе натурных испытаний. Пусть известен отклик реальной системы Y_Q^* при нагрузке G, параметрах Q^* и $Y_Q^* = \varphi^*(Q^*, G^*)$. Модель представляет собой аппроксимирующую зависимость $Y_k = \varphi(Q_k, G)$, $k = \overline{1, N}$ (k и N – соответственно номер и число опытов на модели и реальной системе), найденную в ходе эксплуатации СТС по результатам наблюдений входных воздействий Q^*_k, $k = \overline{1, N}$, и выхода $Y^*_{Q_k}$ при заданных значениях G^*. Чаще всего исследователи проверяют адекватность модели по одному из трёх способов: сравнение средних значений, определение дисперсий отклонений откликов модели от среднего значения откликов системы; нахождение максимального значения абсолютных отклонений откликов модели от откликов СТС.

При первом способе проверяется гипотеза о близости средних значений каждого n-го компонента откликов модели $\overline{Y}_n$ известным средним значениям n-го компонента откликов СТС $\overline{Y}^*_{Q_k}$. Проводят N_1 опытов на СТС и формируют выборки значений $\{Y^*_{Q_{nk}}\}$, $k = \overline{1, N}_1$. Выполняют N_2 опытов на модели системы и получают по тем же n-м компонентам откликов модели выборки значений $\{Y_{nk}\}$; $k = \overline{1, N}_2$. Обычно стараются, чтобы объёмы выборок были одинаковы ($N_1 = N_2$), но в ряде случаев натурные эксперименты весьма дороги и поэтому $N_1 > N_2$. По выборкам вычисляются оценки математического ожидания и дисперсии откликов модели и системы с помощью следующих соотношений:

$$\overline{Y}_n = \frac{1}{N_2}\sum_{k=1}^{N_2} Y_{nk};\quad D_n = \frac{1}{N_2 - 1}\sum_{k=1}^{N_2}(Y_{nk} - \overline{Y}_n)^2;$$
$$\overline{Y}^*_{Q_n} = \frac{1}{N_1}\sum_{k=1}^{N_1} Y^*_{Q_{nk}};\quad D^*_n = \frac{1}{N_1 - 1}\sum_{k=1}^{N_1}(Y^*_{Q_{nk}} - \overline{Y}^*_{Q_n})^2. \qquad (2.7)$$

Основой проверки гипотезы является разность $E_{1n}=(\overline{Y}_n-\overline{Y}^*_{Q_n})$, оценкой дисперсии которой будет

$$D_{on}=((N_1-1)D_n+(N_2-1)D^*_n)/(N_1+N_2-2). \quad (2.8)$$

Величины E_{1n} и D_{on} являются статистиками независимыми, поэтому можно использовать t-статистику:

$$t_n=(\overline{Y}_n-\overline{Y}^*_{Q_n})\sqrt{\frac{N_1N_2}{D_{\rho n}(N_1+N_2)}}. \quad (2.9)$$

При числе степеней свободы $\nu=N_1+N_2-2$ и уровне значимости $\alpha=0{,}05$ по таблицам распределения Стьюдента находят критическое значение ($t_{кр}$). Если выполняется неравенство $t_n\le t_{кр}$, то гипотеза о близости средних значений n-го компонента откликов модели и СТС принимается. Только при близости откликов по всем компонентам векторов Y и Y^*_Q можно говорить об адекватности модели и СТС по первому способу.

При втором способе для каждой n-й компоненты откликов проверяется гипотеза о значимости различий оценок двух дисперсий D^*_n и D_{on}. Первая дисперсия D^*_n определяется из соотношения (2.7), для нахождения второй дисперсии используют формулу

$$D_{on}=\sum_{k=1}^{N_2}(Y_{nk}-\overline{Y}_{Q_n})^2\Big/(N_2-1). \quad (2.10)$$

Для сравнения дисперсий составляется F-статистика: $F=D_{on}/D^*_n$ и проверяется выполнимость неравенства $D_{on}\ge D^*_n$. По таблицам распределения Фишера находят $F_{кр}$ (при уровне значимости $\alpha=0{,}05$, числе степеней свободы $\nu_1=\nu_2=N_2$; стараются достичь равенства объемов выборок $N_1=N_2$). Выполнение неравенства $F>F_{кр}$ указывает на то, что гипотеза о значимости различий двух оценок дисперсий принимается и отсутствует адекватность между n-ми компонентами откликов модели и реальной системы. Если отсутствует адекватность модели хотя бы по одной из компонент, то считают, что модель неадекватно отображает реальную систему.

При третьем способе проверяется соответствие отклонений откликов модели и реальной СТС по каждой компоненте. Отклонения этих откликов должны быть не более заданной величины. Этот способ применяется при проверке адекватности моделей систем управления. Для каждой n-й компоненты формируют две выборки: $\{Y_{nk}\}$ и $\{Y^*_{Q_{nk}}\}$; $k=\overline{1,N_1=N_2}$. По выборкам

определяют вектор отклонений откликов модели от откликов реальной системы в процентах. Компонентами этого вектора являются

$$\delta Y_n=\max_k\frac{\left|Y_{nk}-Y^*_{Q_{nk}}\right|}{\overline{Y}^*_{Q_n}}\cdot 100\,\%, \quad (2.11)$$

где $\overline{Y}^*_{Q_n}$ – среднее значение n-й компоненты отклика систем, определяемое по формуле (2.7).

Таким образом, δY_n представляет собой максимальное значение отклонений n-го компонента откликов модели в N_2 модельных экспериментах от соответствующих окликов реальной СТС, нормированное средним значением n-го компонента отклика СТС, принимаемого в качестве истинного значения. Задавшись процентным значением допустимого отклонения каждого компонента откликов модели от откликов системы $\delta Y_д$, проверяют выполнимость неравенства $\delta Y_n\le\delta Y_д$. Невыполнение данного неравенства хотя бы по одному компоненту вектора оклика модели ставит под сомнение адекватность модели реальной СТС из-за ограничений и предложений, положенных в основу ИЭ.

Результаты испытания модели СТС также представляют собой важный документ. Подробно описывается план проверки модели, согласованный с лицом, принимающим проектные решения. Приводятся тесты и входные данные для испытания ИМ. По результатам испытания ИМ составляется пояснительная записка, где указываются размеры области изменения параметров ИМ, в которой модель даёт удовлетворительные результаты.

Убедившись в пригодности ИМ для решения поставленных задач, составляют документы для обучения заказчика работе с ИМ. В них должно быть приведено описание процедуры привлечения пользователя к работам с ИМ. Для этого составляются: руководство пользователя ИМ; руководство аналитика СТС; резюме исполнителя, облегчающее интерпретацию результатов лицу, принимающему решение.

Этап моделирования СТС тщательно документируется и сопровождается большим числом иллюстративного материала. Все выводы оформляются в виде отчётов, которые составляют неотъемлемую часть документации этапа эксплуатации и могут служить источником для разработки новой ИМ СТС.

3. ОСНОВЫ ПЛАНИРОВАНИЯ ЭКСПЕРИМЕНТОВ С МОДЕЛЯМИ СТС

3.1. Особенности планирования и организации имитационных экспериментов

После того как установлены цели эксперимента и определена система автоматизации ИЭ, полезно провести предварительное планирование ИЭ. Необходимо иметь подробный план ИЭ для целенаправленного и эффективного получения результатов исследований. Плановые ограничения на время ИЭ и затраты должны быть приведены в соответствие с имеющимися в распоряжении исследователя ресурсами. Чем дороже и сложнее эксперимент, тем большее внимание следует обращать на планирование ИЭ. Зачастую ограничения в ресурсах опытов настолько жёсткие, что не позволяют воспользоваться традиционными статистическими методами. Обычно используются три типа экспериментов: сравнение средних и дисперсий разных альтернатив; определение важности учёта или значимости влияния переменных и ограничений, наложенных на эти переменные; отыскание оптимальных значений на некотором множестве возможных значений переменных.

Эксперименты первого типа являются, как правило, однофакторными и довольно простыми. Исследователь решает вопросы о размере выборки, начальных условиях, наличии или отсутствии автокорреляции. Экспериментам второго типа посвящено большинство книг по планированию и анализу их результатов. Основными методами истолкования результатов этих экспериментов являются дисперсионный и регрессионный анализы. Третий тип экспериментов обычно предполагает использование последовательных методов построения экспериментов. С методиками планирования экспериментов второго и третьего типов рекомендуется ознакомиться в учебном пособии авторов «Основы регрессионного анализа и планирования экспериментов» [3]. Поэтому рассмотрим только технологию организации экспериментов первого типа.

На этапе эксплуатации ИМ исследователю необходимо осуществить обработку результатов ИЭ и представление этих результатов в максимально информативной форме. Любая ИМ не имеет ценности до тех пор, пока она не будет использована теми, для кого разработана. Для большинства руководителей разработки СТС представляют интерес только их проблемы. Поэтому результаты ИЭ должны быть приемлемы для них по критерию надёжности и полезности этой информации для проектирования. Поэтому для создателей ИМ важно уметь представить результаты ИЭ в наглядной форме, широко применяя графические способы анализа данных моделирования.

ИМ должна быть понятной заказчику-пользователю, способной давать ответы на его вопросы, формировать информацию для последующего проектирования или модификации СТС и, наконец, недорогой при её применении. Подача результатов столь же важна, сколь и само проведение работы. Все традиционные методики обработки результатов наблюдений приемлемы и для анализа результатов имитации. С нашей точки зрения, наиболее важными являются следующие вопросы представления результатов ИЭ: исключение резко отклоняющихся значений при натурных экспериментах для получения исходной информации моделирования; выбор системы координат при графическом представлении данных имитации; анализ регрессионной зависимости функции откликов от параметров моделирования.

3.2. Определение требуемого размера выборки при планировании ИЭ

При ограниченном ресурсе времени моделирования первейшей задачей исследования является получение ответа на вопрос: как много выборочных значений следует взять во время моделирования, чтобы обеспечить достаточную статистическую значимость. Поскольку разбросы выборочных значений случайны, то обусловленная ими неточность результата ИЭ определяется размером выборки значений отклика. На практике размер выборки обычно является функцией количества средств, отпущенных на ИЭ.

Многие методы анализа используют предположение о независимости и нормальном распределении значений откликов модели. Это предположение основано на применении центральной предельной теоремы теории вероятностей. Действительно, часто отклик представляет собой сумму большого числа небольших эффектов, и переменная отклика ИМ СТС является результатом аддитивного действия большого числа случайных переменных. Если учесть, что каждое выборочное значение представляет собой также сумму большого числа небольших эффектов, то центральная предельная теорема применима, и есть основание предполагать, что приближённо отклик имеет нормальное распределение. В условиях применимости этой теоремы и при отсутствии автокорреляции можно использовать для определения размера выборки, необходимой для оценивания параметров с заданной точностью, правило автоматической остановки. Оцениваемыми параметрами могут быть средние значения и среднеквадратичные отклонения совокупности.

Правило «автоматической обстановки» основано на методе доверительных интервалов. Этот метод предполагает задание точности представления dn математического ожидания μ_n или b_n дисперсии σ_n^2 у n-й компоненты вектора отклика Y и уровня значимости α, гарантирующего попадание μ_n, σ_n^2

внутрь интервалов ($\overline{Y}_n \pm d_n$), ($D_n \pm b_n$) с вероятностью $\alpha\mu_n(1-\alpha)$. Здесь $\overline{Y}_n$ и D_n представляют собой среднее значение и дисперсию, вычисленные по выборке объёма N, и являются оценками соответственно μ_n и σ_n^2.

В ходе испытания и исследования свойств ИМ исследователь определяет векторы точностей (d_1, ..., d_n, ..., d_L) или (b_1, ..., b_n, ..., b_L) представления компонент вектора отклика Y. Выполнение правила «автоматической остановки» представляет собой итеративную процедуру, суть которой состоит в следующем. До начала проведения серии экспериментов известны уровень значимости α и векторы точностей представления компонент вектора отклика b_n или d_n. Из априорных сведений или опыта устанавливается количество начальных экспериментов N_1, необходимых для получения выборок значений компонент откликов модели $\{Y_{nk}\}$; $n = \overline{1,L}$; $k = \overline{1,N_1}$. Если исследователь предполагает, что в ходе исследования свойств ИМ число опытов было слишком большим, то в качестве начального значения полагают $N_1 = 5$. Далее алгоритм выбора числа экспериментов N состоит из следующих шагов.

Шаг 1. По выборке $\{Y_{nk}\}$ по формулам (2.7) находят средние значения $\overline{Y}_n$ и дисперсии D_n.

Шаг 2. Для очередного номера n компоненты вектора откликов модели Y определяют достигнутую точность оценок $\overline{Y}_n$ и D_n при выполнении N_1 экспериментов. Здесь возможны следующие случаи.

Если выборка малого объёма ($N_1 \le 30$), то для вычисления доверительного интервала используют t-статистику, имеющую распределение Стьюдента $d_{1n} = t_{\text{кр}}\sqrt{D_n/(N_1-1)}$, где $t_{\text{кр}}$ – критическое значение t-статистики при (N_1-1) степенях свободы и заданном уровне значимости α.

Если размер выборки большой ($N_1 > 30$), то для вычисления доверительного интервала μ_n используют двухстороннюю статистику нормированного нормального распределения $d_{1n} = Z_{\alpha/2}\sqrt{D_n/N_1}$, где $Z_{\alpha/2}$ – значение нормированного нормального распределения при уровне значимости $\alpha/2$.

Если нормальность Y_n предположить нельзя, но N_1 большое, то применяют неравенство Чебышева

$$P\{|\overline{Y}_n - \mu_n| \ge h\,\sigma_n/\sqrt{N_1}\} \le 1/h^2, \qquad (3.1)$$

где h – некоторая наперёд заданная константа, означающая число среднеквадратичных отклонений, удовлетворяющих исследователя. В этом случае доверительный интервал можно с достаточной точностью вычислить по формуле

$$d_{1n} = \sqrt{D_n/N_1(1-\alpha)} \qquad (3.2)$$

При оценивании дисперсии задача отыскания оценки D_n с достоверностью $(1-\alpha)$ имеет вид

$$P\{(1-b_n)\sigma_n^2 \le D_n \le (1+b_n)\sigma_n^2\} = 1-\alpha\,. \qquad (3.3)$$

Поскольку N_1 достаточно велико, то эту статистику аппроксимируют нормальным распределением и получают следующую формулу для определения достигнутой точности оценки σ_n^2 в N_1 опытах:

$$b_{1n} = Z_{\alpha/2}\sqrt{2/(N_1-1)}\,, \qquad (3.4)$$

где $Z_{\alpha/2}$ – значение нормированного нормального распределения при заданном уровне значимости $\alpha/2$.

Шаг 3. Сравнивают достигнутую точность d_{1n} или b_{1n} оценок μ_n или σ_n^2 в N_1 опытах с заданным значением d_n или b_n. Если выполняется неравенство $d_{1n} \le d_n$ или $b_{1n} \le b_n$, то отмечают, что по n-й компоненте вектора откликов модели Y достигнута требуемая точность оценки в N_1 экспериментах. Если данное неравенство не выполняется, то переходят к шагу 4. При выполнении данного неравенства переходят к шагу 5.

Шаг 4. Выполняют ещё один модельный эксперимент. N_1 увеличивают на единицу ($N_1 \equiv N_1 + 1$) и переходят к шагу 1.

Шаг 5. Проверяют, все ли компоненты вектора откликов проверены на удовлетворение точности оценки μ_n или σ_n^2. Если проверена достижимость точности по всем компонентам вектора точностей (d_1, ..., d_L) или (b_1, ..., b_L), то эксперименты завершаются. В противном случае меняют номер компоненты вектора откликов модели и переходят к шагу 2.

Достижимость заданной точности d_n или b_n оценки некоторой статистики (или группы статистик) может быть одним из условий окончания ИЭ.

3.3. Определение интервалов изменения параметров ИМ и исключение источников ошибок имитации

В любом ИЭ важно определить интервалы изменения параметров ИМ. До начала имитации необходимо найти предельные значения статистик моделирования. Существуют три основных фактора, влияющих на выбор интервалов изменения параметров модели: необходимость получения одинаковой относительной точности статистик на разных участках области изменения параметров ИМ; характер функции отклика; назначение ИЭ.

Если анализ погрешностей показывает, что на каком-то участке моделируемого процесса данные вызывают сомнения, то в ходе имитации на этом

участке необходимо увеличить частоту фиксации данных. Необходимо стремиться к тому, чтобы точность функции отклика модели была на всех участках одинаковой. Для этого необходимо соответствующим образом спланировать эксперимент. Если известно, что в ИЭ имеются особенности, которые можно обнаружить при получении данных в регулярной последовательности, используют классический последовательный план. Все параметры ИМ полагают постоянными, а один из них изменяют по всей области значений. Выбор интервалов между соседними значениями осуществляется с учётом баланса точностей. Аналогично изменяют второй параметр, третий и т. д.

Затем составляют план изменения параметров в ИЭ. В теории планирования экспериментов компоненты вектора параметров моделирования X называют факторами, а их значения называют уровнями факторов. Перед планированием ИЭ устанавливаются границы области определения факторов. Необходимо при этом учитывать следующие типы ограничений: принципиальные, которые неустранимы из-за их физической сущности; обусловливаемые технико-экономическими соображениями; связанные с условиями проведения ИЭ. Каждая комбинация уровней факторов является точкой в многомерном пространстве параметров, называемым факторным пространством. Построение плана эксперимента сводится к выбору экспериментальных точек, симметричных относительно основного уровня.

В зависимости от вида функции отклика $Y=\varphi(X, G)$ исследователь может располагать различными (априорными) сведениями об области наилучших значений Y. Априорная информация об этой области существенно влияет на выбор основного уровня по каждому фактору. Если у исследователя имеются сведения о координатах только одной точки и нет информации о границах факторов, то ему ничего не остаётся, как рассматривать эту точку в качестве основного уровня. Когда границы изменения факторов известны исследователю и он знает, что наилучшие значения X находятся внутри области изменения факторов, то в качестве основного уровня он может выбрать любую из точек факторного пространства. Хуже обстоит дело, когда известная точка лежит на границе факторного пространства. В этом случае исследователю приходится выбирать основной уровень со сдвигом от наилучших значений Y. Если же исследователю известно, что имеется несколько наилучших значений Y, то в качестве основного уровня он может выбрать любую случайную точку внутри факторного пространства. Наконец, исследователю до постановки ИЭ известна подобласть в факторном пространстве, где исследуемый процесс протекает оптимально и эксперимент ведётся только для уточнения места этого оптимума. Тогда в качестве основного уровня он выбирает центр этой подобласти.

Назначение ИЭ также существенно влияет на методику выбора интервалов изменения параметров модели. Если предметом имитации является поиск узких месть в функционировании СТС или выбор гипотез о механизме явлений, то применяется многоуровневое факторное планирование. В этом случае на выбор уровней факторов влияет относительная точность фиксации статистик имитации на разных участках изменения параметров ИМ. Когда предметом моделирования является либо поиск оптимальных условий функционирования СТС, либо выбор существенных для управления СТС параметров, либо нахождение вида функции, аппроксимирующей поведение СТС, то задача сводится к выбору для каждого фактора трёх его уровней (нижнего, основного, верхнего). В этом случае выбирается интервал варьирования фактора (число, прибавление которого к основному уровню даёт верхний уровень фактора, а вычитание – нижний уровень фактора). На выбор интервалов варьирования факторов накладываются ограничения сверху и снизу. Интервал варьирования не может быть меньше той погрешности, с которой исследователь фиксирует уровни, чтобы они были различимы. Но интервал варьирования не может быть настолько большим, чтобы эти уровни оказались за пределами области изменения факторов. Для выбора интервала варьирования исследователь обычно использует следующую априорную информацию: сведения о точности фиксации факторов; данные о кривизне функции отклика (линейная или нелинейная); результаты оценки чувствительности отклика модели Y_k к изменениям параметров модели X. Поэтому часто исследователи проводят серию «затравочных» экспериментов для знакомства с исследуемой СТС и получения недостающей априорной информации о влиянии выбранных интервалов варьирования факторов на вид искомой функции откликов.

При составлении плана анализа результатов ИЭ необходимо предусмотреть несколько проверок точности и приемлемости результатов имитации. В ряде случаев для определения источников погрешностей можно использовать уравнение баланса точностей. Суть подобной методики рассмотрим на следующем примере. Пусть в ИЭ измеряются переменные A, B, X, Y, которые связаны между собой уравнением сохранения значений между парами переменных: $A \cdot B = X \cdot Y$. Установлено, что из-за неточности описания реального процесса одна из этих переменных (неизвестно, какая из них) является причиной систематической погрешности. Для коррекции алгоритма ИМ нужно выявить не только виновника погрешности, но и место её возникновения в алгоритме модели. Пусть в ходе имитации не меняется значение A, а остальные переменные изменяются следующим образом: $X \to mX$; $Y \to nY$;

$B \to m \cdot B$. Уравнение баланса при этих условиях имеет вид: $AmnB = mXnY$. Далее используется следующее правило. Если одна из переменных в уравнении баланса, например, может быть представлена в виде суммы $(A + \psi(A))$, где $\psi(A)$ – систематическая погрешность, то эту переменную можно обнаружить, рассматривая поочерёдно случаи с фиксированным значением каждой переменной. Переменная, относительная погрешность изменения которой при фиксированном её значении не меняется, и является причиной систематической погрешности. Исключением является случай, когда переменная имеет пропорциональное приращение ошибки $A + kA$ (k – постоянная величина). В этом случае погрешность предлагаемым способом обнаружить невозможно.

Часто для поиска погрешности ИЭ используется предварительный эксперимент, по результатам которого исследователь строит график зависимости $Y = f(X)$. Система координат и сама функция f выбираются такими, чтобы график был линейным или хотя бы не имел большой кривизны вблизи начала координат для последующей экстраполяции графика. Обычно используются линейные или полулогарифмические шкалы, позволяющие строить прямую зависимость отклика Y от вектора параметров X.

Например, пусть исследователю известно, что один из графиков линейной зависимости отклика имеет систематическую погрешность. Необходимо определить, какой из графиков верен, и оценить при этом эту погрешность. Пусть для простоты каждый набор результатов ИЭ характеризуется одинаковым показателем степени, хотя сами графики не совпадают. Однако известно, что в области малых знаний зависимость $Y = f(X)$ должна проходить через начало координат. Тогда для идентификации и оценки систематической ошибки погрешности ИЭ строят графики в области малых значений. Тот график, который не удовлетворяет условию прохождения через начало координат, и имеет систематическую погрешность. Отметим также, что среднее значение Y при $X = 0$ и является оценкой этой погрешности.

3.4. Технология эксплуатации имитационных и вероятностных моделей СТС

Любая ИМ не имеет ценности до сих пор, пока не будет использована теми, для кого она разработана. Поэтому информация, получаемая в ходе эксплуатации ИМ, должна быть приемлемой для них. Критерии приемлемости прежде всего включают в себя надёжность и полезность информации для проектирования СТС. Исследователь должен понимать, как необходимо поступить или использовать результаты ИЭ в трудных ситуациях. Если ему не ясно, как эти данные могут помочь ему принимать проектные решения, то он

их будет просто игнорировать, и вся работа по созданию ИМ окажется безрезультатной. Поэтому очень важно уметь представить результаты ИЭ в наглядной форме, применяя при этом графические способы анализа данных. Наконец, любая ИМ должна позволять разработчикам СТС оценивать те решения, которые удовлетворяют их собственным понятиям рациональности СТС, и возможные результаты применения других стратегий проектирования. Отсюда следует важность их личного участия в создании ИМ.

Таким образом, ИМ должна давать информацию, которая может быть использована при проектировании СТС, быть понятной заказчику-пользователю, способной давать разумные ответы на его вопросы, легко модифицируемой, недорогой при её применении. Подача результатов имитации очень важна. Иногда полезнее организовать серию небольших неформальных обсуждений результатов с заказчиком, чем формализованное изложение результатов моделирования. Поэтому исследователь должен владеть методиками подачи результатов моделирования и их графического представления.

Чтобы задать характеристики поведения компонентов ИМ реальной СТС, зачастую приходится ставить серии экспериментов с прототипами компонент модели. Однако в ряде случаев поведение компонент реальной СТС может резко отличаться от предполагаемого исследователем. В таких случаях исследователь стоит перед проблемой: или игнорировать некоторые измерения как ошибочные, или пересмотреть своё представление о функционировании СТС. Зачастую может помочь процедура отбраковки ошибочных значений, суть которой можно пояснить следующим примером. На рис. 3.1 представлена зависимость $Y = f(X)$, у которой несколько точек резко отклоняются от аппроксимирующей прямой. Ставится вопрос: какие из них можно считать грубыми погрешностями опыта и исключить из дальнейшего рассмотрения. Весьма вероятно, что точка A является сомнительной, поскольку и для меньших, и для больших значений X ординаты близки к аппроксимирующей прямой линии. Точка B может быть ошибочной, но, возможно, она является важной и отражает какой-либо реальный физический

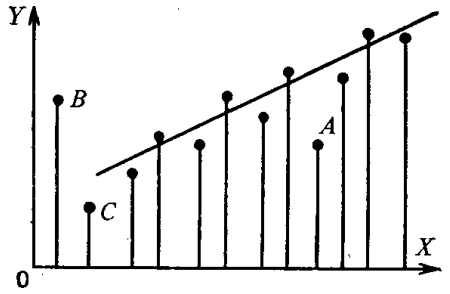

Рис. 3.1. Пример представления данных ИЭ, содержащих резко отклоняющиеся значения от линейной зависимости

эффект (например, переход СТС в область других состояний). Точку C также следует сохранить до тех пор, пока не будут получены дополнительные сведения об откликах СТС в области малых значений. Можно рекомендовать следующее правило отбраковки: отклоняющиеся точки следует исключать, пользуясь статистическими критериями, только в том случае, когда они находятся в средней части области изменения X.

В экспериментальной работе часто может возникать следующая ситуация: имеются данные, которые в определённой системе координат могут быть представлены в виде прямой (уравнение которой можно найти). Тогда первой задачей исследователя является выбор такой системы координат или преобразование выбранной уже системы координат в такую, в которой набор данных моделирования давал бы по возможности прямую линию. Для этой цели строят график в логарифмических координатах (например, $Y=k\cdot X^{\alpha}$ имеет вид $\lg Y=\lg k+\alpha\cdot\lg X$) либо в полулогарифмических координатах (шкала Y – логарифмическая, X – линейная). Иногда гиперболическую функцию $Y=X/(a+bX)$ можно представить в виде прямой, построив в линейных координатах зависимость X/Y от X или $1/Y$ от $1/X$.

Вообще графики позволяют представить данные в наглядной форме при минимальной обработке экспериментального материала и являются средством выдачи максимума информации при минимальном пространстве графика. При графическом представлении данных имитации рекомендуется придерживаться трёх простых правил, позволяющих получить графики с минимальной неопределённостью:

– минимальное деление шкалы графика должно соответствовать вероятной погрешности измеряемой величины;

– по возможности для любых данных ИЭ надо пытаться строить линейный график;

– выбор длины шкалы должен производиться из условия максимального использования формата графической бумаги. В противном случае можно либо не уловить основной характер кривой и не установить закономерность её изменения, либо не получить желаемой точности, поскольку все случайные отклонения сгладятся.

На всех этапах разработки и использования ИМ очень важно документирование как для лица, принимающего решение, так и для самих разработчиков ИМ. Документация на модель должна содержать в себе следующую информацию: точную формулировку цели моделирования; математическое определение, состав предположений и формулировку задачи моделирования; полный набор текущих входных данных моделирования; набор схем программ ИМ; описание используемых стандартных программ с комментариями; имена и адреса разработчиков программ ИМ. По мере того как описание ИМ создаётся, оно постепенно пополняется и детализируется.

На этапе определения необходимости ИЭ составляется документ, содержащий адреса всего состава разработчиков и потребности заказчика ИЭ. По мере изучения СТС составляется второй документ, в котором устанавливается цель моделирования, набор альтернативных решений. Далее составляется план работы над созданием ИМ и распределение функциональных обязанностей каждого из разработчиков ИМ.

На этапе формировки ИМ разрабатываются рабочие документы на весь жизненный цикл моделирования. В них содержится: описание всех деталей моделей; обоснование ИМ; состав предположений, гипотез и ограничений; описание процедур оценки параметров; основные требования к заданию исходных данных для моделирования; список ограничений по использованию ИМ.

На этапе реализации ИМ формируется набор эксплуатационных документов. В том числе приводится описание задачи проектирования СТС и общее описание ИМ. Указывается список допущений, ограничений и предложений, положенных в основу ИЭ.

Результаты испытания модели СТС также представляют собой важный документ. Подробно описывается план проверки модели, согласованный с лицом, принимающим проектные решения. Приводятся тесты и входные данные для испытания ИМ. По результатам испытания ИМ составляется пояснительная записка, в которой указываются размеры области изменения параметров ИМ, в которой модель дает удовлетворительные результаты.

Убедившись в пригодности ИМ для решения поставленных задач, составляют документы для обучения заказчика работе с ИМ. В них должно быть приведено описание процедуры привлечения пользователя к работе с ИМ. Для этого составляются: руководство пользователя ИМ; руководство аналитика СТС; резюме исполнителя, облегчающее интерпретацию результатов лицу, принимающему решение.

Этап моделирования СТС тщательно документируется и сопровождается большим числом иллюстративного материала. Все выводы оформляются в виде отчетов, которые составляют неотъемлемую часть документации этапа эксплуатации и могут служить источником для разработки новой ИМ СТС.

4. СИСТЕМЫ МОДЕЛИРОВАНИЯ, РЕАЛИЗУЮЩИЕ ТРАНЗАКТНЫЙ СПОСОБ ИМИТАЦИИ СТС

4.1. Общая характеристика языка моделирования GPSS

4.1.1. Объекты языка GPSS

В качестве объектов имитации могут выступать различные СТС и ТП, представленные в виде вероятностных СМО: вычислительные системы (ВС), ТП, гибкие автоматизированные производства (ГАП), роботехнические комплексы (РК), системы обработки информации (СОИ), предприятия сферы обслуживания (ПСО), транспортные системы (ТРС); коммуникационные системы связи (КСС) и т. д. Для указанного класса систем выделено конечное множество абстрактных элементов и конечное множество стандартных операций, описывающих связи между элементами. Выделенным множеством элементов и операций установлено соответственно множество объектов языка *GPSS*. Модель СТС на *GPSS* строится путем объединения объектов в некоторую фиксированную логическую блок-схему. Различают 7 категорий и 14 типов объектов (табл. 4.1).

Таблица 4.1. **Категории и типы объектов GPSS**

Категория объектов	Тип объектов	Обозначение
1. Динамическая	1. Транзакты	*TR*
2. Операционная	2. Блоки	БЛ
3. Аппаратная	3. Устройства	*F*
	4. Памяти	*S*
	5. Логические ключи	*L*
4. Вычислительная	6. Арифметические переменные	*V*
	7. Булевые переменные	*BV*
	8. Функции	*FN*
5. Статистическая	9. Очереди	*Q*
	10. Таблицы	*T*
6. Запоминающая	11. Ячейки	*X*
	12. Матрицы ячеек	*M*
7. Группирующая	13. Списки пользователя	*C*
	14. Группы	*G*

Каждому объекту ИМ соответствуют арифметические или логические атрибуты, описывающие состояние объекта в текущий момент модельного

времени t_0. Большинство из атрибутов недоступны для пользователя и являются рабочими переменными УПМ, реализующей операторы языка *GPSS*. Атрибуты, к которым в ИМ можно обращаться, называются стандартными числовыми атрибутами (СЧА). Основными объектами ИМ на языке *GPSS* являются транзакты (*TR*) и блоки (БЛ). *TR* описывают единицы исследуемых потоков (требования на обслуживание), поступающих в систему. БЛ задают логику функционирования ИМ и определяют пути движения *TR* между объектами аппаратной категории (*F*, *S*, *L*). Все изменения состояний ИМ (события C_{ij}) происходят в результате входа *TR* в БЛ и выполнения блоками следующих функций: создание (генерация) и уничтожение *TR*, изменение числовых атрибутов объектов, задержка *TR* на определенный интервал времени τ_{ij}, изменение маршрута движения *TR*.

