МОСКОВСКИЙ ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ

В. М. Кирюхин

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ

Москва 1990

ГОСУДАРСТВЕННЫЙ КОМИТЕТ СССР
ПО НАРОДНОМУ ОБРАЗОВАНИЮ

МОСКОВСКИЙ ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ
ИНЖЕНЕРНО-ФИЗИЧЕСКИЙ ИНСТИТУТ

В.М. Кирюхин

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ

Утверждено
редсоветом института
в качестве учебного пособия

Москва 1990

УДК 681.51.001.57

К и р ю х и н В.М. Имитационное моделирование сложных систем: Учебное пособие. М.: МИФИ, 1990. — 60 с.

Излагаются современные методы построения имитационных моделей сложных систем. Проводятся основные типы алгоритмических спецификаций систем и их элементов, используемых на этапе формализации. Рассматриваются особенности построения имитационных моделей в рамках концепций состояний, событий и процессов. Особое внимание уделено вопросам формализации и реализации инвариантных компонентов имитационных моделей. Приведен также анализ современных систем моделирования и особенности их использования при программировании имитационных моделей.

Пособие предназначено для студентов и специалистов, занимающихся изучением вопросов имитационного моделирования сложных систем в различных областях науки и техники.

Редактор Е.Г. Станкевич
Техн. редактор Е.Н. Кочубей
Корректор Г.А. Станкевич

Тем. план 1990 г., поз. 52

Подписано в печать 14.02.91 Формат 60x84 1/16 Печ.л. 3,75
Уч.-изд.л. 4,0 Тираж 300 экз. Заказ 2453
Цена 20 коп. Изд. № 102—1

Московский инженерно-физический институт. Типография МИФИ.
115409, Москва, Каширское шоссе, 31

ВВЕДЕНИЕ

Повсеместное использование ЭВМ и, как следствие, увеличение за счет этого интеллектуальных возможностей человека в производственной, исследовательской и управленческой сферах предопределило интенсивное развитие методов моделирования. Среди них имитационное моделирование занимает одно из важных мест. Невозможно перечислить все области науки и техники, где применяется имитационное моделирование. Более того, при проектировании сложных систем, анализ которых не основывается на методологии, абстрагирующейся от изменений во времени, только имитационное моделирование может обеспечить успешное решение стоящих перед исследователями задач.

Повышенный интерес к имитационному моделированию вызвал появление обширной литературы в этой области. Особенно большое число публикаций посвящено рассмотрению отдельных систем и языков имитационного моделирования и использованию их при решении задач определенного класса. Однако этого явно недостаточно, чтобы на практике хорошо ориентироваться среди сотен существующих систем моделирования. Более того, большое многообразие объектов исследования требует разработки единых принципов построения имитационных моделей и их использования в процессе научных исследований.

Опыт создания имитационных моделей показал, что наибольшие трудности возникают у разработчиков на этапе формализации исследуемых объектов или систем. В данном учебном пособии предпринята попытка обобщить часто используемые в этой области подходы и методы и установить соответствие между существующими формализмами и методами их реализации,основанными на применении широко распространенных в нашей стране систем моделирования.

Первая глава посвящена рассмотрению общих положений теории имитационного моделирования. Конкретизируются понятия имитационного моделирования и имитационной модели. Рассматриваются основы формализации систем с использованием концепций состояний, событий и процессов.

Во второй главе рассматриваются основные типы алгоритмических спецификаций, используемых при формализации объектов моделирования. Наиболее подробно изложены спецификации систем, описываемых дифференциальными уравнениями, дискретно-событийных и дискретно-временных систем.

Третья глава посвящена вопросам реализации моделирующего процессора в имитационных моделях. Рассматриваются алгоритмы отображения времени, различные типы управления процессом моделирования. Важное место занимают алгоритмы функционирования программных имитаторов дискретно-событийных систем.

В четвертой главе приведен анализ широко распространенных в нашей стране языков и систем моделирования. Рассматриваются вопросы реализации выбранных на этапе формализации алгоритмических спецификаций с использованием различного типа языков моделирования.

Глава 1. ОБЩИЕ ПОЛОЖЕНИЯ ТЕОРИИ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ

Широкое распространение имитационного моделирования в науке и технике и, как следствие, большой интерес к ним со стороны различных специалистов требуют конкретизации этого понятия, определения возможностей такого подхода и путей решения возникающих при этом задач. Однако существующая по этим вопросам литература не может дать исчерпывающие ответы ввиду отсутствия систематизации принципов построения имитационных моделей, как, например, это было сделано в химии после открытия периодической системы элементов. Но тем не менее, имеющиеся научные результаты позволяют по ряду вопросов внести определенную ясность.

1.1. Имитационное моделирование как метод научного познания

Что же такое имитационное моделирование? Чтобы иметь по этому поводу более ясное представление, сделаем ряд уточнений. Во-первых, термин "имитация" (*"imitatio"*) имеет смысл как подражание, копирование. В этой связи следует уточнить, что копируется или чему производится подражание в ходе моделирования.

Во-вторых, очень часто имитационные модели рассматриваются как модели, описывающие поведение во времени сложных динамических систем. Однако такое понимание является неправомерным, так как моделирование объектов во времени осуществлялось и раньше, но не называлось имитационным. В этом плане термин "имитационное" следует рассматривать не как определение класса изучаемых объектов, а как определение способа моделирования.

В-третьих, термин "имитационное моделирование", на первый взгляд, может вызвать недоумение. Действительно, термины "моделирование" и "имитация" являются синонимами. Однако если иметь в виду моделирование объектов, которое осуществляется посредством моделирования интеллектуальных процессов, то все становится на свои места.

Вышесказанное подтверждает тот факт, что имитационное моделирование образует специфический вид познавательной деятельности. Возникает вопрос: почему до недавнего времени имитационное моделирование как таковое не существовало? Ответ на этот вопрос заключается в следующем.

Имитационное моделирование как вид познавательной деятельности может осуществляться только благодаря взаимодействию трех видов познания: логического, физического и семиотического. Первый включает процедуры, по которым воспроизводятся модели объектов познания или управления, а также интеллектуальные процессы решения задач на основе моделей. Физические средства познания представляются природными органами познания человека и техническими средствами. Семиотическое познание включает различные диалоговые системы, позволяющие человеку формулировать свои задачи, оперировать с моделями, обращаться с техническими средствами.

Идея объединения природных органов познания человека и технических устройств при выполнении познавательных процессов возникла давно. В исследованиях таких ученых, как Р. Луллий, Т. Лейбниц, Б. Паскаль, Я. Баббедж и др., описаны устройства, которые были способны моделировать математические и логические операции. Однако адекватная реализация этой идеи оказалась возможной только на основе применения современных технических средств, в частности ЭВМ.

Несмотря на широкое распространение имитационного моделирования в различных областях науки и техники до сих пор нет единого мнения в точном определении этого понятия. Более того, существуют ошибочные взгляды на имитационное моделирование, которые заключаются в следующем.

Очень многие считают, что имитационное моделирование представляет собой область применения ЭВМ или вообще совпадает с ним. На самом деле это не так, так как имитационное моделирование может осуществляться и на гидродинамических, механических или электронных системах. Сам факт использования ЭВМ ни в коем случае не определяет имитационного моделирования.

Не следует также понимать, что имитационное моделирование — это просто экспериментирование на ЭВМ. Например, если речь идет о проведении опытов с программой, которая реализует какой-либо метод математического программирования с целью его исследования, то нельзя говорить об имитационном моделировании.

Широкое использование в имитационном моделировании случайных или псевдослучайных чисел способствовало формированию неправильного мнения, что этот факт является неотъемлемой его частью. Это привело в свою очередь к отождествлению понятий имитационного и стохастического моделирования. На самом деле можно привести много примеров неиспользования случайных чисел в имитационном моделировании, например, в случае непрерывного моделирования, равно как и примеры их использования, но к имитационному моделированию не относящиеся. В частности, метод Монте-Карло не вытекает из метода имитационного моделирования, не включает в себя этот метод и не включается в него, хотя и имеет с ним много общего.

Существует также ошибочная формулировка, определяющая имитационное моделирование как процесс изучения с помощью ЭВМ динамических систем, описываемых системами дифференциальных уравнений. С одной стороны, такая формулировка не охватывает, например, большой класс систем массового обслуживания, а с другой стороны, использование традиционных численных методов также далеко от имитационного моделирования, так как в этом случае мы имеем дело не с моделью исследуемого объекта, а с программой численного решения данной совокупности дифференциальных (интегродифференциальных) уравнений.

Очень часто используется ошибочное представление об имитационном моделировании как о процессе построения моделей. Однако существует много примеров построения моделей, которые к имитационному моделированию не относятся. Например, популярные у детей механические игрушки производятся не для того, чтобы с помощью экспериментов на них получать информацию о том, что эти игрушки из себя представляют.

Из большого многообразия имеющихся в литературе определений понятия имитационного моделирования мы остановимся на одном, кото-

рое было дано О.-И.Далом в 1966 г. [1] и уже в течение длительного времени не может быть опровергнуто, несмотря на неоднократные проверки. Суть его заключается в следующем.

Имитационное моделирование – метод исследования, основанный на том, что изучаемая динамическая система заменяется ее имитатором и с ним проводятся эксперименты с целью получения информации об изучаемой системе.

Встречаемое в данном определении понятие "имитатор" представляет собой технический объект, позволяющий воспроизводить модель объекта исследования и реализовывать процедуры решения задач имитационного моделирования на его основе.

Возникновение имитационного моделирования существенно расширило исходные концептуальные схемы познания. Так, длительное время непрерывные модели противопоставлялись моделям дискретным. Долгое время сама природа используемых при моделировании моделей делила специалистов на два лагеря – приверженцев "дискретных" и "непрерывных" моделей. В соответствии с этим происходило даже деление мира на непрерывные и дискретные части, и термины "моделирование непрерывной системы" и "моделирование дискретной системы" неявно предполагали, что используемая система является или непрерывной, или дискретной соответственно. Развитие теории имитационного моделирования позволило объединить эти компоненты. Более того, появились работы, показывающие, что многие системы, традиционно моделируемые с помощью исчисления и потому называемые "непрерывными системами", могут быть успешно замоделированы с помощью арифметических операций и поэтому с полным правом отнесены к дискретным моделям.

До недавнего времени моделирование противопоставлялось теории игр. Имитационное моделирование способствует взаимопроникновению этих направлений науки. Более того, многие вопросы, исследуемые в теории имитационного моделирования, находят там непосредственное применение.

Долгое время специалисты по использованию дискретных моделей не допускали возможности объединения с другими методами, например, с методами исследования операций. Однако используемые в имитационном моделировании методы попадают в разряд схем оптимизации. Дальнейшее развитие в этом направлении позволило объединить системы оптимизации и имитационного моделирования с дискретными моделями и получить обнадеживающие результаты.

Развитие имитационного моделирования выявило много общего с методами управления базами данных. Более того, появились новейшие

СУБД с интерактивным алгоритмическим языком, основанным на понятиях имитационного моделирования – объекты, признаки, совокупности и события.

Возможный вклад имитационного моделирования в развитие экспертных систем, систем программирования, искусственного интеллекта, равно как и в общую теорию систем, уже не вызывает сомнений. Но это в свою очередь, требует дальнейшего развития теории имитационного моделирования путем взаимопроникновения и обогащения различных дисциплин.

1.2. Понятие имитационной модели

Понятие модели сформировалось достаточно давно и в современной науке стало настолько привычным, что сама потребность выяснения содержания этого понятия почти перестала осознаваться. Однако возникновение имитационных моделей потребовало определения их места среди существующих типов моделей и уточнения их содержания с целью обнаружения неклассических тенденций описания изучаемых объектов, а также ликвидации опасности возникновения недоразумений или вульгаризации.

Понимание сущности имитационной модели может быть основано на том факте, что имитационное моделирование как вид познавательной деятельности осуществляется только благодаря взаимодействию трех видов познания: логического, физического и семиотического. С учетом этого, в отличие от других типов моделей [2], имитационную модель можно представить в виде совокупности алгоритмической (математической) модели объекта и имитатора, который воспроизводит поведение объекта в соответствии с этой моделью и имеет средства взаимодействия с пользователем в процессе моделирования.

Если понятия математической и алгоритмической моделей достаточно хорошо определены, то понятие имитатора требует уточнения.

Во-первых, имитаторы реализуются на различных физических принципах, как правило, отличных от моделируемого объекта. В частности, ими могут быть механические, гидродинамические или электронные системы. В настоящее время все большее число имитаторов реализуют на ЭВМ.

Во-вторых, в имитационной модели имитатор должен являться действительно моделирующей системой, т.е. обладать средствами взаимодействия с пользователем. Из этого следует, что не всякий имитатор обуславливает имитационную модель. Простейшим примером этого является дет-

ская механическая игрушка-имитатор. Более сложным примером является имитация с помощью конкретной операционной системы несуществующей (виртуальной) ЭВМ, удовлетворяющей всем требованиям пользователя. В этом случае ЭВМ, управляемая реальной операционной системой, является действительно имитатором, однако метод имитационного моделирования и имитационная модель здесь не реализуются. Операционная система позволяет лишь лучше организовать работу реальной ЭВМ, а не выявить новые свойства несуществующей (виртуальной) ЭВМ. Однако если такой имитатор позволяет использовать его для проведения экспериментов с целью выявления реакции виртуальной ЭВМ на различные ситуации, то в этом случае мы говорим об имитационной модели и имитационном моделировании.

Имитатор может в явном или неявном виде включать алгоритмическую (математическую) модель моделируемого объекта. Например, в качестве имитатора физического маятника может использоваться электронная схема в виде колебательного контура. Математическая модель вида

$$a_2 \frac{d^2x}{dt^2} + a_1 \frac{dx}{dt} + a_0 x = 0;$$

$$x(t_0) = x_0, \quad \left.\frac{dx}{dt}\right|_{t=t_0} = 0$$

в этом случае является составной частью соответствующей имитационной модели, однако она присутствует в неявной форме, так как здесь нет необходимости преобразовывать ее в алгоритмический вид: колебательный контур функционирует сам в соответствии с этим описанием.

Другое дело, если мы используем в качестве имитатора ЭВМ. В этом случае наличие алгоритмической модели является необходимым условием, так как именно алгоритмическая модель будет определять имитационную модель конкретного объекта в силу универсального характера самой ЭВМ. Более того, эта алгоритмическая модель должна быть реализована в программном виде, который позволит ЭВМ в процессе функционирования отображать моделируемые свойства исследуемого объекта.