Объекты аппаратной категории служат для описания единиц оборудования ТС. Воздействуя на эти объекты, *TR* могут изменять их состояние и влиять на движение других *TR*. Устройства *F* описывают оборудование, которое в любой момент времени может быть занято только одним *TR* (обычная одноканальная СМО), а также быть прервано поступлением более приоритетного *TR*. Памяти *S*, представляя собой многоканальные устройства, описывают оборудование, которое может использоваться несколькими *TR* одновременно (многоканальная СМО). Логические ключи используются для блокировки или изменения движения *TR* в зависимости от ранее наступивших в ИМ событий C_{ij}.

Объекты вычислительной категории описывают связи между элементами ТС, задаваемые с помощью аналитических или логических соотношений. Они могу служить как для задания вероятностных законов распределения случайной величины в ИМ, так и для численного или логического описания условий, определяющих движение транзактов. В эту категорию объектов входят две группы переменных – *V* и *BV*, а также допустимый набор функций *FN*. Статистические объекты обеспечивают вычисление и предоставление в стандартном виде значений показателей эффективности ТС, предоставляемой в виде СМО (средних значений, стандартных отклонений, эмпирических функций распределения параметров СМО). Очереди *Q* обеспечивают исследователя статистикой для поиска узких мест в моделируемой системе. Таблицы *T* содержат итоги предварительной обработки стандартной статистики.

Объекты категории **запоминающих** служат для условий моделирования, хранения, накопления и обработки информации, получение которой не предусмотрено стандартными средствами GPSS. Эту информацию можно хранить в виде двух типов объектов : ячеек *X* и матриц *M*, элементами которой являются ячейки. Объекты **группирующей** категории содержат информацию о транзактах, находящихся в ИМ. Двигаясь по ИМ от блока к блоку, *TR* могут приводить к наступлению таких событий C_{ij} , как поступление *TR* в

СМО, занятие и освобождение места в накопителе, занятие и освобождение канала обслуживания, прерывание обслуживания требования с более низким приоритетом, совпадение значений числовых атрибутов нескольких *TR*. При этом соответствующие *TR* помещаются в один из пяти списков: текущих или будущих событий (по отношению к t_0); прерываний; синхронизируемых друг с другом *TR*; список пользователя (т. е. *TR*, которые удалены пользователем из списка текущих событий).

4.1.2. Правила составления программ ИМ на GPSS

В программах на *GPSS* для описания объектов используются **операторы** и **карты описания**. На этапе разработки блок-схемы ИМ можно использовать графические обозначения объектов ТС. Моделирующая программа на языке *GPSS* записывается в операторной форме, которая затем преобразуется компилятором интерпретационного типа в загрузочный модуль. Запись операторов и карт описаний имеет единую форму и состоит из поля метки для записи символического адреса (имени) БЛ или объекта; поля операции для записи типа блока, объекта, управляющей карты; поля операндов, включающего от 0 до 9 операндов, разделенных запятыми и служащих для задания информации, необходимой для выполнения операций. Обычно эти операнды обозначают в порядке записи буквами *A*, *B*, *C*, ..., *H*, *I*. При записи операторов и карт можно использовать фиксированный и свободный формат. При фиксированном формате имеет место следующее распределение колонок по полям: 2–6-я колонки (поле метки); 8–18-я колонки (поле операции); 19–71-я колонки (поле операндов). Символ «*» в первой колонке означает строку-комментарий. При свободном формате необходимо выполнение следующих правил:

– поле метки начинается с 1-й колонки (наличие символа «*» также означает строку-комментарий);

– поля в записи отделяются друг от друга пробелами;

– некоторые типы блоков могут иметь расширенное поле операции.

Различают системные числовые атрибуты и стандартные числовые атрибуты объектов (СЧА). Системные числовые атрибуты характеризуют процесс имитации в целом. СЧА определяют состояние объектов ИМ в момент t_0. Каждый объект имеет свой набор СЧА. Все числовые атрибуты имеют символические имена. Имя СЧА включает мнемоническое обозначение, задающее тип атрибута (см. табл. 4.1) и порядковый номер объекта в ИМ либо его символическое имя. Объекты всех типов в ИМ нумеруются УПМ автономно для каждого типа в порядке их встречаемости в ИМ. Основными СЧА транзакта являются **параметры** и **приоритет** *TR*. Каждый *TR* может иметь от 0 до 1020 параметров одного из четырех форматов соответственно с именами PF_i, PH_i, PB_i, PL_i (где i – номер параметра). Выбор количества и назначение параметров осуществляется пользователем при формализации ТС на

языке GPSSV. В ИМ могут присутствовать *TR*, имеющие различный приоритет (от 0 до 127). У каждого блока имеется два СЧА: w_i – номер транзакта, находящегося в блоке с номером i в текущий момент времени t_0; N_i – общее количество *TR*, поступивших в i-й блок в течение одного ИЭ на ЭВМ.

4.1.3. Организация потоков транзактов в GPSS

Организация потоков *TR*, поступивших в ИМ на обслуживание, обеспечивается оператором создания требований *GENERATE A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, *I*. Операторы *A*, ..., *I* указывают следующие характеристики входного потока *TR*: среднее время поступления транзактов (*A*), модификатор среднего времени (*B*), время задержки первого транзакта (C), количество создаваемых *TR* (*D*), приоритет *TR* (*E*), *F*, ..., *I* – количество и формат параметров *TR*. Если в поле *F* записан 0, то у *TR* нет параметров.

Для задержки *TR* на заданный интервал времени используется оператор: *ADVANCE A*, *B*.

Оператор *TERMINATE A* уничтожает (удаляет из ИМ) транзакты. Здесь операнд *A* указывает число, на которое уменьшается содержимое счетчика числа завершений, значение которого задается в управляющей карте *START* в поле операндов *A*.

Оператор *ASSIGN A*,*B*,*C*,*D* служит для задания значений параметров транзактов, где:

A – номер изменяемого параметра с указанием режима изменения, накопления (+), вычитания (–), замещения (нет символов в знаковом разряде);

B – целое число, изменяющее значение параметра;

C – номер функции-модификатора значения, указанного в поле *B*;

D – формат изменяемого параметра (*PF*, *PH*, *PB*, *PL*).

Оператор *PRIORITY A* изменяет приоритет транзакта, здесь *A* – значение приоритета (от 0 до 127), присваиваемого *TR*.

Объект в *GPSS* типа «устройство» является аналогом канала обслуживания в СМО. Занятие канала (устройства) требованием *TR* осуществляется с помощью оператора *SEIZE A*, а освобождение канала по окончанию обслуживания *TR* – оператором *RELEASE A*. Здесь A- номер (имя) занимаемого (освобождаемого) *TR* устройства. Устройство занимается и освобождается в момент входа *TR* в соответствующий блок. *TR*, заставшие устройство занятым, ждут его освобождения в очереди.

Блок *PREEMPT A*,*B*,*C*,*D*,*E* приостанавливает обслуживание транзакта, ранее занятого устройства, и дает возможность захватить устройство следующему транзакту. Здесь A – номер (имя) устройства, работа которого прерывается входящим в блок *TR*; B – режим прерывания; C – адрес помещения прерванного *TR*; *D* – параметр прерванного *TR* (остаток времени его обслуживания); *E* – указатель на возможность повторного захвата устройства прерванным *TR*. Транзакт, вошедший в блок *RETURN A*, снимает прерывание на

устройстве с именем *A*. Блоки *PREEMPT* и *RETURN* часто применяются для моделирования циклических операций в ТС.

Памяти описывают оборудование, обслуживающее одновременно несколько требований (многоканальные СМО). Для задания объема памяти используется карта описания *n STORAGE A*. Здесь *n* – номер (имя) памяти, A – объем памяти. Эта карта помещается в начале программы ИМ на *GPSSV* перед первым оператором *GENERATE*. Изменение состояния памяти в ИМ осуществляется с помощью операторов занятия памяти *ENTER A*, *B* и освобождения памяти *LEAVE A*,*B*. Здесь A – номер (имя) памяти, B – число единиц памяти, занимаемое (освобождаемое) *TR* при входе в блоки.

Изменение маршрутов движения транзактов обеспечивается операторами и блоками *GATE*, *TRANSFER*, *TEST*, *LOOP*. Транзакты, входящие в соответствующие блоки, далее продвигаются УПМ не к следующему блоку (согласно блок-схемы ИМ), а к блокам, адреса которых определяются либо указываются в этих операторах.

4.1.4. Изменение маршрутов движения транзактов в GPSSV

Блок *GATE XXX A*, *B* разрешает движение *TR* (в основном либо альтернативном направлениях) при определенном состоянии оборудования (устройств, памятей, ключей). Существуют два режима работы блока: отказа и безусловного перехода. Здесь *XXX* – мнемоническое изображение перехода; A – номер (устройства, памяти, ключи); B – режим перехода (альтернативный адрес). В режиме отказа *GATE* не пропускает *TR*, если соответствующий объект не находится в требуемом состоянии, а направляется по альтернативному адресу *B*.

Оператор *TRANSFER A*, *B*, *C* обеспечивает четыре вида переходов *TR*:

TRANSFER B; – безусловный переход *TR* на блок с именем B;

TRANSFER BOTH B, *C*; – условный переход с одним альтернативным адресом; здесь B – основной адрес, а C – альтернативный адрес;

TRANSFER ALL B,*C*,*D*; – условный переход с несколькими альтернативами, где B – основной адрес, C – альтернативный адрес, *D* – константа для вычисления следующего альтернативного адреса;

TRANSFER A,*B*,*C*; – статистический переход с заданной вероятностью A; B – основной адрес; C – альтернативный адрес.

4.1.5. Организация вычислений и накопление результатов имитации

Связи между элементами ИМ, выраженные математическими соотношениями, на языке *GPSSV* описываются с помощью переменных (арифметических и булевских) и функций (непрерывных и дискретных). Арифметическая переменная (АП) – это арифметическое выражение, включающее в себя различные СЧА, знаки операций и скобки. Карта описания АП помещается в

начале программы. В булевских переменных употребляются три вида операций: логические, условные, булевые. Для обозначения логических операций используются мнемонические обозначения *j*-го объекта: устройство не прервано (FNI_j); устройство прервано (FI_j); устройство занято (FU_j); устройство не занято (FNU_j); устройство занято либо прервано (F_j); память заполнена (SF_j); память не заполнена (SNF_j); память пуста (SE_j); память не пуста (SNE_j); логический шаг установлен/сброшен (LS_j/LR_j). Условные операции имеют традиционные обозначения: *G*, *L*, *E*, *NE*, *LE*, *GE*. При использовании булевской переменной данные символы заключаются в апострофы.

Язык GPSSV не располагает средствами для проведения сложных аналитических расчетов; результаты операций берутся с точностью до целых. Поэтому для того, чтобы проводить аналитические вычисления, необходимо протабулировать аналитическую функцию. Затем результаты табулирования вводятся в модель картой

n FUNCTION A, *B*;

где *n* – номер (имя) функции; A – аргумент функции (СЧА); B – указывает тип функции. Каждая карта *FUNCTION* сопровождается картами последовательности пар значений (X_i, Y_i); где *i* – номер точки в табулируемой функции, значения аргумента X_i и функции Y_i.

При написании этих карт необходимо соблюдать правила:

– запись начинается с 1-й по 71-ю колонку;

– координаты точки X_i, Y_i разделяют запятыми, а пары – разделителем «/»;

– координаты одной точки должны быть заданы на одной карте;

– значения X_i, Y_i могут быть не целыми числами.

В процессе имитации в среде GPSSV можно сохранять и накапливать необходимые значения с помощью ячеек памяти *SAVEVALUE A*, *B*, *C*, где *A* – номер ячеек с указанием режима изменения содержимого ячейки; *B* – число либо СЧА, используемое для изменения содержимого ячейки; *C* – тип форматов информации.

4.1.6. Дополнительные возможности GPSSV

Для создания **копий** *TR* используется оператор *SPLIT A*,*B*; где A – количество копий; B – адрес, по которому направляется копия *TR*.

Для объединения определенного числа транзактов одного семейства используется оператор *ASSEMBLE A*, где *A* – число объединяемых транзактов. Транзакты задерживаются до тех пор, пока их не наберется *n* штук, затем все они уничтожаются, кроме первого, который далее переходит к следующему блоку.

Для синхронизации движения двух транзактов из одного семейства ис-

пользуются два сопряженных блока *MATCH*. Например:

AAA MATCH BBB

. . .

BBB MATCH AAA

Блок с меткой *AAA* будет ожидать в этом блоке прихода члена того же семейства в блок *BBB*.

Для определения количества объектов (из заданного множества объектов), удовлетворяющих заданному условию, используется блок *COUNT XXX A*, *B*, *C*, *D*, *E*; где *A* – номер параметра с указанием формата (*PF*, *PH*, *PB*); *B*, *C* – соответственно нижняя и верхняя границы диапазона изменения номеров объектов, для которых проверяется условие: *D* – СЧА, значение которого сравнивается со значением СЧА объекта, указанного в поле *E*; *E* – СЧА анализируемых объектов.

Для сбора и обработки статистики по очередям используются блоки *QUEUE*, *DEPART*: *QUEUE A*, *B*; где *A* – номер (имя) очереди, в которую заносится *TR* при невозможности войти в следующий за блоком *QUEUE* блок ; *B* – число занимаемых *TR* в очереди;

DEPART A, *B*, где *A*, *B* – аналогичны предыдущему, *B* – количество освобожденных мест в очереди.

Средства *GPSSV* позволяет получить эмпирические таблицы абсолютных и относительных частот попадания исследуемой случайной характеристики в заданные интервалы значений. Карта описания таблицы имеет вид :

n TABLE A, *B*, *C*, *D*,

где *n* – номер (имя) таблицы; *A* – табулируемый СЧА (аргумент таблицы); *B* – верхняя границы первого интервала; *C* – ширина интервала ; *D* – число интервалов. Помимо таблицы частот одновременно вычисляются оценки среднего и среднеквадратичного отклонений аргумента таблицы с помощью оператора *TABULATE A*, *B*; где *A* – номер таблицы; *B* – число добавляемых в соответствующий интервал единиц.

Для организации ИЭ пользователь должен задать УПМ *GPSSV* дополнительную информацию с помощью управляющих карт. Карта *SIMULATE A*; является первой картой программы ИМ и указывает на необходимость начала процесса имитации, который продлится А минут. Карта *START A*; помещается в конце входных данных, указывая таким образом, что входные данные для имитации заданы и можно начинать имитацию. Здесь А- число транзактов, которые должны пройти через модель до выдачи окончательной статистики.

4.2. Особенности организации и возможности системы моделирования GPSSV/PC

4.2.1. Дополнительные возможности GPSSV/PC

СМ GPSSV/PC является развитием GPSS для ПЭВМ. В GPSSV/PC все фазы создания и эксплуатации ИЭ (редактирование, компоновка, построение загрузочного модуля, выполнение и диагностика ошибок ИМ) объединены в одну фазу – сеанс имитации. Это означает, что как только программа ИМ вводится в систему, она проходит все стадии создания ИМ. УПМ по мере ввода операторов ИМ строит соответствующую структуру данных рабочей области ОП ПЭВМ. Если вводятся новые операторы ИМ с клавиатуры, они автоматически присоединяются к составу ИМ. Это исключает необходимость выполнения заново всех фаз сеанса при корректировке ИМ. Все числа в ИМ хранятся в форме, позволяющей снять ограничения на длину и точность представления чисел в модели. По сравнению с *GPSSV* добавлены новые возможности интерфейса, предоставлены возможности ввода информации с клавиатуры для записей, содержащих синтаксические ошибки, с диагностическим сообщением УПМ пользователю. Распознаются команды и операторы ИМ. Обеспечена автоматическая разметка полей в строке описания операторов. Имеется встроенный редактор строк, что позволяет модифицировать ИМ в процессе имитации. Используются изменяемые функциональные клавиши. Возможно управление процессом имитации с помощью команд *START*, *STEP* и *STOP*. Для контроля за процессом моделирования используются графические окна и микроокна. Имеется встроенная функция *HELP*, содержащая описания команд и операторов языка.

Аналогично *GPSSV* объекты *GPSS*/*PC* разделяются на те же **категорий и 14 типов. Модель ТС на языке *GPSS*/*PC* строится путем объединения объектов в некоторую блок-схему. Основой *GPSS*/*PC* являются программные модули, описывающие функционирование объектов языка, и УПМ, называемое «симулятор». Числовые атрибуты также делятся на системные и стандартные. Имя СЧА состоит из двух частей: символического обозначения, идентифицирующего тип объектов и тип информации об объекте, а также порядкового номера или символического имени объекта. Атрибуты могут использоваться в качестве операндов практически в любом числе блоков, входить в большинство операторов описания объектов, в то же время они обеспечивают пользователю доступ к характеристикам состояния системы в процессе имитации.

4.2.2. Операторы GPSS/PC

Различают операторы языка моделирования и системные команды

GPSS/PC. Используются три типа операторов языка: операторы блоков; операторы описания объектов (операторы описания данных); управляющие операторы. Возможности GPSS/PC как диалоговой системы ИЭ реализуются с помощью специальных команд. По форме описания оператора GPSS/PC и GPSSV аналогичны. В исходном тексте ИМ блоки описываются с помощью операторов описания блоков. При обработке исходного текста ИМ интерпретатор GPSS/PC присваивает блокам последовательные идентифицирующие номера. При необходимости непосредственного обращения к какому-либо блоку модели необходимо пользоваться идентификатором блока в виде метки. Поскольку в GPSS/PC отсутствует спецификация данных по типу формата, то в ряде операторов (*GENERATE*, *LINK*, *LOOP*, *ASSIGN*, *SAVEVALUE*, *COUNT*, *SELECT*) опущены операнды, связанные с указанием типа формата.

Для описания таких типов объектов, как памяти, таблицы, функции и переменные, используются **операторы описания объектов**. Они практически аналогичны соответствующим операторам языка *GPSSV*, поэтому рассмотрим некоторые особенности их использования в *GPSSV*/*PC*.

Оператор описания памяти имеет лишь один формат записи: <имя> *STORAGE A*; где <имя> – имя памяти, а *A* – объем памяти. Память нельзя удалить, но можно перераспределить, используя оператор *STORAGE* с тем же именем.

Оператор описания таблицы имеет вид: <имя> *TABLE A*,*B*,*C*,*D*; где <имя> – имя таблицы; *A* – СЧА (аргумент таблицы); *B* – верхняя граница первого интервала; *C* – ширина интервала; *D* – число интервалов. Таблица может быть переопределена другим оператором *TABLE* с той же меткой.

В GPSS/PC поддерживаются те же 5 типов функций (непрерывная, дискретная и табличная числовая, дискретная и табличная атрибутивная), что и в GPSSV. Операторы описания этих функций имеют похожий вид: <имя> *FUNCTION A*, *B*. При этом имеется три типа переменных: арифметические переменные, арифметические с «плавающей точкой», булевые. Операторы описания имеют похожий вид:

<имя> *VARIABLE A*;

<имя> *FVARIABLE A*;

<имя> *BVARIABLE A*,

где <имя> – имя (номер) переменной; *A* – арифметическое или логическое выражение. В выражениях допускается использование скобок для группировки и обозначения операции умножения.

Для задания условий имитации, таких, как продолжительность моделирования, число прогонов ИМ, порядок и условия сбора статистики в *GPSS*/*PC* используются управляющие операторы: *SIMULATE*, *START*, *RESET*, *CLEAR*, *END*. Использование этих операторов аналогично языку GPSSV, за исключением оператора *END*, который служит для завершения

сеанса работы с системой *GPSS*/*PC* и выхода в операционную систему ПЭВМ.

Для ссылок на блоки, объекты, СЧА, параметры или числовые контакты в *GPSS*/*PC* допускается использование символических имен. Любое имя может включать до 20 символов, начинающихся с буквы. Именами не могут быть: коды операторов, ключевые слова, коды СЧА. При трансляции именам присваиваются уникальные номера, начиная со стартового номера 10000. Имена объектов, параметров, числовых констант, метки блоков могут выступать как операнды. При этом возможна прямая и косвенная адресация.

4.2.3. Управление имитацией в GPSS/PC

Для организации интерактивного взаимодействия пользователя с моделью на GPSS/PC пользователь использует **команды**. Для начала работы с системой GPSS/PC требуется сделать текущий каталог, в котором будет размещена система, и выполнить программу GPSSPC.EXE. После исчезновения титульного листа системы необходимо ввести исходный текст ИМ в рабочий буфер GPSS/PC. Возможны два способа ввода: посредствам чтения текстового файла MS-DOS, содержащего исходный текст, и с помощью встроенного редактора GPSS/PC. Если создается новый текст ИМ с помощью редактора системы, пользователь последовательно набирает строки программы в командной строке окна данных и вводит их клавишей (*ENTER*). Редактор, добавляя новые строки в буфер, ориентируется на их номера. При этом ввод ошибочных строк заблокирован.

Для редактирования существующего текста программы ИМ можно использовать следующие команды:

EDIT A; – корректировка строки с номером, указанным в поле A;

DELETE A; – удаление строк, указанных в полях A и B (от A до B).

После корректировки программы ИМ можно перенумеровать все строки программы в рабочем буфере *GPSS*/*PC*. Для этой цели используется команда:

RENUMBER [*A*], [*B*];

где *A* и *B* – соответственно номера первой строки и шаг нумерации. Для просмотра в окне данных части или всей программы, содержащейся в рабочем буфере, служит команда:

DISPLAY [*A*], [*B*];

где *A* и *B* соответственно номера первой и последней строк отображаемой части программ.

Откорректированная в рабочем буфере GPSS/PC ИМ может быть сохранена на диске в текущем каталоге DOS с помощью команды:

SAVE A, [*B*], [*C*];

где *A* – имя файла, в который записывается программа ИМ; *B* и *C* – соответственно номера первой и последней строк сохраняемого участка программы ИМ.

Средства интерактивной графики *GPSS/PC* позволяют пользователю наблюдать за ходом процесса имитации и оперативно модифицировать модель или менять условия моделирования как на стадии отладки ИМ, так и во время решения задач исследования моделируемой системы. Для проведения модификаций процесс моделирования временно прерывается с помощью специальных команд. Для безусловного прерывания процесса моделирования служит клавиша [*ESC*]. При прерывании имитации по заданному условию можно пользоваться командами:

STEP A – команда прерывания после прохождения активным *TR* заданного в поле А числа блоков ИМ;

STOP [*A*], [*B*], [*C*] – установка (*ON*) или снятия (*OFF*) условий прерывания при вхождении *TR* с номером, заданным в поле *A*, в блок ИМ с номером. указанным в поле *B*; в поле *C* указывается тип состояния команды (*ON* или *OFF*), по умолчанию принимают *ON*. Команда *STOP* в состоянии *ON* устанавливает условия прерывания, но не запускает ИМ. Для продолжения процесса имитации при нулевом счетчике числа завершений может использоваться команда *CONTINUE*. Моделирование прерывается, когда встречается условие, установленное командами *STEP* или *STOP*. Для снятия условия прерывания необходимо ввести эту же команду *STOP*, но в состоянии *OFF*.

Остановка процесса имитации с целью временного выхода в ОС *DOS* осуществляется по команде *DOS*. Возврат же в прерванную точку процесса имитации происходит по команде *EXIT*. В течение прерывания вся информация о модели на момент остановки хранится в файле выгрузки с именем SWAPGPSS. Содержимое последнего уничтожается после возврата в систему GPSS/PC.

4.2.4. Средства интерфейса GPSS/PC с пользователем

Взаимодействие пользователя с УПМ осуществляется в режиме активного диалога. Для анализа результатов имитации пользователю предоставляют «виртуальные окна» и «микроокна». Под «виртуальным окном» понимается отображение информации о состоянии отдельных объектов ИМ на экране дисплея. Информация может отображаться как статически, так и динамически, при изменении состояния объекта в процессе его использования при имитации. Пользователь может активно вмешиваться в процесс отображения в «виртуальном окне», задавая различные команды из набора команд GPSS/PC и (или) используя специальные поля команд, имеющихся в ряде виртуальных окон. «Микроокно» – это небольшие графические окна в правой части виртуального окна, в которых отображается текущее значение системных или СЧА и соответствующий заголовок. В пределах «виртуального окна» может быть открыто до четырех «микроокон», имеющих фиксированную позицию и размеры. В процессе имитации содержимое «микроокон» меняется динамически при изменении значений, связанных с «микроокнами» переменных.

В распоряжении пользователей *GPSS/PC* предоставляет следующие типы «виртуальных окон»: данных (*DATA WINDOW*); блоков (*BLOCK WINDOW*); устройств (*FACILITIES WINDOW*); многоканальных устройств с памятью (*STORAGE WINDOW*); таблиц (*TABLES WINDOW*); матриц (*MATRICES WINDOW*); позиций (*POSITIONS WINDOW*). Команда открытия виртуального окна имеет вид: *WINDOW A*, [*B*]; где *A* – тип окна ; *B* – содержит имя или номер первого из отображенных объектов. Для открытия или переопределения микроокна можно использовать команду: *MICROWINDOW A*, [*B*], [*C*]; где: *A* – номер микрокоманды (от 1 до 4) ; *B* – аргумент микроокна (или номер параметра); *C* – состояние микроокна (*ON* – открыто, *OFF* – закрыто); *COMMENT* – комментарий (заголовок), относящийся к окну. Работая с окнами, пользователь может запросить «строку трассировки» прохождения *TR* блоков модели. Эта строка появляется в верхней части окна при нажатии клавиш.

Окно данных позволяет обеспечивать ввод и корректировку исходных текстов ИМ, а также отображение информации, связанной с командами *SHOW* и *PLOT*. Окно блоков предназначено для графического отображения блоков модели в виде блок-схем. После выбора команды «окно» появляется меню окна, состоящее из команд *STOP*, *EDIT*, *INSERT*, *DELETE*. **Окно устройств** обеспечивает графическое отображение информации о состоянии объектов GPSS/PC типа «устройство» в процессе имитации (коэффициент использования устройства, количество задержанных, прерванных и ожидающих *TR*; индикаторы доступности и занятости; среднее время пребывания *TR* в устройстве; номер транзакта, занимающего устройства). **Окно памятей** предназначено для графического отображения информации о состоянии объектов типа «память» в процессе имитации (коэффициент использования, количество задержанных *TR*, индикатор доступности, коэффициент использования памяти, число занятых единиц памяти). Меню окна памятей аналогично меню окна устройств. **Окно таблиц** обеспечивает визуализацию значений числовых характеристик и гистограммы распределения частот для анализируемого СЧА объекта, принимаемого в процессе моделирования случайных значений. Параметры таблиц задаются в операторах описания таблиц *TABLE* и *QTABLE*.

После завершения процесса имитации автоматически создается систем-

ный файл с именем REPORT.GPS, содержащий стандартную выходную информацию. Просмотр данного файла может быть осуществлен после выхода в DOS с помощью программы GPSSREPT.EXE. Выходной файл статистики состоит из подразделов, содержащих стандартную статистику об объектах GPSS/PC, используемых в данной модели (*FACILITIY*, *QUEUE*, *STORAGE*). Выходная информация состоит из следующих сегментов : основная информация о результатах работы модели; информация об именах, блоках текущей модели, устройствах, очередях, памяти, таблицах; информация об используемых в ИМ списках пользователя; информация о ячейках памяти.

4.3. Возможности моделирующего комплекса АСИМ

4.3.1. Графическое представление ИМ СТС

При разработке ИМ СТС одной из проблем является обеспечение простого перехода от временных диаграмм (ВД) к ИМ системы. В моделирующем комплексе (МК) АСИМ [1] предложен один из возможных способов такого перехода, сущность которого состоит в следующем. Любую компоненту СТС можно представить прибором массового обслуживания, называемым в АСИМ устройством, поведение которого изображается графически (рис. 4.1). Это поведение устройства от обычного прибора СМО отличается следующим. На входе и выходе возможно изображение любого числа очередей *TR*. Любое устройство может обслуживать любое число *TR* по индивидуальному для каждого из них закону распределения времени обслуживания $F(t_{ij})$ транзакта *i*-го типа на *j*-м элементе сложной системы. Каждое устройство имеет три управляющих входа : «Включить», «Выключить» и «Прервать» устройство (данное или другое). На эти управляющие входы поступают соответствующие управляющие сигналы. Если устройство выключено, на входе его образуются очереди, несмотря на то, что устройство может быть свободным от обслуживания. И это состояние продолжается до прихода сигнала на вход «Включить». При поступлении сигнала на вход «Прервать» переходят в состояние «Выключено», а *TR* возвращается первым в очередь к прибору, где ожидает продолжения обслуживания. На входе и выходе устройства возможно рождение из *TR* сигналов всех типов. На выходе устройства указывается траектория *TR* по модели после окончания его обслуживания *j*-м устройством. Возможен либо вероятностный выбор направления *i*-го *TR* по вероятности P_{ij}, либо детерминированный выбор маршрута его движения. На входе и выходе устройства допускается: размножение *i*-го *TR* на любое указанное заранее число транзактов – копий; формирование из *i*-го *TR* любого указанного заранее числа управляющих сигналов всех типов. В результате изменение состава управляющих сигналов (УС), мест их генерации при постоянном составе устройств позволяет изменить логику функционирования ТС.

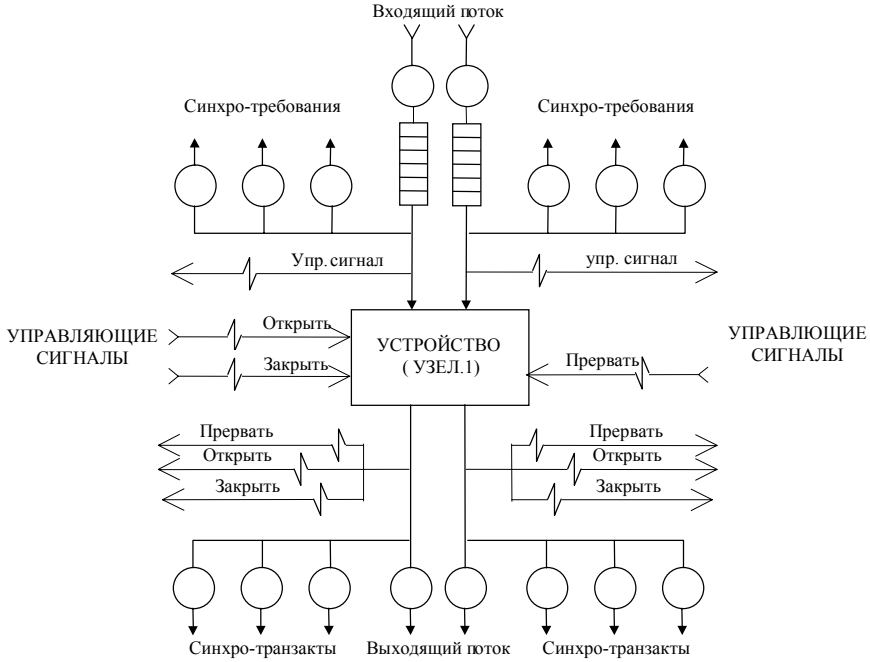

Рис. 4.1. Обобщенный узел сети СМО

Вначале графическое представление ТС записывается набором декларативных утверждений, отражающих все аспекты функционирования ИМ. К их числу относятся: состав устройств объекта имитации, связи между ними, логика функционирования и взаимодействия устройств. Затем полученное описание объекта на входном языке АСИМ автоматически переводится в машинную ИМ. Поэтому задача исследователя состоит в построении графической модели и описания каждого элемента этой графической модели набором декларативных предложений, не представляющих собой текста программы для ЭВМ.

Каждый из элементов ИМ характеризуется набором терминов из теории массового обслуживания: ресурс памяти, входящие потоки, очереди и дисциплины их обслуживания, времена обслуживания транзактов, выходящие потоки, маршруты транзактов, логика управлений состояний устройств. Совокупность описаний всех устройств в соответствии с установленными синтаксическими правилами представляет собой текст, готовый для автомати-

ческого перевода в машинную ИМ.