Важным вопросом при рассмотрении сущности имитационных моделей является определение их места в общепринятой классификации моделей. Так, с точки зрения степени абстрагирования модели от оригинала, все модели подразделяются на две группы: материальные и абстрактные (идеальные). Если материальные модели представляют собой некоторую

естественно или искусственно созданную материальную систему, то абстрактные модели формируются различными способами в сознании людей и отображаются в виде знаковых систем.

Среди материальных моделей наиболее распространенными до недавнего времени были физические модели. Основным отличием таких моделей является сохранение в них частично или полностью физической природы моделируемого объекта. Абстрактные (идеальные) модели в общем случае представляют собой упорядоченную запись символов или знаков. Знаки взаимодействуют между собой не по физическим законам, а согласно правилам, установленным в той или иной области знаний. Широко известными абстрактными моделями являются математические модели.

Важной особенностью материальных моделей является то, что получение информации об оригинале с их помощью осуществляется в результате экспериментальных исследований в процессе их естественного функционирования. В абстрактных моделях получение информации об объекте осуществляется совершенно отличным путем. Например, в случае использования математических моделей этот процесс осуществляется с помощью дедуктивно-математических преобразований соотношений, лежащих в их основе.

Из приведенного выше описания совершенно очевидно, что имитационные модели относятся к материальным моделям и в отличие от физических являются предметно-математическими. Более того, как показали исследования, все предметно-математические модели являются имитационными.

Следует сказать о наиболее распространенной ошибке в отношении имитационных моделей. Так, ряд авторов относят их к математическим моделям, что находится в явном противоречии с рассмотренными ранее определениями имитационного моделирования.

Имитационные модели по типу имитаторов можно разделить на модели с жесткофиксируемой структурой имитатора и модели на основе программируемых имитаторов. Для первых характерно присутствие алгоритмической (математической) модели в неявном виде, для вторых — наоборот. В свою очередь, если в имитационных моделях на основе программируемых имитаторов используются аналоговые вычислительные машины, то такие модели назовем аналоговыми имитационными, если используются ЭВМ, то — цифровыми имитационными. В последнем случае говорят об имитационном моделировании на ЭВМ.

На рис. 1 представлена классификация, отображающая место имитационных моделей в общей структуре моделей. Понятно, что данная

классификация является неполной и может быть расширена с учетом других целей.

Рис. 1. Обобщенная классификация моделей

Поскольку при решении задач проектирования и управления сложными системами наибольший интерес представляет имитационное моделирование на ЭВМ, то в дальнейшем речь будет идти именно об этом типе имитационных моделей. В этой связи для краткости термин "ЭВМ" будем опускать, каждый раз имея его в виду.

1.3. Основы формализации систем при построении имитационных моделей

Особенность имитационного моделирования сложных систем обуславливает потребность в специальных способах построения и анализа имитационных моделей. В частности, составной характер сложной системы диктует представление ее модели в виде тройки $\langle \mathcal{A}, S, T \rangle$, где $\mathcal{A}$ – множество элементов модели, включая элементы внешней среды; S – множество допустимых связей между элементами; T – множество рассматриваемых моментов времени.

Каждый элемент $A_i \in \mathcal{A}$ характеризуется набором атрибутов a_1, a_2, ..., a_k, значения которых будут определять изменения, происходящие в этом элементе. Изменение значений величин a_1, a_2, ..., a_k может осуществляться либо под воздействием внешних по отношению к модели факторов, либо в соответствии с заранее определенными в модели правилами. Совокупность величин a_1, a_2, ..., a_n, определяемых в каждый момент времени $t \in T$ внешними факторами, назовем входными переменными и будем обозначать их x_1, x_2, ..., x_n, причем $n < k$.

В оставшемся подмножестве названных атрибутов выделим те из них, которые аккумулируют информацию, необходимую для прогноза будущей динамики элемента A_i. В теории имитационного моделирования эти величины называются переменными состояния. Из этого определения следует, что если из величин a_1, a_2,..., a_k переменными состояния являются a_{i1}, a_{i2}, ..., a_{im}, где $m \leq k-n$, то по значениям этих переменных $a_{i1}(t), a_{i2}(t),..., a_{im}(t)$ в момент времени $t \in T$ можно определить значения величин $a_{i1}(t')$, $a_{i2}(t'),..., a_{im}(t')$ в момент времени $t' \in T$ для любых t и t', таких что $t' > t$.

Наряду с переменными состояния и входными переменными введем в рассмотрение выходные переменные. Если входные переменные элемента A_i определяют характер воздействия на этот элемент других элементов множества $\mathcal{A}$, то выходные переменные в свою очередь определяют воздействие на них со стороны элемента A_i. Следует отметить, что подмножества переменных состояния и выходных переменных могут перекрываться, т.е. одни и те же выходные переменные могут быть одновременно и переменными состояния (рис. 2).

Рис. 2. Структура переменных моделируемой системы

В процессе имитационного моделирования на каждом этапе, соответствующем моменту времени t, осуществляется вычисление новых значений переменных состояния и выходных переменных на основе текущих значений входных переменных и предыдущих значений переменных состояния. Для реализации этих функций при формализации объекта выделяются оператор перехода из одного состояния в другое и оператор выхода. Рассмотрим суть этих операторов.

Пусть $x = (x_1, x_2, \ldots, x_n)$ – вектор входных переменных, $q = (q_1, q_2, \ldots, q_m)$ – вектор переменных состояния и $y = (y_1, y_2, \ldots, y_l)$ – вектор выходных переменных. Тогда оператор, позволяющий определять значения переменных состояния в следующий момент времени моделирования, назовем оператором перехода из одного состояния в другое или просто оператором перехода. Обозначим его символом "δ".

Оператор δ формирует вектор переменных состояния q в момент времени t_i в случае изменения состояния модели, т.е.

$$q(t_i) = \delta(x(t_i), q(t_{i-1})), \qquad (1.1)$$

где t_i, $t_{i-1} \in T$ и в течение интервала $[t_{i-1}, t_i)$ состояние модели не изменяется.

Следует отметить, что изменение состояния модели может осуществляться не только под воздействием входных сигналов или внешних факторов, а также под воздействием внутренних факторов. В этом случае оператор перехода будет состоять из двух компонент, т.е.

$$q(t_i) = \begin{cases} \delta'(x(t_i), q(t_{i-1})); \\ \delta''(q(t_{i-1})), \end{cases} \qquad (1.2)$$

где второй компонент определяет эту ситуацию.

Оператор, позволяющий определять значения выходных переменных в текущий момент моделирования, назовем оператором выхода и обозначим его f. В общем случае этот оператор можно записать в виде

$$y(t_i) = f(x(t_i), q(t_{i-1})), \qquad (1.3)$$

хотя в более простом случае имеет место следующая запись:

$$y(t_i) = f(q(t_i)). \qquad (1.4)$$

Формализация моделируемого объекта, основанная на выделении множеств входных, выходных переменных и переменных состояния, а также соответствующих операторов перехода и выхода, играет важную

роль в имитационном моделировании. На основе этого сформировалась концепция формализации, которая названа в имитационном моделировании концепцией состояний.

Концепция состояний играет важную роль на этапе формализации. Одним из важнейших условий использования ее на практике является адекватное определение состояний моделируемого объекта. Такое определение должно включать достаточно информации о предыстории элемента, чтобы для каждого текущего состояния прошлое статистически не влияло на прогнозирование его будущего поведения.

Следует также отметить, что переменные состояния в имитационных моделях используются для выполнения ряда специальных функций. К ним относятся:

инициализация имитационной модели, т.е. задание начальных условий моделирования;

прерывание и последующее восстановление имитационного моделирования при выполнении определенных условий или сбое ЭВМ;

отображение траектории изменения значений соответствующих переменных в процессе моделирования.

Опыт использования концепции состояний при построении имитационных моделей показал достаточную трудоемкость процесса формализации в этом случае, особенно при моделировании сложных систем. Попытки структуризации названных компонент имитационной модели с целью упрощения этого процесса привели к возникновению концепции событий и концепции процессов.

Основу концепции событий составляет понятие "события", которое определим следующим образом. Событие — это непротяженное во времени действие, связанное с изменением его состояния. Момент времени, когда объект изменяет свое состояние в соответствии с данным событием, называется моментом наступления этого события. В свою очередь, переход из одного состояния в другое определяет возникновение нового события.

События в модели могут либо отражать реальные действия, происходящие в системе, либо искусственно вводиться. В первом случае происходящие события имеют вполне определенный физический смысл — отказы элементов, окончание или начало некоторых операций и т.п. Именно последовательность указанных событий составляет существо функционирования системы. В этом случае динамика модели фактически повторяет динамику системы, т.е. переход от одного события к другому.

Во втором случае события вводятся искусственно вследствие необходимости представить некоторый непрерывный процесс в дискретной

ЭВМ. Характерным примером является любой численный метод интегрирования дифференциальных уравнений. В подобных моделях моменты наступления событий определяются шагом интегрирования. Шаг интегрирования либо назначается постоянным, либо, если это необходимо, изменяется в соответствии со специально установленным механизмом.

Использование концепции событий позволяет структуризировать операторы перехода и выхода путем выделения в них компонентов, являющихся общими для обработки отдельных событий или группы событий. Особенно это становится важным, если число всевозможных событий, характеризующих динамику функционирования моделируемого объекта, достаточно велико.

Одной из наиболее трудных процедур в формализации моделируемых объектов с использованием концепции событий является установление взаимосвязей между различными событиями. Очень часто, особенно когда их число велико, связи между событиями оказывается довольно трудно систематизировать. Это, в свою очередь, порождает путаницу, а нередко и непонимание процесса функционирования системы в целом. Стремление упорядочить сложную систему взаимосвязей между событиями, временами их возникновения и условиями их реализации привело к возникновению концепции процессов.

"Процесс" – это наиболее часто употребляемый и наименее точно определенный термин, используемый в любой области науки и техники. С самых общих позиций процесс – это некоторая полезная работа, совершаемая по заданной программе и приводящая к определенному планируемому результату.

В каждой конкретной предметной области понятие процесса определяется исходя из особенностей формализации объектов, принадлежащих этой области. Так Вирт – создатель языка Паскаль – определил процесс через понятие "действия": "Если действие можно разложить на части, то оно называется процессом... То, что выполняет действие согласно заданным инструкциям, является процессором".

В терминологическом толковом словаре фирмы IBM процесс определен как "систематизированная последовательность операций, направленная на получение конкретного результата". С точки зрения системотехника определение процесса мало отличается от приведенных и заключается в следующем. "Процесс – это выполнение программы. Он по своей сути активен, т.е. способен производить действие. Процесс от программы отличает то, что последняя по существу пассивна, т.е., расположенная на некотором носителе данных, она не способна производить никаких действий" [1].

Понятие процесса в имитационном моделировании обычно связывается с двумя типами действий: протяженными и непротяженными во времени. Любое конкретное событие является результатом непротяженного во времени действия, так как оно происходит мгновенно в силу сделанных нами предположений. В отличие от этого протяженное во времени действие характеризуется парой событий, первое из которых отмечает его начало, а второе – завершение, причем между ними могут происходить другие события. Продолжительность действия в этом случае определяется моментами времени наступления событий начала и завершения действия.

Исходя из этого, введем понятие процесса в терминах определенного набора событий, связанных с функционированием системы. Пусть в системе происходят события c_1, c_2, ..., c_n, образующие множество C. Тогда запись вида

$$P: (c_{i1} \rightarrow c_{i2} \rightarrow \ldots \rightarrow c_{ik}).$$

будет определять процесс с именем P путем задания последовательности следующих друг за другом событий $c_{ij} \in C$. Множество C назовем алфавитом процесса, а множество событий, в которых процесс участвовал, – протоколом поведения процесса.

Понятие процесса является концептуально более емким понятием, чем понятие состояния или события. В случае процессного описания появляется возможность декомпозиции множества событий на подмножества, связанные с изменением определенных атрибутов или переменных модели. Осуществляемая в этом случае структуризация множества переменных связывается с выделением в системе самостоятельных элементов, выполняющих вполне определенные функции в процессе ее функционирования. Это позволяет из множества событий C, определяющих динамику системы в целом, каждому выделенному i-му элементу поставить в соответствие "свои" события, которые образуют подмножество C_i, где

$$C = \bigcup_{i=1}^{n} C_i .$$

В свою очередь конечная последовательность событий, принадлежащих подмножеству C_i, будет описывать процесс P_i, соответствующий выполняемым этим элементом действиям.

Поскольку при процессном описании действие каждого элемента определяется парой событий его начала и завершения, а между ними могут происходить другие события, то удобно ввести в рассмотрение еще одно понятие, называемое активностью. Элементарное действие, выполняемое элементом в течение времени между двумя ближайшими события-

ми, называется активностью и играет важную роль в имитационном моделировании. Фактически, логически связанный набор активностей образует процесс, причем сама активность выступает как простейший процесс.

Формализация объектов моделирования на уровне активности находит широкое применение на практике, особенно при построении имитационных моделей систем массового обслуживания. В этом случае структуризация осуществляется путем объединения двух связанных с выделенным элементом событий – началом элементарного действия и его окончанием. Элементы, над которыми эти действия осуществляются, называются транзактами. Транзакты сами никаких действий не производят, а элементарные действия или активности связываются с изменением их атрибутов.

Особое значение в концепции процессов приобретают вопросы организации их взаимодействия. Взаимодействие процессов состоит в передаче сообщений и является специальным классом событий, наступление которых может требовать одновременного участия более одного независимо описанного процесса. Совокупность действий, приводящих к возникновению взаимодействия, называется фазой процесса. Процесс в этом случае можно представить как совокупность фаз, которые последовательно располагаются на оси времени.

Вопросы организации взаимосвязей событий в процессе имитационного моделирования представляют самостоятельный интерес, независимо от используемой концепции формализации моделируемого объекта. Более того, они тесно связаны с продвижением времени в модели, поскольку в каждый момент моделирования выполняются следующие функции:

формирование значений переменных состояния и выходных переменных на основе текущих значений входных переменных и предыдущих значений переменных состояния;

определение интервала времени относительно текущего времени, в течение которого определенные значения переменных состояния не изменятся при отсутствии входных воздействий;

определение момента времени появления следующего события в модели или следующего момента времени моделирования.

Названные вопросы будут рассмотрены в третьей главе настоящего пособия.

1.4. Отображение реального времени при имитационном моделировании

В теории имитационного моделирования вводится в рассмотрение три типа времени: реальное, модельное или системное и машинное. Реаль-

ное время — это время, в котором функционирует моделируемая система, т.е. время, определяемое ходом обычных часов.

Модельное или системное время — это время, значение которого определяется в имитационной модели специальной переменной. Принимаемые в процессе моделирования значения этой переменной отображают ход часов реального времени.