4.3.2. Средства АСИМ для создания ИМ

Каждый элемент ТС на языке АСИМ описывается комбинацией из следующих девяти предложений: устройство; входящий поток; очередь; механизм обслуживания; время отказа; механизм дообслуживания; выходящий поток. Модель представляется в виде сети СМО, состоящей из узлов (устройств) и связей между ними (поток *TR*, называемых требованиями). Проводится декомпозиция сети на отдельные узлы, а затем каждый узел отождествляется девятью языковыми конструкциями (предложениями АСИМ). Первое предложение идентифицирует узел сети, а остальные описывают его характерные особенности. Общий текст состоит из текста описаний каждого узла и завершается предложением ВРЕМЯ МОДЕЛИРОВАНИЯ.

Предложение УСТРОЙСТВО идентифицирует каждый из узлов с помощью указателя имени устройства ТИП, который может иметь смысловую нагрузку, и указателя приоритета устройства ИНДЕКС. При моделировании систем с ненадежными элементами, в которых возможно использование резервных устройств, включаемых при выходе из строя основного устройства, резервное устройство задается с помощью ключевого слова РЕЗЕРВ в предложении УСТРОЙСТВО и последующего описания идентификатора резервного устройства.

Предложение ВХОДЯЩИЙ ПОТОК характеризует процесс поступления требований на вход обслуживающего устройства (ОУ). Описание входного потока *TR* состоит из перечисления источников, поставляющих *TR* на обслуживание. Источники *TR* подразделяются на внесистемные и внутрисистемные. Внесистемные источники задают порядок поступления *TR* извне и определяются уникальным номером в одном из возможных законов распределений (равномерным, экспоненциальным, нормальным, Эрланга, Вейбулла, Парето, табличным). При исследовании СМО с приоритетами каждому источнику присваивается определенный класс и приоритет согласно предусмотренным в АСИМ дисциплинам обслуживания. Для регулирования процесса поступления *TR* конструкции ПЕРИОД ЗАПУСКА задаются активный и пассивный периоды работы источника транзактов. Внутрисистемные источники задают порядок поступления *TR* из других узлов сети. При моделировании систем с ненадежными элементами и резервированием для описания входов дополнительно используются конструкции ДЛЯ ОГРАНИЧЕНИЙ. Наличие этой конструкции указывает на то, что поступление *TR* от этих источников на резервное ОУ будут планироваться только в случае отказа основного устройства. Для имитации систем с внутренним управлением МК АСИМ позволяет переводить ОУ из состояния ГОТОВО в состояние

ЗАНЯТО. Перевод узла из одного состояния в другое производится сигналами «включить» и «выключить». Генерация этих сигналов может быть проведена любым из внутренних потоков *TR* в момент их поступления на вход ОУ. Задание режима генерации сигналов *TR* производится посредством конструкции С РАЗДЕЛЕНИЕМ НА СИГНАЛЫ ТИПА.

Предложение МЕХАНИЗМ ОБСЛУЖИВАНИЯ задает совокупность правил, которые определяют порядок обработки *TR*, поступающих на устройство. Предложение содержит описатели дисциплин обслуживания и ряд дополнительных указателей в случае моделирования систем с отказами. Возможны следующие дисциплины обслуживания *TR* в очереди: *FIFO*, *LIFO*, *RANDOM*, круговые алгоритмы, приоритетные с отказами. Для приоритетных дисциплин указывается тип приоритета с помощью указателя ДИНАМИЧЕСКИЙ или ВНЕШНИЙ. Кроме того, возможна дополнительная модификация приоритета с помощью указателей: АБСОЛЮТНЫЙ, ОТНОСИТЕЛЬНЫЙ, СМЕШАННЫЙ, АБСОЛЮТНО-ОТНОСИТЕЛЬНЫЙ, МНОГОУРОВНЕВЫЙ. Заданием конструкции С ПОТЕРЯМИ или ПОВТОРНОЕ ОБСЛУЖИВАНИЕ указывается порядок обработки *TR* после прерывания его обслуживания из-за прихода более приоритетного. При этом характер повторного обслуживания задается конструкцией С НОВЫМ ВРЕМЕНЕМ или С ПРЕЖНИМ ВРЕМЕНЕМ.

Предложение ОЧЕРЕДЬ описывает состав входных очередей к устройству. Возможно задание трех типов очередей к ОУ с помощью указателей: ОБЩАЯ, ОТДЕЛЬНАЯ, СМЕШАННАЯ. Общая очередь предполагает постановку в одну очередь *TR* от всех источников, перечисленных в предложении ВХОДЯЩИЙ ПОТОК. На формирование очередей могут налагаться ограничения: длина очереди, время пребывания в очереди, вероятности постановки транзактов в очередь. Для задания этих ограничений в предложении имеются следующие указатели: ДЛИНА, ВРЕМЯ, ВЕРОЯТНОСТЬ. При отказе от обслуживания *TR* либо покидает систему, либо поступает на обслуживание к ОУ, описанному следом за ограничениями. На устройствах с ограниченными очередями при их переполнении может быть задан специальный режим конструкцией БЕЗ ПОТЕРЬ. В этом случае *TR*, застав ОУ занятым и очередь переполненной, не покидает предыдущее устройство до тех пор, пока не появится свободное место в очереди.

Предложение МЕХАНИЗМ ПОДКЛЮЧЕНИЯ описывает различные стратегии подключения свободных устройств. Возможны следующие стратегии подключения, задаваемые конструкцией: В ПОРЯДКЕ ПРИОРИТЕТА, В ПОРЯДКЕ ОСВОБОЖДЕНИЯ, В СЛУЧАЙНОМ ПОРЯДКЕ, В ПОРЯДКЕ ГРУППОВОГО ПРИОРИТЕТА. В некоторых СМО поступивший *TR* тотчас же не может быть принятым на обслуживание, так как для включения или

подготовки ОУ к работе требуется некоторое время «разогрева». Вид «разогрева» указывает пользователь в конце предложения с помощью соответствующих конструкций: РАЗОГРЕВ 1-ГО РОДА и РАЗОГРЕВ 2-ГО РОДА. В этих конструкциях используются указатели типа РАСПРЕДЕЛЕНИЕ. Для моделирования СМО с малой нагрузкой вводится описатель ПОРОГ ВКЛЮЧЕНИЯ. В этом режиме ОУ начинает обслуживание, когда на входе скопилось число транзактов, превышающее пороговое значение.

Предложение ВРЕМЯ ОБСЛУЖИВАНИЯ описывает длительность обслуживания *TR* устройством. Описатель РАСПРЕДЕЛЕНИЕ задает тип и параметры закона распределения длительности обслуживания *TR*. Поскольку в СМО широко распространено квантование, то с помощью конструкции С РЕЖИМОМ КВАНТОВАНИЯ задается квант времени, выделяемый для обслуживания.

Предложение ВРЕМЯ ОТКАЗА используется для описания сетей с ненадежными элементами. Задается длительность безотказной работы, распределенная по закону, указанному описателем РАСПРЕДЕЛЕНИЕ. В течение определенного времени отказавшее ОУ может быть восстановлено, после чего процесс обслуживания ОУ возобновляется. Длительность восстановления также является случайной величиной, распределенной по закону, указанному вторым описателем РАСПРЕДЕЛЕНИЕ. Требования, поступившие на вход отказавшего ОУ, заполняют очередь по обычным правилам, если для последнего не было задано резервное ОУ.

Предложение МЕХАНИЗМ ДООБСЛУЖИВАНИЯ описывает судьбу *TR*, если в момент его обслуживания произошел отказ ОУ. Это предложение имеет смысл только при наличии предложения ВРЕМЯ ОТКАЗА. Предложение конструируется из описателей: ТРЕБОВАНИЯ ДООБСЛУЖИВАЮТСЯ, ТРЕБОВАНИЯ ТЕРЯЮТСЯ, ТРЕБОВАНИЯ ПОВТОРНО ОБСЛУЖИВАЮТСЯ. В случае повторного обслуживания используются описатели: С ПРЕЖНИМ ВРЕМЕНЕМ, С НОВЫМ ВРЕМЕНЕМ. Характер повторного обслуживания задается описателями: В ПЕРВУЮ ОЧЕРЕДЬ, НА ОБЩИХ ОСНОВАНИЯХ.

Предложение ВЫХОДЯЩИЙ ПОТОК применяется в тех случаях, когда после завершения этапа обслуживания транзакт является источником новых *TR*. Процесс деления *TR* задается конструкцией С РАЗДЕЛЕНИЕМ НА ЗАЯВКИ. Правила планирования новых *TR* на дальнейшую обработку отличны от стандартных. Маршруты их следования могут быть как детерминированными, так и вероятностными и задаются комбинациями конструкций СПИСОК УСТРОЙСТВ и С ВЕРОЯТНОСТЬЮ. Особенностью выходящего потока является свойство управляющих сигналов «включить» и «выключить», переводящих ОУ в состояние ГОТОВО или ЗАНЯТО. Задание управляющих

сигналов осуществляется с помощью конструкции С РАЗДЕЛЕНИЕМ НА СИГНАЛЫ ТИПА. Задается также способ определения дальнейшего маршрута следования обслуженных транзактов. Имеется возможность выбора маршрута на основе вероятностей, приписанных каждому из альтернативных устройств. Множество устройств и их весов составляют условный список, а задание случайного способа планирования осуществляется посредством конструкции ПЛАНИРОВАНИЕ ПО ВЕРОЯТНОСТИ.

Таким образом, в распоряжение пользователя предоставляются декларативный неалгоритмический язык описания стохастических сетей СМО, который имеет следующие особенности:

– наличие средств управления структурой сети, технологии связей алгоритмов маршрутизацией *TR* стартстопного режима работы узла сети;

– возможность описания систем с ненадежными элементами (учитываются отказы, резервирование и восстановление устройства сети);

– широкий спектр законов распределения входящих потоков и дисциплины обслуживания, часто встречающиеся в практике.

4.3.3. Средства АСИМ для эксплуатации ИМ

Этап исследования свойств ИМ обеспечивается в МК АСИМ наличием предложения ВРЕМЯ МОДЕЛИРОВАНИЯ, задающим интервал времени, в течение которого необходимо имитировать поведение СМО. Наличие в этом предложении специальной конструкции СБОР СТАТИСТИКИ В СТАЦИОНАРНОМ РЕЖИМЕ обеспечивает автоматическое наблюдение за моментом входа контролируемого параметра ИМ в стационарный режим путем оценки его среднего значения. Контроль может осуществляться на любом ОУ, входящем в состав ИМ. Для этого фиксируются идентификаторы контролируемого ОУ посредством конструкции УСТРОЙСТВО. Конструкцией ПАРАМЕТР задается временной интервал Δt, по истечении которого происходит вычисление среднего значения параметра. Проверка ведется по критерию Вилкоксона, который требует задания уровня значимости с помощью конструкции ДОВЕРИТЕЛЬНАЯ ВЕРОЯТНОСТЬ. Для достижения необходимой надежности заключения о стабилизации этой оценки вводится коэффициент подтверждения, который позволяет отложить до следующей проверки заключение о достижении стационарного режима. Для этой цели используется указатель КОЭФФИЦИЕНТ ПОДТВЕРЖДЕНИЯ.

В ряде случаев, кроме требования сбора статистики, в стационарном режиме необходимо оценить характер переходного процесса в ИМ. Наличие в предложении ВРЕМЯ МОДЕЛИРОВАНИЯ конструкции ПЕЧАТЬ ГРАФИКА СРЕДНЕГО позволяет осуществлять в конце имитации выдачу на печать графика контролируемого параметра.

Процесс управления узлами в сложных сетях, как правило, трудно реализовывать средствами СМО. В МК АСИМ для этой цели используются: управляющие сигналы «включить», «выключить», «прервать»; синхронизирующие транзакты с помощью операции деления транзактов как на входе ОУ (в предложении ВХОДЯЩИЙ ПОТОК), так и на выходе ОУ (в предложении ВЫХОДЯЩИЙ ПОТОК). Синхронизирующие *TR* из основной модели ТП поступают в блок управления. Блок управления генерирует ответные сигналы, воздействующие на структуру и состояния ОУ основной модели ТП путем включения, выключения и прерывания соответствующих устройств. В отличие от способа организации управления на основе алгоритмических языков (типа *GPSS*/*PC*) здесь вопросы конструирования схем управления не затеняются алгоритмическим представлением, что существенно облегчает поиск необходимого варианта управления сетью СМО.

Генерация новых транзактов в СМО осуществляется посредством включения управляемых источников *TR*. Число транзактов регулируется количеством сигналов «включить источник». Тем самым возможна имитация группового поступления *TR* на обслуживание. Управляемые источники могут быть сгруппированы на одном устройстве. В ряде случаев при построении ИМ с помощью МК АСИМ необходимо объединить множество копий в один транзакт (осуществить «сборку» *TR*). Для реализации операции сборки *TR* при описании каждого конкретного входа на устройство, на котором производится сборка *TR*, используются дополнительные устройства.

Печать стандартной статистики имитации проводится по окончании имитации для всех структурных элементов сети. К основным стандартным статистикам ОУ относятся: количество *TR*, получивших отказ в обслуживании; время работы и простоя; коэффициенты загрузки и простоя; коэффициент занятости; время простоя из-за отказов ОУ.

По очередям формируются статистики: количество транзактов, побывавших в очереди; количество *TR*, находящихся в ней; количество *TR*, получивших отказ в обслуживании; максимальная и средняя длина очереди; среднее время пребывания *TR* в очереди. К выходной информации, характеризующей ИМ в целом, относятся среднее время пребывания *TR* в системе и количество *TR* данного типа, прошедших через систему. Выдаются также характеристики распределений любой заказанной случайной величины в ИМ: оценка моментов, эксцесса, асимметрии, доверительная область с заданными коэффициентами доверия, табличная функция распределения.

5. АВТОМАТИЗАЦИЯ ЭТАПОВ ПОСТРОЕНИЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ В СИСТЕМЕ МОДЕЛИРОВАНИЯ MICIC

5.1. Основная идея применения и построения ИМ в СМ MICIC

В основу СМ MICIC положена концепция многоуровневого представления объектов моделирования (ОМ). Это дает возможность работать с ИМ пользователям разной квалификации и степени принятия управленческих решений. Уровни пользователей определяются целями работы с ИМ. Можно выделить четыре цели: создание концептуальной модели и формального описания ОМ, программирование модели, конструирование и эксплуатация ИМ. В соответствии с ними пользователи СМ MICIC логически подразделяются на аналитиков, разработчиков, конструкторов и исследователей модели.

Исследователь модели рассматривает ОМ как «черный ящик», имеющий входы (параметры) и выходы (отклики). Для проведения ИЭ служит интегрированная среда (ИС) СМ MICIC. В ней имеются оперативные средства задания параметрической нагрузки, просмотра результатов, мониторинга, постановки и реализации ИЭ.

Конкретную структуру ИМ задает конструктор также в СМ MICIC. Он рассматривает ОМ как систему массового обслуживания. Ее отображением является орграф, в котором вершины – это устройства (приборы массового обслуживания), а дуги – потенциальные пути перемещения транзактов (заявок на обслуживание). ИМ развивается во времени путем взаимодействия транзакта и устройства. Между устройствами (на дугах) никаких функциональных действий не происходит. ИС СМ MICIC позволяет создавать и уничтожать элементы структуры модели, видоизменять связи между ними, переводить их в различные состояния, получать разнообразную справочную информацию.

Разработчик модели программирует на языке моделирования MICIC компоненты ИМ дискретного типа. Он проводит декомпозицию структуры ИМ, выделяя различные по механизму обслуживания пары устройство – транзакт. Для каждой такой пары составляется орграф активностей, отображающий последовательную смену функциональных действий взаимодействия компонентов в модельном времени. Таким образом, разработчик модели должен на языке моделирования (ЯМ) MICIC определить параметры и отклики ИМ, компоненты ИМ и их параметры, активности контактов компонентов. По своему синтаксису ЯМ MICIC «погружен» в язык программирования Си. Это обеспечивает процесс программирования и отладки модели всем сервисом, который предоставляет пакет *Borland C*. На рис. 5.1 схемати-

чески представлены этапы создания загрузочного модуля и отладки программы имитационной модели средствами *Borland C*.

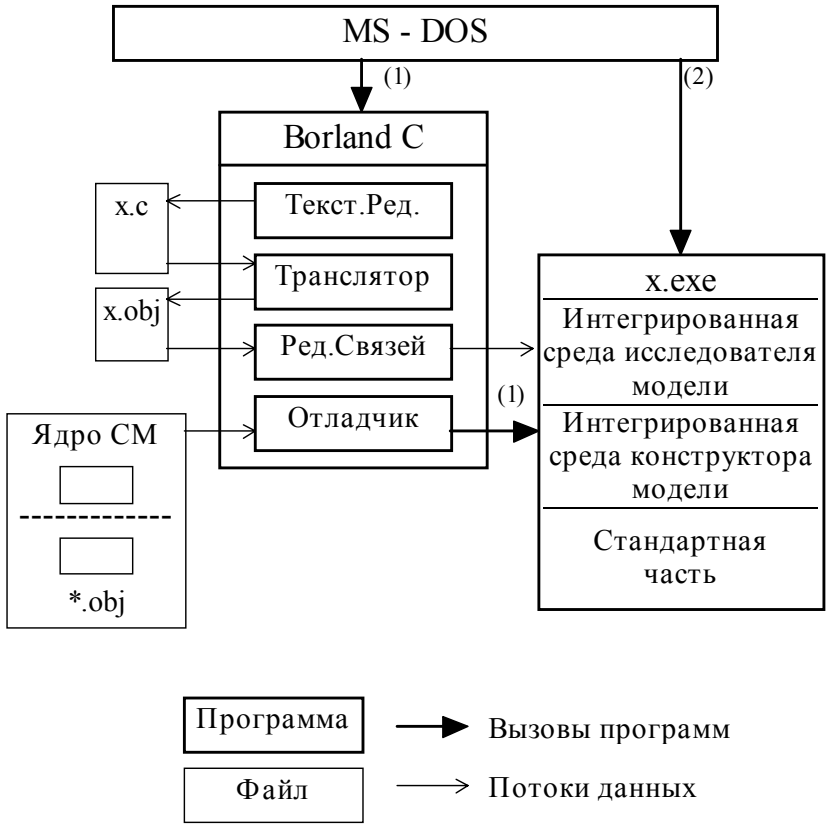

Рис. 5.1. Схема получения загрузочного модуля ИМ в СМ MICIC

Из схемы видны два способа вызова программы ИМ:

1) из отладчика Borland C (используется при отладке текста ИМ);

2) из MS DOS (используется как звено технологической цепочки при решении задач системного анализа).

Таким образом, во втором случае загрузочный модуль ИМ представляет собой СМ конечного пользователя на двух уровнях: исследователя и конструктора модели. Определение структуры и имитация различных сценариев функционирования системы обеспечиваются диалоговыми средствами корректировки параметров и изменения состояния компонентов в ИС СМ MICIC.

Интегрированная среда исследователя ИМ позволяет выполнить следующие функции:

– корректировка параметров ИМ;
– просмотр значений откликов ИМ в динамике имитации;
– управление списком событий ИМ;
– организация мониторинга процесса имитации;
– постановка и реализация планов эксперимента на ПЭВМ;
– экспорт результатов многопрогонного эксперимента на ИМ.

При этом при мониторинге имитации обеспечивается контроль за состоянием ИМ СТС в ходе имитации путем использования различных графических объектов. Исследователю предоставляются возможности перезагрузки и сохранения промежуточных состояний ИМ для продолжения имитации с прерванного места в ИМ. При экспорте результатов ИЭ данные ИМ преобразуются в *dbf* или *txt* форматы для последующей их обработки другими программами исследователя.

Интегрированная среда конструктора ИМ предоставляет следующие возможности организации ИЭ:

– создание и удаление элементов структуры ИМ;
– изменение состояний элементов и связей между ними;
– генерации сигналов к элементам ИМ;
– просмотр и корректировка параметров элементов ИМ;
– управление очередями в ИМ;
– получение разнообразной справочной информации.

Аналитик модели разрабатывает концептуальную модель и формальное описание СТС, опираясь на базовую схему формализации ЯМ MICIC. Данный этап невозможно автоматизировать. Как правило, от его качественного выполнения зависит достоверность выходной выборки. Задачей аналитика также является интерпретация результатов ИЭ. Для этого используются средства экспорта выборки из СМ MICIC в наиболее распространенные форматы и дальнейшая обработка данных пакетами *OLAP*, имеющимися в распоряжении пользователя.

Таким образом, совокупность усилий перечисленных выше специалистов приводит к выдаче экспертных оценок и рекомендательных решений руководителям высшего звена управления, заказавших проект на моделирование СТС.

5.2. Базовая схема формализации системы моделирования MICIC

Под базовой схемой формализации понимают совокупность понятий, которые используются для построения формального описания моделируемой системы и непосредственно представлены в языке моделирования. На уровне конструктора модели объект моделирования есть система массового обслуживания, компонентами которой являются обслуживающие устройства и очереди на обслуживание (статические элементы модели) и транзакты (динамические элементы). Мы хотим обратить внимание на различие между

понятиями компонент и элемент имитационной модели. Оно напоминает различие между классом и объектом в концепции объектно-ориентированного программирования. Компонент есть совокупность набора параметров и механизма обслуживания транзакта на устройстве. Конкретные элементы любого компонента модели типа «устройство» называются версиями устройства, а элементы типа «транзакт» – копиями транзакта. Элементы одного и того же компонента модели совпадают по набору параметров и механизму обслуживания, но различаются по конкретным значениям, присвоенным параметрам элементов.

Механизм обслуживания транзакта на устройстве представляет собой процесс (т. е. множество многофункциональных активностей, объединенных в виде связного ориентированного графа, в котором дуги показывают пути передачи управления между активностями – вершинами), состоящий по крайней мере из двух активностей (начальной и конечной). Начальная активность при поступлении транзакта на свободное устройство или при освобождении устройства с непустой очередью реализует передачу управления на первую активность, аппроксимирующую непосредственно обслуживание (при отсутствии таковой управление сразу передается на конечную активность). Конечная активность завершает процесс, освобождая обслужившее устройство и перемещая транзакт в очередь к следующему устройству. Никаких других ограничений на структуру процесса и механизма обслуживания между крайними активностями не накладывается (т. е. допускаются вместе с линейными участками условные переходы и циклы).

Рассмотрим свойства введенных элементов.

– Устройство многоканально, т. е. в любой момент модельного времени на устройстве можно одновременно обслуживать более одного транзакта, но не более заранее заданного количества. Взаимодействующая пара – версия устройства и копия транзакта – называется каналом.

– Устройство может функционировать в одном из двух режимов: сервер, процесс. Они различаются тем, что устройство-процесс не принимает транзакты на обслуживание. Очевидно основное назначение таких устройств – генерация внешних потоков транзактов.

– Устройство полиморфно, т. е. выбор соответствующего механизма обслуживания происходит при попадании транзакта на устройство и, естественно, реализуется начальной активностью.

– Устройство-сервер может принимать сигналы типа «Открыть» и «Закрыть». Первый сигнал разрешает принятие на обслуживание новых транзактов из очереди, а второй, наоборот, запрещает. Данные сигналы никаким образом не влияют на транзакты, которые начали обслуживаться до приема этих сигналов.

– Устройство-процесс может принимать сигналы «Пассивизировать», «Активизировать». Первый сигнал временно запрещает выполнение очередной активности и убирает ее из списка событий, а второй, соответственно, возвращает эту активность в список и передает на нее управление в текущий момент модельного времени, если активность оказалась просроченной.

– Транзакт может принимать сигналы «Задержать», «Отпустить», «Покинуть», «Повторить», «Уничтожить». Первый сигнал временно запрещает выполнение очередной активности канала и убирает ее из списка событий, а второй, соответственно, возвращает эту активность в список и передает на нее управление в текущий момент модельного времени, если активность оказалась просроченной. Фактически эти сигналы являются аналогами сигналов «Активизировать»/ «Пассивизировать» для устройства-сервера. Однако они подаются на транзакты, чтобы исключить неоднозначность, связанную с многоканальностью устройства. Третий и четвертый типы сигналов обеспечивают переход соответственно на конечную и начальную активности механизма обслуживания. В случае получения последнего сигнала транзакт поглощается системой моделирования.

– Транзакт обладает встроенными приоритетом и указателем направления движения по устройствам модели. Конечная активность переправляет транзакт на очередное устройство по значению этого указателя. Во время обслуживания на любом устройстве это значение можно изменять сколь угодно много раз.

– Транзакт обладает встроенным транзактным буфером. Это позволяет группе транзактов передвигаться по модели как единому элементу. При появлении на устройстве транзактов, находящихся в транзактном буфере, процесс-механизм обслуживания не запускается. Данный механизм позволяет достаточно просто описывать на детальном уровне поведение, например, транспортных систем, где потоки пассажиров и/или грузов накладываются на также динамические потоки транспортных средств.

– Транзакты сохраняемы, т. е., приходя на обслуживание на занятое или закрытое устройство, транзакт автоматически становится во встроенную к нему очередь.

– Очередь упорядочена по неубыванию приоритета и времени прибытия транзакта в очередь.

Описанный подход к формализации СТС с помощью вышеприведенных базовых элементов носит название транзактно-процессный способ представления объекта моделирования.

5.3. Методика построения имитационной модели для системы моделирования MICIC с помощью языка программирования СИ

5.3.1. Назначение методики

Одной из составных частей системы моделирования (СМ) MICIC являет-

ся язык разработчика элементов ИМ. Он позволяет описывать программы обслуживания устройств, классов транзактов и очередей. Однако, как правило, разработкой ИМ занимаются специалисты, достаточно уверенно владеющие языком программирования Си. Поэтому для них идеальным решением является возможность использования функций языка моделирования при соблюдении некоторой дисциплины построения программы ИМ. Далее, получив объектный модуль ИМ и скомпоновав его с ядром СМ MICIC, исследователи СТС могут работать в интегрированной среде СМ в соответствии с руководством пользователя.

5.3.2. Определение глобальных данных имитационной модели

В СМ MICIC под глобальными данными понимают входы и выходы ИМ, т. е. параметры и отклики. В программе ИМ глобальные данные описываются с помощью уникальных для каждой ИМ структур переопределенных типов *GlobalPar* и *GlobalRes* с именами *GPar* и *GRes* соответственно. За описанием параметров ИМ может следовать их определение (начальная инициализация) стандартным для языка Си способом. Глобальные данные могут иметь следующие типы (в скобках приведен их контекст в программе ИМ на Си):

1) целый (*int*);
2) вещественный (*float*);
3) устройство (*dvC* *);
4) транзакт (*tns* *);
5) целый расширенного диапазона (*long*);
6) строковый (*string*).

Строковые данные по длине не могут превышать 15 символов, поскольку тип «*string*» переобъявлен следующим образом:

typedef Char string [16].

Данные ИМ будем подразделять на массивы и простые переменные. В программе ИМ запрещается использование массивов, имеющих более семи измерений.

ПРИМЕР 5.3.1. ИМ имеет следующие параметры и отклики: *i*1 – переменную целого типа; *d*1 – переменную типа устройство; *t*1 – массив типа устройство; *s*1 – массив типа строка; *f*1 – переменную вещественного типа; *l*1 – массив расширенного целого типа. Определить параметры и описать отклиёки, выбрав произвольно границы массивов.

```
typedef struct {
        int i1;
        dvc *d1;
        tns *t1[3][2];
        string s1[4];
}GlobalPar GPar =
 {200,NULL,NULL,NULL,NULL,NULL,NULL,NULL,
```



```
"строка 1","строка 2","строка 3","строка 4"};

typedef struct {
        float f1;
        long l1[4];
} GlobalPar GRes;
```

Особенностью технологических оболочек (интегрированной среды (ИС)) СМ MICIC является возможность просмотра и корректировки глобальных данных ИМ. Это достигается за счет введения следующих макроопределений:

IEGlobal (data_type) – описание заголовка глобальных данных, где data_type принимает значение Prm для параметров и Res – для откликов;

IEVar (name,type) – описание простой переменной ИС с именем name (любая не взятая в кавычки последовательность из не более 15 ASCII символов) и типом type, принимающим одно из значений: INT, FLOAT, DEVICE,TRANSACT,LONG,STRING;

IEArray (name,type,measures) – описание массива ИС с именем name, типом type и одномерным целым массивом measures, в котором находятся границы массива name по каждому измерению и который должен быть определен заранее.

ПРИМЕР 5.3.2. Определить глобальные данные для ИМ из примера 5.3.1.

```
int m1[]={3,2},m2[]={4};
IEGlobal (Prm) = {
        IEVar (целое1,INT),
        IEVar (устройство1,DVC),
        IEArray (транзакт1,TNS,m1),
        IEArray (строка1,STRING,m2)
}
IEGlobal (Res) = {
        IEVar (веществ1,FLOAT),
        IEArray (длинный1,LONG,m2)
}.
```

Таким образом, чтобы обеспечить эффективное управление данными в ИМ для СМ MICIC, необходимо их двойное описание. Первый раз глобальные данные описываются для обращения к ним из программы ИМ, а второй раз определяются в соответствии с шаблоном примера 5.2.2 для совершения операций над ними из ИС СМ MICIC.

5.3.3. Описание параметров компонентов ИМ

Параметры каждого компонента ИМ должны быть описаны в соответствии с приведенным в предыдущем разделе принципом дублирования. Единственное исключение состоит в том, что для параметров каждого компонен-

та вводится отдельный тип, который как раз и является параметром макроопределения *IEGlobal*.

ПРИМЕР 5.3.3. Пусть некоторый компонент модели с именем «Генератор заявок» имеет следующие параметры: интервал между заявками; устройство, на которое подаются заявки, и его номер очереди; частотное распределение вероятностей для определения типа заявки. Описать параметры компонента для программы ИМ и для ИС.

Описание параметров для программы ИМ:

```
typedef struct {
        float Interval;
        dvc *Receiver;
        int numQuene;
        float Frequency [10];
} ReqGenerator;
```

Описание параметров для ИС:

```
int m1[]={10};
IEGlobal (ReqGenerator) = {
        IEVar(интервал,FLOAT),
        IEVar(получатель,DVC),
        IEVar(номер_очереди,INT),
        IEArray(частота,FLOAT,m1)
}.
```

Если у компонента ИМ параметры отсутствуют, то не требуется никакого описания.

5.3.4. Определение компонентов ИМ

Чтобы можно было совершать операции над компонентами ИМ в ИС СМ MICIC, необходимо определить их типы в программе ИМ с помощью следующих описателей:

IEKomponents – заголовок определения;

IEKmp (name,type,Kind) – определение компонента ИМ с именем name (любая не взятая в кавычки последовательность из не более 15 ASCII символов) ; здесь

type – переопределенный тип структуры описания параметров для программы ИМ,а Kind принимает значение PRC для устройств-процессов, SRV – для обслуживающих устройств, TNS – для транзактов;

IEKmp0 (name,Kind) – определение компонента ИМ, не имеющего параметров.

Вообще, в программе ИМ для обращения к ее компонентам можно не вводить никаких дополнительных конструкций, так как тип компонента есть порядковый номер его определения в макросе IEKomponents. Однако для описания более мобильных программ ИМ рекомендуется ввести перечисление следующего вида:

enum {KOMPONENT_1, KOMPONENT_2, ... , KOMPONENT_N},

где KOMPONENT_i – некоторый идентификатор *i*-го компонента в макросе IEKomponents.

ПРИМЕР 5.3.4. ИМ модель включает следующие компоненты:

1) генератор заявок с параметрами: интервал между заявками и частотное распределение вероятностей для определения типа заявки, время начала работы;

2) обслуживающий прибор с параметром «время обслуживания»;

3) поглотитель без параметров;

4) заявка с параметром «тип заявки».

Определить компоненты ИМ.

```
typedef struct {
        float Interval;
        float Frequency[10];
        float StartTime;
} ReqGnerator;
int m1[]={10};
IEGlobal (ReqGnerator) = {
        IEVar(интервал,FLOAT),
        IEArray(частота,FLOAT,m1)
        IEVar(ВремяНачала,FLOAT)
}
typedef struct {
        float ServiceTime;
} Server;
IEGlobal (Server) = {
        IEVar(ВремяОбслуживания,FLOAT)
}
typedef struct {
        int Type;
} Request;
IEGlobal (Request) = {
        IEVar(тип,INT)
}
IEKomponents = {
        IEKmp(ГенераторЗаявок,ReqGenerator,PRC),
        IEKmp(Сервер,Server,SRV),
        IEKmp0(Поглотитель,SRV),
        IEKmp(Заявка,Request,TZT)
}
```

```
enum {
        REQGENERATOR, SERVER, ABSORBER, REQUEST
};
```

5.3.5. Структура программы ИМ

Программу ИМ на языке программирования Си для СМ MICIC рекомендуется писать, придерживаясь нижеприведенной схемы: #inClude <MICIC.h>; определение (описание) глобальных данных ИМ (п. 5.2); описание параметров компонентов (п. 5.3); определение компонентов ИМ (п. 5.4); BEGIN (name); перечисление активностей; последовательность активностей ИМ; последовательность обязательных функций ИМ; определение массивов, активностей, очередей; END.

Рассмотрим те части схемы, которые не упоминались в предыдущих разделах. Функциональная часть программы ИМ ограничивается двумя макроопределениями BEGIN и END. Параметром макроса BEGIN является имя ИМ для ИС, т. е. последовательность не более 15 ASCII символов, не взятых в кавычки.

Под активностью будем понимать реализованный в виде функции средствами языка программирования Си некоторый алгоритм, аппроксимирующий определенные функциональные действия компонента и заканчивающийся вызовом функции преобразования списка событий. Выполнение активности приводит к свершению очередного события в ИМ, а следовательно, и к изменению состояния ИМ.

В СМ MICIC [4] цепочка активностей составляется либо для каждой пары: тип обслуживающего устройства – тип обслуживающегося транзакта, либо для каждого типа устройства – процесса. Способ обращения к данным ИМ является стандартным для языка Си. Рекомендуется определять активность по следующей схеме:

```
byte activity (void) {
        DServPar(Dtype,ptr1);
        TServPar(Ttype,ptr2);
        тело активности
        NextActivity(NUMActivity);
функция преобразования списка событый
        return InterruptionSign;
}
```

Активность возвращает признак прерывания процесса имитации InterruptionSign переопределенного типа byte. Этот признак может принимать одно из двух значений: CONTINUE или BREAK , соответственно продолжать или прервать моделирование. Если разработчик модели должен получить доступ к параметрам соответственно обслуживающего устройства

или обслуживаемого транзакта, то необходимо использовать макроопределения DServPar() и TServPar(). Переменные ptr1 и ptr2 есть указатели на параметры компонента, структура описания которых имеет переопределенные типы Dtype и Ttype (см. п. 5.3). Тело активности содержит раздел описания локальных переменных активности и раздел, состоящий из последовательности операторов языка Си, вызовов библиотечных функций языка Си, функций моделирования СС для СМ MICIC (см. п. 5.6), а также собственных функций разработчика ИМ. Эта последовательность аппроксимирует функциональную часть активности (т. е. совокупность функциональных действий, приводящих к появлению нового события в ИМ). Если id является идентификатором параметра ИМ, отклика ИМ, параметра компонента, локальной переменной активности, то их значения есть соответственно GPar.id, GRes.id, ptr->id, id.

На разработчика модели возложена также обязанность формирования последовательности переходов между активностями процесса «механизм обслуживания». Это достигается с помощью макроса NextActivity, чьим параметром является номер NUMActivity той активности, которая следует после выполняющейся активности компонента. Подчеркнем особо, что активность NUMACtivity является следующей после текущей в локальном времени описываемого компонента. Порядком смены активностей в модельном времени управляет подсистема имитации с помощью списка событий.

Разработчик ИМ должен с помощью функции преобразования списка событий подсказать только, когда следует вызвать активность NUMACtivity. Другими словами, этими функциями разработчик указывает, сколько единиц модельного времени требуется на реализацию функциональной части активности. Принадлежность функции к группе преобразующих список событий отдельно оговаривается в п. 5.6.

Осталось раскрыть вопрос о начале и завершении каждой цепочки активностей. Чтобы инициализировать цепочку, разработчик ИМ должен определить массив начальных активностей (см. ниже).Завершение цепочки зависит от того, сколько единиц модельного времени приходится на функциональную часть его конечной активности. Если она является единовременной, т. е. не требуется сдвига локальной временной координаты компонента, то макроопределение NextActyvity() для конечной активности отсутствует, а в качестве функции преобразования списка событий вызывается StopService(). Во всех остальных случаях в конечной активности указывается, что следующей выполняемой активностью будет StopService(), т. е. вставляется макроопределение NextActivity(STOPSERVICE).

Для устройств, работающих в режиме процесс, StopServiCe() заменяется на Stopprocess(), а STOPSERVICE на STOPPROCESS.

К обязательным функциям языка описания ИМ для СМ MICIC относятся следующие функции:

void Constructor(void); предназначена, как правило, для определения структуры модели и ее начального состояния. ИС СМ MICIC позволяет вызвать эту функцию единственный раз по нажатию клавиши F2 из главного меню.

void AfterStep(void); предназначена для определения функциональных действий, которые должны выполняться при каждой смене модельного времени.

void AfterStop(void); предназначена для определения функциональных действий, которые должны выполняться при каждой остановке имитационного эксперимента.

void PrintState(FILE*); предназначена для сохранения промежуточных данных о состоянии имитационной модели в виде текстового файла с именем, совпадающем с именем загрузочного модуля ИС ИМ, и расширением «.out». Вызывается каждый раз по нажатию клавиши F5 из главного меню ИС СМ MICIC.

void BaCKground(void); предназначена для создания собственного фона при работе в ИС СМ. Для корректной работы разработчику программы ИМ следует знать, что графическая система инициализирована в режиме VGAMED, т. е. 640x350 пикселов и 16 цветов. Причем пользователь должен рисовать изображение в прямоугольной области с координатами {(0; 14) – (639; 335)}.

За последовательностью обязательных функций располагаются определения следующих массивов: активностей; начальных активностей цепочек; очередей.

1. Массив активностей.

Его заголовком является макроопределение Activities, а его элементы представляют собой имена всех активностей. Первыми должны быть определены четыре активности из подсистемы имитации: EOSimulation, StartService, StopService, StopProcess. Таким образом, массив всех активностей выглядит следующим образом:

```
Activities = {
        EOSimulation, StartService, StopService, StopProcess,
        Activity_5,  Activity_6,  ... ,
        Activity_K
};
```

Как в случае с компонентами ИМ, для описания более мобильных программ ИМ при формировании цепочки активностей рекомендуется ввести перечисление следующего вида:

```
enum {
        EOSIMULATION, STARTSERVICE,
        STOPSERVICE, STOPPROCESS,
```



```
        ACTIVITY_5,  ACTIVITY_6,  ... ,
        ACTIVITY_K
};
```

где ACTIVITY__i – некоторый идентификатор i-й активности в макросе ACtivities. Подчеркнем, что согласно схеме это перечисление определяется до текста первой активности в программе ИМ.

2. Массив начальных активностей цепочек.

Элемент массива содержит три поля: номер начальной активности цепочки, номер типа обслуживающего устройства в макроопределении IEKomponents, номер типа обслуживающегося транзакта (там же). Вместо номера транзакта иногда можно ставить макросы PROCESS, ALL, OTHERS. Первый макрос PROCESS должен указываться для устройств, работающих в режиме «процесс». Макрос ALL экономит память, если все транзакты должны обслуживаться с помощью одной и той же цепочки. Наконец, пусть некоторые фиксированные типы транзактов взаимодействуют с устройством уникальным образом, что описывается стандартно. Остальные же типы транзактов должны обслужиться по одинаковой цепочке для всех активностей. Тогда на месте типа транзакта ставится макрос OTHERS.

Заголовком массива служит макроопределение InitialACtivities. За ним располагаются тройки макросов, если придерживаться данной методики. Порядок элементов массива практически не играет существенной роли. Некоторого ускорения процесса имитации можно достигнуть, расположив в начале массива элементы, соответствующие наиболее часто выполняемым механизмом обслуживания. Однако записи с третьим полем, равным OTHERS, рекомендуется располагать последними.

3. Массив очередей.

В СМ MICIC существует три встроенных механизма обслуживания очередей (т. е. способов нахождения того транзакта из очереди, который должен отправиться на обслуживание следующим): Fifo (первый пришел – первый обслужен), Lifo (последним пришел – первый обслужен), Priority (приоритетным является транзакт, имеющий минимальное значение некоторого своего параметра). Кроме этого, два и более устройств могут иметь одну и ту же очередь, называемую совместной. Одно из таких устройств должно обладать одним из трех встроенных или одним из определенных разработчиком ИМ механизмом обслуживания очередей. Остальные устройства извлекают транзакты, пользуясь функцией Joint.

Если разработчику ИМ достаточно стандартных возможностей СМ MICIC, то массив очередей выглядит следующим образом:

```
StandardQueues;
```

Каким образом определяются уникальные механизмы очередей и их массивы описано в п. 5.3.6.

сивы описано в п. 5.3.6.

ПРИМЕР 5.3.5. Привести схему программы ИМ для примера 5.3.1. Заметим, что компоненты и их параметры были определены в примере 5.3.1.

```
BEGIN (ТривиальнаяСМО)
enum {
        EOSIMULATION, STARTSERVICE,
        STOPSERVICE,   STOPPROCESS,
        GENERATOR1,   GENERATOR2,
        REQ_ON_SERVER, REQ_ON_ABSORBER
};
byte Generator1(void){
        DServPar(RegGenerator,ptr);
        // тело первой активности пусто
        NextActivity(GENERATOR2);
        функция изменения локальной временной
        координаты на время ptr -> StartTime
        return CONTINUE;
}
byte Generator2(void){
        DServPar(RegGenerator,ptr);
        //генерация очередной заявки
        NextActivity(GENERATOR2);
        // можно и не указывать, если очередная
        // активность совпадает с текущей
        функция изменения локальной временной
        координаты на время ptr -> Interval
        return CONTINUE;
}
byte Req_on_Server(void){
        DServPar(Server,ptr);
        указать,что направлением движения
        является устройство ABSORBER
        NextActivity(STOPSERVICE);
        функция изменения локальной временной
        координаты на время ptr -> ServiceTime
        return CONTINUE;
}

byte Req_on_Absorber(void){
        указать об окончании движения
        пересчет откликов ИМ
```



```
        StopService(ptr);
        return CONTINUE;
}
void Constructor(void) {
        . . . . .
}
void AfterStep(void){  }
void AfterStep(void){  }
void Background(void){  }
void PrintState(FILE*){  }

//определение массива активностей
Aktivities={
        EOSimulation, StartProcess, StopService,   StopProcess,
        Generator1,   Generator2,   Req_on_Server, Req_on_Absorber
};

//определение массива начальных активностей фрагментов
InitialActivities={
        GENERATOR1, REQGENERATOR, PROCESS,
        REQ_ON_SERVER,SERVER,REQUEST,
        REQ_ON_ABSORBER,ABSORBER,REQUEST
};

//определение массива очередей
StandardQueues;
END
```

5.3.6. Описание функций языка моделирования

1. Функция преобразования списка событий

Согласно методу имитационного моделирования, аппроксимированные активностями функциональные действия компонент, приводящие к некоторому событию в ИМ, реализуются в некоторый момент модельного времени. В свою очередь в СТС реальные функциональные действия происходят в непрерывном времени. Чтобы отразить функционирование компонент, каждая активность завершается вызовом функции, указывающей, сколько единиц модельного времени проходит от одного до другого события. В результате очередная активность перемещается вперед по списку событий и будет вызвана в требующийся момент модельного времени. Ее заголовок выглядит следующим образом:

void WaitTime(float Time);

Time – модельное время реализации активности.

К функциям, преобразующим список событий, относятся также некоторые частные случаи функций изменения структуры ИМ.

ПРИМЕР 5.3.6. Определить активность Generator1 примера 5.3.5.

```
byte Generator1(void){
    DServPar(RegGenerator,ptr);
    NextActivity(GENERATOR2);
    WaitTime(ptr->StartTime);
    return CONTINUE;
}
```

2. Доступ к компоненту ИМ и его параметрам

В подавляющем большинстве случаев параметрами приводимых в следующих пунктах функций являются либо сам активный компонент, либо парный компонент. Поэтому имеет смысл ввести последовательность макроопределений, облегчающих локализацию нужных элементов модели.

Tself – возвращает указатель (tns *) на активизированный транзакт.

DServ – возвращает указатель на обслуживающее устройство (точнее на устройство, очередная активность которого вызвана из подсистемы имитации).

TServ – возвращает указатель на обслуживающийся транзакт.

VServ – возвращает номер версии обслуживающего устройства.

Kmp(kk) – возвращает номер компонента в массиве IEKomponents для элемента KK, который может быть типа устройства или транзакта, т.е. dvc * или tns *.

KPar(type,kk) – возвращает указатель на параметры, определяемые структурой типа type (см. п. 5.3), элемента kk (см. предыдущий макрос); этот указатель может появляться в выражениях как с левой, так и с правой стороны от знака присваивания.

DHead(word kmp) – возвращает указатель (dvc *) на первое устройство в списке компонентов типа kmp; если список пуст, то возвращенное значение-NULL.

THead(word kmp) – возвращает указатель (tns *) на первый транзакт в списке компонентов типа kmp; если список пуст, то возвращенное значение-NULL.

Next(kk) – возвращает указатель на элемент, следующий за элементом kk (типа dvc * или tns *) в списке компонентов того же типа, что и kk; если kk – последний элемент списка, то возвращенное значение – NULL.

Device(word kk, word num) – возвращает указатель (dvc *) на устройство типа kk с номером версии num. Если такого устройства нет, то возвращается NULL.

Serving(tns *tt) - возвращает указатель (dvc *) для транзакта tt.

3. Функции выбора транзактом маршрута

MoveTo(dvc *dd)

Это макроопределение модифицирует встроенный в транзакт указатель направления движения так, что транзакт, покинув обслуживающее устройство, станет в очередь к устройству dd.

Если необходимо, чтобы транзакт поглотился СМ по окончанию обслуживания на устройстве, то это макроопределение используется следующим образом:

Moveto(NULL)

dvc *Movement(tns *tt)

С помощью данного макроопределения пользователь получает доступ к встроенному в транзакт tt указателю направления движения. Может быть использовано как с левой, так и с правой стороны от оператора присваивания.

4. Функции изменения состояния ИМ

void Instal(word Kmp,word VersionQuantity,float InitTime,
word Maxtns,byte QueType,void *ptr);

Функция Instal вводит в ИМ VersionsQuantity версий устройств (word определен как unsigned int) типа Kmp (номер в массиве IEKomponents). Для устройств, работающих в режиме, процесс InitTime есть модельное время активизации процесса, а остальные два параметра не имеют значения. Для обслуживающих устройств, наоборот, InitTime не играет роли, а максимально возможное количество обслуживающихся одновременно транзактов задается параметром Maxtns. Механизм обслуживания очередей к устройству представляет собой номер QueType в массиве очередей (см. п. 5.6). Указатель Ptr содержит адрес, начиная от которого, находятся начальные значения параметров устройств. Пример записи:

void Run(word Kmp,word CopiesQuantity,dvc *dd, byte Priority, word buflen, void *ptr);

По данной функции генерируется подсистемой имитации CopiesQuantity новых транзактов, принадлежащих компоненту ИМ с номером Kmp в массиве IEKomponents. Они направляются на обслуживание к устройству dd, имея внутренний приоритет Priority. Указатель ptr содержит адрес, начиная от которого находятся начальные значения параметров рожденных транзактов. Параметр buflen задает объем транзактного буфера для генерируемых транзактов.

word OpenDvc(word num, dvc *dd);

Перевести num версий устройства типа Kmp(dd), начиная с dd, работающих в режиме обслуживания , в открытое состояние. Возвращается реальное количество переведенных в указанное состояние версий.

word CloseDvc(word num, dvc *dd);

Перевести num версий устройства типа Kmp(dd), начиная с dd, работающих в режиме обслуживания, в закрытое состояние. Возвращается реальное количество переведенных в указанное состояние версий.

word ActivDvc(word num, dvc *dd);

Перевести num версий устройства типа Kmp(dd), начиная с dd, работающих в режиме обслуживания, в активное состояние. Возвращается реальное количество переведенных в указанное состояние версий.

word PassivDvc(word num, dvc *dd);

Перевести num версий устройства типа Kmp(dd), начиная с dd, работающих в режиме обслуживания, в пассивное состояние. Возвращается реальное количество переведенных в указанное состояние версий.

word DelayTns(word num, tns *tt);

Транзакты, начиная от tt (num копий), и обслуживающие их устройства приостанавливают выполнение своих активностей.

word ReleaseTns(word num, tns *tt);

Транзакты, начиная от tt (num копий), и обслуживающие их устройства продолжают выполнение своих активностей с прерванного места.

word LeaveTns(word num, tns *tt);

Транзакты, начиная от tt (num копий), покидают устройства, на которых они обслуживались.

word RestartTns(word num, tns *tt);

Транзакты, начиная от tt (num копий), должны повторить обслуживание сначала на тех устройствах, где находятся.

word KillTns(word num, tns *tt);

Уничтожить транзакты, начиная от tt (num копий).

ЗАМЕЧАНИЕ. Если вызов функций DelayTns(1,TServ), LeaveTns(1, TServ), RestartTns(1,TServ), KillTns(1,TServ), PassivDvc(1, DServ) следует в приведенном контексте, то они изменяют список событий. Поэтому перед ними необходимо указывать следующую активность, а после нее ставить оператор return (см. п. 5.5).

5.Функции изменения состояния транзактного буфера

word BufferVolume(tns *tt);

Возвращает объем транзактного буфера для транзакта tt.

tns *Place(tns *tt, word i);

Возвращает транзакт, находящийся на месте *i* в транзактном буфере

транзакта tt.

word FindPlace(tns *tt);

Возвращает номер свободного места в транзактном буфере транзакта tt. Если все места заняты, то возвращается MAXUINT.

word TakePlace(tns *tt, que *qq, dvc *dd);

Транзакт, находящийся внутри элемента qq очереди к устройству dd, размещается в транзактном буфере транзакта tt. Если все места заняты, то возвращается MAXUINT.

byte FreePlace(tns *tt, word i, dvc *dd);

Транзакт, находящийся на месте *i* в транзактном буфере транзакта tt, отправляется в очередь к устройству dd.

6.Функции генерации псевдослучайных потоков

double rnd(void);

Возвращает псевдослучайные числа, равномерно распределенные на полуинтервале [0;1).

double Urnd(double m,double h);

Возвращает псевдослучайные числа, равномерно распределенные на полуинтервале [m-h,m+h).

double Nrnd(double m,S2);

Возвращает псевдослучайные числа, нормально распределенные со средним *m* и среднеквадратичным отклонением *S*2.

double Ernd(double m);

Возвращает псевдослучайные числа, экспоненциально распределенные со средним *m*.

double Hi2rnd(int ν);

Возвращает псевдослучайные числа, χ^2-распределенные с ν степенями свободы.

double Brnd(int ν1, int ν2);

Возвращает псевдослучайные числа, бета-распределенные с ν1 и ν2 степенями свободы.

double Frnd(int ν1, int ν2);

Возвращает псевдослучайные числа, *F*-распределенные с ν1 и ν2 степенями свободы.

double Trnd(int ν);

Возвращает псевдослучайные числа, *t*-распределенные с ν степенями свободы.

unsigned UCrnd(unsigned k);

Возвращает псевдослучайные числа, выбираемые из множества 0...*k* – 1 с

равными вероятностями.

unsigned WCrnd(unsigned k,double p[]);

Возвращает псевдослучайные числа, выбираемые из множества 0...k – 1 с вероятностями p[0]...p[k – 1],где сумма всех значений массива p равна 1.

unsigned Grnd(double p);

Возвращает псевдослучайные числа согласно геометрическому распределению с вероятностью p.

unsigned Prnd(double m);

Возвращает псевдослучайные числа согласно распределению Пуассона со средним m.

7. Справочные функции

Для оперативного контроля за развитием процессов в модели можно использовать следующий набор функций.

float ModelTime(void);

Функция возвращает текущее значение модельного времени.

word MaxTD (dvc *dd);

Функция возвращает максимально возможное количество транзактов на устройстве dd.

word NumTD (dvc* dd , word kk);

Функция возвращает количество транзактов типа kk на устройстве dd. Если второй параметр равен MAXUINT, то возвращается общее количество транзактов.

word NumTQ (dvc *dd , byte num , word kk);

Функция возвращает количество транзактов типа kk в очереди с номером num к устройству dd. Если последний параметр равен MAXUINT, то возвращается общее количество транзактов в очереди.

float BirthTime (tns *tt);

Функция возвращает значение времени рождения копии транзакта tt.

float ServingTime (tns *tt);

Функция возвращает значение времени начала обслуживания или прибытия в очередь транзакта tt.

char *Name (word kk);

Функция возвращает строковое имя по номеру компонента в массиве IEKomponents.

word Version (dvc *dd);

Функция возвращает номер версии устройства dd.

word Priority(tns *tt);

Данное макроопределение дает доступ к приоритету транзакта и может быть использовано с любой стороны от знака присваивания.

6. ОСНОВЫ ПРИНЯТИЯ РЕШЕНИЙ В ЗАДАЧАХ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ

6.1. Виды стратегий исследователя при анализе процессов, протекающих в СТС

Рассмотренные ранее типы математических моделей представляют собой как инструмент анализа ситуаций в СТС, так и основное средство исследований при проектировании СТС. Управление СТС с помощью ММ, проектирование устройств или компонентов СТС предполагают, как правило, достижение определённой цели. Для этого исследователям или проектировщикам СТС приходится принимать определённые решения. Однако только в простейших случаях удаётся указать шкалу (целевую функцию), значения которой измеряют качество решения. В сложных ситуациях качество решения не может быть оценено единственной функцией. Поэтому механизм рационального выбора в таких случаях требует некоторой дополнительной косвенной информации, позволяющей сравнивать возможные альтернативы решений. Очень часто возникает необходимость принимать решения, для которых не удаётся полностью учесть предопределяющие их условия, а также последующее их влияние (эффект неопределённости) на функционирование СТС. Например, любое планирование производства обычно связано с подобными факторами неопределенности. В таких ситуациях ответственность за принимаемые решения очень велика. Поэтому исследователи стремятся к оптимальному использованию имеющейся информации относительно поставленной задачи, чтобы, взвесив все возможные варианты решения, постараться найти среди них наилучший. Учёт возможных последствий от принятия решений на основе математического моделирования процессов в СТС определяет актуальность изучения основ принятия решений в составе курса» Исследование операций».

Обычно исследователи могут придерживаться разных стратегий при анализе процессов, протекающих в СТС. В зависимости от имеющейся в их распоряжении осведомительной информации и от наличия ресурсов для организации исследований возможны разные комбинации вариантов анализа СТС (стратегий изучения СТС и оценки последствий от принятия решений). На выбор стратегии исследования операций влияют следующие основные факторы:

– наличие информации об исследуемом процессе;

– величина риска потерь от ошибочных решений;

– размеры ресурсов для организации исследований на ММ;

– количество и иерархия целей, преследуемых исследователем при выборе решения.

Если информации о СТС недостаточно, то имеет место принятие решений в условиях неопределённости. Когда же при этом необходимо учитывать возможный риск от ошибочных решений, то перед исследователем стоит дополнительная задача – принятия решений в условиях риска. Если количество целей большое и они зачастую противоречат друг другу, то исследователю приходится иметь дело с многоцелевым выбором решений. Как правило, наиболее трудными являются задачи векторной оптимизации, решение которых предполагает нахождения оптимального соотношения параметров управления СТС. Наконец, размеры ресурсов, которыми располагает исследователь при использовании ММ ИСО, играют существенную роль при выборе решения. На практике при решении большинства практических задач ИСО исследователи ощущают недостаток ресурсов для постановки исследований, анализа результатов и принятия обоснованного решения.

Исходя из вышеизложенного, в данном разделе рассмотрены наиболее часто используемые традиционные методы принятия решений, ставшие уже классическими. По нашему мнению, без владения основами принятия решений, излагаемыми ниже, методики исследования операций будут неэффективными и слабо используемыми на практике. Поэтому курс исследования операций завершается элементарными основами по принятию решений. Учитывая ориентацию данного пособия на подготовку специалистов технических университетов, основное внимание уделяется наиболее употребительным алгоритмам принятия решений.

6.2. Принятие решений в условиях неопределенности и риска

6.2.1. Основная формальная структура принятия решений. Матрица решений. Оценочная функция

Принятие решений представляет собой выбор одного из некоторого множества вариантов: $E_i \in E$. Условимся, что каждый вариант E_i вырабатывает некоторую количественную оценку e_i. Будем искать вариант решения с наибольшим значением e_i, полагая, что e_i характеризует такие величины, как полезность, надежность, выигрыш, прибыль. Таким образом, выбор оптимального варианта производится с помощью критерия

$$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \max_i e_i\}. \qquad (6.1)$$

Рассмотренный случай, когда каждому варианту решения соответствует единственное внешнее состояние (случай детерминированных решений), с точки зрения его применения является простейшим.

При решении большинства практических задач каждому допустимому варианту решения E_i могут соответствовать вследствие различных внешних условий различные внешние состояния F_j и количественные оценки решений

e_{ij}. Рассмотрим пример, иллюстрирующий подобную ситуацию.

Пусть требуется изготовить изделие, долговечность которого зависит от вида материала, из которого оно состоит, и внешних условий, связанных с той или иной степенью нагрузки при эксплуатации изделия. Нагрузки считаются известными. Требуется определить вид материала, из которого целесообразно изготовить изделие .

Варианты решений в данном примере таковы:

E_1 – выбор вида материала из соображений максимальной долговечности;

E_m – выбор вида материала из соображений минимальной долговечности;

E_i – промежуточные решения ($i = \overline{2, m-1}$).

Внешние условия, учет которых необходим, следующие:

F_1 – условия, обеспечивающие максимальную долговечность;

F_n – условия, обеспечивающие минимальную долговечность;

F_j – промежуточные условия ($j = \overline{2, n-1}$).

Под результатом решения e_{ij} будем понимать оценку, соответствующую варианту решения E_i и условию F_j и характеризующую экономический эффект (прибыль), полезность или надежность изделия.

Ситуация, соответствующая описанному примеру, характеризуется следующей матрицей решений $\|e_{ij}\|$:

	F_1	F_2	…	F_n
E_1	e_{11}	e_{12}	…	e_{1n}
E_2	e_{21}	e_{22}	…	e_{2n}
…	…	…	…	…
E_m	e_{m1}	e_{m2}	…	e_{mn}

По данной матрице необходимо выбрать тот вариант решения, которому соответствует наилучший результат, но так как неизвестно, какое из внешних условий может наступить, необходимо принимать во внимание все оценки e_{ij}. Таким образом, первоначальная задача максимизации согласно критерию (6.1) должна быть теперь заменена другой.

Процедура выбора в случае нескольких внешних состояний может быть представлена по аналогии с применением критерия (6.1). При этом матрица решений $\|e_{ij}\|$ дополняется некоторым столбцом, т. е. каждому варианту E_i приписывается некоторый результат e_{ir}. Проблема в том, какой смысл вложить в результат e_{ir}. Оценочные функции можно вводить различным образом. Если, например, последствия каждого из альтернативных решений характеризовать комбинацией из его наибольшего и наименьшего результатов, то можно принять:

$$e_{ir} = \min_j e_{ij} + \max_j e_{ij}, \qquad (6.2)$$

наилучший в этом смысле результат имеет вид

$$\max_i e_{ir} = \max_i \{\min_j e_{ij} + \max_j e_{ij}\}. \quad (6.3)$$

Определяя таким образом желаемый результат, *лицо, принимающее решение* (ЛПР), исходит из компромисса между оптимистическим и пессимистическим подходами.

Целесообразность применения той или иной оценочной функции определяется комплексом условий. Приведем некоторые другие примеры оценочных функций.

Оптимистическая позиция:

$$\max_i e_{ir} = \max_i \{\max_j e_{ij}\}, \quad (6.4)$$

следуя которой, ЛПР становится на точку зрения азартного игрока, делая ставку на то, что при любом его решении внешняя среда будет находиться в максимально благоприятном состоянии.

Позиция нейтралитета:

$$\max_i e_{ir} = \max_i \{\frac{1}{n}\sum_{j=1}^{n} e_{ij}\}; \quad (6.5)$$

в этом случае ЛПР исходит из того, что все встречающиеся отклонения результата решения от «среднего» случая допустимы, и выбирает решение, оптимальное с этой точки зрения.

Позиция пессимиста:

$$\max_i e_{ir} = \max_i \{\min_j e_{ij}\} \quad (6.6)$$

ориентируется на то, что выпадет наименее благоприятный случай и приписывает каждому из альтернативных вариантов наихудший из возможных результатов. После этого ЛПР выбирает самый выгодный вариант, т. е. ожидает наилучшего результата при наихудшем состоянии внешней среды.

Позиция относительного пессимизма:

$$\min_i e_{ir} = \min_i \max_j \{\max_i e_{ij} - e_{ij}\}; \quad (6.7)$$

в этом случае для каждого варианта решения ЛПР оценивает потери по сравнению с определенным по каждому варианту наилучшим результатом, а затем из совокупности наихудших результатов выбирает наилучший согласно данной оценочной функции.

Влияние исходной позиции ЛПР на эффективность результатов можно интерпретировать, исходя из наглядных представлений. Простейшим здесь является графическое изображение на плоскости, соответствующее двум

внешним состояниям при m вариантах решения.

Введем прямоугольную систему координат. По оси абсцисс отложим значения результатов решений e_{i1}, соответствующие внешнему состоянию F_1, а по оси ординат – значения e_{i2}, соответствующие состоянию F_2.

Каждый вариант решения E_i, таким образом, соответствует точке (e_{i1}, e_{i2}), $i=\overline{1,m}$ на плоскости. Все m точек (e_{i1}, e_{i2}) лежат внутри прямоугольника, стороны которого параллельны координатным осям, а противоположные вершины – утопическая точка с координатами ($\max e_{i1}$, $\max e_{i2}$) и антиутопическая точка с координатами ($\min e_{i1}$, $\min e_{i2}$), $i = \overline{1,m}$. Данный прямоугольник называется полем полезности решений.

Чтобы сравнить варианты решений с точки зрения их качества, назовем вариант E_i не худшим, чем вариант E_j, если для соответствующих точек (e_{i1}, e_{i2}) и (e_{j1}, e_{j2}) выполняются неравенства: $e_{i1} \geq e_{j1}$ и $e_{i2} \geq e_{j2}$, причем решение E_i считается лучшим, если хотя бы одно из двух неравенств является строгим.

Очевидно, что при таком определении не все варианты решения допускают сравнение друг с другом, так как в общем случае существуют варианты решений E_i и E_j, такие, что, например, $e_{i1} < e_{j1}$ и $e_{i2} > e_{j2}$. Это означает, что в поле полезности решений установлено отношение частичного порядка.

Выберем в поле полезности некоторую точку РТ. С помощью прямых, параллельных координатным осям, разобьем плоскость на четыре части и обозначим их ***I***, ***II***, ***III***, ***IV***. В случае произвольной размерности эти части будем называть *конусами*. Все точки конуса ***I*** в смысле введенного выше порядка являются лучшими, чем точка РТ, и, соответственно, все точки конуса ***III*** являются заведомо худшими, чем точка РТ. Поэтому конус ***I*** называется *конусом предпочтения*, а конус ***III*** – *антиконусом*. Оценка же точек конусов ***II*** и ***IV*** является неопределенной, поэтому эти конусы носят название *конусов неопределенности*. Для точек конусов неопределенности оценки получаются только с помощью выбранного критерия принятия решений (рис. 6.1).