Понятие машинного времени связано лишь с временными характеристиками работы ЭВМ в процессе исполнения модели. Это время определяется в общем случае эффективностью алгоритма моделирования и быстродействием ЭВМ. Например, работа моделируемой системы в течение года, часа или суток может имитироваться на ЭВМ в течение нескольких минут или секунд, в то же время работа быстропротекающих процессов может имитироваться на ЭВМ в течение длительного периода.

Процесс отображения реального времени в имитационных моделях имеет ряд специфических особенностей, обусловленных как дискретным характером обработки информации на ЭВМ, так и свойствами моделируемой системы. В общем случае все объекты существуют в непрерывном времени. Однако в процессе моделирования это время T^0 мы заменяем множеством T действительных чисел $t_1, t_2, \ldots,$ которое линейно упорядочено и имеет привычную метрику.

С каждым моментом модельного времени t_i в модели связывается возникновение тех или иных событий. Последовательность возникновения этих событий должна соответствовать динамике моделируемого объекта. С этой целью на этапе формализации мы должны также определить:

упорядоченное множество дискретных временных элементов T ;

метрику, задающую длину интервалов времени;

отображение, обуславливающее соответствие между реальным временем и его эквивалентом в модели;

алгоритм продвижения модели во времени.

При формировании множества моментов времени моделирования T в теории имитационного моделирования можно выделить два подхода. Первый подход основан на выделении моментов времени наступления событий в процессе функционирования объекта и называется дискретно-событийным подходом. Такой подход характерен для моделирования дискретных процессов и систем.

Второй подход основан на формировании моментов времени, рассматриваемых в процессе моделирования, в соответствии с выражением

$$t_{i+1} = t_i + \Delta t \qquad (1.5)$$

и используется тогда, когда не удается в моделируемом процессе выделить реально происходящие в нем события. В этом случае события вводятся искусственно и определяются дискретным характером обработки информации на ЭВМ. Данный подход широко используется при моделировании непрерывных процессов и называется дискретно-временным.

Следует также отметить, что при имитационном моделировании сложных систем оба подхода к определению множества T могут использоваться вместе. Так в многокомпонентных моделях подмножества T^i, где $T = \bigcup_i T^i$, могут определяться для каждого i-го элемента модели одним из указанных подходов. Непротиворечивость такого описания множества T заключается в том, что оба подхода исповедуют дискретный принцип описания моментов времени моделирования. Однако от того, какой подход выбран, будет зависеть алгоритм отображения реального времени в имитационной модели, о котором речь пойдет ниже.

Поскольку множество T – упорядоченное множество дискретных временных элементов, то отношение линейного упорядочения ($\overset{\leqslant}{t}$) имеет следующий смысл для любых $t_1, t_2, t_3 \in T$:

1) $t_1 \overset{\leqslant}{t} t_1$;
2) $t_1 \overset{\leqslant}{t} t_2$ и $t_2 \overset{\leqslant}{t} t_1$ для $t_1 = t_2$;
3) $t_1 \overset{\leqslant}{t} t_2$ и $t_2 \overset{\leqslant}{t} t_3$ для $t_1 \overset{\leqslant}{t} t_3$;
4) для любых t_1 и $t_2 \in T$ либо $t_1 \overset{\leqslant}{t} t_2$ или $t_2 \overset{\leqslant}{t} t_1$.

Метрика $d(t_i, t_j)$ представляет собой функцию, позволяющую моментам времени $t_i, t_j \in T$ поставить в соответствие величину интервала, определяемого ими. Естественно, что данная функция зависит от единицы измерения времени в модели. Одно дело, если мы измеряем время сутками, другое – если в качестве единицы измерения взята микросекунда.

В общем случае метрика d обладает следующими свойствами для любых моментов времени $t_j, t_k, t_i \in T$:

1) $d(t_j, t_k) > 0$ для $j \neq k$;
2) $d(t_j, t_k) = 0$ для $j = k$;
3) $d(t_j, t_k) = d(t_k, t_j)$;
4) $d(t_j, t_i) \leqslant d(t_j, t_k) + d(t_k, t_i)$.

Если выбрана единица измерения времени в имитационной модели, то любой интервал времени будет кратен этой всличине. В свою очередь множество T может быть представлено подмножеством целых неотрицательных чисел I. В этом случае будет справедливо соотношение

$$d(t_j, t_k) = 1 + d(t_{j+1}, t_k),$$

где $\ldots\ t_{j-1}, t_j, t_{j+1}, \ldots, t_{k-1}, t_k, t_{k+1}\ldots$ – последовательность возможных значений моментов времени, построенная исходя из выбранной единицы времени.

Изменение состояния имитационной модели может осуществляться только при наступлении определенных событий в соответствующие моменты времени моделирования и с учетом отображения $T^Q \to T$. Можно выделить два типа событий – внешние и внутренние. Если при возникновении внешних событий происходит изменение атрибутов элементов модели в соответствии с оператором δ', как указано в (1.2), то внутренние события обрабатываются оператором δ''. В этом случае возникает вопрос: как долго в новом состоянии будет находится система при отсутствии воздействия извне, т.е. внешнего события?

Для ответа на данный вопрос в дополнение к ранее названным операторам в состав имитационной модели вводится так называемый оператор времени, который определяется на этапе формализации моделируемого объекта с учетом выбранного подхода к формированию множества T. При использовании дискретно-временного подхода этот оператор базируется на выражении (1.5), где величина Δt определяется исходя из указанных ранее соображений. В случае использования дискретно-событийного подхода этот оператор имеет более сложный вид, зависящий от специфики функционирования объекта.

Оператор времени может входить в состав оператора перехода или может существовать в виде самостоятельного компонента. В каждом конкретном случае это определяется выбранной концепцией формализации объекта и стремлением разработчика имитационной модели упростить процесс ее разработки.

Возникновение событий в процессе имитационного моделирования имеет в общем случае стохастический характер: события различных типов могут возникать в заранее непредсказуемые моменты времени, значения которых определяются оператором времени в процессе исполнения модели. Определение взаимосвязи возникающих событий с целью обеспечения логики функционирования объекта в имитационной модели осуществляется алгоритмом продвижения модели во времени, базирующемся на операторе планирования событий. Оператор планирования упорядочивает будущие события по времени их возникновения и на основе этого определяет, какое событие будет следующим.

На этапе формализации объекта моделирования достаточно определить только тип алгоритма продвижения модели во времени в силу его относительной инвариантности. Разработка этой части имитационной модели представляет самостоятельный интерес и будет рассмотрена в последующих разделах пособия.

Глава 2. АЛГОРИТМИЧЕСКИЕ СПЕЦИФИКАЦИИ ФОРМАЛИЗУЕМЫХ СИСТЕМ

Попытки упорядочить действия разработчиков имитационных моделей на этапе формализации объектов моделирования привели к появлению типовых спецификаций систем, позволяющих описание любой системы свести к одной из них. С одной стороны, это определяет конкретные пути анализа объектов моделирования независимо от их принадлежности к той или иной сфере деятельности человека, а с другой — позволяет сразу сориентироваться на методы их реализации в рамках существующих систем программирования или моделирования.

К сожалению, в отечественной литературе этому вопросу уделено недостаточное внимание. Наиболее изученной формальной схемой на которую ссылаются многие авторы, является агрегатная схема, предложенная Н.П. Бусленко. В дальнейшем мы попытаемся развить полученные в настоящее время научные результаты в области имитационного моделирования на более широкий класс объектов.

2.1. Иерархия спецификаций систем

В практике построения имитационных моделей наиболее часто используется следующая иерархия спецификаций систем, отображающих различным образом ее внутреннюю структуру:

1-й уровень– простейшая спецификация систем (C_1);

2-й уровень – обобщенная спецификация систем (C_2);

3-й уровень – спецификация структурированных систем (C_3);

4-й уровень – спецификация систем на уровне взаимосвязанных подсистем (C_4);

Простейшая спецификация систем представляется в виде пятерки:

$$C_1 = \langle T, X, \Omega, Y, F \rangle$$

и используется в случае, когда с помощью отображения $F: X \times T \to Y$ множеству значений входных переменных X можно поставить в соответствие множество выходных переменных Y на всем множестве моментов времени моделирования T и множестве входных процессов $\Omega = \{\omega: T \to X\}$. Поскольку при моделировании сложных систем спецификация C_1 используется редко, то основное внимание уделим спецификациям C_2, C_3 и C_4.

Обобщенная спецификация систем имеет следующий вид:

$$C_2 = \langle T, X, \Omega, Q, Y, \delta, \lambda \rangle, \qquad (2.1)$$

где T – множество моментов времени моделирования; X – множество

значений входных переменных; Y – множество значений выходных переменных; Ω – множество входных процессов, причем $\Omega \in X \times T$; Q – множество значений переменных состояния; δ – функция, определяющая переход системы из одного внутреннего состояния в другое; λ – функция выхода, определяющая значения выходных переменных.

Множество T определяет модельное время и служит для упорядочивания событий в модели, каждое из которых отображает совокупность изменений, происходящих в динамической системе в определенный момент ее существования. В общем случае $T \subset R^+$ или $T \subset I^+$, где R^+ – множество неотрицательных действительных чисел, I^+ – множество неотрицательных целых чисел.

Множество X определяет взаимодействие системы с внешней средой. В общем случае $X \subset R^n$, где n – число входных переменных $(n \in \{1, 2, 3, \ldots\})$ и может быть либо дискретным, либо непрерывным. В случае дискретного множества используется описание вида $X \cup \{\emptyset\}$, где $\emptyset$ определяет отсутствие входного события.

Множество Y отличается от множества X своим назначением. Если посредством входных переменных внешняя среда воздействует на систему, то выходные переменные определяют воздействие со стороны системы на внешнюю среду. Все остальное, сказанное относительно множества X , справедливо и для множества Y .

Для описания характера изменения во времени входных воздействий используется понятие входного процесса, который представляет собой отображение $\omega: T \to X$. Если $T = [t_н, t_к]$, то $\omega: [t_н, t_к] \to X$, где $t_н$ – начальный момент времени, $t_к$ – конечный момент времени, причем $t_н < t_к$. Если X – дискретное множество, то в этом случае $\omega: [t_н, t_к] \to X \cup \emptyset$ и $\omega(t) = \emptyset$ означает отсутствие входного сигнала в момент времени $t \in T$, за исключением конечного множества моментов времени $\{t_1, t_2, \ldots, t_n\} \subset [t_н, t_к]$.

Множество Q определяет возможные внутренние состояния системы и используется исключительно для целей моделирования. Необходимость введения переменных состояния связана с наличием предыстории системы или памяти системы, определяющей ее поведение в будущем.

Преобразование $\delta: Q \times \Omega \to Q$ определяет функцию перехода системы из одного состояния в другое. Если в момент $t_i \in T$ система находится в состоянии $q \in Q$ и на вход системы поступает входной сегмент $\omega: [t_i, t_f] \to X$, то состояние системы в момент $t_f \in T$ будет определяться значением функции $\delta(q, \omega)$. Эта функция обладает композиционным свойством, т.е.

$$\delta(q, \omega) = \delta(\delta(q, \omega_1), \omega_2),$$

где ω_1 – часть входного сегмента ω, определяемая отображением $\omega_1 : [t_i, t] \to X$, а ω_2 – оставшаяся часть ω, определяемая отображением $\omega_2 : [t, t_f] \to X$ при условии $t \in [t_i, t_f]$.

Выходная функция в общем случае определяется отображением $\lambda : X \times Q \times T \to Y$. Однако наиболее часто это отображение имеет более простой вид, т.е. $\lambda : Q \to Y$, позволяющий определять значения выходных сигналов по состоянию системы в момент времени t.

Отличие структурированного описания системы от рассмотренного ранее заключается в том, что входящие в состав спецификации

$$C_3 = \langle T, X', \Omega', Q', Y', \delta', \lambda' \rangle \qquad (2.2)$$

множества и функции могут быть структурированы, т.е. представлены с помощью декартовых произведений элементарных множеств и функций.

В общем случае структурированное множество представляет собой описание следующего вида:

$$\mathcal{A}' = \langle \mathcal{A}, D, \{\mathcal{A}_a \mid a \in D\}, i \rangle$$

где $\mathcal{A}$ – структурируемое множество; D – упорядоченное множество имен выделенных подмножеств $\{a_1, a_2, \ldots\}$; $\mathcal{A}_a$ – выделенное в $\mathcal{A}$ подмножество, соответствующее имени a, $a \in D$; i – функция назначения или структуризации, которая определяется отображением $i : \mathcal{A} \to \prod_{a \in D} \mathcal{A}_a^{\mathcal{A}_a}$, где $\prod$ – знак декартового произведения.

Для определенного таким образом понятия структурированного множества можно ввести понятие структурированной функции. В частности, отображение одного структурированного множества в другое будет определять структурированную функцию. Например, если множество $\mathcal{A}'$ структурируется множеством $\mathcal{A}$, а множество $\mathcal{B}'$ структурируется множеством $\mathcal{B}$, то функция $f : \mathcal{A} \to \mathcal{B}$ тоже структурирована и представляет собой семейство функций f_b, соответствующих каждому имени b из B.

Спецификация системы на уровне связанных подсистем имеет следующий вид:

$$C_4 = \langle D, \{C_a\}, \{I_a\}, \{Z_a\} \rangle, \qquad (2.3)$$

где D – множество имен выделенных подсистем; C_a – спецификация подсистемы с именем a $(a \in D)$; I_a – подмножество элементов множества D $(I_a \subset D)$, оказывающих влияние на подсистему с именем a; Z_a – функция взаимодействия $Z_a : \prod_{b \in I_a} Y_b^{Y_b} \to X_a$, определяющая воздействие со стороны других подсистем на подсистему с именем a.

Из представленного описания системы видно, что первые два компонента $\langle D, \{C_a\} \rangle$ определяют множество элементов (подсистем) системы, причем $C_a = \langle T, X_a, Q_a, Y_a, \delta_a, \lambda_a \rangle$, а другие два компонента $\langle \{I_a\}, \{Z_a\} \rangle$ определяют схему связи подсистем.

Рассматривая различные уровни описания систем, следует отметить их тесную взаимосвязь. В частности, использование процедуры декомпозиции позволяет перейти от описания системы на i-м уровне к описанию системы на j-м уровне при $i < j$. В свою очередь, используя процедуру агрегирования, можно с j-го уровня описания системы перейти на i-й уровень при тех же соотношениях между i и j. Правомерность таких преобразований показана в известных работах по имитационному моделированию. Более того, можно доказать эквивалентность таких преобразований.

2.2. Типовые алгоритмические спецификации систем

Формальное представление моделируемой системы во многом облегчается благодаря существованию хорошо изученных дискретных и непрерывных абстрактных систем: систем, описываемых дифференциальными уравнениями; автоматных систем; систем массового обслуживания; агрегативных систем; "потоково-уровневых" систем и т.п. Эти системы могут быть применены для формализации весьма широкого класса объектов с учетом самых различных требований. При этом необходимо помнить, что создаваемое описание должно полностью отражать концепцию исследователя относительно процесса функционирования исследуемого объекта.

В общем случае можно выделить три разновидности спецификации (2.1) с помощью которых описываются все известные абстрактные системы:

спецификацию систем, описываемых дифференциальными уравнениями (ДУ-спецификация);

спецификацию дискретно-событийных систем (ДС-спецификация);

спецификацию дискретно-временных систем (ДВ-спецификация).

Из названий спецификаций видно, что каждая из них предназначена для описания систем, имеющих определенный характер изменения значений переменных во времени. Например, ДУ-спецификация используется для формализации систем, в которых значения переменных изменяются непрерывно во времени. ДС-спецификация ориентирована на описание систем, в которых изменения значений переменных связаны с наступлением определенных событий и происходят в дискретные моменты времени, распределенные произвольно на временной оси. Что касается ДВ-спецификации, то она характерна для систем, в которых все изменения происходят в строго определенные моменты времени, определяемые некоторым приращением Δt. Поскольку эти моменты времени вычисляются с использованием выражения $t_i = t_0 + i\Delta t$, то каждому значению

t_i можно поставить в соответствие целочисленную величину, а множество T преобразовать в целочисленное.

В табл. 1 приведены основные характеристики трех вышеназванных спецификаций. Особенности их использования при построении имитационных моделей будут рассмотрены в последующих разделах пособия.

Таблица 1

Компоненты спецификации	Тип спецификации системы		
	ДУС	ДСС	ДВС
T	Действительные числа	Действительные числа	Дискретные целые числа
X, Q, Y	Действительные числа	Произвольные числа	Произвольные числа
Ω	Кусочно-непрерывные сегменты	Дискретно-событийные сегменты	Последовательности
$Q \times T$, $Y \times T$	Непрерывные сегменты	Кусочно-постоянные сегменты	Последовательности

2.2.1. Спецификация систем, описываемых дифференциальными уравнениями

В общем случае ДУ-спецификация относится к классу неалгоритмических спецификаций, так как для получения траекторий изменения состояний системы и значений выходных переменных необходимо использовать либо аналитическое решение, либо какой-нибудь метод численного интегрирования. ДУ-спецификация имеет следующий вид:

$$D = \langle X, Q, Y, f, \lambda \rangle, \qquad (2.4)$$

где X – множество значений входных переменных; Q – множество значений переменных состояния; Y – множество значений выходных переменных; f – функция, реализующая отображение $f: Q \times X \to Q$, путем описания соответствующих производных; λ – выходная функция, реализующая отображение $Q \to Y$.

При выполнении условий Липшица представленная в ДУ-спецификации система уравнений имеет единственное решение. В этом случае данная спецификация может быть преобразована в более общую спецификацию типа (2.1) следующим образом:

$$C_Д = \langle T_Д, X_Д, \Omega_Д, Q_Д, Y_Д, \delta_Д, \lambda_Д \rangle,$$

где $t \in T_\Delta : [t_0, \infty] \subset R^+;$

$X_\Delta = X : R^m, \; m \in I^+;$

$\Omega_\Delta = \{\omega : [t_0, t_0 + \tau] \to X, \tau \to 0\};$

$Q_\Delta = Q : R^n, \; n \in I^+;$

$Y_\Delta = Y : R^l, \; l \in I^+;$

$\delta_\Delta = f :$ { если система дифференциальных уравнений имеет единственное решение $\Phi(t)$, то преобразование $f : Q \times X \to Q$ может быть получено из этого решения };

$\lambda_\Delta = \lambda : Q_\Delta \to Y_\Delta.$

2.2.2. Спецификация дискретно-событийных систем

Понятие дискретно-событийной системы объединяет в себе системы, обладающие одним общим свойством – все они базируются на концепции события. В частности, такими являются системы:

основанные на формировании следующего события;

основанные на планировании активностей;

основанные на взаимодействии процессов.

Спецификация дискретно-событийных систем имеет следующий вид

$$DC = \langle X, S, Y, \delta, \lambda, ta \rangle, \qquad (2.5)$$

где X – множество внешних событий; S – множество внутренних состояний системы; Y – множество значений выходных переменных; δ – спецификация функций перехода системы из одного состояния в другое; λ – выходная функция; ta – временна́я функция, определяющая момент наступления следующего события в системе.

Временна́я функция представляет собой отображение $ta : S \to R^+$, определяющее период времени, в течение которого система будет находиться в состоянии s ($s \in S$). Для того чтобы отразить тот факт, что система будет находится в состоянии s в течение определенного интервала времени с момента его наступления, введем понятие множества обобщенных состояний системы Q, которое определим следующим образом:

$$Q = \{(s, e) \mid s \in S, \; 0 \leq e < ta(s)\},$$

Здесь e – период времени, в течение которого система находится в состоянии s на текущий момент моделирования; $ta(s)$ – период времени нахождения системы в состоянии s при отсутствии внешних событий.

С учетом сказанного конкретизируем спецификацию функций перехода системы из одного состояния в другое. Она будет состоять из двух компонент: описания внутренней функции перехода $\delta^{\emptyset}$ и внешней функции перехода $\delta^{вн}$. Функция $\delta^{\emptyset}$ представляет собой отображение $\delta^{\emptyset}: S \to S$, которое определяет новое состояние системы $\delta^{\emptyset}(s)$, в которое она перейдет из состояния s после истечения времени $ta(s)$ и при отсутствии внешних событий. Функция $\delta^{вн}$ представляет собой отображение $\delta^{вн}: Q \times X \to S$, которое в отличие от внутренней функции перехода $\delta^{\emptyset}$ определяет новое состояние системы $\delta^{вн}(s,e,x)$ при наступлении внешнего события x $(x \in X)$ до момента времени, определяемого величиной $ta(s)$.

Поскольку моменты наступления событий в системе определяются элементами упорядоченного по возрастанию множества $T = \{t_0, t_1, t_2, \ldots, t_k\}$, то для формализации дискретно-событийных систем удобно ввести в рассмотрение пустой элемент события – $\emptyset$, который совместно с множеством X будет определять всевозможные внешние состояния, т.е. $X^{\emptyset} = X \cup \emptyset$. В этом случае входной процесс будет описываться отображением $\omega: T \to X^{\emptyset}$.

Выходная функция λ представляет собой отображение $\lambda: S \to Y$ и определяет значения выходных переменных в соответствующие моменты времени.

Между спецификацией дискретно-событийных систем и обобщенной спецификацией систем (2.1) также можно установить соответствие, которое заключается в следующем:

$$C_{DC} = < T_{DC}, X_{DC}, \Omega_{DC}, Q_{DC}, Y_{DC}, \delta_{DC}, \lambda_{DC} >,$$

где T_{DC} – множество моментов времени, определяемых наступлением как внешних, так и внутренних событий системы;

$$X_{DC} = X^{\emptyset};$$

$$\Omega_{DC} = \{ \omega: T_{DC} \to X_{DC} \};$$

$$Q_{DC} = \{ (s,e) / s \in S, \ 0 \leq e \leq ta(s) \};$$

$$Y_{DC} = Y;$$

$$\delta_{DC} = \delta;$$

$$\lambda_{DC} = \lambda.$$

При определении функции перехода системы из одного состояния в другое необходимо учитывать следующее. Находясь в начальный момент

времени $t_н$ в состоянии $q=(s,c)$, система C_{DC} будет изменять свои состояния как при наступлении внешних событий, т.е. при воздействии процесса $\omega: [t_н, t_k] \to X_{DC}$, так и при наступлении внутренних событий. Пусть $\tau_1, \tau_2, \ldots, \tau_n$ – моменты времени наступления внешних событий (если множество внешних событий пусто, то $\tau_1 = t_k$), а $t_1, t_2, \ldots, t_m$ – моменты наступления внутренних событий $s_1, s_2, \ldots, s_m$ в период $[t_н, \tau_1]$. Кроме того, момент наступления начального состояния s_0 обозначим t_0, и этот момент может предшествовать начальному времени моделирования $t_н$.

С учетом принятых положений для вычисления величин t_i и s_i $(i=\overline{0,m-1})$, будут иметь место следующие выражения:

$$t_0 = t_н - e;$$
$$t_{i+1} = t_i + ta(s_i);$$
$$s_0 = s;$$
$$s_{i+1} = \delta^{\emptyset}(s_i).$$

Обобщенное состояние системы в момент времени τ_1 будет определяться соотношением $q' = (s_m, \tau_1 - t_m)$ до наступления события x_1 и соотношением $q_1 = (\delta^{t_н}(q', x_1), 0)$ – после наступления события x_1. Аналогично определяются состояния системы в моменты времени $\tau_2, \tau_3, \tau_4, \ldots, \tau_n$. Окончательно, в момент времени t_k состояние системы будет равным $\delta(q, \omega)$.

Среди всех возможных состояний следует выделить такие, для которых справедливо утверждение $ta(s) = \infty$. Поскольку в этом случае состояние системы может быть изменено только под влиянием внешнего воздействия, то такие состояния будем называть пассивными, и для них определение функции $\delta^{\emptyset}$ не требуется.

Все отличные от пассивных состояния системы будем называть активными. Более того, если для активных состояний выполняется соотношение $ta(s) = 0$, то назовем их мгновенными.

2.2.3. Спецификация дискретно-временных систем

Основной характеристикой дискретно-временных систем является изменение состояний системы в дискретные моменты времени, определяемые выражением $t_i = t_н + i\Delta t$, где $t_i \in [t_н, t_k]$ и $i=\overline{1,k}$. В частности, к таким системам относятся алгоритмические машины Тью-

ринга и Поста, автоматные и аналогичные им системы. Соответствующая спецификация таких систем имеет вид

$$DB = \langle X, Q, Y, \delta, \lambda \rangle, \qquad (2.6)$$

где X, Q, Y – множества значений входных переменных, переменных состояния и выходных переменных соответственно; δ – функция перехода системы из одного состояния в другое в моменты времени, определяемые приращением Δt; λ – выходная функция.

В отличие от ранее рассмотренных систем функция перехода δ в этом случае определяется отображением $\delta : Q \times X \to Q$ и характеризует изменение состояний системы в последовательные моменты времени, соответствующие целочисленным переменным, принадлежащим множеству I^+. В свою очередь, множество I^+ является аналогом множества T при принятых ранее допущениях.

Оператор выхода λ определяет значения выходных переменных в соответствующие моменты времени $t \in T$ или $i \in I^+$ и в зависимости от типа автомата может описываться либо отображением $\lambda : Q \to Y$ (например, для автоматов Мура), либо отображением $\lambda : X \times Q \to Y$ (например, для автоматов Мили). Поскольку между различными типами автоматов существует взаимно однозначное соответствие, то в дальнейшем за основу может быть положено любое преобразование.

Соответствие между обобщенной спецификацией систем (2.1) и ДВ-спецификацией устанавливается следующим образом:

$$C_{DB} = \langle T_{DB}, X_{DB}, \Omega_{DB}, Q_{DB}, Y_{DB}, \delta_{DB}, \lambda_{DB} \rangle,$$

$$T_{DB} = I^+;$$

$$X_{DB} = X;$$

$$Q_{DB} = Q;$$

$$Y_{DB} = Y;$$

$\Omega_{DB} = X \times I^+$, т.е. отображение ω будет формировать последовательность $\omega(t_\mu), \omega(t_\mu + 1), \ldots, \omega(t_k)$ значений входных переменных на интервале $[t_\mu, t_k]$; $\delta_{DB} = \delta(q, \omega)$; $\lambda_{DB} = \lambda$.

Учитывая тот факт, что значения входных переменных формируются в последовательные моменты времени, для данного типа систем часто вводят в рассмотрение расширенную функцию перехода $\delta' : Q \times X^* \to Q$, которая с функцией $\delta(q, \omega)$ имеет следующую связь:

$$\delta(q,\omega) = \delta'(q, x^*),$$

где x^* – последовательность значений входных переменных, принадлежащая последовательности $\omega(t_\mu), \omega(t_\mu+1), \ldots, \omega(t_k)$, а совокупность последовательностей x^* образует множество X^*.

Для этого случая будут также справедливы соотношения:

$$\delta'(q, \Lambda) = q;$$

$$\delta'(q, x\, x^*) = \delta'(\delta(q,x), x^*),$$

где Λ – пустая последовательность входных сигналов; $x \in X$, $x^* \in X^*$.

Следует также отметить, что поведение дискретно-временных систем не зависит от выбора начального момента моделирования, т.е. такие системы являются инвариантными относительно времени.

2.3. Спецификации структурированных и модульных систем

При построении имитационных моделей сложных систем очень редко удается использовать описанные ранее спецификации типа (2.4), (2.5) и (2.6). Во многих случаях с целью упрощения процесса формализации сложная система декомпозируется на более простые подсистемы на основе структуризации или модульного представления. Особенно это характерно для дискретно-событийных систем, представляющих с этой точки зрения наибольший интерес. В силу важности этих вопросов рассмотрим, как осуществляются структуризация и модульное представление сложных систем и используемые для этих целей спецификации.

2.3.1. Структуризация дискретно-событийных систем

Алгоритмическая спецификация дискретно-событийных систем (2.5) характерна для применения концепции состояний в процессе формализации объектов моделирования. Использование концепции событий или процессов приводит к необходимости структуризации компонент, определенных в ней. Для этого случая структурированная дискретно-событийная спецификация (СДС-спецификация) имеет следующий вид:

$$СДС = \langle D, \{S_a\}, \{I_a\}, \{F_a\}, æ \rangle, \qquad (2.7)$$

где D – множество имен компонент; S_a – множество последовательных состояний компонента с именем $a\,(a \in D)$; I_a – подмножество элементов множества D, связанных с компонентом $a\,(a \in D)$ определенными от-

ношениями; F_a – функция перехода, определяющая отображение $(S_a \times R^+)^{n+1} \to (S_a \times R^+)^{n+1}$ при $n = |I_a|$, задающим число элементов в подмножестве I_a, $a \in D$; $æ$ – функция, определяющая порядок выбора элементов из множества D, т.е. $æ: D^\alpha \to D$ и D^α означает подмножество множества D.

В рамках данной спецификации описание системы с точностью до следующего события базируется на введении переменной σ, которая определяет оставшееся время нахождения элемента в текущем состоянии, причем $\sigma \in R^+$ и $\sigma = ta(s) - e$. При активизации какого-либо элемента происходит изменение значений (s_j, σ_j) у элементов, которые с ним связаны, с помощью функций перехода.