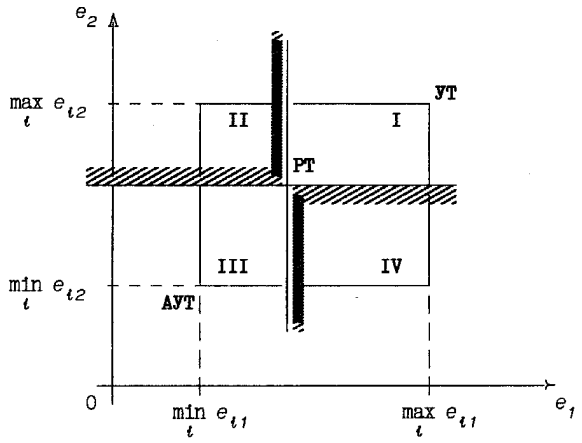

Рис. 6.1. Поле полезности решений

6.2.2. Критерии принятия решений в условиях неопределенности и риска. Анализ ситуации принятия решений

Минимаксный критерий (ММ-критерий). Этот критерий использует оценочную функцию, соответствующую позиции крайнего пессимизма:

$$Z_{MM} = \max_i e_{ir},\ e_{ir} = \min_j e_{ij}, \tag{6.8}$$

т. е. множество оптимальных решений E_0 определяется соотношением

$$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \max_i \min_j e_{ij}\}. \tag{6.9}$$

Выбранные таким образом варианты полностью исключают риск. Однако это достоинство стоит некоторых потерь. Применение ММ-критерия бывает оправдано, если ситуация характеризуется обстоятельствами:

– о возможности появления состояний F_j ничего не известно;

– решение реализуется один или очень малое число раз;

– необходимо исключить какой бы то ни было риск.

Критерий Севиджа (*S*-критерий). Оценочная функция критерия Севиджа имеет вид

$$Z_S = \min_i e_{ir} = \min_i \max_j \{\max_i e_{ij} - e_{ij}\} \tag{6.10}$$

и множество оптимальных вариантов решения строится следующим образом:

$$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \min_i e_{ir}\}. \tag{6.11}$$

Для понимания величины $a_{ij} = \max_i e_{ij} - e_{ij}$ нужно трактовать как дополнительный выигрыш, если вместо варианта E_i в состоянии F_j выбрать другой, оптимальный для этого внешнего состояния результат.

Условия для применения критерия Севиджа такие же как и для ММ-критерия.

Критерий Байеса-Лапласа (*BL*-критерий). Пусть q_j – вероятность появления внешнего состояния F_j, тогда для критерия Байеса-Лапласа оценочная функция примет вид

$$Z_{BL} = \max_i e_{ir},\ e_{ir} = \sum_{j=1}^{n} e_{ij} q_j\ , \tag{6.12}$$

т. е. $$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \max_i \sum_{j=1}^{n} e_{ij} q_j\ \wedge \sum_{j=1}^{n} q_j = 1\}.$$

Применение критерия рекомендуется, если ситуация характеризуется следующим образом:

– вероятности появления состояний F_j известны и не зависят от времени;

– решение реализуется (теоретически) бесконечно много раз;

– для малого числа реализаций решения допускается некоторый риск.

Критерий Ходжа-Лемана (*HL*-критерий). Этот критерий опирается на *BL*-критерий и *MM*-критерий. С помощью параметра ν выражается степень доверия к использованному распределению вероятностей. Если это доверие велико, то акцентируется *BL*-критерий, в противном случае предпочтение отдается *MM*-критерию.

Оценочная функция определяется равенством

$$Z_{HL} = \max_i e_{ir},$$

$$e_{ir} = \nu \sum_{j=1}^{n} e_{ij} q_j\ + (1-\nu) \min_i e_{ij},\ 0 \le \nu \le 1, \tag{6.13}$$

т. е. $$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \max_i [\nu \sum_{j=1}^{n} e_{ij} q_j\ + (1-\nu) \min_i e_{ij}] \wedge 0 \le \nu \le 1\}.$$

Критерий Ходжа-Лемана предъявляет к ситуации принятия решения следующие требования:

– вероятности появления состояний F_j неизвестны, но некоторые предположения о распределении вероятностей возможны;

– принятое решение теоретически допускает бесконечно много реализаций;

– при малых числах реализаций допускается некоторый риск.

Критерий Гурвица (*HW*-критерий). Стараясь занять наиболее уравновешенную позицию, ЛПР также может воспользоваться критерием Гурвица, оценочная функция которого находится как средневзвешенное между точками зрения предельного оптимиста и крайнего пессимиста:

$$Z_{HW} = \max_i e_{ir},$$

$$e_{ir} = c \min_i e_{ij} + (1-c) \max_j e_{ij},\ 0 \le c \le 1, \tag{6.14}$$

т. е. $$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \max_i [c \min_i e_{ij} + (1-c) \max_j e_{ij}] \wedge 0 \le c \le 1\}.$$

Чаще всего весовой множитель $c = 0{,}5$. Критерий предъявляет к ситуации принятия решений следующие требования:

– о вероятности появления состояний ничего не известно;

– решение реализуется лишь малое количество раз;

– допускается некоторый риск.

Критерий Гермейера (*G*-критерий). Критерий Гермейера ориентирован на величины потерь, т. е. при его применении предполагается, что e_{ij} – отрицательные.

В качестве оценочной функции *G*-критерия выступает

$$Z_G = \max_i e_{ir},$$

$$e_{ir} = \min_i e_{ij} q_j. \qquad (6.15)$$

G-критерий имеет следующее множество решений:

$$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \max_i \min_i e_{ij} q_j \wedge e_{ij} < 0\}.$$

Поскольку, например, при решении целого ряда производственных и экономических задач преимущественно имеют дело с ценами и затратами, то условие отрицательности оценок e_{ij} обычно выполняется. Если среди e_{ij} имеются положительные величины, то путем преобразования $e_{ij} - a$ при подходящем выборе $a > 0$ матрица решений преобразуется к отрицательному виду, однако следует учитывать, что оптимальное решение может зависеть от величины a.

G-критерий некоторым образом обобщает *MM*-критерий, а в случае равномерного распределения q_j ($q_j = 1/n, j = \overline{1,n}$) они становятся идентичными.

Условия применимости *G*-критерия таковы:

– вероятности появления состояний F_j известны;

– допускается некоторый риск;

– решение может реализовываться как малое, так и большое число раз.

Если функция распределения известна с малой надежностью, и числа реализаций малы, то при использовании *G*-критерия, вообще говоря, имеется неоправданно большой риск.

Составной *BL*(*MM*)-критерий. Стремление получить критерии, которые бы лучше приспосабливались к имеющейся ситуации, чем все до сих пор рассмотренные, привело к построению так называемых составных критериев. Исходным для построения данного был *BL*-критерий. Вследствие того, что распределение $\boldsymbol{q} = (q_1, ..., q_n)$ устанавливается эмпирически и потому известно не точно, происходит, с одной стороны, ослабление критерия, а с помощью заданных границ для риска и посредством *MM*-критерия обеспечивается соответствующая свобода действий.

Зафиксируем прежде всего задаваемое *MM*-критерием опорное значение

$$Z_{MM} = \max_i \min_j e_{ij} = e_{i_0 j_0},$$

где i_0, j_0 – оптимизирующие индексы для рассматриваемых вариантов решений и, соответственно, состояний.

Посредством некоторого заданного или выбираемого уровня допустимого риска определим некоторое множество согласия, являющееся подмножеством множества индексов $\{i, ..., m\}$:

$$I_1 = \{i \mid i \in \{i, ..., m\} \wedge (e_{i_0 j_0} - \min_j e_{ij}) \le \varepsilon_{\text{доп}}\}. \qquad (6.16)$$

Величина $\varepsilon_i = e_{i_0 j_0} - \min_j e_{ij}$ для всех $i \in I_1$ характеризует наибольшие возможные потери по сравнению со значением, задаваемым *MM*-критерием. С другой стороны, в результате такого снижения открываются возможности для увеличения выигрыша по сравнению с тем, который обеспечивается *MM*-критерием.

Поэтому мы рассматриваем также некоторое выигрышное подмножество

$$I_2 = \{i \mid i \in \{i, ..., m\} \wedge (e_{i_0 j_0} - \min_j e_{ij}) = \varepsilon_i \le \max_j e_{ij} - \max_j e_{i_0 j}\}.$$

Тогда в множество-пересечение $I_1 \cap I_2$ соберутся только такие варианты решений, для которых, с одной стороны, в определенных состояниях могут иметь место потери по сравнению с состоянием, задаваемым *MM*-критерием, но зато в других состояниях имеется по меньшей мере такой же прирост выигрыша. Теперь оптимальными в смысле составного *BL*(*MM*)-критерия будут решения из множества

$$E_0 = \{E_{i0} \mid E_{i0} \in E \wedge e_{i0} = \max_{i \in I_1 \cap I_2} \sum_{j=1}^{n} e_{ij} q_j \wedge \sum_{j=1}^{n} q_j = 1\}.$$

Применение *BL*(*MM*) критерия бывает целесообразным, если:

– вероятности появления состояний F_j неизвестны, однако имеется некоторая априорная информация в пользу какого-либо определенного распределения;

– необходимо считаться с появлениями различных состояний как по отдельности, так и в комплексе;

– допускается ограниченный риск;

– принятое решение реализуется один раз или многократно.

BL(*MM*)-критерий хорошо приспособлен для построения практических решений прежде всего в области техники и может считаться достаточно надежным.

Однако задание границы риска и, соответственно, оценок риска не учитывает ни число применений решения, ни иную подобную информацию. Условие $\varepsilon_i \le \max_j e_{ij} - \max_j e_{i_0 j}$ существенно в тех случаях, когда решение реализуется один или малое число раз.

6.2.3. Пример применения классических и производных критериев в задаче принятия решений в условиях неопределенности

Пусть некоторую технологическую установку требуется подвергнуть проверке с приостановкой ее эксплуатации. Из-за этого на некоторое время будет, естественно, приостановлен и выпуск продукции. Если же существующая неисправность не будет вовремя обнаружена, то это приведет к еще

большим потерям, поскольку технологическая установка выйдет из строя.

У руководства предприятия есть возможность выбора одного из следующих альтернативных вариантов решения:

E_1 – осуществить полную проверку оборудования с привлечением специалистов-ремонтников со стороны;

E_2 – провести проверку и возможный ремонт своими силами;

E_3 – вообще отказаться от какой-либо проверки и не приостанавливать выпуск продукции.

После длительного срока эксплуатации установка может находиться в одном из следующих состояний:

F_1 – неисправностей нет, и установка может продолжать работать без какого-либо ремонта;

F_2 – требуется незначительный ремонт отдельных деталей;

F_3 – дальнейшая эксплуатация установки возможна лишь после капитального ремонта.

Накопленный на предприятии опыт позволил составить следующую матрицу решений, элементы которой отрицательны, поскольку включают в себя затраты на проверку и устранение неисправностей, а также затраты, связанные с потерями в выпускаемой продукции и с поломкой установки:

	F_1	F_2	F_3
E_1	–20	–22	–25
E_2	–14	–23	–31
E_3	0	–24	–40

Применяя *MM*-критерий, получаем, что следует проводить полную проверку: $E_0 = E_1$. Этого и следовало ожидать, так как данный критерий соответствует позиции крайнего пессимиста и исключает какой-либо риск, который в данной ситуации при отсутствии информации о вероятностях возможных состояний установки сопряжен, например, с ее поломкой в случае отказа от проверки и продолжения ее эксплуатации при имеющихся серьезных неисправностях.

Если предположить, что все возможные состояния установки равновероятны ($q = 1/3$), то при применении *BL*-критерия будет рекомендовано решение E_3 – отказ от проверки. Если применить *S*-критерий, то в качестве оптимального будет рекомендовано принять решение E_2 – провести проверку оборудования без привлечения специалистов со стороны.

Итак, воспользовавшись теоретическими рекомендациями, мы мало что выиграли, поскольку ситуация осталась неопределенной: каждый из критериев рекомендует свой вариант решения. Но следует помнить о том, что различные критерии связаны с различными аспектами ситуации, в которой решение принимается. Поэтому прежде чем воспользоваться тем или иным критерием, необходимо тщательно проанализировать ситуацию принятия

решения и только потом выбрать подходящий критерий. Если принимаемое решение относится к сотням работающих установок с одинаковыми параметрами и если информация о вероятностях состояний F_j достаточно точна, то целесообразно воспользоваться *BL*-критерием. Если число реализаций решения на практике невелико, то больший вес приобретают более осторожные рекомендации *S*- или *MM*-критериев.

Если рассмотреть ситуацию, когда состояние F_3 – серьезная неисправность установки – наиболее вероятно, например $q_1 = q_2 = 1/4$, $q_3 = 1/2$, то тогда и *BL*-критерий и *MM*-критерий рекомендуют провести полную проверку установки.

Применяя производные критерии для принятия решения по данной проблеме, получим следующие результаты:

Критерий Гурвица. При $c = 0{,}5$ рекомендуется отказаться от проверки (E_3). При $c > 0{,}57$ в качестве рекомендуемого будет выступать уже решение E_1.

Критерий Ходжа-Лемана. При $\nu = 0{,}5$ и $q_1 = q_2 = q_3 = 1/3$ по *HL*-критерию рекомендуется воспользоваться решением E_1 – выполнить полную проверку установки. Лишь при $\nu > 0{,}94$ рекомендуются менее осторожные варианты решений – E_2 или E_3.

Критерий Гермейера. Также рекомендует в случае равномерного распределения состояний установки придерживаться более осторожного варианта решения E_1.

Составной *BL*(*MM*)-критерий. Данный критерий является одним из наиболее гибких критериев и довольно часто может применяться на практике при решении конкретных технических задач. *BL*(*MM*)-критерий при $q_1 = q_2 = q_3 = 1/3$ в большинстве случаев при незначительном уровне допустимого риска также указывает на осторожный вариант E_1, как на оптимальный. Вариант E_3 (отказ от проверки) принимается этим критерием лишь при $\varepsilon_{доп} > 15$, однако во многих технических и хозяйственных задачах уровень допустимого риска бывает намного ниже, составляя лишь незначительный процент от возможных затрат.

6.3. Принятие решений в многоцелевых задачах производства и планирования деятельности предприятий

6.3.1. Традиционные методы принятия решений в многокритериальных задачах

При решении большинства задач проектирования, планирования и управления техническими и экономическими системами возникает необходимость оптимизации этих систем по совокупности противоречивых критериев эффективности их функционирования.

Такая оптимизация получила название векторной или многокритериальной. Ее отличительной особенностью является наличие не одного оптимального решения, как в задачах с одним критерием эффективности, а целого множества недоминируемых решений (множества Парето), каждое из которых может быть выбрано в качестве оптимального. Центральная проблема задач векторной оптимизации – выбор одного «оптимального в некотором смысле» решения. Объективный выбор одного решения из множества недоминируемых (компромиссных) решений невозможен без участия ЛПР, без получения от него информации о его предпочтениях.

В общем виде задачи векторной оптимизации могут быть записаны следующим образом:

$$\begin{aligned} &f_i(x) \to \max,\ i = \overline{1,k}\ , \\ &f_i(x) \to \min,\ i = \overline{k+1,m}\ , \\ &x \in G \subset \Re^m. \end{aligned} \qquad (6.17)$$

Понятие оптимального решения заменяется для таких задач понятием *эффективного решения*. Решение представляет собой эффективное решение многокритериальной задачи, если не существует решения, не уступающего ему по всем критериям и превосходящего его хотя бы по одному из них.

Рассмотрим некоторые часто применяемые на практике методы многокритериальной оптимизации.

Метод выделения главного критерия. Определяется главный критерий (предположим $f_1(x)$) и задача (6.17) преобразуется в следующую:

$$\begin{aligned} &f_1(x) \to \max, \\ &f_i(x) \ge f_i^*,\ i = \overline{2,k}\ , \\ &f_i(x) \le f_i^*,\ i = \overline{k+1,m}\ , \\ &x \in G \subset \Re^m. \end{aligned}$$

Метод последовательных уступок. Критерии эффективности располагаются в порядке уменьшения степени важности: f_{i1}, f_{i2}, ..., f_{in}. Допустим, что соответствующая нумерация была осуществлена в самом начале при постановке задачи (6.17) и, кроме того, допустим, что для всех i $f_i(x) \to \max$. Алгоритм получения решения сводится к следующему. Вначале находится решение, обращающее в максимум главный критерий f_1. Затем из практических соображений назначается некоторая «уступка» Δf_1. Требуя выполнения неравенства $f_1 \ge f_1^* - \Delta f_1$, где $f_1^* = \max f_1$, находим такое решение x, при котором $f_2(x) \to \max$. Далее снова назначается «уступка» по критерию f_2, с помощью которой можно максимизировать f_3 и т. д.

Метод «составного» критерия. ЛПР определяет важность каждого критерия, а дисперсия выражается весом критерия f_i. Затем формулируется составной критерий:

$$u(x) = \sum_{i=1}^{m} \gamma_i f_i(x) \to \max, \qquad (6.18)$$

где γ_i – вес i-го критерия, $\gamma_i > 0$, если $f_i(x) \to \max$, $\gamma_i < 0$, если $f_i(x) \to \min$.

Несмотря на удобную форму записи, «составные» критерии имеют существенные недостатки, связанные с произволом в выборе весов γ_i, а также с тем фактом, что недостатки эффективности по одним критериям могут компенсироваться за счет преимуществ по другим критериям.

Нормативные методы векторной оптимизации. Нормативные методы являются своего рода обобщением рассмотренных выше методов и состоят в предварительном получении нормативов ξ_{f_i}, $i = \overline{1,m}$ на основе приближенного решения многоцелевой задачи и приближения к этим нормативам по некоторой заданной метрике $\rho(f(x), \xi_f) \to \min$, где $\rho(f(x), \xi_f)$ может быть определено различными способами, например:

$$\rho_1^2(f(x),\xi_f) = \sum_{i=1}^{m} \left[f_i(x) - \xi_{fi} \right]^2 ;$$

$$\rho_2(f(x),\xi_f) = \sum_{i=1}^{m} \left| f_i(x) - \xi_{fi} \right| ;$$

$$\rho_3(f(x),\xi_f) = \max_i \left| f_i(x) - \xi_{fi} \right| .$$

Методы логического объединения критериев. Предположим, что критерии $f_1(x)$, $f_2(x)$, ..., $f_m(x)$ могут принимать только два значения: 0 или 1:

$$f_i(x) = \begin{cases} 1, \text{если } i\text{-я цель достигнута}, \\ 0, \text{если } i\text{-я цель не достигнута}. \end{cases}$$

Тогда обобщенный критерий может быть записан:

– в виде конъюнкции критериев $f_i(x)$, если общая цель состоит в выполнении всех целей одновременно, т. е. $F(x) = \prod_{i=1}^{n} f_i(x)$;

– в виде дизъюнкции критериев, когда общая цель достигается, если достигнута хотя бы одна частная цель, т. е. $F(x) = 1 - \prod_{i=1}^{n} (1 - f_i(x))$

Рассмотрим пример, иллюстрирующий применение метода уступок для решения многокритериальных задач.

Пример. Найти компромиссное решение задачи при условии, что откло-

нение по первому критерию от максимального значения составляет 50 %:

$$f_1 = 3x_1 + 2x_3 \,(\max);$$

$$f_2 = x_1 + 2x_2 + x_3 \,(\min);$$

$$\begin{cases} -2x_1 - x_2 + 5x_3 \le 6; \\ x_1 - 2x_3 \le 2; \\ 2x_2 - x_3 \le 5; \\ x_i \ge 0 \ \ (j = \overline{1,3}). \end{cases}$$

Поскольку данная задача является задачей линейного программирования, то на каждом шаге для решения соответствующих однокритериальных задач можно воспользоваться симплекс-методом. Решим однокритериальную задачу по первому критерию. Составляем симплекс-таблицу

БП	1	$-x_1$	$-x_2$	$-x_3$
x_4	6	-2	-1	5
x_5	3	**1**	0	-2
x_6	5	0	2	-1
f_1	0	-3	0	-2

Поскольку данный план не удовлетворяет условию оптимальности, то, находя разрешающий элемент и применяя преобразование Гаусса-Жордана, строим такую последовательность симплекс-таблиц:

БП	1	$-x_5$	$-x_2$	$-x_3$
x_4	10	2	–1	1
x_1	2	1	0	–2
x_6	5	0	2	–1
f_1	6	3	0	–8

БП	1	$-x_5$	$-x_2$	$-x_4$
x_3	10	2	–1	1
x_1	33	5	–2	2
x_6	15	2	1	1
f_1	86	19	–8	8

БП	1	$-x_5$	$-x_6$	$-x_4$
x_3	25			
x_1	52			
x_2	15			
f_1	260	35	8	16

Максимальное значение целевой функции f_1 достигается, таким образом, для плана $\vec{x}^* = (x_1^*; x_2^*; x_3^*) = (52; 15; 25);\ f_1 = 206$.

Делая уступку на 50 %, получаем: $f_1 = 0{,}5 \cdot 206 = 103$ и вводим дополнительное ограничение: $3x_1 + 2x_3 \ge 103$.

Решаем теперь однокритериальную задачу для второй целевой функции с учётом дополнительного ограничения. Получаем такую последовательность

симплекс-таблиц:

БП	1	$-x_1$	$-x_2$	$-x_3$
x_4	6	–2	–1	5
x_5	2	1	0	–2
x_6	5	0	2	–1
x_7	-103	–3	0	–2
f_2	0	–1	–2	–1

БП	1	$-x_5$	$-x_2$	$-x_3$
x_4	10	2	–1	1
x_1	2	1	0	–2
x_6	5	0	2	–1
x_7	–97	3	0	–8
x_4	2	1	–2	–3

БП	1	$-x_5$	$-x_2$	$-x_4$
x_3	10	2	-1	1
x_1	22	**15**	-2	2
x_6	15	2	1	1
x_7	-17	19	-8	8
f_2	32	7	-5	3

БП	1	$-x_1$	$-x_3$	$-x_2$
x_3	97/8			
x_1	105/4			
x_6	137/8			
x_2	17/8			
f_2	341/8	-39/8	-5/8	-2

Таким образом, при заданных условиях задачи эффективным является следующий план:

$$\vec{x}^* = (x_1^*; x_2^*; x_3^*) = (105/4;\ 17/8;\ 97/8);\ f_1 = 103;\ f_2 = 341/8.$$

6.3.2. Многошаговая человеко-машинная процедура решения задач многокритериальной оптимизации, основанная на методе ЛП$_\tau$-последовательностей

Процесс решения задач векторной оптимизации предполагает диалог с ЛПР для получения дополнительной информации о его предпочтениях уже в ходе решения проблемы. Поэтому программные комплексы, реализующие процесс принятия решений в многокритериальных задачах на основе вычислительных алгоритмов и диалога с ЛПР, называются *человеко-машинными процедурами* (*ЧМП*).

Прежде чем перейти собственно к изложению метода ЛП$_\tau$-последовательностей, лежащего в основе соответствующей ЧМП, рассмотрим кратко соответствующие математические понятия и факты.

Рассмотрим единичный n-мерный куб K^n, состоящий из точек с декартовыми координатами $x_1, x_2, \ldots, x_n$,

$$P = (x_1, x_2, \ldots, x_n),$$

удовлетворяющими неравенствам $0 \le x_j \le 1$, при $j = \overline{1,n}$.

Кубические решетки. Обычно полагают, что наиболее равномерное зондирование такого куба обеспечивает кубическая решетка, состоящая из $N^n = \boldsymbol{M}$ точек с координатами

$$\left(\frac{i_1 + \frac{1}{2}}{M}; \frac{i_2 + \frac{1}{2}}{M}; \ldots; \frac{i_n + \frac{1}{2}}{M} \right),$$

где $i_1, ..., i_n$ – независимо принимают все значения 0, 1, ..., $M-1$. Однако это неверно. Такая решетка оптимальна только в одномерном случае при $n=1$. Уже при $n=2$ она не очень хороша, а с увеличением n «равномерность» ее быстро ухудшается.

Сравним двумерные сетки, изображенные на рис. 6.2. В обоих случаях каждому из 16 элементарных объемов принадлежит одна и только одна точка сетки, так что, казалось бы, равномерность расположения точек обеих сеток примерно одинакова. Ситуация, однако, изменится, если потребуется исследовать функцию $f(x_1, x_2)$, определенную в K^2, которая сильно зависит лишь от одного аргумента: например, $f(x)=f(x_1)$. В этом случае, вычислив значения функции f в точках кубической решетки, мы получим лишь четыре различных значения, каждое повторенное четыре раза; а при расчете $f(x)$ в точках улучшенной сетки получим 16 значений, дающих гораздо лучшее представление о диапазоне изменения функции $f(x)$.

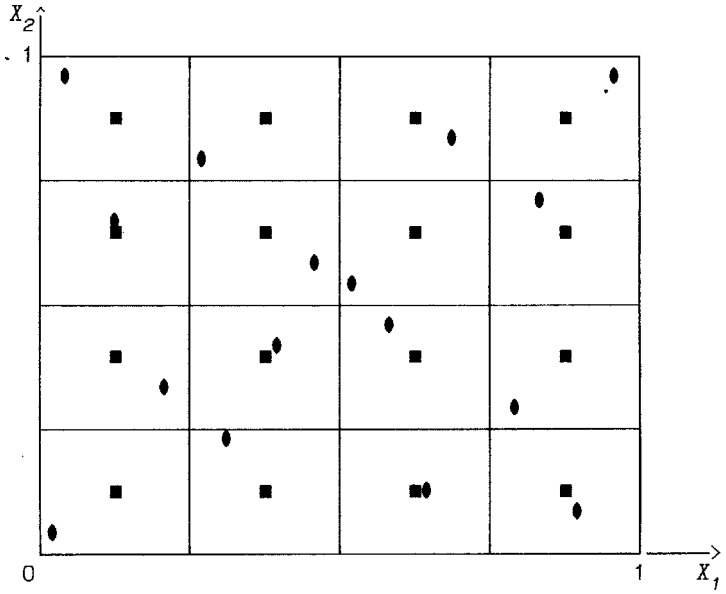

Рис. 6.2. Кубическая решетка (■) и улучшенная сетка (•) при $n=2$ ($N=16$)

В многомерном случае кубическая решетка оказывается еще хуже, так как «потеря информации» при вычислении $f(x_1, ..., x_n)$ может еще больше возрасти: вычислив $N^n=M$ значений функции $f(x)$, мы получим всего $M=N^{1/n}$ различных значений.

Равномерно распределенные последовательности точек. Пусть $P_1, ..., P_i, ...$ – последовательность точек, принадлежащих R^n . Выберем в R^n

произвольный n-мерный параллелепипед П со сторонами, параллельными координатным граням. Обозначим через $S_N(\Pi)$ количество точек P_i с номерами $1 \le i \le N$, принадлежащих П.

Определение. *Последовательность точек* $P_1, ..., P_i, ...$ *называется равномерно распределенной* (*р.р.*) *в* R^n, *если для любого* П

$$\lim_{N\to\infty} \frac{S_N(\Pi)}{N} = V_\Pi, \tag{6.19}$$

где V_Π – *объем* (*n-мерный*) *параллелепипеда* П.

Можно доказать, что если G – произвольная область, расположенная в R^n и имеющая объем V_Π, то из (6.19) вытекает

$$\lim_{N\to\infty} \frac{S_N(G)}{N} = V_G. \tag{6.20}$$

Соотношение (6.20) показывает, что при достаточно больших N количество точек последовательности, принадлежащих G, пропорционально объему G: $S_N(G) \sim NV_G$.

Легко также доказать, что проекции точек р.р. последовательности на любую m-мерную грань куба K^n при $m < n$ образуют р.р. последовательность в K^m.

Несмотря на то, что определение и первые примеры р.р. последовательностей были указаны Г. Вейлем еще в 1916 году, использование таких последовательностей в вычислительной математике началось только в 60-х годах, когда удалось построить последовательности, для которых скорость сходимости в (6.19) при $N\to\infty$ близка к наилучшей, а равномерность расположения наблюдается начиная с небольших N.

Использование в качестве сеток начальных участков $P_1, ..., P_N$ р.р. последовательности имеет еще одно достоинство: количество точек сетки может быть удвоено добавлением еще N точек $P_{N+1}, ..., P_{2N}$. При использовании кубических решеток удвоение M вынуждает увеличить количество точек сразу в 2^n раз; а замена M на $M+1$ заставляет все точки новой сетки считать заново.

Простейший поиск. Предположим, что функция $F(P)$ кусочно-непрерывна в K^n , и требуется приближенно найти точку $\hat{P}$ такую, что

$$F(\hat{P}) = \min_{P\in K^n} F(P).$$

Приближенных методов отыскания минимума функции очень много, но в большинстве своем это все локальные методы, сходимость которых гарантируется лишь в достаточно малой окрестности минимума. Если же речь идет

о нахождении глобального минимума, то выбор методов поиска гораздо более ограничен.

Рассмотрим простейший случайный поиск, который состоит в следующем. В K^n выбираем N независимых случайных точек $\Gamma_1, ..., \Gamma_N$, равномерно распределенных в K^n (здесь равномерное распределение понимается в теоретико-вероятностном смысле). Среди значений $F(\Gamma_1), ..., F(\Gamma_N)$ находим наименьшее $F(\Gamma_{i0}) = \min\limits_{1 \le i \le N} F(\Gamma_i)$ (если таких несколько, то выбираем любое из них). И считаем, что $F(\Gamma_{i0}) \approx \min F(P)$, $\Gamma_{i0} \approx \hat{P}$.

Сходимость такого поиска доказывается достаточно просто. Пусть H-произвольная окрестность единственной точки $\hat{P}$, и объем V_H положителен. Так как вероятность $P\{\Gamma \in H\} = V_H$, то вероятность того, что хотя бы одна из точек $\Gamma_1, ..., \Gamma_N$ попадет в H, равна $1-(1-V_H)^N$ и при $N \to \infty$ стремится к 1. Следовательно, при достаточно больших N вероятность попадания хотя бы одной пробной точки в любую окрестность точки минимума $\hat{P}$ как угодно близка к 1.

Легко показать, что в качестве пробных точек в простейшем поиске можно использовать точки $P_1, \ldots, P_i, \ldots$, образующие р.р. последовательность.

Поиск будет тем лучше, чем более равномерно расположены в K^n пробные точки (если, конечно, нет никакой предварительной информации о положении минимума). Случайны ли они или нет – не столь важно.

ЛП$_\tau$-поиск. И. М. Соболь и Р. Б. Статников предложили использовать в качестве пробных точки ЛП$_\tau$-последовательности, которые являются наиболее равномерно распределенными среди всех известных в настоящее время последовательностей. Многочисленные эксперименты, проведенные с целью сравнения ЛП$_\tau$-поиска с простейшим случайным поиском, показали преимущество ЛП$_\tau$-поиска, хотя количественные характеристики «выигрыша» от его применения меняются в зависимости от рассматриваемых задач.

Наиболее перспективным оказалось применение ЛП$_\tau$-поиска при решении задач следующих двух классов. Во-первых, это задачи, в которых одновременно требуется оценить максимумы и (или) минимумы нескольких функций, заданных в K^n, так как это можно сделать по одним и тем же пробным точкам. Во-вторых, это задачи, в которых для отыскания глобального экстремума многоэкстремальной функции используются локальные методы оптимизации. Для того чтобы не попасть вместо глобального в какой-нибудь из локальных экстремумов, приходится повторять локальный поиск много раз, начиная из различных начальных точек; очевидно, что начальные точки должны быть равномерно расположены в K^n. Самым эффективным способом выбора начальных точек для подобных задач оказалось использование точек ЛП$_\tau$-последовательности.

Предположим, что задана математическая модель исследуемой или проектируемой системы, и модель эта зависит от n параметров $a_1, ..., a_n$. Слова «задана математическая модель» означают, что имеются формулы (или готовые программы), позволяющие по заданному набору $a_1, ..., a_n$ вычислить любые интересующие нас характеристики системы. Если функционирование системы описывается дифференциальными уравнениями, то в качестве параметров можно выбирать коэффициенты или начальные значения этих дифференциальных уравнений.

Пространство параметров. Пространством параметров называется n-мерное пространство, состоящее из точек A с декартовыми координатами $(a_1, ..., a_n)$. Таким образом, каждой точке A пространства параметров соответствует конкретный набор параметров $(a_1, ..., a_n)$ и наоборот.

Как правило, проектировщики могут указать разумные пределы изменения каждого из параметров, которые мы будем называть *параметрическими ограничениями*

$$a_j^* \le a_j \le a_j^{**}, (j = \overline{1,n}). \qquad (6.21)$$

Эти ограничения выделяют в пространстве параметров параллелепипед $\Pi = \{A \,|\, (6.21)\}$, объем которого (n-мерный объем) равен произведению $V_\Pi = (a_1^{**} - a_1^*) \ldots (a_n^{**} - a_n^*)$. В дальнейшем нас будут интересовать только точки A, принадлежащие П, так как только им соответствуют системы, параметры которых удовлетворяют ограничениям (6.21).