В случае описания системы на уровне планируемых активностей подмножество I_a объединяет как совокупность элементов, которые оказывают влияние на элемент с именем a, так и совокупность элементов, на которые элемент с именем a оказывает влияние. Структурированная функция перехода в этом случае представляется парами (c_a, f_a), где c_a – предикат, определяющий условия воздействия на элемент с именем a элементов подмножества I_a, а f_a – функция, определяющая воздействие элемента a на элементы подмножества I_a. При активизации элемента с именем a происходит проверка выполнения условия c_a и если это условие выполняется, то в соответствии с функцией f_a осуществляется изменение значений (s_j, σ_j) элементов, связанных с ним. В противном случае, все остается без изменений до очередной проверки при наступлении следующего события.

Описание систем на уровне взаимодействующих процессов осуществляется аналогично системам с планируемыми активностями. Единственным отличием является представление каждой функции перехода F_a в виде конечного множества поименованных пар "условие-действие". Изменение состояний соответствующих элементов осуществляется в соответствии со значением, так называемой фазовой переменной, диапазон изменения которой определяется совокупностью возможных меток этих пар. При активизации соответствующего элемента по значению фазовой переменной определяется пара (c_j, f_j) и затем выполняются точно такие же операции, как и в системах с планированием активностей.

2.3.2. Модульное представление многокомпонентных систем

В настоящее время модульный или агрегатный подход при построении имитационных моделей сложных систем находит все более широкое распространение на практике. В значительной степени это связано с тем, что создание имитационной модели на заключительном этапе сводится к

разработке сложного программного комплекса, и здесь концепция модульности при программировании оказывается очень полезной для большинства современных программных систем.

Понятие модуля в проектировании программного обеспечения достаточно хорошо известно. В имитационном моделировании в это понятие вкладывается тот же смысл, однако, учитывая специфику решаемых задач, определим модуль или агрегат как алгоритмическую конструкцию, которая формально определяет элемент системы с помощью одной из спецификаций типа (2.4), (2.5), (2.6) или (2.7).

Объединение модулей в единое целое путем определения взаимодействия между ними образует композиционную агрегативную систему. Рассмотрим спецификацию таких систем для случая описания отдельных модулей с помощью ДС-спецификации.

Спецификация модульных дискретно-событийных систем имеет вид

$$MC = \langle D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, \mathfrak{R} \rangle, \qquad (2.8)$$

где D – множество имен модулей; M_i – спецификация i-го модуля $(i \in D)$; I_i – подмножество элементов множества D, связанных по входу с i-м модулем $(I_i \subset D, i \notin I_i)$; Z_{ij} – функция взаимодействия, определяющая воздействие i-го модуля на j-й модуль $(i \in D, j \in I_j)$; $\mathfrak{R}$ – функция выбора модулей, определяющая порядок их исполнения при одновременной активизации.

Для рассматриваемого случая спецификация i-го модуля M_i имеет аналогично спецификации (2.5) следующий вид:

$$M_i = \langle X_i, S_i, \delta_i, ta_i \rangle.$$

В свою очередь функция Z_{ij} определяется отображением Z_{ij} : $S_i \to X_j$, а функция $\mathfrak{R}$ – отображением $\mathfrak{R} : D' \to D$, где D' является подмножеством множества D.

В качестве примера рассмотрим спецификацию композиционной дискретно-событийной системы, состоящей из двух модулей, изображенных на рис. 3. Она имеет следующий вид:

$$MC_{12} = \langle \{1,2\}, \{M_1, M_2\}, \{I_1, I_2\}, \{Z_{12}, Z_{21}\}, \mathfrak{R} \rangle;$$
$$M_1 = \langle X_1, S_1, \delta_1, ta_1 \rangle;$$
$$M_2 = \langle X_2, S_2, \delta_2, ta_2 \rangle;$$
$$I_1 = \{2\};$$
$$I_2 = \{1\};$$

$$Z_{12} = S_1 \to X_2 \ ;$$
$$Z_{21} = S_2 \to X_1 \ ;$$
$$æ(\{1,2\}) = 1 \ .$$

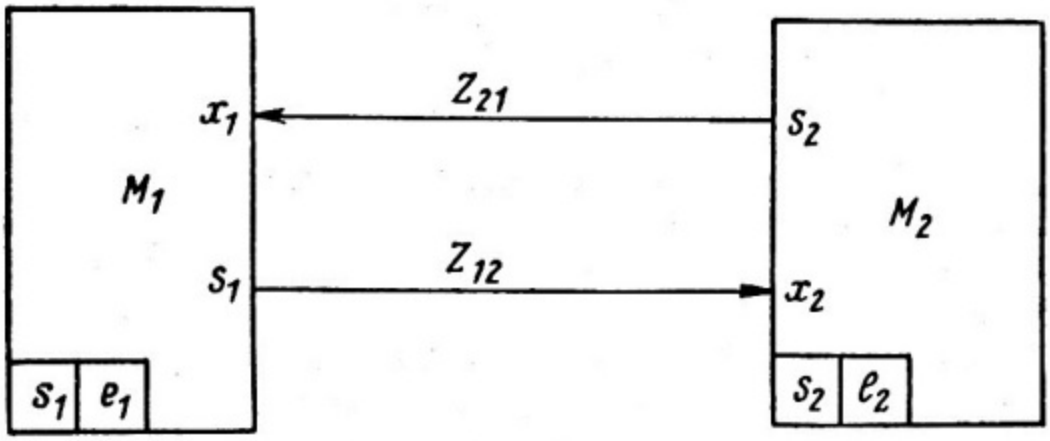

Рис. 3. Пример формализации композиционной дискретно-событийной системы

2.3.3. Преобразование структурированных систем в модульные

Сравнительный анализ различных подходов к формализации систем позволяет сделать заключение об эквивалентности выразительной мощности структурированных дискретно-событийных систем и модульных систем. Действительно, пусть $MC = \langle D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, æ \rangle$ – рассмотренная ранее модульная спецификация системы, а $CDC = \langle D', \{S'_i\}, \{I'_i\}, \{F_i\}, æ \rangle$ – соответствующая той же системе дискретно-событийная спецификация. Поскольку обе эти спецификации описывают одну и ту же систему, то для них справедливы следующие соглашения:

$D = D'$;

для каждого $i \in D$ множество последовательных состояний S'_i совпадает с соответствующим множеством S_i в спецификации M_i ;

$I'_i = I_i \cup \{j\}$, где $\{j\}$ определяет совокупность элементов, которые оказывают влияние на i-й элемент;

$æ' = æ$;

функция перехода F_i определяется исходя из соотношений:

$$F_i((s_1,\sigma_1),\ldots,(s_n,\sigma_n)) = ((s_1^*,\sigma_1^*),\ldots,(s_n^*,\sigma_n^*)),$$

где

$$(s_j^*,\sigma_j^*) = \begin{cases} (\sigma_j^{\emptyset}(s), ta_j(s^*)) & \text{– для случая } j = i; \\ (\delta_j^{bn}(s_j, ta_j(s_j) - \sigma_j, x_{ij}), ta_j(s_j^*)) & \text{– в противном случае;} \end{cases}$$

s, s^* – соответственно текущее и следующее состояние i-го элемента и $x_{ij} = z_{ij}(s)$.

Таким образом, чтобы перейти от описания структурированных дискретно-событийных систем к описанию модульных систем, необходимо в рамках структурированной спецификации выделить элементы, позволяющие определить их взаимодействие на уровне "вход-выход" и представить для каждого случая соответствующие спецификации, исходя из приведенных выше соглашений. В простейшем случае такая замена отображена на рис. 4, из которого видно, что, если в структурированной системе возникает необходимость в изменении значения v, принадлежащего одной структурной единице (1), со стороны другой структурной единицы (2), то при переходе к модульному представлению происходит формирование двух отдельных модулей путем расширения существующего описания функцией взаимодействия z и функцией перехода δ^{bn}

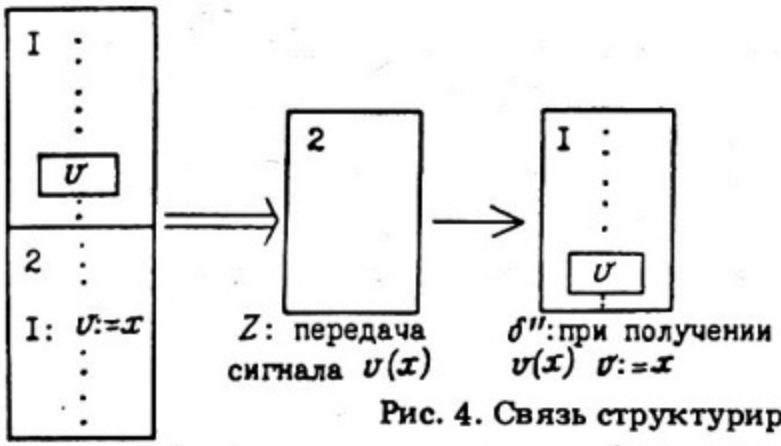

Рис. 4. Связь структурированных и модульных (агрегативных) систем

Традиционно в имитационном моделировании сложных систем более широкое распространение на практике получила спецификация типа (2.7), позволяющая относительно просто описывать системы с точностью до следующего события, на уровне планируемых активностей и взаимодействующих процессов. Однако в последнее время все чаще используются спецификации модульных (агрегативных) систем. Такие

преимущества этого подхода, как простота конструирования моделей из набора отлаженных модулей и возможность использования нисходящей стратегии, определяют его привлекательность и дальнейшее развитие.

Глава 3. ФОРМАЛИЗАЦИЯ СРЕДСТВ МОДЕЛИРОВАНИЯ В ИМИТАЦИОННЫХ МОДЕЛЯХ

Во всех представленных ранее спецификациях в явной или неявной форме присутствует оператор определения следующего события. Например, в ДС-спецификации таким оператором является оператор ta. Помимо этого в процессе имитационного моделирования возникает задача подсчета модельного времени, которая при использовании структурированных спецификаций или модульного представления систем становится достаточно сложной. Для решения этих проблем в имитационных моделях, как правило, выделяется специальный программный блок, который называется программным имитатором.

Относительная инвариантность программного имитатора по сравнению с другими компонентами имитационных моделей приводит к необходимости разработки специальных методов его реализации, которые требуют отдельного рассмотрения. Остановимся на этих вопросах более подробно в силу недостаточной их освещенности в существующих публикациях по имитационному моделированию.

В общем случае программный имитатор выполняет следующие функции:

продвижение модельного времени;

планирование событий в модели;

управление процессом реализации спланированных событий, или процессом моделирования;

обеспечение различных режимов моделирования с отображением промежуточных и конечных результатов.

Рассмотрим реализацию названных функций в зависимости от принятой на этапе формализации концепции моделирования.

3.1. Способы продвижения модельного времени

Продвижение модельного времени в имитационных моделях тесно связано с формированием моментов времени моделирования. Исходя из этого можно выделить два способа решения этой проблемы. Первый способ основан на использовании принципа "Δt", когда текущее время

моделирования t_i изменяется в соответствии с выражением (1.5). Второй способ основан на принципе особых состояний, когда модельное время определяется последовательностью возникающих в модели событий.

В случае использования принципа особых состояний оператор планирования реализуется на основе специальной информационной структуры в виде списка, которая часто называется календарем событий. Каждая запись в этом списке содержит имя (тип) события, время его возникновения и ряд дополнительных признаков. Наличие подобного календаря позволяет обеспечить в процессе моделирования правильный "ход часов модельного времени" путем упорядочивания событий по времени их появления и выбора из списка событий с наименьшим временем.

В процессе имитационного моделирования календарь постоянно обновляется: в него вносятся записи новых событий, обусловленных закономерностями работы системы, и исключаются события с минимальным временем или события, которые в соответствии с логикой работы системы необходимо отменить. Временная упорядоченность списка при этом строго сохраняется. Если в календаре возможно появление нескольких событий, имеющих одно и то же время наступления, то порядок их выбора, как правило, определяется путем назначения приоритетов, которые также должны быть указаны в записях событий.

В случае использования в модели для отображения реального времени принципа "Δt", что характерно для моделирования непрерывных систем, формирование календаря связано лишь с включением событий, через интервалы Δt. Понятно, что использование календаря событий здесь приводит только к существенному увеличению времени моделирования. Поэтому при имитационном моделировании непрерывных систем календарь событий не используется, и планирование в этом случае сводится только к пересчету модельного времени в соответствии с выражением (1.5).

Несколько видоизменяется механизм продвижения модельного времени при использовании концепции процессов. В этом случае вводятся понятия локального и глобального времени. Локальное время отображает продвижение модельного времени внутри каждого процесса, и основное назначение его – фиксация моментов наступления фазовых событий, определяющих взаимодействие процессов между собой. В свою очередь, глобальное время отображает поведение во времени всей совокупности взаимодействующих процессов и определяет текущее время функционирования всей системы в целом.

Продвижение модельного времени в случае процессного описания систем осуществляется в два этапа. Сначала изменяется локальное время активных процессов, а затем – глобальное время. Если изменение локального времени происходит лишь при наступлении внутренних событий процессов, то глобальное время переопределяется только в моменты возникновения фазовых событий. В обоих случаях механизм продвижения времени может быть основан либо на принципе "Δt", либо на принципе особых состояний.

Наиболее часто при формировании глобального времени используется принцип особых состояний. В этом случае с помощью оператора планирования осуществляется упорядочивание всех будущих фазовых событий, и без календаря событий здесь не обойтись.

Использование принципа "Δt" при формировании глобального времени возможно только при продвижении локальных времен процессов таким же образом. Однако и здесь может возникнуть потребность в календаре событий, когда некоторые подмножества Γ^i оказываются несовместными.

3.2. Управление процессом моделирования

Управление процессом моделирования в значительной степени определяется оператором планирования событий и способом продвижения модельного времени. В зависимости от этого в имитационных моделях можно выделить три типа управления: императивное, интеррогативное и автоматическое.

Императивное управление основано на назначении времени наступления следующего события. Управление в этом случае будет заключаться в следующем:

упорядочивании будущих событий по времени в календаре событий;

выборе события с минимальным временем наступления;

коррекции модельного времени;

активизации программного модуля, осуществляющего преобразования, соответствующие наступлению выбранного события.

Императивное управление является основным методом управления в имитационных моделях, основанных на использовании событийного принципа моделирования. Определение в этом случае временного оператора обеспечивает достаточно простую и эффективную реализацию этого метода управления. Основные сложности в этом случае возникают с организацией работы календаря событий, о чем будет сказано ниже.