Так как данный метод основан на зондировании параллелепипеда П конечным числом пробных точек, то без необходимости расширять границы (6.21) не рекомендуется.

Функциональные ограничения. Кроме параметрических ограничений обычно в условия задачи включаются функциональные ограничения

$$c_l^* \le f_l(A) \le c_l^{**}, (l = \overline{1,t}). \qquad (6.22)$$

Здесь $f_l(A)$ – некоторые функции от параметров $A = (a_1, ..., a_n)$. Они могут быть заданы явно. Но если, например, функционирование системы описывается дифференциальными уравнениями, то $f_l(A)$ часто представляют собой функционалы, зависящие от интегральных кривых этих уравнений. Предположим, что все функции $f_l(A)$ непрерывны в П.

Обозначим через G подмножество параллелепипеда П, состоящее из точек A, удовлетворяющих ограничениям (6.22):

$$G = \{A \,|\, (6.21), (6.22)\}.$$

Множество G может быть любым замкнутым множеством. Единственное ограничение: объем G должен быть положительным ($V_G > 0$).

Можно сказать, что требование $V_G > 0$ исключает из рассмотрения задачи с функциональными ограничениями в форме равенств, например, $f(A) = c$.

Впрочем в некоторых случаях удается разрешить систему ограничений вида

$$f_l(a_1, ..., a_n) = c_l, \ (l = \overline{1,t}\ , \ t < n),$$

относительно $a_{t+1}, ..., a_n$: $a_j = \varphi_j(a_1, ..., a_t; c_1, ..., c_t)$, $(j = \overline{t+1, n}\ )$.

Тогда можно рассматривать задачу в t-мерном пространстве параметров $(a_1, ..., a_t)$ без этих ограничений, а значения $a_{t+1}, ..., a_n$ считать известными функциями $a_j = \varphi_j$ от $a_1, ..., a_t$.

Критерии качества. Критерием качества называется характеристика системы, которая связана с ее качеством монотонной зависимостью. Иными словами, при прочих равных условиях система тем лучше, чем больше (меньше) значение критерия.

Для простоты записи в дальнейшем будем предполагать, что все заданные критерии $\Phi_1(A), ..., \Phi_k(A)$ желательно уменьшить: $\Phi_\nu(A) \to \min$.

Следовательно, чем меньше $\Phi_\nu(A)$, тем лучше система (при прочих равных условиях).

Формально, любой критерий можно привести к такому виду, заменяя, если это нужно, Φ_ν на $1/\Phi_\nu$ или на $-\Phi_\nu$. Однако делать это совсем не обязательно: конструктору удобнее оперировать привычными реальными величинами. Как видно будет из дальнейшего, алгоритм выбора критериальных ограничений легко реализовать и тогда, когда некоторые из критериев желательно максимизировать. Относительно функций $\Phi_\nu(A)$ будем предполагать, что они непрерывны в П.

Сформулировать математическую оптимизационную задачу при наличии нескольких критериев качества совсем непросто, ибо критерии эти часто противоречат друг другу. Например, уменьшая вес машины (что часто очень желательно), мы в то же время уменьшаем ее прочность (что как раз нежелательно).

Считают, что все дело в удачном выборе решающего критерия качества $\Phi_\nu(A)$, который «должен» соединить в себе значения и важность каждого из индивидуальных критериев $\Phi_1(A), ..., \Phi_k(A)$. Однако замена нескольких критериев единым – проблема сложная и не всегда разрешимая. В большинстве реальных задач такой подход себя не оправдывает, так как при грубом выборе $\Phi(A)$ решение математической задачи об отыскании точки $\widetilde{A}$, в которой $\Phi(\widetilde{A}) = \min\limits_{A \in G} \Phi(A)$, оказывается практически плохим из-за того, что некоторые из значений $\Phi_\nu(\widetilde{A})$ превышают допустимые (по мнению проектировщиков) пределы. Чтобы избежать такой ситуации, необходимо ввести критериальные ограничения

$$\Phi_\nu(A) \le \Phi_\nu^{**}, \ (\nu = \overline{1,k}\ ). \qquad (6.23)$$

Критериальное ограничение Φ_ν^{**} – это худшее значение критерия, которое проектировщик считает приемлемым.

Пусть D – множество точек A, которые удовлетворяют всем ограничениям (6.21), (6.22), (6.23): $D = \{A \mid (6.21) - (6.23)\}$, так что $D \subseteq G \subseteq \Pi$; если множество D непустое, то оно замкнуто. Естественно назвать D множеством допустимых точек, ибо если сформулировать задачу об отыскании точки $\widetilde{A}$ такой, что

$$\Phi_\nu(\widetilde{A}) = \min_{A \in D} \Phi(A)\ , \qquad (6.24)$$

то решение этой задачи всегда существует и конструктора устраивает: как бы ни был выбран решающий критерий $\Phi(A)$, все значения $\Phi_\nu(A)$ удовлетворяют ограничениям (6.23).

Таким образом, главная трудность при переходе к математической задаче (6.24) состоит в выборе критериальных ограничений и в обеспечении непустоты множества допустимых точек D. Требования, предъявляемые к множеству D, такие же, как и требования к множеству П.

Диалоговый алгоритм метода ЛП$_\tau$ -последовательностей.

В основе алгоритма лежит численное исследование (зондирование) пространства параметров проектируемой системы. Исследование проводится в три этапа.

1-й этап: составление таблиц испытаний. Этот этап выполняется ЭВМ без вмешательства человека. Последовательно выбираются N пробных точек $A_1, ..., A_N$, равномерно расположенных в П. В каждой из точек A_i рассчитывается система и вычисляются значения всех критериев $\Phi_1(A_i)$, $\Phi_2(A_i)$, ..., $\Phi_K(A_i)$, По каждому критерию составляется таблица испытаний, в которой значения $\Phi_\nu(A_1)$, ..., $\Phi_\nu(A_N)$ расположены в порядке возрастания:

$$\Phi_\nu(A_{i1}) \le \Phi_\nu(A_{i1}) \le ... \le \Phi_\nu(A_{iN}) \qquad (6.25)$$

и указаны номера $i_1, i_2, ..., i_N$ соответствующих пробных точек (свои для каждого Φ_ν). Такие таблицы представляют собой аналог статистических вариационных рядов. При $N \to \infty$ наименьшее значение $\Phi_\nu(A_{i1})$ стремится к $\min \Phi_\nu(A)$, а наибольшее $\Phi_\nu(A_{iN}) \to \max \Phi_\nu(A)$.

Но таблица испытаний показывает не только приближенные значения максимума и минимума $\Phi_\nu(A)$ в области G: по таблице можно судить о частоте тех или иных значений $\Phi_\nu(A)$.

2-й этап: выбор критериальных ограничений. Этот этап предполагает вмешательство ЛПР (специалиста-проблемщика).

Стоит подчеркнуть, что режим диалога очень удобен для проблемщика: он не должен «комбинировать», уменьшая одни критерии за счет других: ему показывают одну таблицу испытаний и предлагают назначить одно ог-

раничение, затем повторяют то же с другой таблицей испытаний. Конструктор заинтересован в том, чтобы все Φ_ν^{**} были по возможности меньше, но необходимо понимать, что если выбирать Φ_ν^{**} неоправданно малыми, то множество допустимых точек окажется пустым.

3-й этап: проверка непустоты *D*. Этот этап также выполняется автоматически, без вмешательства человека. Фиксируем какой-нибудь из критериев, например, $\Phi_1(A)$ и рассмотрим соответствующую ему таблицу испытаний. Пусть s – количество значений в этой таблице, удовлетворяющих выбранному критериальному ограничению Φ_ν^{**}, так что

$$\Phi_1(A_{i1}) \le \Phi_1(A_{i2}) \le \ldots \le \Phi_1(A_{is}) \le \Phi_1^{**}. \qquad (6.26)$$

Путем перебора значений всех критериев в точках $A_{i1},\ldots,A_{is}$ нетрудно проверить, есть ли среди этих точек хотя бы одна такая, в которой справедливы одновременно все неравенства (6.26)

$$\Phi_\nu(A_{ij}) \le \Phi_\nu^{**}, (\nu = \overline{1,k})$$

(при $n = 1$ можно не проверять). Если такая точка A_{ij} существует, то множество D, определенное неравенствами (6.21) – (6.23), непустое, и задача (6.24) разрешима.

В противном случае следует вернуться ко второму этапу и потребовать от конструктора уступок при назначении Φ_ν^{**}. Если такие уступки невозможны, то необходимо вернуться к первому этапу и увеличить количество N пробных точек, чтобы повторить второй и третий этапы с таблицами испытаний большего объема.

Наконец, если при неоднократном увеличении N точки A_{ij}, принадлежащие D, не обнаруживаются, то есть все основания считать, что выбранные критериальные ограничения Φ_ν^{**} несовместны. Конечно, нельзя категорически исключить возможность того, что в некоторой точке *A'*, отличной от всех пробных точек $A_1, \ldots, A_N$, все неравенства (6.21) – (6.23) выполнены; однако если даже такая точка *A'* существует, то ее окрестность, в которой эти неравенства сохраняются, очень мала и практически система, соответствующая точке *A'*, будет неустойчивой (неконструктивной).

Выбор пробных точек. По декартовым координатам точек $ЛП_\tau$-последовательности $Q_0, Q_1, \ldots, Q_i, \ldots$: $Q_i = (q_{i,1}, \ldots, q_{i,n})$ вычисляются декартовы координаты точки $A^{(i)} = (a_1^{(i)}, \ldots, a_n^{(i)})$, принадлежащей параллелепипеду П:

$$a_j^{(i)} = a_j^* + (a_j^{**} - a_j^*) q_{i,j}, (j = \overline{1,n}). \qquad (6.27)$$

При $A = A^{(i)}$ рассчитываем проектируемую систему и проверяем выполнение функциональных ограничений. Если они выполнены, то точка $A = A^{(i)}$ отбирается в качестве пробной точки в G и вычисляются все $\Phi_\nu(A)$; в противном случае точка $A = A^{(i)}$ отбрасывается. Используемые на первом этапе пробные точки – это первые N отобранных таким образом точек. Эти пробные точки при $N \to \infty$ образуют последовательность, равномерно распределенную в G.

Функциональные ограничения и псевдокритерии. При традиционном подходе к многокритериальным задачам нередко пытаются сократить количество критериев, заменяя их функциональными ограничениями. Например, встречается рекомендация выбрать один из критериев в качестве решающего, а на остальные наложить ограничения (при этом предполагается, что эти ограничения задаются априорно, хотя, как мы уже отмечали, обоснованно задать их совсем непросто).

С точки зрения методики применения метода $ЛП_\tau$-последовательности желательно поступать иначе: если функциональное ограничение

$$c_l^{**} \le f_l(A) \le c_l^{**}$$

не абсолютное, т. е. если конструктор допускает, что c_l^* и (или) c_l^{**} могут быть изменены, то стоит вместо этого ограничения ввести псевдокритерий, например $\Phi_{k+1} = f_l(A)$ (не критерий, ибо здесь нет монотонной зависимости от качества). Однако разумные ограничения для Φ_{k+1} можно будет выбрать, изучив таблицу испытаний этой величины.

Если по мнению конструктора, значение $\overline{c_l}$ для величины $f_l(A)$ было бы «весьма хорошим», то в качестве псевдокритерия удобно ввести величину

$$\Phi_{k+1} = |f_l(A) - \overline{c_l}|.$$

Тогда для Φ_{k+1} можно будет выбрать лишь одностороннее ограничение вида $\Phi_{k+1}(A) \le \Phi_{k+1}^{**}$, и при этом окажется, что

$$c_l^* = \overline{c_l} - \Phi_{k+1}^{**}, \; c_l^{**} = \overline{c_l} + \Phi_{k+1}^{**}.$$

Возможность использования псевдокритериев – важное достоинство данного метода. Это позволяет во многих случаях выбирать не произвольные, а обоснованные функциональные ограничения.

Таблицы испытаний. Таблицы испытаний часто встречаются в инженерной практике. Особенность таблиц, построенных по $ЛП_\tau$-методу, в том, что испытания равномерно распределены в области G пространства параметров. Благодаря этому таблицы позволяют получить правильное представление о распределении значений каждой из функций $\Phi_\nu(A)$ при $A \in G$ и гарантируют достаточно подробный просмотр любой наперед заданной части G, когда $N \to \infty$.

Если количество пробных точек N велико, то вместо просмотра всей таблицы испытаний можно ограничиться просмотром ее части, содержащей M

наилучших значений ($M < N$):

$$\Phi_v(A_{i1}) \le \Phi_v(A_{i2}) \le \ldots \le \Phi_v(A_{iM}).$$

Такая таблица называется усеченной таблицей испытаний. Чрезмерное усечение таблиц может оказаться причиной пустоты множества D, но это будет обнаружено на третьем этапе диалога, и тогда, возвращаясь ко второму этапу, следует увеличить объем таблиц.

Диалоговый алгоритм метода может быть наглядно представлен в виде блок-схемы (рис. 6.3).

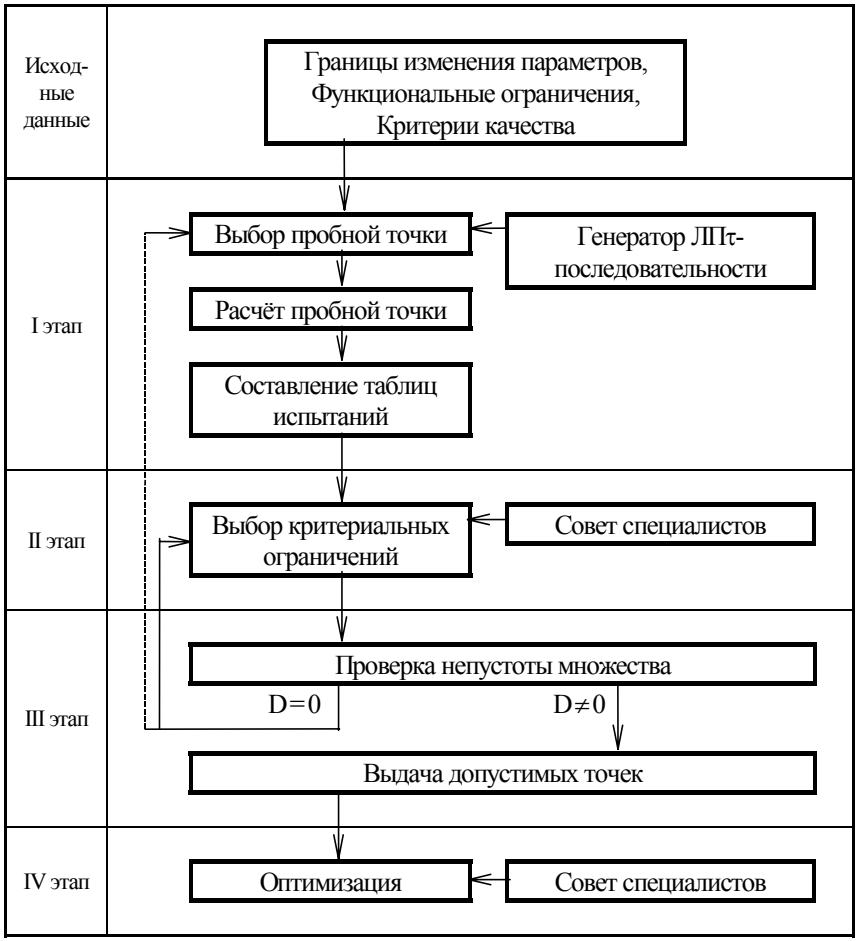

Рис. 6.3. Схема диалогового алгоритма метода $ЛП_\tau$-последовательности

6.3.3. Неформализуемые задачи многокритериального принятия решений. Использование экспертных оценок

Вводные замечания. В исследовании операций и науке об управлении разработано много методов и моделей, механически применяемых для решения сложных проблем. Самые большие неудачи этих наук – в сфере обучения и адаптации, политике и разрешении конфликтов. Дело в том, что ни одна достаточно сложная проблема не встречается в таком виде, в каком люди пытаются ее предвидеть и осознать, строя строго формализованные математические модели.

Экономика представляет собой еще один пример сложности. Частые неудачи в прогнозировании экономических процессов подтверждают, что сложность, свойственная социоэкономическому поведению, может превышать пределы наших интеллектуальных возможностей: многие долгосрочные прогнозы всего на несколько лет выливаются не более чем в обоснованные догадки. Исследователи часто применяют в экономическом прогнозировании методы линейного программирования для нахождения наилучших решений в задачах, включающих не десятки, а сотни и даже тысячи искомых переменных, предполагая, что все они линейно влияют на качество функционирования исследуемой сложной системы (но, например, даже размеры кучи камней больше, чем сумма габаритов всех ее камней). Более того, оптимизация на основе строго формализованной модели вынуждает специалиста по планированию концентрировать внимание на одной или нескольких целях, фактически исключая из рассмотрения остальные.

В 60 – 70-е годы быстрое развитие компьютерной техники открыло новые горизонты моделирования и прогнозирования. Кроме того, это была «золотая эра» исследования операций (ИСО), когда специалисты этой области знаний, основываясь на строго формализованных моделях, «дирижировали» всем, начиная от бомбежки Северного Вьетнама и кончая построением национального бюджета США.

Однако с конца 70-х годов для исследователей все более очевидным становился тот факт, что большинство задач, связанных с изучением сложных экономических и социальных систем, не может быть строго формализовано. Более того, зачастую строгая математическая формализация задачи приводит к ее упрощению и, как следствие, к получению результатов, являющихся далеко не оптимальными. Именно в этот период возрос интерес к так называемым методам экспертного оценивания, одним из которых является метод анализа иерархий (МАИ), разработанный американским ученым Т. Л. Саати.

Метод анализа иерархий (метод Т. Л. Саати). Он состоит в декомпозиции проблемы на все более простые составные части и дальнейшей обработке последовательности суждений ЛПР по парным сравнениям. В результате может быть выражена относительная степень (интенсивность) взаимодействия элементов в иерархии проблемы. МАИ включает в себя процедуры де-

композиции проблемы, синтеза множественных суждений эксперта, получения приоритетных критериев и нахождения альтернативных решений. Метод базируется на следующих принципах.

1. Принцип идентичности и декомпозиции. Данный принцип предусматривает структурирование проблемы в виде иерархии или сети, что является первым этапом применения МАИ. Иерархия считается полной, если каждый элемент заданного уровня связан со всеми элементами последующего уровня. Простейшая полная иерархия проблемы многокритериального выбора включает в себя следующие три уровня:

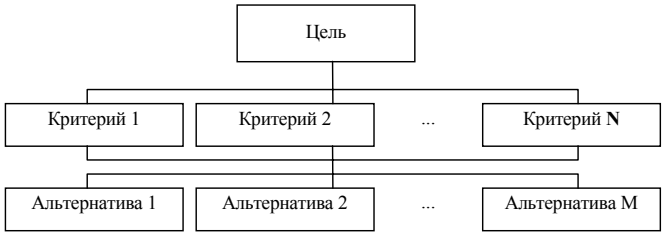

Рис. 6.4. Иерархии проблем многокритериального выбора

2. Принцип дискриминации и сравнительных суждений. Чтобы установить приоритеты критериев, получить оценки для альтернативных решений в МАИ используется метод парных сравнений: строятся матрицы парных сравнений $\boldsymbol{A} = \| a_{ij} \|$, где $a_{ij} = \varphi_i / \varphi_j$, φ_i – «вес» i-го элемента иерархии. Очевидно, что $a_{ii} = 1$, $a_{ij} = 1/a_{ji}$.

При заполнении матриц парных сравнений ЛПР рекомендуется пользоваться следующей шкалой относительной важности для a_{ij}:

Таблица 6.1. **Шкала относительной важности**

a_{ij}	Пояснения
1	Равная важность сравниваемых элементов иерархии
3	Умеренное превосходство i-го элемента иерархии над j-тым
5	Существенное или сильное превосходство i-ого элемента
7	Значительное превосходство i-го элемента
9	Очень значительное превосходство i-го элемента
2, 4, 6, 8	Промежуточные степени превосходства

Следует помнить, что между собой сравниваются элементы, принадлежащие к одному уровню иерархии. Сравнение происходит по степени их соответствия конкретному элементу вышестоящего уровня. Таким образом, для проблемы, обладающей приведенной выше простой иерархией, необходимо составить $N+1$ матрицу парных сравнений (одну – для сравнения элементов второго уровня, т. е. критериев, по степени их важности для ЛПР при достижении цели) и N матриц – для сравнения элементов третьего уровня,

т. е. альтернативных решений, по степени их соответствия каждому из N критериев.

3. Принцип синтеза приоритетов. Итак, будем считать, что построены матрицы парных сравнений: одна – для второго уровня иерархии, а на каждом последующем уровне – столько матриц парных сравнений, сколько элементов содержит предшествующий уровень иерархии. Какую информацию содержат эти матрицы?

Для каждой матрицы мы можем рассчитать *локальные приоритеты* сравниваемых элементов. Каждой строке матрицы ставим в соответствие геометрическое среднее ее элементов. Суммируя полученные результаты, делим геометрические средние каждой из строк матрицы на эту сумму. В результате получаем локальные приоритеты соответствующих сравниваемых элементов.

Важно также вычислить так называемый *индекс согласованности* (ИС) суждений по каждой матрице:

$$\text{ИС} = (\lambda_{\max} - n)/(n-1), \tag{6.28}$$

где n – размерность матрицы, а $\lambda_{\max}$ считается следующим образом: вначале суммируется каждый столбец суждений, затем сумма первого столбца умножается на величину первой компоненты нормализованного вектора приоритетов, сумма второго столбца – на вторую компоненту и т. д., затем полученные числа суммируются.

Теперь необходимо сравнить ИС с той величиной, которая получилась бы при случайном выборе суждений по нашей шкале: 1/9 ... 9. Значения этой величины – случайной согласованности (СС) – представлены в следующей таблице:

Размер матрицы	1	2	3	4	5	6	7	8	9	10
Случайная согласованность	0	0	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49

Определяя ИС и СС, находим отношение согласованности

$$\text{ОС} = \text{ИС}/\text{СС}. \tag{6.29}$$

Если для конкретной матрицы окажется, что $\text{ОС} > 0{,}17$, то можно утверждать, что суждения эксперта, на основе которых заполнена исследуемая матрица, сильно рассогласованы, и ему надлежит заполнить матрицу заново, более внимательно используя при этом шкалу парных сравнений.

Теперь обратимся непосредственно к принципу синтеза приоритетов. Приоритеты синтезируются, начиная со второго уровня вниз. Локальные приоритеты альтернатив перемножаются на приоритеты соответствующих критериев предшествующего уровня и суммируются по каждому элементу в соответствии с критериями. Приоритеты элементов второго уровня умножаются на единицу.

Использование метода МАИ может быть проиллюстрировано на следующем примере. Предположим, что некоторая крупная преуспевающая фирма ставит перед собой цель строительства своего филиала в одной из стран с так называемой «переходной экономикой». Пусть в качестве таковых определены Египет, Турция, Хорватия, Беларусь и Россия. Цель строительства – получение доступа к зарубежным рынкам сбыта и снижение издержек производства за счет более низкой оплаты труда в этих странах. При этом не сбрасываются со счета и потенциальные издержки: некоторая потеря контроля за управлением, преобладание неквалифицированной рабочей силы, риск изменения политических и экономических условий в выбранной стране.

Воспользовавшись методом Саати для решения данной проблемы, надлежит в первую очередь четко определить те потенциальные выгоды и издержки, которые необходимо учитывать. Допустим, что в результате получены следующие иерархии выгод и издержек (рис. 6.5, 6.6):

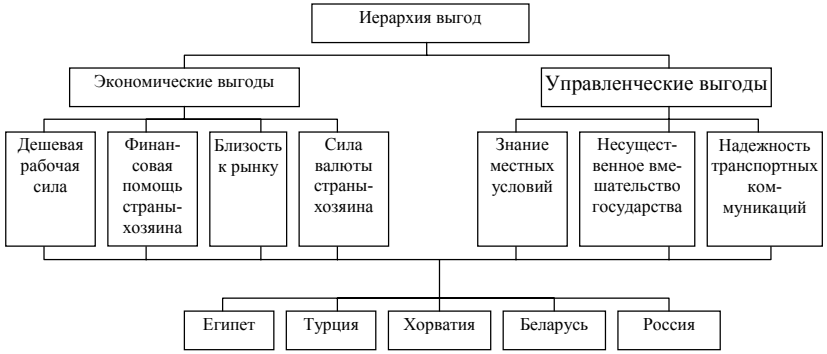

Рис. 6.5. Иерархия выгод

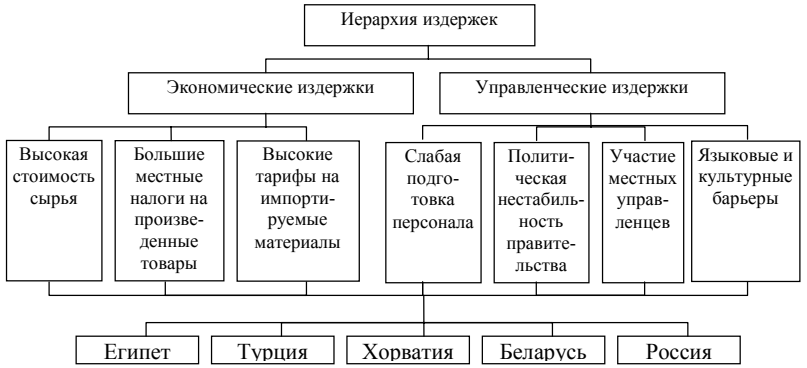

Рис. 6.6. Иерархия издержек

После создания иерархии проблемы необходимо приступить к заполнению матриц парных сравнений. Матрица парных сравнений для второго уровня первой иерархии имеет следующий вид (предположим, что эксперт фирмы заполнил ее с учетом интересов и суждений своих и руководства):

Выгоды	Экономические	Управленческие
Экономические	1	3
Управленческие	1/3	1

Из вида заполненной матрицы следует, что эксперт при решении проблемы отдает предпочтение (хотя и незначительное) достижению экономических выгод перед управленческими. После этого для данной матрицы по описанной выше методике рассчитываются локальные приоритеты и ее согласованность.

Приведем здесь незаполненные матрицы парных сравнений для третьего уровня – уровня критериев:

Важность критерия при достижении экономических выгод	Дешевая рабочая сила	Финансовая помощь страны-хозяина	Близость к рынку	Сила валюты страны-хозяина
Дешевая рабочая сила	1			
Финансовая помощь страны-хозяина		1		
Близость к рынку			1	
Сила валюты страны-хозяина				1

Важность критерия при достижении управленческих выгод	Знание местных условий рынка	Несущественное вмешательство государства	Надежность транспортных коммуникаций
Знание местных условий рынка	1		
Несущественное вмешательство государства		1	
Надежность транспортных коммуникаций			1

Что касается последнего, четвертого, уровня, то для него необходимо составить семь (по числу критериев) элементов вышестоящего уровня матриц для сравнения стран-альтернатив предполагаемого строительства филиала по степени их соответствия каждому критерию:

Дешевая рабочая сила	Египет	Турция	Хорватия	Беларусь	Россия
Египет	1				
Турция		1			
Хорватия			1		
Беларусь				1	
Россия					1

и т. д.

Надежность транспортных коммуникаций	Египет	Турция	Хорватия	Беларусь	Россия
Египет	1				
Турция		1			
Хорватия			1		
Беларусь				1	
Россия					1

После того как все эти матрицы будут заполнены, нужно проверить согласованность суждений эксперта при заполнении каждой из них и в случае удовлетворительного значения ОС по этим матрицам рассчитать локальные приоритеты сравниваемых объектов. Зная локальные приоритеты всех элементов иерархии, можно переходить к этапу синтеза глобальных приоритетов. Таким образом будут получены глобальные приоритеты стран-альтернатив с точки зрения выгод строительства в них филиала фирмы.

Повторяя описанные выше действия для иерархии издержек, получим глобальные приоритеты стран-альтернатив с точки зрения возможных издержек строительства филиала. И, наконец, вычислив отношения приоритетов выгод к приоритетам издержек по каждой из стран, определим ту страну, для которой это отношение является максимальным. Это и будет та страна, которая в наибольшей степени удовлетворяет требованиям фирмы.

Данный подход, основанный на методе МАИ, опирается на рассмотрении доходов и издержек одновременно. Это выгодно отличает его от подходов, опирающихся в основном лишь на учет доходов. Но единственный критерий величины доходов – не очень подходящая основа для сравнения, поскольку чем больше ресурсов будет потрачено, тем больше могут быть доходы. Однако ресурсы в большинстве случаев ограничены. С другой стороны, если специалисты по планированию, оценивая возможные проекты, выбирают лишь те из них, которые требуют минимальных инвестиций, то можно скатиться к подходу «ничего неделания» или, что более реально, производить незначительные действия, не способствующие существенным прогрессивным сдвигам.

Рассмотрим в качестве примера еще одну проблему. Допустим, что перед неким правительственным комитетом, в ведении которого находится проблема строительства мостов и туннелей, встал вопрос: построить или нет туннель или мост через крупную реку, на которой в настоящее время работает частный паром. Допустим, что эксперты комитета, комплексно подходя к решению данной проблемы, опираясь на метод МАИ, сумели разработать следующие иерархии выгод и издержек (рис. 6.7, 6.8).

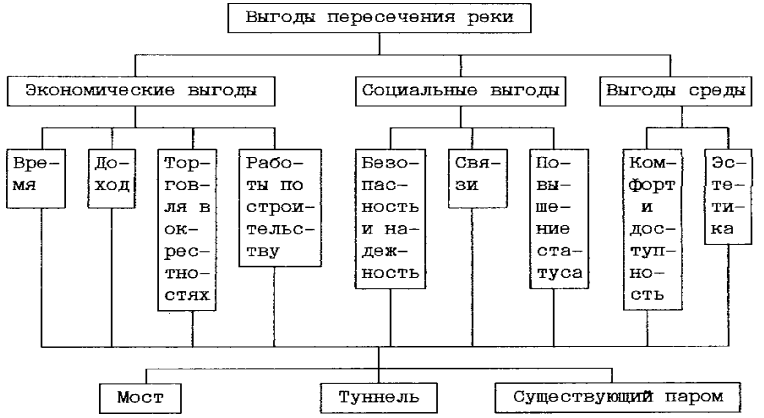

Рис. 6.7. Иерархия выгод пересечения реки

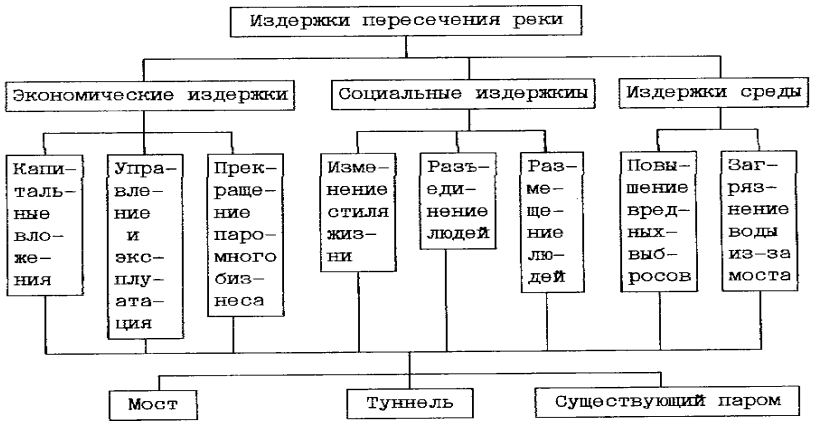

Рис. 6.8. Иерархия издержек пересечения реки

После применения процедуры, реализующей метод Саати, решение может быть рекомендовано к принятию для сравнения выгод и издержек по каждому из возможных решений. Уточним понимание конкретных выгод и издержек, приведенных в иерархиях.

Выгоды. Экономические факторы, влияющие на выбор, содержат выгоды, связанные с выигрышем во времени при передвижении по новому мосту или туннелю по сравнению со временем переправы на пароме. Увеличение транспортного потока в районе может также принести ощутимый доход от эксплуатации дорог при введении платы за проезд, что положительно должно сказаться на местном бюджете. Более интенсивное движение будет способствовать и развитию торговли как в окрестностях моста, так и вдоль всей

дороги, будут построены новые бензоколонки, магазины и рестораны. Возникнет также возможность дополнительного привлечения местного населения на строительные работы. Нельзя сбрасывать со счетов и социальные выгоды проекта: мост или туннель может обеспечить большую безопасность и надежность переправы по сравнению с паромом, а также будет способствовать большему количеству пересечений реки для посещения родственников, друзей, в целях посещения музеев, выставок и т. д.; строительство моста или туннеля может привести также к повышению статуса населенного пункта. Выгоды среды напрямую связаны с социальными выгодами и большим психологическим комфортом жителей.

Издержки. Как и выгоды, издержки, связанные с выбором той или иной альтернативы переправы через реку, включают факторы экономического и социального плана, а также факторы среды. Основные экономические издержки: капитальные вложения на строительство, затраты на управление и эксплуатацию, а также последствия свертывания уже налаженного паромного бизнеса. При планировании социальных последствий следует просчитать возможность отрицательного влияния последствий разрушения существующего стиля жизни. Издержки, связанные со средой, должны учитывать возможный вред, причиняемый экосистеме каждой из альтернатив.

Допустим, что при вычислении приоритетов альтернативных проектов экономические факторы имели бóльший приоритет, и глобальные приоритеты по выгодам и издержкам следующие:

	Мост	Тунель	Паром
Выгоды	0,57	0,36	0,07
Издержки	0,36	0,58	0,05

В предыдущем примере при выборе проектов мы опирались на критерий «стоимость – эффективность», определяя проект с наибольшим отношением выгод к издержкам. В данной задаче таким проектом является строительство моста. Однако поскольку паромная переправа уже существует, то представляется также важным учесть критерий сравнения приращения выгод (0,57 – 0,07) с приращением издержек (0,36 – 0,05), т. е. имеем $0{,}5/0{,}31 > 0{,}07/0{,}05$. Таким образом, и с точки зрения этого подхода строительство моста – наиболее предпочтительная из всех альтернатив.

Для решения более сложных проблем, иерархия которых не может быть сведена к 3- или 4-уровневой структуре, возможна следующая их декомпозиция по иерархии.

В вершине иерархии устанавливается единственный элемент – фокус – формулировка исследуемой проблемы.

Во второй (не обязательный) уровень следует включать различные *экономические, политические и социальные силы*, влияющие на исход.

Третий уровень – *акторы*, которые реально влияют на ситуацию путем

манипулирования этими силами.

Четвертый уровень – преследуемые *цели* каждого актора.

Пятый (не обязательный) уровень включает *политики акторов*, посредством которых они пытаются достичь своих целей.

Шестой уровень – *альтернативные возможные сценарии или исходы*, за которые борется каждый актор ради достижения своих целей.

Седьмой уровень – *обобщенный исход* как результат реализации и взаимодействия возможных альтернативных сценариев развития проблемы.

Методы ЭЛЕКТРА, Подиновского и порядковой оптимизации в задачах экспертного выбора. Как уже отмечалось, выделение множества Парето при решении многокритериальных задач довольно часто является лишь предварительным этапом процесса принятия решений, поскольку при достаточно большом исходном множестве вариантов множество Парето также оказывается недопустимо большим для того, чтобы ЛПР мог осуществить окончательный выбор без затруднений самостоятельно. Следовательно, выделение множества Парето можно рассматривать лишь как предварительный этап оптимизации, и налицо проблема дальнейшего сокращения этого множества. Собственно говоря, те методы многокритериальной оптимизации, которые нами уже рассматривались ранее, также посвящены проблеме сужения множества Парето. В данном разделе рассмотрим методы оптимизации, основанные на построении бинарного отношения предпочтения, более сильного, чем отношение Парето.

Методы ЭЛЕКТРА. Группа методов (ЭЛЕКТРА *I*, ЭЛЕКТРА *II*, ЭЛЕКТРА *III*) была разработана коллективом французских ученых, возглавляемым профессором Б. Руа. В этих методах бинарное отношение предпочтения, более сильное, чем отношение Парето, строится следующим образом.

Для каждого из n критериев (предполагается, что критерии числовые) определяется вес – число, характеризующее важность соответствующего критерия, которое тем больше, чем важнее для ЛПР соответствующий критерий. Эти веса могут быть определены либо ранжированием, либо, например, по методу Саати. Для того чтобы определить, превосходит альтернативный вариант $x = (x_1, ..., x_n)$ вариант $y = (y_1, ..., y_n)$, где x_i, y_i – значения i-го критерия, сообщаемые ему вариантами x и y соответственно, производятся следующие действия.

Множество I критериев разбивается на три подмножества:

– $I^+(x, y)$ – критерии, по которым x превосходит y;

– $I^=(x, y)$ – критерии, по которым x и y имеют одинаковые оценки;

– $I^-(x, y)$ – критерии, по которым y превосходит x.

Далее определяется относительная важность P^+_{xy}, $P^=_{xy}$, P^-_{xy} каждого из этих подмножеств. Устанавливается также некоторый порог c и считается, что вариант x превосходит вариант y только в том случае, когда некоторая

функция, называемая *индексом согласия*, удовлетворяет условию

$$f(P_{xy}^{+},\ P_{xy}^{=},\ P_{xy}^{-}) \geq c. \qquad (6.30)$$

Вид функции f определяется по-своему для каждой модификации метода ЭЛЕКТРА.

Условие (6.30) является необходимым, но не достаточным условием превосходства x над y. В методах ЭЛЕКТРА формулируются дополнительные условия, предназначенные учитывать не только порядок следования оценок x и y по критериям, но и значения модулей разностей $x_i - y_i$. Эти условия, называемые *индексом несогласия*, могут быть записаны в виде

$$d_{xy} \leq d, \qquad (6.31)$$

где d – пороговое значение индекса несогласия d_{xy}; d_{xy} для каждой модификации метода ЭЛЕКТРА определяются по-своему.

Таким образом, отношение предпочтения R определяется следующим образом:

$$x\,R\,y \Leftrightarrow f(P_{xy}^{+},\ P_{xy}^{=},\ P_{xy}^{-}) \geq c \wedge d_{xy} \leq d. \qquad (6.32)$$

Особенность методов ЭЛЕКТРА состоит в том, что в них несколько отступают от традиционных методов выделения подмножества недоминируемых вариантов. Следуя теории игр, их создатели предлагают несколько расширить это подмножество путем выделения в исходном множестве некоего *ядра*, все элементы которого несравнимы между собой, а любой вариант, в ядро не вошедший, доминируется хотя бы одним элементом ядра.

Выделение ядра на множестве исходных вариантов является заключительным этапом методов ЭЛЕКТРА. Дальнейшее сужение ядра может быть достигнуто заданием других, более жестких ограничений в условиях (6.30) – (6.31), т. е. увеличением порогового значения индекса согласия c и уменьшением порогового значения индекса несогласия d.

Опишем более конкретно применение данного метода.

Во всех модификациях метода ЭЛЕКТРА на первом этапе с помощью ЛПР определяются веса критериев – положительные действительные числа, которые тем больше, чем важнее для ЛПР соответствующий критерий. Такой подход, конечно, имеет существенный недостаток – неоднозначность определения весовых коэффициентов. Однако полностью избежать субъективных оценок в процедуре принятия решений невозможно, следует лишь с большой тщательностью подходить к определению весов. Здесь можно воспользоваться, например, процедурой описанного выше метода Саати. Пусть при назначении весов критериям, по которым предстоит выбрать автомобиль, от ЛПР получена следующая информация: цена (критерий 1) важнее комфортности (критерий 2), а та, в свою очередь, важнее скоростных качеств

(критерий 3) и внешнего вида автомобиля (критерий 4). Кроме того, критерии 3 и 4 имеют одинаковую важность, а рассматриваемые совместно, имеют большую важность, чем критерий 1 (цена). Таким образом, ЛПР сообщил информацию о критериях качественного типа, и на ее основе необходимо назначить веса критериев p_i, $i = 1, 2, 3, 4$ так, чтобы выполнялись соотношения

$$p_1 > p_2 > p_3 = p_4, \quad p_3 + p_4 > p_1.$$

Ясно, что, например, решение $p_1 = 5$, $p_2 = 4$, $p_3 = p_4 = 3$, далеко не единственное. И хотя описанную неоднозначность при переводе чисто качественной информации о критериях в числовую полностью устранить невозможно, использование метода Саати будет способствовать более корректному выбору весов критериев.

Далее определяются важности групп критериев $I^{+}(x, y)$, $I^{-}(x, y)$ и $I(x, y)$ для каждой пары сравниваемых альтернатив x и y:

$$P_{xy}^{*} = \sum_{i \in I^{*}(x,y)} p_i\ ,\ \ {*} \in \{+, -, =\}.$$

В качестве условия в методе ЭЛЕКТРА *I* предлагается рассматривать выражение вида

$$(P_{xy}^{+} + P_{xy}^{=}) / \sum_{i=1}^{n} p_i > c_1,\ \ (0{,}5 \leq c_1 \leq 1), \qquad (6.33)$$

в методе ЭЛЕКТРА *II* – выражение вида

$$P_{xy}^{+} / P_{xy}^{-} > c_2,\ (c_2 \geq 1). \qquad (6.34)$$

Следует отметить, что условие (6.33) можно применять лишь тогда, когда сравнение альтернатив происходит в строгих шкалах (множество $P_{xy}^{=}$ пусто) или когда число совпадающих оценок у различных вариантов достаточно мало по сравнению с n. В другом случае отношение предпочтения может оказаться симметричным (x лучше y и y лучше x одновременно). Поэтому, если используются нестрогие шкалы, то лучше пользоваться условием (6.34).

Использование порядковых отношений, т. е. отношений, основанных лишь на порядковой информации о сравниваемых альтернативных вариантах, связано с двумя существенными проблемами.

Первая, присущая всему классу порядковых отношений, – это то, что незначительный выигрыш по одному критерию может сопутствовать большому проигрышу по другому критерию. Например, если $n = 5$, $x = (10, 10, 10, 1, 1)$, $y = (9, 9, 9, 10, 10)$ и все критерии имеют одинаковую важность, то при $c = 1$ вариант x превосходит y в соответствии с зависимо-

стью (6.7), хотя преимущество x над y по первым трем критериям весьма незначительно, а по двум последним критериям x значительно уступает y. Чтобы как-то избежать возможных ситуаций в ЭЛЕКТРА и используется условие (6.31). Используя это условие, мы определяем некоторую область несравнимости – область несогласия D, такую, что для любых вариантов x и y из того, что $(x, y) \in D$, следует, что x и y – не сравнимы. Если, например, $D = \{(x, y) : \exists\, i = 1, 2, ..., n : x_i - y_i > 5\}$, то это означает, что y не может доминировать x, если уступает ему более пяти единиц хотя бы по одной компоненте (критерию).

Вторая сложность, возникающая при использовании порядковых отношений и их модификаций, связана с возможностью появления циклов, т. е. таких ситуаций, когда x^1 лучше, чем x^2; x^2 лучше, чем x^3; ...; x^{k-1} лучше, чем x^k, а вот x^k, в свою очередь, лучше x^1. В связи с этим при использовании порядковых отношений необходимо помнить о возможности возникновения подобных ситуаций и избегать их. Методы ЭЛЕКТРА данную проблему не решают. Если говорить о методе Саати, то наличие процедуры проверки согласованности матриц парных сравнений как раз и нацелено на то, чтобы избежать подобных ситуаций.

В заключение описания метода ЭЛЕКТРА приведем иллюстративный пример. Пусть в исходном множестве альтернативных вариантов, сравниваемых по пяти критериям, определены следующие семь, не доминируемых по Парето:

$x^1 = (5, 3, 2, 7, 2)$ $\quad$ $x^5 = (1, 6, 6, 4, 5)$
$x^2 = (4, 2, 3, 5, 1)$ $\quad$ $x^6 = (2, 7, 5, 2, 6)$
$x^3 = (3, 4, 1, 6, 3)$ $\quad$ $x^7 = (6, 5, 6, 3, 4).$
$x^4 = (7, 1, 4, 1, 7)$

Применим метод ЭЛЕКТРА для того, чтобы, получив у ЛПР дополнительную информацию, сократить число вариантов, которое будет предложено ему для окончательного выбора.

1-й этап. От ЛПР получается информация о сравнительной важности критериев. Пусть ЛПР сообщил, что:

– критерии 1 и 2 имеют одинаковую важность;

– критерии 3, 4 и 5 имеют также одинаковую важность;

– каждый из первых двух критериев важнее каждого из оставшихся.

Пусть в соответствии с этой информацией критериям назначены веса

$$p_1 = p_2 = 2,\ p_3 = p_4 = p_5 = 1.$$

2-й этап. Строим матрицу 7×7, в которой элемент a_{ij} определяется следующим образом:

$$a_{ij} = P^{+}_{x^i x^j} / P^{-}_{x^i x^j}\ .$$

Допустим, что в качестве порогового значения индекса согласия выбрано

на основе консультаций с ЛПР $c_2 = 1{,}25$. Как видно из таблицы, приведенной ниже, любой из семи вариантов доминируется хотя бы одним из остальных:

—	6	1,3	0,75	0,75	0,75	0,17
0,17	—	0,75	0,75	0,75	0,75	0,17
0,75	1,3	—	0,75	0,75	0,75	0,17
1,3	1,3	1,3	—	0,75	0,75	0,75
1,3	1,3	1,3	1,3	—	0,4	1,3
1,3	1,3	1,3	1,3	2,5	—	0,75
6	6	6	1,3	0,75	1,3	—

Поэтому без учета индекса несогласия подмножество оптимальных вариантов оказалось бы пустым.

3-й этап. С помощью ЛПР устанавливается индекс несогласия. Пусть $D = \{(x, y) : x_i - y_i \geq 5\}$. В этом случае один из вариантов – x^7 – оказывается недоминируемым, оптимальным будет считаться также и вариант x^5, который несравним с x^7.

Таким образом, применение метода ЭЛЕКТРА позволило более полно учесть мнение ЛПР и сократить исходное множество не доминируемых по Парето решений до двух элементов. Однако следует отметить, что группа методов ЭЛЕКТРА не лишена традиционных недостатков, присущих многим современным методам многокритериальной оптимизации.

Метод Подиновского. Метод Подиновского также имеет своей целью построение более сильного, нежели паретовское, бинарного отношения предпочтения. Как и в ЭЛЕКТРА, для этого используется дополнительная информация о сравнительной важности критериев. Однако основное и существенное отличие метода Подиновского состоит в том, что качественная информация о критериях, получаемая от ЛПР, не преобразуется в количественную. Автору метода впервые в практике многокритериальной оптимизации удалось освободиться от необходимости ввода весовых коэффициентов важности критериев, вносящих большую неопределенность в решение задачи.

Информация о сравнительной важности критериев задается совокупностью сообщений ЛПР типа:

– критерий i важнее, чем критерий j ($i\,\mathrm{B}\,j$);

– критерии i и j равноценны ($i\,S\,j$);

– набор критериев ($i_1, ..., i_l$) важнее, чем набор ($j_1, ..., j_m$);

– наборы критериев ($i_1, ..., i_l$) и ($j_1, ..., j_m$) равноценны по важности.

Построенное на основании информации о важности критериев бинарное отношение предпочтения позволяет существенно сузить множество Парето. Так, если имеется информация о том, что все n критериев равноценны, то при большом числе сравниваемых вариантов это позволяет сузить паретовское множество приблизительно в $n!$ раз.

Рассмотрим применение метода Подиновского для решения описанной

выше задачи в наиболее благоприятном случае, когда все критерии для ЛПР равноценны. Тогда, следуя методу Подиновского, нам необходимо упорядочить оценки каждого из альтернативных вариантов (например, по убыванию) и среди полученных векторов выбрать в качестве оптимальных не доминируемые по Парето. Упорядочив оценки, получаем:

$$\tilde{x}^1 = (7, 5, 3, 2, 2) \qquad \tilde{x}^5 = (6, 6, 5, 4, 1)$$
$$\tilde{x}^2 = (5, 4, 3, 2, 1) \qquad \tilde{x}^6 = (7, 6, 5, 2, 2)$$
$$\tilde{x}^3 = (6, 4, 3, 3, 1) \qquad \tilde{x}^7 = (7, 6, 5, 4, 3).$$
$$\tilde{x}^4 = (7, 7, 4, 1, 1)$$

Среди вновь образованных упорядоченных векторов оценок недоминируемыми по Парето оказались векторы $\tilde{x}^4$ и $\tilde{x}^7$. Следовательно, руководствуясь методом Подиновского, в качестве эффективных решений при равнозначности критериев рекомендуются варианты x^4 и x^7.

Метод Подиновского в описанном виде может быть применен только в случае однородности критериев, т. е. критериев, значения которых принадлежат одному и тому же множеству. Примером однородных критериев может служить, например, множество суждений одинаково компетентных экспертов, оценивающих варианты по одной и той же шкале. В этом случае получил вариант x оценки экспертов $x_1 = a$, $x_2 = b$ или $x_1 = b$, $x_2 = a$. Сложности появляются, когда критерии оказываются неоднородными, что бывает довольно часто. При неоднородных критериях определение их сравнительной важности сводится по существу к определению коэффициентов важности критериев. Это является основным недостатком метода Подиновского, и в этом случае чаще целесообразнее использовать методы ЭЛЕКТРА.

Метод порядковой оптимизации. В основе данного метода лежит аппроксимация изнутри структуры предпочтений ЛПР, описываемой бинарным отношением.

В основе метода порядковой оптимизации лежит следующая процедура:

1. определение упорядочения критериев по важности;

2. нахождение порядковых отношений, удовлетворяющих этому упорядочению;

3. построение некоторого полинома по всем этим порядковым отношениям, который и будет аппроксимацией k штук предпочтений ЛПР.

Для иллюстрации метода вновь рассмотрим пример сравнения семи вариантов по пяти критериям.

Допустим, что в роли ЛПР выступает покупатель автомобиля; он сформулировал пять критериев, которыми будет руководствоваться при выборе: цена (критерий 1), комфортность (критерий 2), фирма-производитель (критерий 3); скоростные качества (критерий 4); внешний вид автомобиля (критерий 5). Пусть в результате опроса ЛПР получена следующая информация о

важности критериев: входящие в группы $L_1 = \{1, 2\}$ и $L_2 = \{3, 4, 5\}$ имеют одинаковую важность, причем каждый критерий из L_1 важнее любого критерия из L_2. Кроме того, после дополнительного уточнения структуры предпочтений покупателя, проведенного на основе его опроса специалистом по маркетингу, было определено, что в качественное понятие «быть лучше» ЛПР вкладывает следующий смысл: «быть лучше – значит быть лучше по первым двум и по любой паре из оставшихся трех критериев». Нетрудно показать, что в этом случае аппроксимирующий полином имеет вид

$$f_{R^*}(u) = u_1 u_2 (u_3 u_4 + u_3 u_5 + u_4 u_5).$$

Если рассматривать предыдущий пример, то не доминируемыми по R^* будут варианты x^4, x^5, x^6, x^7. Чем сильнее будут упорядочены критерии, тем меньшее число альтернативных вариантов будет рассматриваться в качестве эффективных. Пусть, например, удалось упорядочить все критерии, кроме двух последних:

$$\text{крит.1} \rightarrow \text{крит.2} \rightarrow \text{крит.3} \rightarrow (\text{крит.4} \leftrightarrow \text{крит.5}).$$

В этом случае аппроксимирующий полином имеет вид

$$f_{R^*}(u) = u_1 (u_2 + u_3 (u_4 + u_5)),$$

и выбранными окажутся только два варианта: x^4 и x^7. Как видим, вариант x^7 всегда оказывался в числе рекомендуемых ЛПР для окончательного выбора.

Метод порядковой аппроксимации также не лишен недостатков. К ним можно отнести следующие:

1. Процесс получения от ЛПР информации о сравнительной важности критериев достаточно трудоемок. В общем случае мы должны задать ЛПР порядка n^2 вопросов. Хотя надо отметить, что на практике при благоприятных условиях число вопросов к ЛПР может быть снижено до n.

2. ЛПР, не отдавая себе в этом отчета, может давать противоречивую информацию о сравнительной важности критериев.

3. Главный недостаток: информация по сравниваемым альтернативам должна быть представлена в строгих шкалах, иначе составление аппроксимирующего полинома будет крайне затруднено.

Что касается положительных сторон данного метода, то необходимо отметить следующее. Для решения задач выбора в строгих шкалах при сравнительно небольшом числе критериев (5-9) этот метод особенно эффективен. Он дает возможность обоснованно, без внесения произвола, аппроксимировать предпочтения ЛПР. Несомненное достоинство метода также и в том, что он не зависит (в отличие от ЭЛЕКТРА) от числа сравниваемых вариантов. Кроме того, критерии могут иметь произвольную природу и не обязаны быть однородными, как при использовании метода Подиновского.

7. ПРИМЕРЫ ПОСТРОЕНИЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ СТС

7.1. Построение УПМ на языке Си, реализующей процессный способ имитации.

При процессном способе организации имитации любая ИМ СТС представляется последовательностью происходящих внутри ее элементарных событий. При этом каждый элемент СТС $K_i \in \boldsymbol{K}$ характеризуется множеством состояний $\{z_i\}$ (переменных, содержащих информацию, необходимую для прогноза будущей динамики элемента). В каждом элементе K_i в результате выполнения функциональных действий происходят события e_{ij} . Время наступления события τ_{ij} и его содержание полностью определяются состоянием z_i элемента K_i. Реакцией УПМ на наступление события e_{ij} является корректировка модельного времени t_0 и реализация активности a_{ij}. Для реализации на языке Си алгоритма квазипараллелизма в ИМ СТС рассмотрим необходимые для этого структуры данных и схему описания активностей.

Основной управляющей структурой является список событий *AE*, предназначенный для хранения информации, необходимой для активизации очередного для каждого элемента ИМ события.

```
struct Events {
   int   Element;       /* уникальный идентификатор элемента        */
   int   numAct;        /* номер активности в массиве Activity      */
   float TInit;         /* время инициализации активности           */
   struct Events *Next; /* список событий упорядочивается по        */
} *AE ;                 /*       возрастанию TInit                  */
```

Массив активностей Activity определяется следующим образом :

```
typedef  int(*Action)(int);
Action Activity[]={eoSimulation, Act1, Act2, ..., ActN};
```

Параметром активностей является номер элемента. Если выполнение активности должно привести к окончанию моделирования, то соответствующая функция возвращает значение 0. Во всех остальных случаях возвращается ненулевое значение. Поскольку имитационный эксперимент обычно задается на некотором временном интервале, то имеет смысл для обеспечения окончания ИЭ определить специальную функцию, причем как нулевой элемент массива Activity:

```
int eoSimulation (int Element) { return 0; }.
```

Для модификации списка событий целесообразно написать следующие функции:

```
void NewEvent (int Element, int numAct, float TInit) {
            добавить в список AE новое событие,
            сохранив  его упорядоченность по полю TInit;
```



```
            }
void MoveAE (float Step) {
            AE -> TInit+=Step;
            /*вычислить новое время инициализации активности AE->numAct*/
            сдвинуть событие на новое место вперед по списку;
/* как правило, после перемещения  AE будет указывать на новое событие */
}
void DeleteAE (void) {
            /* в некоторых случаях элемент ИМ заканчивает
            функционирование до завершения моделирования (см. e14 рис.1).*/
            удалить заключительное событие элемента после реализации из списка
        AE;
            }
```

Любая активность a_{ij} определяется по одной схеме.

```
int Act_ty (int numElement) {реализация алгоритма  αij  события eij ;
  if (eij    – последнее событие элемента)
    DeleteAE();      /* удаление AE из списка событий */
  else {
     AE->Act=k;    /* номер события ei j+1 в массиве Activity */
     MoveAE(τij ); /* определить новое время активизации элемента */
  }
  return 1;
 }
```

Для алгоритма организации квазипараллелизма в ИМ необходимо ввести еще две переменные:

```
float ModelTime; /* текущее значение модельного времени */
int   eoS;           /* признак окончания моделирования     */
```

Алгоритм можно условно разделить на три части: начало моделирования, имитация, завершение моделирования.

```
/* начало моделирования */
инициализация элементов  Ki  и их начальных состояний (параметров  z i );
NewEvent(0,0,TeoSimulation);  /* определить интервал моделирования */
инициализация начальных событий ei 0 ;

/*      имитация      */
ModelTime=AE->TInit;       eoS=1;
while (eoS) {
while(AE->TInit==ModelTime&eoS)
  eoS=(*(AE->Activity[AE->Act]))(AE->Element);
       /*  смена модельного времени   */
 ModelTime=AE->TInit;
}

/* окончание моделирования */
пересчет откликов модели и их запись на диск.
```

Итак, для правильной работы приведенного алгоритма требуется соблюдение простых правил составления активностей, а именно: каждая активность должна завершаться указанием следующей выполняемой активности и передвижением ее по списку событий на соответствующий интервал модельного времени.

7.2. Примеры построения имитационных моделей на языках моделирования GPSSV и МК АСИМ

Сопоставим возможности двух языков моделирования, основанных на одном и том же способе формализации СТС (транзактом). Для этой цели выберем простейший пример, когда преимущества языка МК АСИМ практически не сказываются из-за того, что нет надобности строить модель управления процессами. Итак, имеется следующая содержательная формулировка технологического процесса обработки деталей в цеху с двумя конвейерными линиями. Нужно построить ИМ обработки потока деталей двумя конвейерами в течение трех циклов длительностью в 3600 с. С некоторой вероятностью ($p = 0{,}25$) осуществить выборочный контроль деталей, обработанных ранее на одном из конвейеров. Конвейеры состоят из буфера деталей (H_i, $i = 1, 2$) и обрабатываемых станков (SR_i). Выборочный контроль деталей осуществляется для деталей, полученных на любом из конвейеров. Детали поступают в цех на первый конвейер, и только при условии, что первый конвейер занят, детали поступают на второй конвейер. Детали поступают в цех с заданной функцией распределения интенсивностей. Целью моделирования является получение статистики: загрузки узлов обработки и контроля деталей; использования буферов обоих конвейеров; времен обработки деталей. Функция распределения входного потока деталей имеет табличный вид.

Построим ИМ на языке моделирования GPSSV. Блок-схема ИМ цеха обработки деталей представлена на рис. 7.1. Здесь И – источник *TR*; H_i – накопители *TR*; SR_i – устройства, обслуживающие TR_i; ПГ – поглотитель *TR*. Конвейера представлены сочетанием накопителя (очереди *TR*) и устройства (обрабатывающего станка). В качестве деталей выступают транзакты, которые поступают из источника И с интенсивностью, заданной экспоненциальной функцией распределения (*EXPON FUNCTION*).

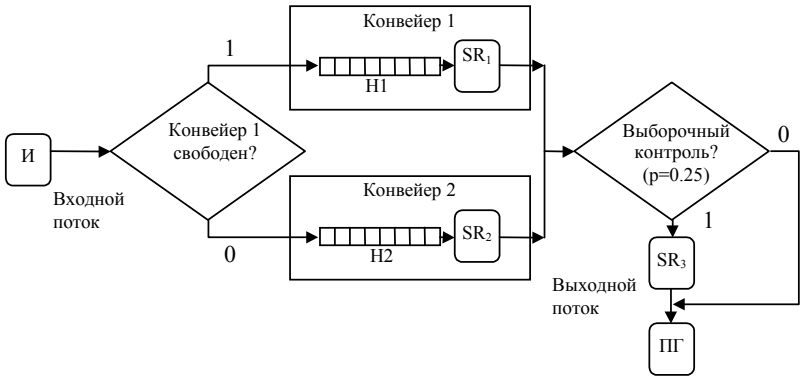

Рис. 7.1. Блок-схема ИМ цеха на GPSSV

Перейдем к сопоставлению программы модели на языке GPSSV. Ниже представлен текст модели. Строки, начинающиеся символами «*», являются комментариями.

```
* МОДЕЛЬ ЦЕХА ОБРАБОТКИ ДЕТАЛЕЙ
      SIMULATE
* ОПИСАНИЕ ФУНКЦИЙ
      EXPON FUNCTION RN1, C24
0,0/.1,.104/.2,.222/.3,355/.4,.509/.5,.69,.0915/.7,1.2/.75,1.38
/8,1.6/.84,1/83/.88,2.12/.9,2.3/.92,2.52/.94,2.82/.95,2.9/.96,3.2
/.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8
* ОПИСАНИЕ ПАМЯТЕЙ
      STORAGE     S1-S2,10
* ОПИСАНИЕ ТАБЛИЦ
      QTIME QTABLE                ACPU,0,60,20
* МОДЕЛИРОВАНИЕ ПРОЦЕССА ОБРАБОТКИ ДЕТАЛЕЙ
      GENERATE    5,FNSEXPON,,,1PB  входной поток
      GATE SNF    1,COMP2         выбор станка
* ОБРАБОТКА ДЕТАЛЕЙ НА СТАНКЕ 1
      ENTER       1               занятие места на станке
      ASSIGN      1,1,,PB         запоминание номера станка
      SEIZE       COMP1           занятие станка
      ADVANCE     8,FNSEXPON      обработка детали
      RELEASE     COMP1           освобождение станка
* НАПРАВЛЕНИЕ ДЕТАЛЕЙ НА ВЫБОРОЧНЫЙ КОНТРОЛЬ
      TRANSFER    .250,NACPU,ACPU
* ВЫПОЛНЕНИЕ ЗАДАНИЙ НА СТАНКЕ 2
COMP2               ENTER 2
      ASSIGN      1,2,,PB
      SEIZE       COMP2           занятие станка
      ADVANCE     12,FNSEXPON
      RELEASE     COMP2
* НАПРАВЛЕНИЕ ДЕТАЛЕЙ НА ВЫБОРОЧНЫЙ КОНТРОЛЬ
      TRANSFER    .750,NACPU
* ВЫБОРОЧНЫЙ КОНТРОЛЬ
ACPU                QUEUE           ACPU
      SEIZE       ACPU
      DEPART      ACPU
      ADVANCE     12,8
      RELEASE     ACPU
* ОСВОБОЖДЕНИЕ БУФЕРОВ СООТВЕТСТВУЮЩЕГО КОНВЕЙЕРА
```

```
NACPU          LEAVE      PB1
    TERMINATE                 выход из системы
* ЗАДАНИЕ ВРЕМЕНИ МОДЕЛИРОВАНИЯ
    GENERATE   3600.,3600.
    TERMINATE 1               прогон 1
    START      1
    CLEAR      1
    START      1              прогон 2
    CLEAR      1
    START      1              прогон 3
    END
```

Построим ИМ того же цеха, но на языке МК АСИМ. Графическая схема ИМ представлена на рис. 7.2.