Интеррогативное управление основано на активизации программных модулей обработки событий в случае выполнения определенных логических условий. Этот тип управления используется тогда, когда заранее невозможно предсказать момент наступления событий.

В процессе интеррогативного управления в каждый текущий момент модельного времени осуществляется:

проверка условий наступления событий;

активизация соответствующих программных модулей обработки спланированных событий ;

коррекция модельного времени.

Интеррогативное управление является наиболее общим типом управления, но в чистом виде в имитационном моделировании используется редко. Связано это с очень неэкономичным потреблением машинного времени. Как только вычисление, управляемое операторами из описания некоторого элемента модели, столкнется с необходимостью проверки выполнения условия, то это условие должно проверяться постоянно, пока не будет выполнено.

Автоматическое управление основано на использовании принципа "Δt" при продвижении модельного времени. В этом случае элементы системы синхронизированы, так как вся система определенный период ждет, а потом все ее элементы проходят свою активную фазу одновременно. Модельное время при этом изменяется в соответствии с выражением (1.5).

Реализация автоматического управления в системах имитационного моделирования в силу своей простоты не предусматривает использование операторов планирования. Такой тип управления характерен для систем, описываемых ДУ-спецификациями (2.4) и ДВ-спецификациями (2.6).

3.3. Спецификации программных имитаторов дискретно-событийных систем

Программные имитаторы, обеспечивающие моделирование дискретно-событийных систем, основаны на использовании императивного управления. Анализ функций, выполняемых имитатором в процессе моделирования, позволяет описать этот элемент имитационной модели ДС-спецификацией. Выходными сигналами имитатора в таком случае будут являться так называемые календарные сигналы, представляющие собой внутрисистемные сообщения о будущих событиях и времени их активизации. В качестве переменных состояния будут выступать два вектора s_1 и s_2 . Вектор s_1 содержит переменную модельного вре-

мени и список переменной длины для фиксации будущих событий. Размерность вектора определяется таким образом, чтобы каждому элементу модели, имитирующему протяженное во времени действие, строго соответствовало определенное место в списке событий. Вектор s_2 имеет постоянную размерность и представляет собой список, часто называемый цепью текущих событий. В этом списке содержатся события, совершающиеся в данный момент времени.

Алгоритм функционирования программного имитатора заключается в обработке входных сигналов таким образом, что описания будущих событий получают отметки в календаре, а описания событий текущего времени размещаются согласно их приоритетам в цепи выходных сигналов. Особо важным вопросом при этом является квазипараллельная обработка событий.

При функционировании программного имитатора соблюдаются следующие специальные условия:

начало имитации связано с поступлением инициирующего сигнала на имитатор; только после активации модели в календарь могут заноситься первые сигналы событий, вырабатываемые элементами системы;

сигнал о наиболее близком по времени реализации событии передается от имитатора к соответствующим элементам системы; одновременно модельное время устанавливается на момент реализации этого события, после чего запись об этом событии в календаре стирается;

активация элемента системы при подаче на его вход сигнала о наступлении события вызывает изменение состояния системы, в результате чего на входе имитатора появляются сигналы о планируемых событиях. Когда все сигналы будут переданы, имитатор активируется.

сигналы, поступившие в имитатор, в определенной последовательности включаются в календарь или же в цепь текущих событий;

первый сигнал в цепи текущих событий считывается и передается в соответствующий элемент, вызывая действия, описанные ранее.

Данный алгоритм функционирования программного имитатора справедлив как для событийного принципа моделирования, так и для процессного. В первом случае, когда система формализуется с помощью спецификации (2.5), имитатор также описывается этой спецификацией. Во втором случае, когда для формализации системы используются спецификации (2.7) или (2.8), появляется возможность структуризации самого имитатора, что также можно отобразить спецификацией (2.7).

Если в процессе формализации системы используется концепция событий, то действия имитатора удобно отобразить в виде схемы, представленной на рис. 5. Из этой схемы видно, что функционирование имитатора определяется переменной состояния s, которая отображает последовательные состояния системы, и переменной t_L, которая определяет момент времени наступления последнего события.

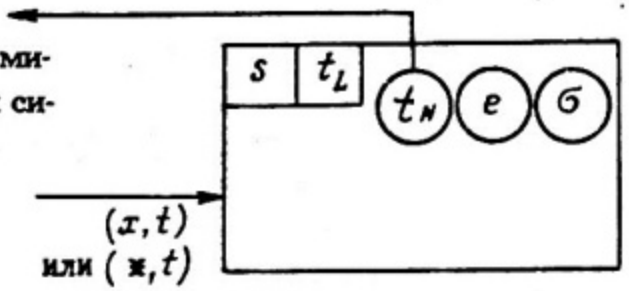

Рис. 5. Схема программного имитатора дискретно-событийных систем

В процессе работы имитатора определяются:

момент времени следующего планируемого события t_N;

время, которое истекло с момента наступления последнего события e;

время, оставшееся до наступления следующего события σ.

Для вычисления величин t_N, e, σ используются следующие выражения:

$$t_N = t_L + ta(s);$$

$$e = t - t_N;$$

$$\sigma = t_N - t = ta(s) - e,$$

где t – глобальное модельное время.

Имитатор взаимодействует с системой посредством входных сигналов – (x,t) или $(*,t)$ и выходного сигнала t_N. Если на вход имитатора поступает сигнал (x,t), где t – глобальное время, определяющее наступление события, связанного с появлением на входе сигнала с именем x, то имитатор отслеживает это событие. В случае отсутствия сигнала на входе, что равносильно поступлению на вход имитатора сигнала $(*,t)$, глобальное время становится равным t_N, передаваемому системе с выхода имитатора.

Если на вход системы поступает входной сигнал x, то значение величины t должно лежать в интервале $t_L \leqslant t \leqslant t_N$, что эквивалентно соотношению $0 \leqslant e \leqslant ta(s)$.

Использование концепции процессов при формализации сложных систем приводит к возможности структуризации самого имитатора. В этом случае в состав имитатора может быть включен координирующий блок, основные функции которого заключаются в синхронизации функ-

шионирования имитаторов отдельных подсистем и управлении внешними событиями.

Схема взаимодействия координирующего имитатора с имитаторами отдельных подсистем представлена рис. 6, на котором в качестве примера изображено взаимодействие имитаторов двух подсистем M_1 и M_2.

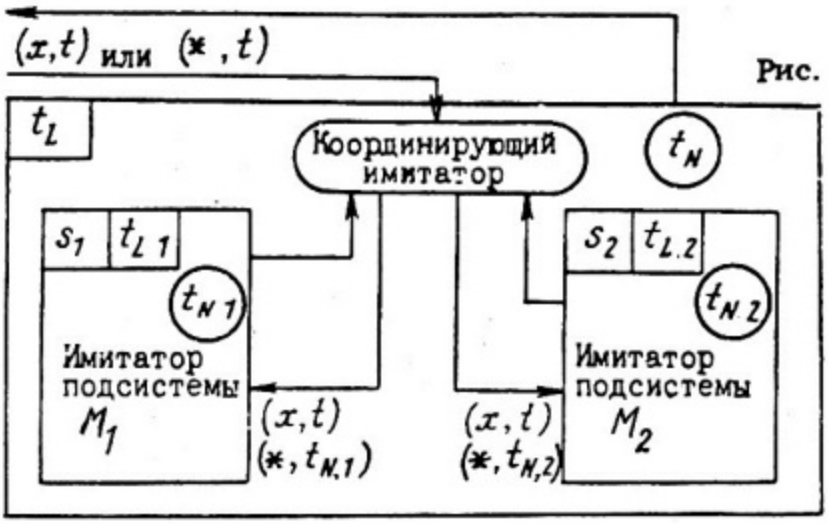

Рис. 6. Схема программного имитатора в случае процессного описания дискретно-событийных систем

Координирующий имитатор имеет такую же структуру, как и имитаторы элементов системы. Его состояниями для приведенного примера являются события-взаимодействия элементов (s_1 , t_{L1}, s_2, t_{L2}) и время наступления последнего события в системе t_L . Все события-взаимодействия записываются в календарь событий координирующего имитатора. На вход координирующего имитатора поступают сигналы (x, t) и $(*, t)$, а выходом является текущее глобальное время модели t_N .

Представленная схема координирующего имитатора используется в случае двухуровневой декомпозиции системы. В свою очередь, каждый из выделенных в системе элементов может быть также представлен в виде взаимодействующих подпроцессов. Дальнейшая структуризация имитатора в этом случае приводит к построению иерархической структуры, изображенной на рис. 7.

3.4. Методы управления календарем событий

При моделировании сложных систем важным вопросом организации функционирования имитационной модели с императивным управлением является обеспечение эффективности работы с календарем событий. В процессе моделирования приходится часто просматривать список событий для нахождения очередного события или вставки новых планируемых событий. Поскольку длина этого списка может быть достаточно большой, то от организации работы с ним в значительной степени зависит время

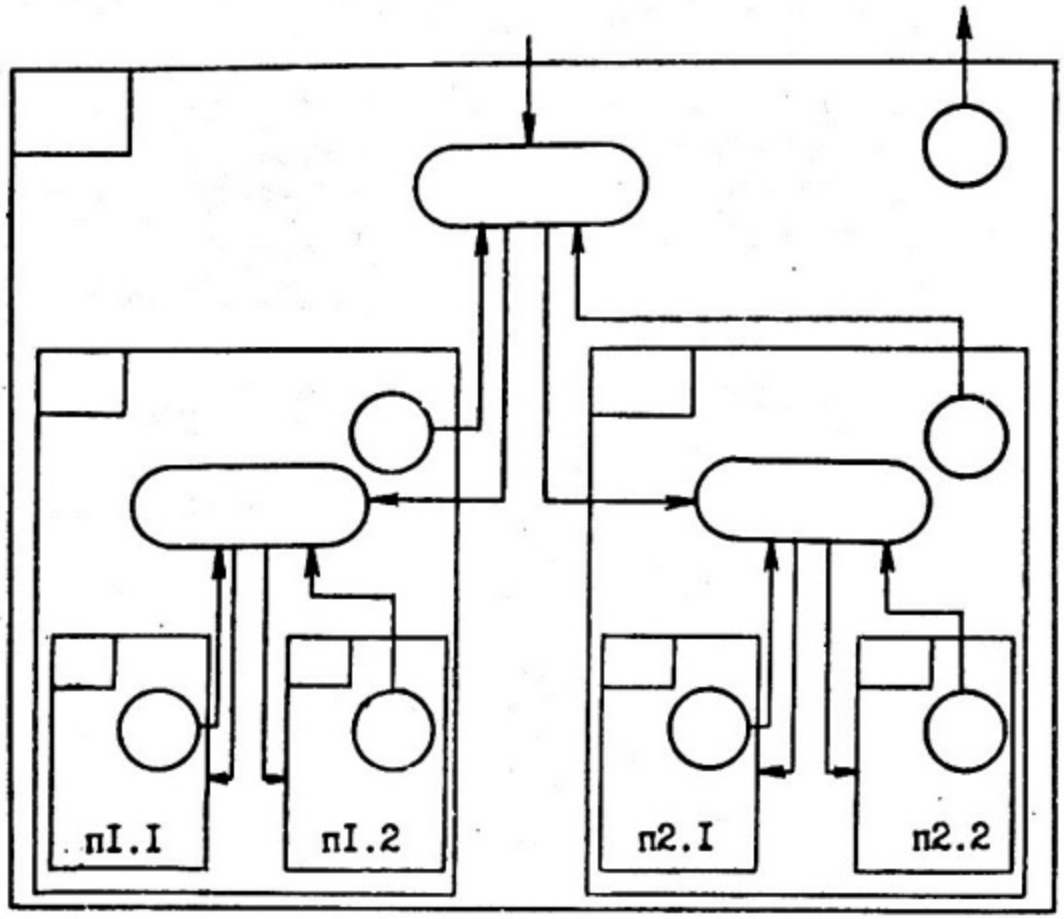

Рис. 7. Иерархическая структура программного имитатора структурированных дискретно-событийных систем

моделирования. По оценкам специалистов это время может достигать 60% от общего времени моделирования.

В общем случае процедура работы с календарем событий включает поиск в списке события с минимальным временем наступления, удаление этого события из списка и включение в список новых событий. Как правило, при включении новых событий происходит упорядочивание списка событий по времени. В результате этого, если список упорядочен в порядке возрастания моментов времени, то событие, время наступления которого минимально, будет всегда находиться в начале списка, и поиск его сведется к выбору первой записи.

Наиболее распространенной структурой списка событий является связанный линейный список (рис. 8,а). Наряду с этим в системах GASP, SIMSCRIPT и GPSS используется двусвязный линейный список (рис. 8,б). Все записи в списке событий упорядочены в порядке возрастания времен наступления соответствующих событий. Процедура включения нового события заключается в последовательном просмотре имеющихся записей с целью определения места включения и запись в список этого события.

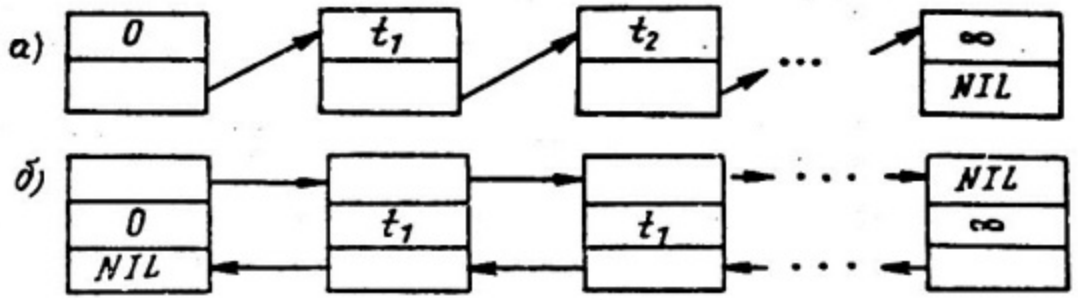

Рис.8. Линейные структуры данных для списка событий

Определение порядка просмотра записей играет важную роль в реализации соответствующих алгоритмов обработки данных в списке будущих событий. Анализ показывает, что эффективность данных алгоритмов определяется исходя из следующих положений.

В общем случае записи в списке будущих событий содержат величины, характеризующие интервалы времени наступления соответствующих событий σ_i $(i=\overline{1,N})$ относительно текущего времени моделирования. Поскольку эти величины являются случайными и упорядочены в порядке возрастания, то расположение их в списке событий будет определяться некоторым распределением, которое мы обозначим G. Тогда величина $G(x)$ будет определять часть записей, для которых $\sigma_i \leqslant x$.

Распределение событий G тесно связано с распределением, которое определяет значения σ_k для записей, сформированных для включения в список в текущий момент времени моделирования. Это распределение, которое мы обозначим F, будет определять, как долго включаемая запись будет находиться в списке будущих событий.

Между названными распределениями F и G существует следующая зависимость:

$$G(x)=\begin{cases}\frac{1}{\mu}\int_0^x [1-F(t)]\,dt, & \text{если } x \geqslant 0;\\ 0 - & \text{в противном случае,}\end{cases}$$

где μ – средняя величина интервалов σ_i $(i=\overline{1,N})$ в списке событий.

В частности, если F является равномерным распределением на интервале (0,1), то

$$G(x)=\begin{cases}0, & \text{если } x \leqslant 0;\\ 2x-x^2, & \text{если } 0 \leqslant x \leqslant 1;\\ 1, & \text{если } x \geqslant 1.\end{cases}$$

Зная вид распределения G, можно оценить объем вычислений, требуемый для включения новой записи в список событий. Например, для вышеприведенного случая, поскольку $G(0,5) = 0,75$, любая новая запись с большей вероятностью будет включаться в конец списка, при этом в целом объем вычислений, требуемый при просмотре списка с начала, будет больше объема вычислений при просмотре списка с конца.

В общем случае, обозначая выраженные в процентах эти величины соответственно % Н и % К, можно получить следующие выражения для их определения:

$$\% H = \int_0^{\infty} G(x)\,dF(x) \cdot 100\%;$$

$$\% K = \int_0^{\infty} [1 - G(x)]\,dF(x) \cdot 100\%.$$

Как показали исследования, для большинства широко используемых на практике распределений F (равномерного, нормального, распределения Эрланга) выполняется соотношение % Н $\geqslant$ % К. Обратная ситуация имеет место, когда F является гиперэкспоненциальным, смешанным экспоненциальным и гамма-распределениями. Если распределение F определяется совокупностью нескольких распределений, то в большинстве случаев выполняется соотношение % Н $\leqslant$ % К.

Линейный связанный список является простейшей структурой данных и поэтому широко используется на практике. Однако при увеличении числа записей, например, при $N > 100$, возникает проблема повышения эффективности процедуры включения. С этой целью предложен ряд других структур данных списка событий и соответствующих им алгоритмов включения новых записей. В частности, заслуживает внимания структура данных в виде индексированного списка (рис. 9). В этой структуре вводится дополнительный список указателей, который логически делит связанный линейный список на ряд частей, для того чтобы уменьшить время включения новых записей.

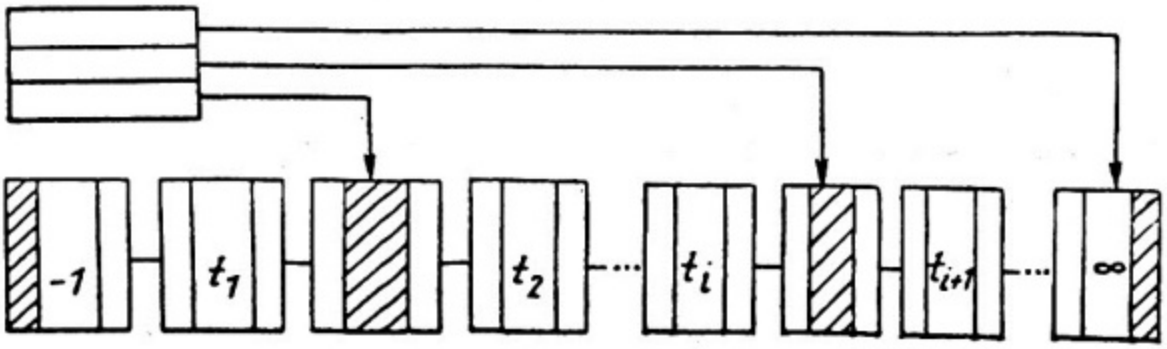

Рис. 9. Структура данных календаря событий в виде индексированного списка

Существует определенное количество модификаций алгоритма включения новой записи с использованием названной структуры данных списка событий. В общем случае параметрами этих алгоритмов являются размер списка указателей и интервал упорядочения. Оптимальный выбор названных параметров осуществляется исходя из размеров всего списка будущих событий и распределений F и G . В частности, интервал упорядочения может выбираться на основе двух условий: либо постоянства временного интервала для каждого подсписка, либо постоянства количества записей в каждом подсписке.

В последнее время большой интерес проявляется к организации списка будущих событий в виде древовидных структур. Наибольшее распространение получили бинарные и П-деревья. Очевидное преимущество этих алгоритмов заключается в том, что даже в наихудшем случае время включения новой записи составляет $o(\log_2 N)$. Показано также, что алгоритмы включения новой записи, построенные на основе использования этих структур, относительно нечувствительных к распределению F .

По понятным причинам достаточно сложно провести полный анализ эффективности используемых в системах имитационного моделирования алгоритмов работы с календарями событий различной структуры. На рис. 10 представлены эмпирические усредненные характеристики времени выполнения операции включения новой записи в список событий τ для трех алгоритмов:

ЛС-алгоритма, основанного на линейном связном списке;

СС-алгоритма, основанного на связном индексированном списке;

Д-алгоритма, основанного на двоичном дереве.

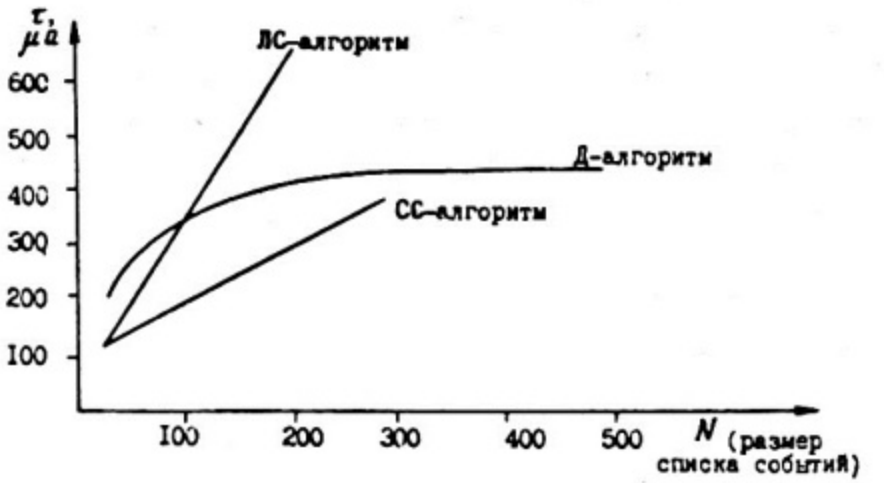

Рис. 10. Сравнительные оценки алгоритмов включения новых записей в список событий

Глава 4. ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ ИМИТАЦИОННЫХ МОДЕЛЕЙ

Программная реализация полученных при формализации алгоритмических спецификаций является заключительным этапом в построении имитационных моделей. Решение этой задачи может быть основано на использовании либо универсальных языков и систем программирования, либо специализированных языков и систем моделирования.

Общепризнанными являются следующие преимущества языков и систем моделирования по сравнению с универсальными [3]:

язык моделирования содержит абстрактные конструкции, непосредственно отражающие алгоритмические спецификации, выбранные на этапе формализации моделируемой системы, что существенно упрощает программирование и позволяет автоматизировать выявление многих ошибок в программах;

системы моделирования имеют в своем составе характерный для них механизм продвижения модельного времени и средства разрешения временных конфликтов;

системы моделирования, как правило, содержат встроенные средства генерации входных процессов и переменных (датчики случайных чисел, генераторы типовых воздействий и т.п.), а также средства автоматизированного сбора, обработки и выдачи результатов моделирования;

языки моделирования имеют средства, упрощающие программирование имитационных экспериментов, в частности, автоматизирующие инициацию (задание начального состояния) и перезапуск модели.

Вместе с тем, можно отметить и присущие языкам и системам моделирования следующие недостатки:

недостаточную распространенность языков моделирования, которые, как правило, не входят в штатное программное обеспечение;

необходимость дополнительного обучения языкам моделирования и, как следствие, недостаток программистов, хорошо владеющих ими;

отсутствие гибкости и широких возможностей, присущих универсальным системам программирования;

ограниченные возможности существующих систем моделирования при создании имитационных моделей сложных систем, вызванных их консервативностью, т.е. отсутствием средств повышения эффективности моделирования, наличием стандартных форм вывода результатов и т.п.

Понятно, что любую имитационную модель можно запрограммировать с использованием универсальных алгоритмических языков. Однако

стремление автоматизировать этот процесс привело к появлению большого числа языков и систем моделирования [1].

В общем случае система моделирования включает:

язык моделирования;

язык управления системой моделирования, реализованный на основе языка управления заданиями пакетных операционных систем или на основе языка команд в интерактивных операционных системах;

программные средства, обеспечивающие трансляцию модели и другие стандартные функции системы моделирования (продвижение модельного времени, генерацию случайных чисел, сбор статистической, информации, вывод результатов и т.д.).

Среди большого числа языков моделирования довольно сложно выделить какое-то базовое подмножество языков, покрывающих основные потребности пользователей. В результате многие разработчики имитационных моделей вынуждены создавать свои оригинальные языки и системы моделирования, ориентированные на решение заданного класса задач. Для этих целей широко используются универсальные языки программирования: ФОРТРАН, ПЛ/1, СИМУЛА-67, ПАСКАЛЬ, СИ. В последнее время находят распространение алгоритмические языки, имеющие развитые средства проблемной ориентации: АДА, МОДУЛА-2 и др.

Количественный рост и разнообразие языков моделирования в значительной степени связаны с большим числом существующих схем формализации и алгоритмизации моделируемых систем. Любая из этих схем может служить основой для разработки нового языка и системы моделирования. Наряду с этим, следует отметить, что со стороны разработчиков многих полезных систем моделирования отсутствует необходимая пользователям поддержка соответствующего программного обеспечения и качественная документация.

Проблема выбора языка и системы программирования при построении имитационной модели является одной из наиболее сложных. Для решения этой задачи познакомимся с основными характеристиками наиболее известных языков моделирования.

4.1. Классификация языков и систем имитационного моделирования

В основу классификации языков и систем моделирования положим следующие основные факторы [3]:

класс моделируемых систем;

базовую для языка моделирования схему формализации;

альтернативные схемы формализации;

синтаксическую основу языка моделирования;

наличие в системе моделирования характерных средств.

В силу различия в организации процессов имитации для различных классов моделируемых систем все языки имитационного моделирования делятся на три самостоятельные группы:

языки непрерывного моделирования;

языки дискретного моделирования;

языки дискретно-непрерывного моделирования.

В [1] приведены сведения о 114 языках непрерывного моделирования, 117 языках дискретного моделирования и 34 языках дискретно-непрерывного (комбинированного) моделирования. Области применения двух первых групп языков практически не пересекаются. Языки комбинированного моделирования используются как для непрерывного, так и для дискретного моделирования.

Под базовой схемой формализации понимается совокупность понятий, которые используются для описания моделируемой системы и непосредственно представлены в языке моделирования. Например, в языках непрерывного моделирования широко применяются системы дифференциальных и интегродифференциальных уравнений, структурные схемы, графовые модели. Среди языков дискретного моделирования различают: языки событий, языки активностей, языки процессов, языки транзактов, основанные на автоматных и сетевых представлениях языки и т.п. Языки комбинированного моделирования могут основываться на агрегатах, КОМБИ-сетях, сочетаниях схем алгоритмизации непрерывного и дискретного моделирования.

Базовая схема формализации предполагает определенную организацию работы программного имитатора. Однако, кроме базовой схемы формализации , могут быть применены альтернативные схемы. Например, язык структурных схем можно использовать для моделирования систем, описываемых дифференциальными уравнениями, дискретные сети можно моделировать на языке транзактов или процессов и т.п.

По синтаксической основе различают языки моделирования, вложенные в базовый язык, расширяющие базовый язык и с собственным синтаксисом. В простейшем случае вложенный язык состоит из обращений к подпрограммам, написанным на базовом языке. В качестве базового языка в свыше 20% языков моделирования используются языки программирования ФОРТРАН, СИМУЛА-67, ПЛ/1. Например, посредством определения класса *simulation* язык процессов вложен в универсальный язык программирования СИМУЛА-67. Базовым языком в *GASP - IV* является язык программирования ФОРТРАН-IV. Следует

отметить мобильность такого рода языков, что обеспечивает легкую переносимость моделей с одной ЭВМ на другую.

Языки, расширяющие базовый язык, реализуются на основе базовой системы программирования, дополненной препроцессором и библиотекой стандартных программ. Препроцессор выполняет трансляцию с языка моделирования в базовый язык, а далее уже используются стандартные средства базовой системы программирования.

Реализация языка с собственным синтаксисом связана с разработкой соответствующего компилятора. Такой тип языков требует значительно больших затрат на создание средств моделирования, хотя не исключена возможность написания компилятора на одном из распространенных языков программирования высокого уровня.

Наличие средств проблемной ориентации в языке моделирования позволяет разрабатывать языки конечных пользователей, вводить макропонятия для упрощения программирования имитационных моделей. В процедурно-ориентированных языках могут использоваться макросы, подпрограммы или процедуры. Широкое распространение в настоящее время находят диалоговые средства для реализации языков такого типа.

В табл. 2 представлены основные свойства наиболее часто используемых в СССР языков моделирования. В столбце "Тип управления временем" содержится информация о типе управления временем: А – автоматическое управление, ИМ – императивное управление, ИН – интерроративное управление.

Приведенные в этой таблице данные позволяют разработчику имитационных моделей провести сравнение существующих систем моделирования. Однако этой информации не достаточно, чтобы сказать, какая система лучше. Ответить на этот вопрос можно только для систем, реализующих один и тот же способ формализации. В противном случае достоинства способа формализации можно отнести ошибочно к достоинствам языка моделирования. Кроме того, каждая из приведенных систем моделирования имеет свою область применения, в пределах которой она обладает преимуществами инструментального либо технологического характера перед другими системами.