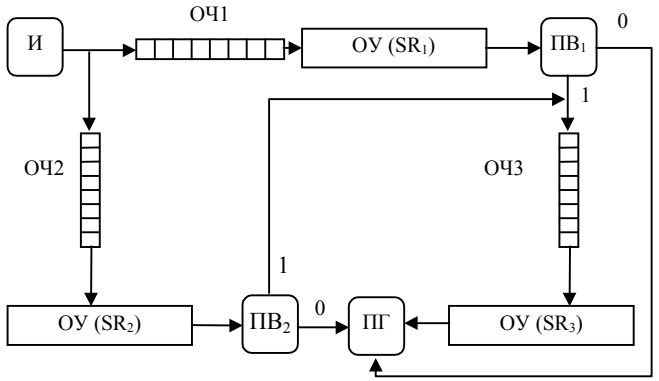

Рис. 7.2. Графическая схема ИМ процесса обработки деталей

Сравнивая рис. 7.1 и 7.2, видим, что обе блок-схемы похожи из-за того, что не задействовано основное достоинство МК АСИМ – описание схем управления. Для данного примера это не нужно делать. Перейдем к составлению текста программы модели. Она представляет собой не программу в традиционном понимании, а набор деклараций.

```
УСТРОЙСТВО (SR,1);
ВХОДЯЩИЙ ПОТОК: (ЧП=1, И) РАСПРЕДЕЛЕНИЕ (RN1,C24);
ОЧЕРЕДЬ 1: ОТДЕЛЬНАЯ ПРИОРИТЕТНАЯ;
МЕХАНИЗМ ОБСЛУЖИВАНИЯ: FIFO;
ВРЕМЯ ОБСЛУЖИВАНИЯ: РАСПРЕДЕЛЕНИЕ ТАБЛИЧНОЕ (0, 0.05, 0.104,
0.22, 0.355, 0.509, 0.69, 0.7, 1.2, 1.38, 1.6, 1.83, 2.12, 2.3, 2.52, 2.82, 2.99, 3.2, 3.5, 3.9,
4.6, 5.3);
ВЫХОДЯЩИЙ ПОТОК: (ИП=1) С ВЕРОЯТНОСТЬЮ 0.25 (SR3);
УСТРОЙСТВО (SR,2);
ВХОДЯЩИЙ ПОТОК: (ИП=1,И) РАСПРЕДЕЛЕНИЕ EXP(RN1,C24);
ОЧЕРЕДЬ: ОТДЕЛЬНАЯ, БЕЗ ПРИОРИТЕТА;
```



```
МЕХАНИЗМ ОБСЛУЖИВАНИЯ: FIFO;
ВРЕМЯ ОБСЛУЖИВАНИЯ: РАСПРЕДЕЛЕНИЕ ТАБЛИЧНОЕ (0, 0.05, 0.104,
0.22, 0.355, 0.509, 0.69, 0.7, 1.2, 1.38, 1.6, 1.83, 2.12, 2.3, 2.52, 2.82, 2.99, 3.2, 3.5, 3.9,
4.6, 5.3);
ВЫХОДЯЩИЙ ПОТОК: (ИП=1) С ВЕРОЯТНОСТЬЮ 0.25 (SR3);
УСТРОЙСТВО (SR,3);
ВХОДЯЩИЙ ПОТОК: (ИП=1,SR1,SR2);
ОЧЕРЕДЬ: ОТДЕЛЬНАЯ;
МЕХАНИЗМ ОБСЛУЖИВАНИЯ: FIFO;
ВРЕМЯ ОБСЛУЖИВАНИЯ: ПОСТОЯННОЕ (0.5);
ВЫХОДЯЩИЙ ПОТОК: (ИП=1);
ВРЕМЯ МОДЕЛИРОВАНИЯ: (3600, 0.1);
ПЕЧАТЬ ТРАФИКА СРЕДНЕГО (ОЧЕРЕДЬ1, ОЧЕРЕДЬ2, ОЧЕРЕДЬ3, УСТР
SR1, УСТР SR2, УСТР SR3) КОЭФФИЦИЕНТ ПОДТВЕРЖДЕНИЯ 3;
```

За счет русского текста с содержательной трактовкой предложений в терминах специалиста по СМО размеры программы на языке МК АСИМ больше. Но у МК АСИМ имеется ряд конструкций и приемов сокращенного описания и использования системы ссылок, которые сокращают текст ИМ на порядок без снижения наглядности и естественной прозрачности текста программы ИМ.

Для иллюстрации указанных средств на рис. 7.3 приведен фрагмент модели СМО, состоящий из управляемых источников ИМ сети и одного из узлов сети, обладающего ресурсом, к которому транзакты выстраиваются в очередь. Управляемые источники ИП = 1, ИП2, ИП3 сгруппированы на устройстве (УПР, 1). Предполагается, что порядок прихода управляющих сигналов на каждый из управляемых источников различен. Так, на источник ИП = 2 одновременно поступают 20 сигналов запуска источника, что приводит к генерации на выходе устройства (УПР, 1) 20 транзактов, поступающих в ИМ сети. На источник ИП = 1 поступают одиночные управляющие сигналы «Включить», поэтому на выходе устройства (УПР, 1) генерируются одиночные *TR* с индекс-потоком, равным 1. На управляемый источник ИП = 3 поступает групповой управляющий сигнал с некоторого узла сети. Допустим, что по этому групповому управляющему сигналу необходимо генерировать 25 транзактов, которые затем поступают в модель сети на очередной этап обслуживания. Не рассматривая подробно структуру и связи узлов модели сети, предположим, что все копии *TR* поступают на устройство (ЭВМ, 1) и требуют захвата ресурса, объем которого перед началом имитации составляет 400 единиц. На входе узла сети (УСТРОЙСТВО (ЭВМ, 1)) имеются три очереди для каждого из типов *TR*. На рис. 7.3 показано, что операция сборки *TR*, рождаемых одним и тем же источником, реализуется с помощью очередей *TR*. Поскольку все *TR* используют единый ресурс, то они выстраиваются в единую очередь на захват части ресурса. После выделения ресурса некоторому *TR* последний поступает в соответствующую очередь

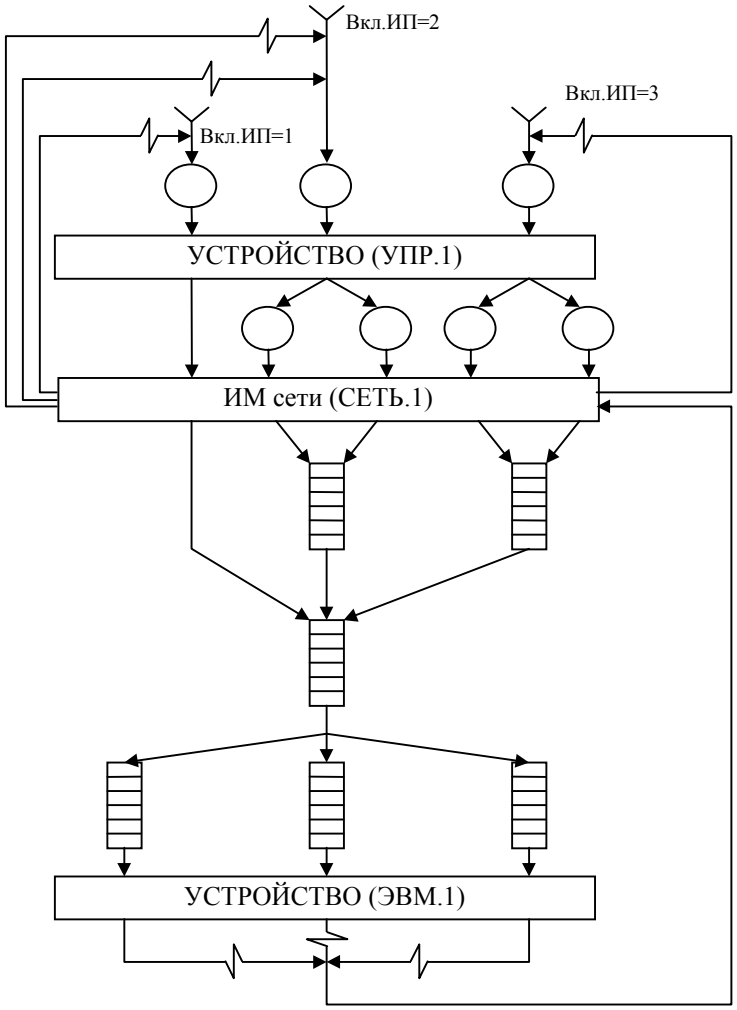

Рис. 7.3. Фрагмент модели СМО с управляемыми источниками

устройства (ЭВМ, 1). Удовлетворение запросов *TR* на ресурс производится по алгоритму, описанному выше. Потоки *TR* обозначены на рис. 7.3 ИП = 1, …, ИП = 3. Таким образом, возможно неявное создание трехуровневой системы очередей (на сборку, к ресурсу на обслуживание). Причем явно описываются только очереди на обслуживание, а остальные очереди формируются УПМ МК АСИМ автоматически. Указателями для организации сборки *TR* служат конструкции вида СБОРКА = ИНДЕКС. Описание приведенного фрагмента сети на языке МК АСИМ имеет вид


```
УСТРОЙСТВО (УПР.1);
ВХОДЯЩИЙ ПОТОК: (ИП=1-3,УПР);
ВРЕМЯ ОБСЛУЖИВАНИЯ: ПОСТОЯННОЕ (0.5);
ВЫХОДЯЩИЙ ПОТОК: (ИП=3) С РАЗДЕЛЕНИЕМ НА ЗАЯВКИ (25 УСТРОЙСТВО (СЕТЬ.1));
<ДАЛЕЕ ТЕКСТ ОПИСАНИЯ ИМ СЕТИ>
УСТРОЙСТВО (ЭВМ.1) РЕСУРС (400);
ВХОДЯЩИЙ ПОТОК: (ИП=1 У(КАНАЛ.1))
РЕСУРС (ПОСТОЯННОЕ(80)), (ИП=2 У (КАНАЛ.1))
СБОРКА=20 РЕСУРС (РАВНОМЕРНОЕ (20,320)), (ИП=3 у (КАНАЛ.1))
СБОРКА=25 РЕСУРС (ТАБЛИЧНОЕ(60,20,0.5,0.3,0.15,0.1,0.05);
ОЧЕРЕДЬ: ОТДЕЛЬНАЯ;
<ДАЛЕЕ СЛЕДУЮТ ОСТАЛЬНЫЕ ПРЕДЛОЖЕНИЯ, СОСТАВЛЯЮЩИЕ ОПИСАНИЕ ДАННОГО УЗЛА>
```

7.3. Пример составления программы имитационной модели на языке моделирования MICIC 2.0 простейшей системы массового обслуживания

Пусть объектом моделирования является следующая система массового обслуживания (рис. 7.4). Имеется *M* источников заявок на обслуживание.

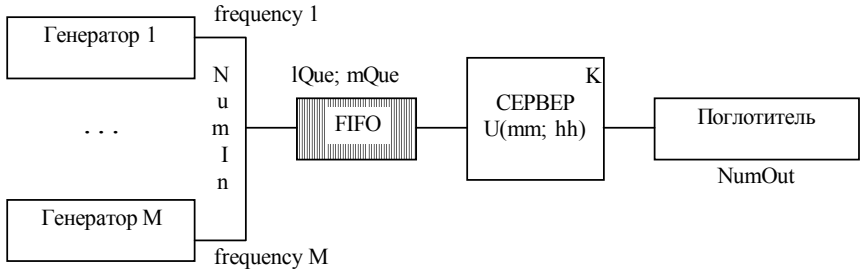

Рис. 7.4. Граф-схема модели простейшей системы массового обслуживания.

Время между выдачей двух заявок распределено по экспоненциальному закону с уникальной для каждого генератора частотой frequency.

Обслуживающий прибор имеет *K* каналов, причем время обслуживания распределено равномерно на отрезке $[mm - hh,\ mm + hh]$ единиц модельного времени. При занятом приборе заявки скапливаются в очереди, откуда извлекаются в соответствии с дисциплиной FIFO. Требуется получить следующие выходные характеристики системы массового обслуживания: общее количество сгенерированных заявок Numin, общее количество обслуженных заявок NumOut, длина очереди lQue, среднее время ожидания в очереди mtQue.

Очевидно, что имитационная модель такой системы может быть построена с помощью трех компонентов типа «устройство», которые назовем «Генератор», «Сервер», «Поглотитель», а также одного компонента типа «транзакт» с названием, например, «Заявка». Для отображения механизма обслуживания всех устройств достаточно одной активности. Ниже приводится полный текст программы имитационной модели простейшей системы массового обслуживания, удовлетворяющей данному описанию, в котором прокомментированы все детали формализации:

```
#include <micic.h>
  // описание глобальных параметров, откликов и собственных
  // параметров компонентов для обращения к ним
  // в Си-тексте программы

struct GlobalPar {
        int M;       // количество генераторов заявок
        int K;       // максимальное количество одновременно
                        // обслуживающихся  заявок на приборе
        float mm,hh;   // среднее и отклонение равномерного
                        // распределения  для  времени обслуживания заявки
} GPar={3,4,5,1.5};   // параметры модели по умолчанию
struct GlobalRes {
        int NumIn;      // общее количество сгенерированных заявок
        int NumOut;   // общее количество обслуженных заявок
        int lQue;        // длина очереди
        float mtQue;   // среднее время ожидания в очереди
} GRes;

        // описание глобальных параметров, откликов и собственных
        // параметров компонентов для обращения к ним
        // в интегрированной среде СМ MICIC

IEGlobal (Prm)={
        IEVar  (КолГенераторов,INT),
        IEVar  (КолКаналов,INT),
        IEVar  (СрВремяОбсл,FLOAT),
        IEVar  (ОтклВремОбсл,FLOAT)
};
IEGlobal (Res)={
        IEVar  (КолВхЗаявок,INT),
        IEVar  (КолВыхЗаявок,INT),
        IEVar  (ДлинаОчереди,INT),
        IEVar  (СрВрОжидания,FLOAT)
};

//описание параметров компонентов ИМ

// Устройство ГенераторЗаявок
typedef struct  {
```



```
float frequency;  // частота поступления заявок в систему
} ReqGenerator;
IEGlobal (ReqGenerator)={
        IEVar (Частота,FLOAT)
};

// Устройство Сервер
typedef struct  {
        int nReq;  // количество заявок, прибывших на обслуживание
} Srvr;
IEGlobal (Srvr)={
        IEVar (КолЗаявок,INT)
};

// Устройство  Поглотитель – параметры отсутствуют

// Класс транзактов Заявка – параметры отсутствуют

//определение компонентов модели
IEKomponents={
        IEKmp  (ГенераторЗаявок,ReqGenerator,PRC),
        IEKmp  (Сервер,Srvr,SRV),
        IEKmp0 (Поглотитель,SRV),
        IEKmp0 (Заявка,TNS)
};

//определение порядковых номеров
//компонентов в макросе IEKomponents
enum {
        REQGENERATOR, SERVER, ABSORBER, REQUEST
};

BEGIN (ПростейшаяСМО)

//определение  номеров активностей
enum {
        EOSIMULATION, STARTSERVICE,
STOPSERVICE, STOPPROCESS,
        RGEN,          SRVR,                ABSORB
};

void Constructor(void) {
        byte i;
        // определение параметров элементов по умолчанию
        ReqGenerator RG_[3]={        0.25, 0.3, 0.4 };
        Srvr SR_={0};
   // определение структуры модели
 for(i=0;i<GPar.M;i++)
  Instal(REQGENERATOR, 1, Ernd(RG_[i].frequency), 0, 0, RG_+i);
 Instal(SERVER, 1, 0.0, GPar.K, 0, &SR_);
 Instal(ABSORBER, 1, 0.0, 1, 0, NULL);
```

```
}
byte RGen(void) {
        DServPar(ReqGenerator,a);
 Run(REQUEST,1,Device(SERVER,1),1,0,NULL);  // очередная заявка
 GRes.NumIn++;          // корректировка
 GRes.lQue++;           // откликов модели
 WaitTime(Ernd(a->frequency));
 return CONTINUE;
}
byte Srvr0(void) {
        DServPar(Srvr,a);
 GRes.lQue--;   // корректировка длины очереди
    // пересчет среднего времени ожидания в очереди
 GRes.mtQue *= a->nReq++;
 GRes.mtQue += ModelTime()-ArrivalTime(TServ);
 GRes.mtQue /= a->nReq;
 // обслуживание отображается единственной активностью
 MoveTo(Device(ABSORBER,1));
 NextActivity(STOPSERVICE);
 WaitTime(Urnd(GPar.mm,GPar.hh));
 return CONTINUE;
}
byte Absorb(void) {
 GRes.NumOut++;  // корректировка отклика
 MoveTo(NULL);   // указание на поглощение заявки
 StopService();  // непосредственный вызов конечной активности
 return CONTINUE;
}
// массив активностей
Activities={
        EOSimulation, StartService, StopService, StopProcess,
        RGen,         Srvr0,        Absorb
};
// массив начальных активностей
InitialActivities={
        RGEN,REQGENERATOR,PROCESS,
        SRVR,SERVER,ALL,
        ABSORB,ABSORBER,ALL
};
// определение вспомогательных функций
void AfterStep(void) {
}
void AfterStop(void) {
}
void Background(void) {
```



```
int x,y,dx,dy,prh=20,cy=190;
int i,j;
char str[]="ГЕНЕРАТОРЫ",s[2]=" ";
setlinestyle(0,0,3);
setcolor(0);
settextstyle(0,HORIZ_DIR,2);
settextjustify(LEFT_TEXT,BOTTOM_TEXT);
outtextxy(90,60,"СТРУКТУРА ИМИТАЦИОННОЙ МОДЕЛИ");
for(i=0;i<10;i++) {
        s[0]=str[i];
        outtextxy(15,110+20*i,s);
 }
x=40;dx=50;dy=50;
for(i=95;i<=235;i+=140) {
        y=i;
        line (x,y,x,y+dy);
        line (x,y,x+dx,y+dy/2);
        line (x,y+dy,x+dx,y+dy/2);
        line (x+dx,y+dy/2,x+dx+prh,y+dy/2);
        outtextxy(x+10,y+35,(i<100)?"1":"M");
 }
dy=25;
for(j=0;j<=1;j++) {
        circle(65,cy+j*dy,2);
        circle(65,cy-j*dy,2);
 }
dy=70;x+=dx+prh;
line(x,cy-5,x,cy-dy);
line(x,cy+5,x,cy+dy);
line(x,cy-5,x+30,cy-5);
line(x,cy+5,x+30,cy+5);
outtextxy(80,295,"NumIn");
for(i=140;i<=370;i+=230) {
        x=i;dx=40;dy=20;y=cy-dy/2;
        rectangle(x,y,x+dx,y+dy);
        line(x+dx,cy,x+dx+prh,cy);
        outtextxy(x-5,y-5,"l,t");
        line(x+25,y-27,x+41,y-27);
        x+=dx/2;dx/=6;
        for(j=0;j<=2;j++) {
                line(x+dx*j,y,x+dx*j,y+dy);
                line(x-dx*j,y,x-dx*j,y+dy);
         }
        x=i+40+prh;dy=60;dx=50;y=cy-dy/2;
        rectangle(x,y,x+dx,y+dy);
        line(x+dx,cy,x+dx+prh,cy);
        outtextxy(x+5,y+dy+25,"m,h");
```

```
                outtextxy(x+32,cy-10,"k");
                outtextxy(x+18,cy+11,(i==140)?"1":"N");
        }
        line(350,cy,350+prh,cy);
        x=310;dx=10;
        for(j=1;j<=3;j+=2) {
                circle(x+j*dx,cy+10,2);
                circle(x-j*dx,cy+10,2);
        }
        outtextxy(215,140,"C E P B E P Ы");
        x=500;y=150;dx=120;dy=80;
        rectangle(x,y,x+dx,y+dy);
        outtextxy(515,265,"NumOut");
        outtextxy(x+22,y+dy/2-3,"ПОГЛО");
        outtextxy(x+15,y+dy/2+22,"ТИТЕЛЬ");
        settextjustify(LEFT_TEXT,TOP_TEXT);
}
void PrintState(FILE *pout) {
fprintf(pout,"Общее количество сгенерированных заявок : %8d\n",GRes.NumIn);
fprintf(pout,"Общее количество  обслуженных заявок    : %8d\n",GRes.NumOut);
fprintf(pout,"Длина очереди                           : %8d\n",GRes.lQue);
fprintf(pout,"Среднее время ожидания в очереди        : %8.3f\n",GRes.mtQue);
fprintf(pout,"=====================================\n\n\n");
}
// определение FIFO-очереди
StandardQueues;
END
```

7.4. Построение ИМ метрополитена на языке MICIC 2.0

7.4.1. Содержательное описание объекта имитации

Для реализации ИМ предлагается рассмотреть функционирование метрополёитена, состоящего из двух пересекающихся линий (рис. 7.5). О каждой линии известно: количество станций на ней, количество курсирующих поездов и время передвижения между станциями. Постоянными принимаются времена остановки на станциях и перехода с одного направления на обратное. Будем считать, что вместимость всех поездов одна и та же, и они равномерно распределены по линии. Последнее означает, что интервал между появлениями со-

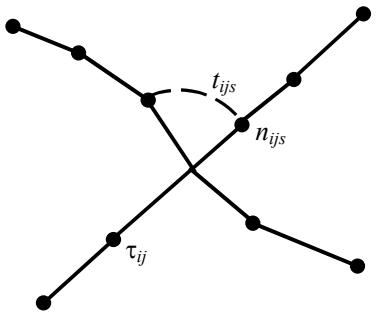

Рис. 7.5. Схема метрополитена

седних поездов в любой точке линии $\tau = 2T/N$, где T – время движения (включая остановки) по линии в одном направлении, N – количество поездов на линии.

Пусть пассажиры, прибывающие на остановку St_i ,образуют экспоненциальный поток с параметром l_i , причем вероятности p_{ij} того, что пассажирам со станции St_i необходимо прибыть на станцию St_j, принимается постоянным в течение ИЭ. Если St_j и St_i принадлежат разным линиям, то пассажир, прибывая на пересадочную станцию, переходит на другую линию, затрачивая на это время, нормально распределенное с параметрами m и $S2$, и возмущая тем самым экспоненциальный поток пассажиров на пересадочных станциях. Для небольшого упрощения алгоритмов активностей сделаем правдоподобные заключения о том, что $p_{ij} = 0$, если St_i и St_j находятся на разных линиях и St_j – пересадочная станция.

Единицей модельного времени будем считать 1 секунду. Длительность ИЭ задается с помощью переменной *TSimulation*. В качестве откликов модели выберем следующие характеристики:

1. Средний коэффициент загрузки поездов на линии:

$$K_i = \sum_{j=1}^{N_i} k_{ij} = \sum_{j=1}^{N_i} \left(\frac{1}{T_{ij} N_p} \sum_{s=1}^{S_{ij}^0} n_{ijs} t_{ijs} \right) \tag{7.1}$$

где $i = 1, 2$ – номер линии; N_i – количество поездов на линии; N_p – вместимость поезда; K_{ij} – коэффициент загрузки поезда j на линии i; S_{ij}^0 –общее количество отрезков между остановками, пройденными поездом (рис. 7.5); t_{ijs} – время прохождения отрезка s; n_{ijs} – количество пассажиров в поезде на отрезке s; $T_{ij} = \sum_{s=1}^{S_{ij}^0} t_{ijs}$ – общее время движения поезда.

Очевидно, что для подсчета K_{ij} требуется использовать рекуррентные формулы

$$\sigma_s = \sigma_{s-1} + n_{ijs} * t_{ijs;} \tag{7.2}$$

$$\theta_s = \theta_{s-1} + t_{ijs.} \tag{7.3}$$

2. Среднее время ожидания поезда на остановке

$$\varpi_i = \frac{1}{N_{\varpi_i}} \cdot \sum_{j=1}^{N_{\varpi_i}} \tau_{ij} \tag{7.4}$$

где $i = 1, \ldots, N_s$ – количество остановок на двух линиях; $N\omega_i$ – количество пассажиров, вошедших в поезд на остановке St_i; τ_{ij} – время ожидания поезда пассажиром j на остановке St_i;

В программе ω_i будут высчитаны с помощью рекуррентных соотноше-

ний:

$$\tau_j = \tau_{j-1} + \tau_{ij}; \quad (7.5)$$

$$N\omega_j = N\omega_{j-1} + 1. \quad (7.6)$$

7.4.2. Формальное описание ИМ метрополитена для СМ MICIC

Согласно концепции многоуровневого представления объекта моделирования в системе моделирования MICIC, рассмотрим описание ИМ двух пересекающихся линий метро с точек зрения исследователя, конструктора и разработчика ИМ.

Исследователь модели должен определить параметры и отклики ИМ на основе содержательного описания.

Параметры ИМ:

1. Количество станций на линии.
2. Общее количество поездов на линии.
3. Номер пересадочной станции на линии.
4. Время движения между станциями.
5. Количество мест в поезде.
6. Время стоянки поезда на станции.
7. Параметр экспоненциального потока пассажиров на станциях.
8. Распределение вероятностей пунктов назначения пассажиров.
9. Параметры нормального распределения для времени, затрачиваемого на переход.

Отклики ИМ:

1. Коэффициенты загрузки поездов на линиях.
2. Время ожидания поезда на станциях.

Конструктор модели, основываясь на базовой схеме формализации системы моделирования MICIC, должен раскрыть содержимое «черного ящика», представив структуру модели в виде граф-схемы. Чтобы не загромождать структуру ИМ повторяющимися деталями, разделим общую граф-схему на две части. Первая из них отображает взаимодействие статических элементов (линий и станций) с динамическими элементами – поездами (рис. 7.6).

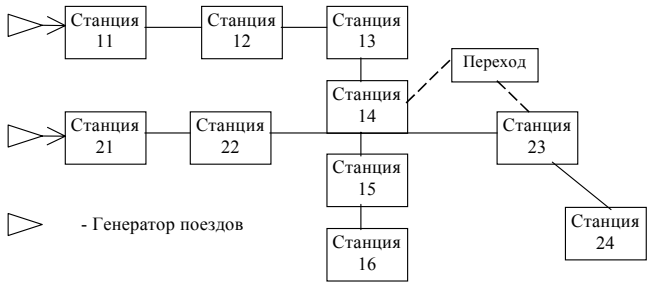

Рис. 7.6. Граф-схема перемещения поездов между станциями

Вторая часть включает взаимодействия динамических элементов ИМ (поездов и пассажиров) на станции (рис. 7.7).

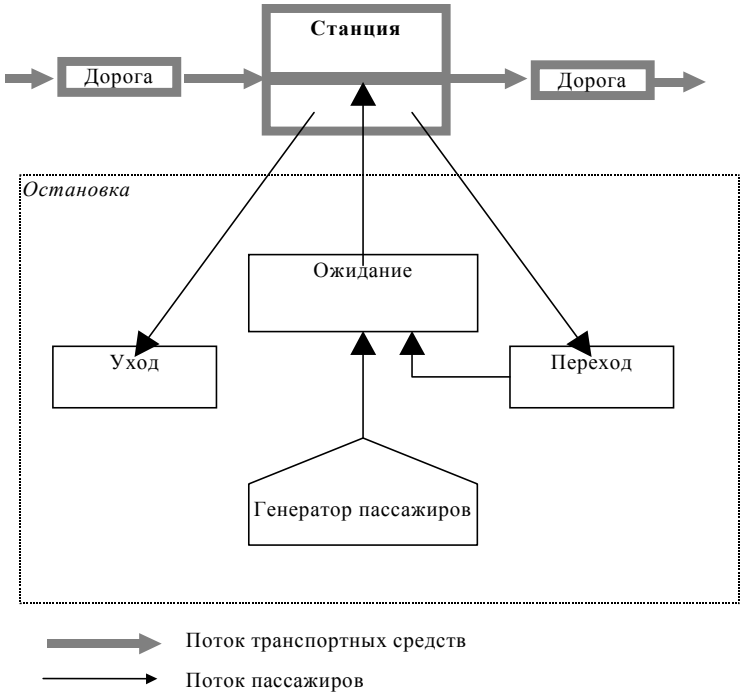

Рис. 7.7. Граф-схема взаимодействия пассажиров с поездами на остановке

На основании декомпозиции ИМ введем следующие типы ее компонентов, представив их в виде табл. 7.1.

Наконец, разработчик имитационной модели должен каждой паре транзакт – устройство (входящая дуга графа - вершина) поставить в соответствие граф активностей. В данном случае для любой пары граф активностей представляет собой крайний случай: одну вершину - активность. Это соответствие приведено в табл. 7.2, причем алгоритмы активностей по сравнению с моделью метро, реализованной средствами языка программирования Си, не изменяются. Они упрощаются из-за использования языка моделирования системы моделирования MICIC. В частности, исчезают списки динамических и массивы статических элементов модели, функции преобразования списка событий, обработка массива пассажиров в поезде и другое. Аналогичные действия осуществляются с помощью таких функции языка моделирования, как Run(), WaitTime(), MoveTo(), обработки транзактного буфера и, разумеется, NextActivity(), StopService(), StopProcess().

Таблица 7.1. **Назначение и параметризация компонентов модели метро.**

Компонент	Назначение	Параметры
Устройство Генератор поездов	Отправляет поезда на линию в начале рабочего дня, равномерно распределяя поезда по линии; когда все поезда выедут на линию, генератор пассивизирует сам себя	Временной интервал между двумя соседними поездами на линии Счетчик отправленных на линию поездов
Устройство Дорога	Соединяет две соседние станции; одновременно на дороге может находиться любое количество поездов	Время, затрачиваемое на переезд до следующей станции
Устройство Станция	Принимает поезда для выхода пассажиров, достигших пункта назначения, и входа новых пассажиров в поезд; вышедшие пассажиры в зависимости от типа станции и своих целей могут передвигаться далее не более чем в двух направлениях: исчезать из модели или переходить на другую линию	
Устройство Генератор пассажиров	Имитирует прибытие очередного пассажира на остановку метро, порождая транзакт с соответствующими концептуальной модели начальными значениями	Распределение вероятностей по перемещению пассажира из текущей станции к любой другой
Устройство Ожидание	Накапливает пассажиров в очереди до прибытия очередного поезда и статистику по ним; всегда закрыто	Количество вошедших в поезд пассажиров Суммарное время ожидания поезда пассажирами
Устройство Переход	Имитирует переход пассажира на другую линию	Среднее значение и дисперсия времени на переход между станциями
Устройство Уход	Реализует удаление из системы моделирования прибывшего в пункт назначения пассажира	
Транзакт Поезд	Отображает движение поезда в модели метро; обладает транзактным буфером, заполняемым транзактами – пассажирами	Количество пассажиров в поезде Средний коэффициент загрузки поезда
Транзакт Пассажир	Имитирует перемещение пассажиров между станциями	Время прибытия в метро Конечный пункт назначения

Таблица 7.2. **Соответствие между компонентами модели и механизмом обслуживания**

Транзакт	Устройство	Активность
	Генератор поездов	TrainStart
	Генератор пассажиров	PassengerArrival
Поезд	Дорога	TrainMovement
Поезд	Станция	TrainArrival
Пассажир	Переход	ChangeLine
Пассажир	Уход	StopService

Машинный вариант данной имитационной модели для системы моделирования MICIC предлагается реализовать самостоятельно по образу приведенного выше примера простейшей сети массового обслуживания.

ЗАКЛЮЧЕНИЕ

Данная часть завершает изложение типовых методик курса исследования операций. Последовательно, от первой до третьей части материал пособия подавался во всё возрастающей полноте и требовал постоянного накопления знаний как по прикладной математике, так и по технологии программирования на ПЭВМ. Быть может, число практических примеров не столь велико и разнообразно, как этого хотелось бы. Однако авторы сознательно уменьшали количество примеров и за счёт этого каждому примеру уделили достаточно внимания для обеспечения глубины рассматриваемых вопросов. В результате все примеры можно рассматривать в качестве базы для выполнения лабораторных и расчётно-графических работ.

Предлагаемое пособие будет также полезно аспирантам и соискателям в их практической работе над диссертациями по различным предметным направлениям. Пособие будет полезно и тем инженерам-практикам, которые используют имитацию в своей практической работе. Зачастую многие специалисты и аспиранты обращаются к имитационному методу исследований, не представляя при этом тех трудностей, с которыми они встретятся, и совершают множество ошибок методологического характера. Затем наступает разочарование, иногда преждевременное. Поэтому в данном пособии авторы обращают внимание на возможные трудности, возникающие перед исследователями при решении задач исследования операций. Для их преодоления авторы представляют для знакомства целый спектр аналитических, вероятностных и имитационных моделей и на их примере дают методики, позволяющие преодолевать указанные трудности.

Авторы благодарят рецензентов за полезные замечания, способствовавшие улучшению качества пособия и облегчению усвоения излагаемого материала.

Все замечания по содержанию и оформлению пособия будут приняты авторами с благодарностью.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. *Максимей И. В.* Имитационное моделирование на ЭВМ. –М.: Радио и связь, 1989. – 230 с.

2. *Киндлер Е.* Языки моделирования. – М.: Энергоатомиздат, 1985. – 288 с.

3. *Жогаль С. П., Жогаль С. И., Максимей И. В.* Основы регрессионного анализа и планирования экспериментов. Учеб. пособие. – Гомель: ГГУ, 1997. – 94 с.

4. *Maximey I. V., Levchuk V. D., Sukach E. I.* Program Technological Complex of Simulation Modeling. Applied Modelling and Simulation. Proceeding International AMSE Conference. Ukraine, September 30. October 2, 1993. AMSEPRES. P.40 – 45.

О Г Л А В Л Е Н И Е