4.2. Выбор языка имитационного моделирования

Существующее многообразие языков моделирования приводит разработчиков имитационных моделей, особенно начинающих, к достаточно сложной проблеме выбора: какой язык является наиболее подходя-

Таблица 2

Название	Базовый язык программирования	Тип моделей	Способ формализации	Тип управления временем	Реализация на ЭВМ типа
Динамо [4]	ФОРТРАН	Непрерывные	Системы дифференциальных уравнений	А	БЭСМ-6
МикроДинамо	ПАСКАЛЬ	—"—	—"—	—"—	IBM PC
GPSS [5]	—	Дискретные	Транзанты	ИМ, ИН	ЕС ЭВМ
СМДП-В [6]	—	—"—	—"—	—"—	—"—
GPSS-PC	—	—"—	—"—	—"—	IBM PC
МОДЕЛЬ-6 [1]	—	Дискретные	Транзанты	ИМ, ИН	БЭСМ-6
СИМУЛА-67 [7]	АЛГОЛ	Дискретные	Процессы	ИМ	ЕС ЭВМ
СИМСКРИПТ [1]	ФОРТРАН	Дискретные	События	ИМ	ЕС ЭВМ
СИМСКРИПТ-11 [1]	—	—"—	—"—	—"—	ЕС ЭВМ
GASP-IV [8]	ФОРТРАН	Дискретные	События	А, ИМ	ЕС ЭВМ
DUCM [1]	ПЛ/1	Дискретные	События	ИМ	ЕС ЭВМ
СИМФОР [9]	ФОРТРАН	Дискретные	События	ИМ	СМ ЭВМ
ПЛИС [1]	—	Дискретные	События	ИМ	ЕС ЭВМ
СЛЭНГ [1]	АЛГОЛ	Дискретные	Процессы	ИМ	БЭСМ-6
НЕДИС [8]	—	Дискретно-непрерывные	События, Процессы, Системы дифференциальных уравнений	А, ИМ, ИН	ЕС ЭВМ, БЭСМ-6

щим для решения поставленной задачи. Как показал опыт, определяющими факторами здесь являются:

достаточность и полнота описательных возможностей языка моделирования и легкость в его изучении;

наличие удобных инструментальных средств, упрощающих программирование имитационных моделей;

возможность функционирования соответствующей системы моделирования на имеющихся в распоряжении разработчиков вычислительных средствах и обеспеченность всей необходимой документацией.

Описательные средства языков моделирования позволяют в рамках выбранной концепции формализации идентифицировать и задавать параметры моделируемой системы и внешних воздействий, алгоритмы функционирования и управления, алгоритмы отображения реального времени, режимы и требуемые результаты моделирования. По структуре и правилам программирования языки моделирования подобны процедурно-ориентированным алгоритмическим языкам высокого уровня. Они имеют тот или иной набор операторов, сопровождаемых соответствующими операндами. Но операторы языков моделирования предопределяют выполнение более сложных процедур. Кроме того, их следует рассматривать как формализованный базис создания алгоритмических моделей моделируемых объектов.

При выборе языка моделирования важно определить степень его универсальности. Как правило, чем больше универсальность, тем проще схема формализации, используемая для разработки модели. Например, подавляющая часть универсальных дискретных и дискретно-непрерывных языков моделирования предопределяет необходимость построения модели на базе простейших объектов систем массового обслуживания, таких как обслуживающий прибор, очередь, поток заявок и т.п.

Стремление разработчиков систем моделирования предоставить возможности учитывать в модели специфические особенности моделируемых объектов приводит к усложнению применения универсальных языков моделирования. С одной стороны, это достигается путем введения определенной избыточности в язык моделирования, с другой стороны, в распоряжение пользователей предоставляются средства самостоятельного расширения путем программирования недостающих ему процедур на определенном алгоритмическом языке и последующего включения их в систему.

В этой связи понятно желание разработчиков имитационных моделей ориентироваться на языки моделирования систем определенной предметной области. Хороший предметно-ориентированный язык моделирова-

ния представляет пользователю достаточные средства реализации выбранной им концепции формализации, соответствующей предметному мышлению в той области, для которой предполагается его использование. Требуемые трудозатраты здесь также уменьшаются, однако уменьшается и диапазон моделируемых систем.

С точки зрения приведенных выше факторов, определяющих выбор универсального языка моделирования, наиболее приемлемыми являются языки GPSS и СИМУЛА-67. По данным Ассоциации по вычислительным машинам ACM эти языки входят в число тринадцати наиболее важных языков программирования. Этот факт говорит о том, что семантические возможности языков GPSS и СИМУЛА-67 обеспечивают пользователя подходящими средствами формализованного описания многих сложных систем.

В настоящее время широко используются реализации языков GPSS и СИМУЛА-67 на ЭВМ единой серии. Имеются реализации этих языков на персональных ЭВМ, в частности, IBM PC и совместимых с ними.

Желание расширить возможности языка GPSS привело к созданию нескольких отечественных систем имитационного моделирования на базе ЕС ЭВМ. Например, система ПЛИС разработана на базе языков GPSS и ПЛ/1. Входной язык этой системы обладает концептуальной преемственностью по отношению к языку GPSS. Однако его возможности существенно шире, поскольку допускают использование всех средств универсального алгоритмического языка ПЛ/1 в ОС ЕС.

Другим примером является пакет прикладных программ СМДП-В, в котором обеспечена возможность обращения из программы, написанной на языке GPSS, к подпрограммам, написанным на языке ассемблера, или к банку данных на логическом уровне. Кроме того, в пакете СМДП-В имеются интерактивные средства работы с моделью.

Среди универсальных языков моделирования находят применение языки СЛЭНГ, СИМСКРИПТ, НЕДИС, GASP-IV, СИМФОР, однако недостаточная обеспеченность качественной документацией и удобными средствами отладки программ сдерживает их распространение среди разработчиков имитационных моделей.

Следует также отметить, что попытки создать в противовес универсальным языкам моделирования широко распространенные предметно-ориентированные системы пока не увенчались успехом. Основная причина этого – ориентация на моделирование ограниченного класса систем. Как правило, такого рода системы разрабатываются для решения

определенного, достаточно узкого, класса задач моделирования, и, более того, со стороны разработчиков в большинстве случаев отсутствует необходимая пользователям поддержка соответствующего программного обеспечения, что не позволяет ее полностью отторгнуть от разработчиков.

Несмотря на сказанное имитационное моделирование развивается по пути создания проблемно-ориентированных имитационных систем. Построение модели в рамках этих систем фактически сводится к конструированию модели из множества моделирующих элементов, функций и условий их использования, а также способов параметризации включаемых в модель элементов. Более того, имея данные о каждом входящем в модель элементе и описание связей между ними, этот процесс можно автоматизировать, автоматически генерируя программу имитации.

Развитие проблемно-ориентированных систем моделирования позволяет говорить о формировании целостной технологии имитационного моделирования, с помощью которой организуются действия исследователя на всех этапах моделирования, начиная от изучения предметной области и выделения моделируемой проблемной ситуации и кончая построением и реализацией планов машинных экспериментов для анализа поведения моделируемой системы.

Определенные успехи в этом направлении уже имеются [3], однако говорить о решении всех существующих здесь проблем еще рано. Предстоит еще решить многие задачи, связанные с созданием и использованием эффективных стандартных приемов и соответствующих программных и инструментальных средств, с помощью которых теоретические результаты могут быть освоены в возможно более широких областях применений.

4.3. Особенности программирования времени в системах имитационного моделирования

Программирование времени является наиболее сложной проблемой при написании имитационной модели на одном из языков моделирования. Если реализация операторов перехода и операторов выхода осуществляется средствами, характерными для любых систем программирования, то планирование следующих событий в рамках каждой системы моделирования осуществляется по-своему. С одной стороны, это зависит от принятого в системе механизма управления моделированием, а с другой — от имеющихся в языке операторов, определяющих выразительную мощность данной системы в рамках принятой в ней концепции формализации.

Как уже отмечалось ранее, при отображении реального времени в модели должны быть реализованы операторы вычисления моментов времени наступления будущих событий и операторы планирования следую-

щего события. В системах моделирования с императивным и интерроративным управлением это делается по-разному. Рассмотрим решение этих вопросов на примере наиболее распространенных языков СИМУЛА-67 и *GPSS*, позволяющих охватить оба типа управления моделированием.

Язык СИМУЛА-67 ориентирован на использование императивного управления. В рамках системного класса *Simulation* имеются средства для фиксации моделируемого времени, работы с календарем событий, активизации и пассивизации объектов.

При любом количестве параллельно функционирующих в системе объектов в каждый момент машинного времени выполняются операторы программы, моделирующие функции одного объекта. Такой объект называется текущим, а выполнение текущим объектом участка программы – активной фазой.

На один и тот же момент модельного времени могут быть запланированы события нескольких объектов. Эти объекты, в отличие от текущего, называются приостановленными. Наряду с этим объект может быть и пассивным, когда точно не известен момент наступления его следующей фазы. Если объект исключается из моделирования, то он считается завершенным.

Во время активной фазы объект может выполнять либо операторы, не требующие изменения модельного времени (арифметические, присваивания, присоединения, проверки условий, переходов к другим операторам и т.д.), либо операторы планирования событий на текущий или какой-нибудь будущий момент модельного времени. Кроме того, здесь могут использоваться операторы перехода объекта в пассивное или завершенное состояние.

К операторам планирования событий в языке СИМУЛА-67 относятся следующие операторы:

hold (t) – переводит объект в приостановленное состояние, и в календаре на момент времени $t_{mod} + t$ планируется следующая фаза;

activate x, *activate x at t*, *activate x delay t*, *activate x after y*, *activate x before y* – переводят пассивный объект со ссылкой x в активное или приостановленное состояние в зависимости от содержащейся в этом операторе информации (y – ссылка на другой объект или его фазу);

reactivate x – переводит объект со ссылкой x любого типа, кроме завершенного, в активное состояние и имеет такие же разновидности как и оператор *activate*.

Включение нового объекта в процесс моделирования осуществляется с помощью генератора объектов классов, являющихся подклас-

сами класса *process*. Созданный в этом случае объект является пассивным. Он может быть активизирован сразу с помощью оператора *activate* или через некоторое время, путем использования объектной переменной, которая помнит ссылку на нее.

Если нужно приостановить функционирование объекта на заранее неизвестное время, можно запомнить ссылку на этот объект в объектной переменной или занести его в соответствующий список, а затем перевести объект в пассивное состояние одним из следующих операторов:

passivate – заканчивает активную фазу текущего объекта и переводит объект в пассивное состояние, после чего управление передается очередному объекту из календаря;

wait(s) – заносит в конец специального списка пассивных объектов *s* ссылку на текущий объект и выполняет для него оператор *passivate*;

cancel(x) – исключает из календаря событий объект со ссылкой *x*, находящийся в приостановленном состоянии, или выполняет те же действия, что и оператор *passivate*, если объект со ссылкой *x* является текущим; если объект со ссылкой *x* пассивен или завершен, то никаких действий не производится;

terminate(x) – переводит объект со ссылкой *x* из любого состояния в завершенное; если *x* – текущий объект, то управление передается следующему объекту из календаря событий.

На рис. 11 представлена схема изменения состояний объектов при использовании описанных ранее операторов. Номера, указанные на рисунке, соответствуют следующим операторам: 1 – *cancel*; 2 – *passivate*; 3 – *activate* и *reactivate*; 4 – *hold*; 5 – *reactivate*; 6 – выход из тела класса (через *end*); 7 – *terminate*; 8 – завершение активной фазы текущего объекта.

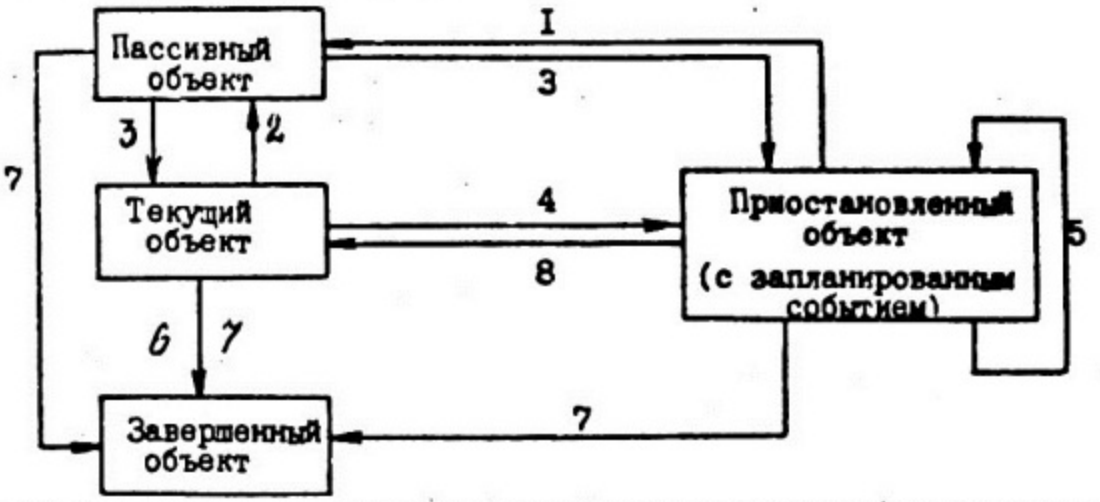

Рис. 11. Схема изменения состояний объектов при использовании соответствующих операторов языка СИМУЛА-67

Для того чтобы узнать, в каком состоянии находится объект в текущее время моделирования, в языке СИМУЛА-67 предусмотрены две булевские процедуры *idle* и *terminated*, которые принимают значение *true*, если объект находится в пассивном или завершенном состоянии соответственно. Кроме того, для определения момента времени активизации, запланированного в календаре события, можно использовать процедуру *evtime*, а ссылку на следующий объект в календаре событий можно получить с использованием ссылочной процедуры *nextev*.

Язык моделирования *GPSS* сочетает в себе императивное и интеррогативное управление. Как правило, системы моделирования, имеющие такую комбинированную систему управления или планирования, содержат помимо календаря будущих событий еще одну или несколько дополнительных информационных структур, организованных в виде списка. В системе моделирования *GPSS* такими списковыми информационными структурами являются пять видов цепей: текущих событий, будущих событий, пользователя, прерывания и парности.

В основе формализации объектов моделирования средствами языка *GPSS* лежит понятие транзакта. Транзакты могут создаваться, обрабатываться и уничтожаться в соответствии с логикой моделирования. Действия над транзактами описываются с помощью активностей, которые отображают функционирование аппаратно-ориентированных, статистических, вычислительных и операционных динамических объектов, определенных в языке.

Описание активностей средствами языка *GPSS* основано на выделении нескольких действий, определяющих продвижение транзакта. Первое из них означает попытку контакта транзакта с активностью, второе — продолжение контакта, третье — окончание взаимодействия транзакта с активностью. Сказанное отображается в языке *GPSS* цепочками операторов *SEIZE - ADVANCE - RELEASE* и *ENTER - ADVANCE - LEAVE*.

Функции планирования активностей или событий, определяющих эти активности, и продвижение модельного времени реализуются интерпретатором *GPSS*. Основную роль при этом играют цепи текущих и будущих событий. Функционирование имитационной модели в этом случае осуществляется следующим образом.

В каждый текущий момент времени моделирования интерпретатор осуществляет просмотр цепи текущих событий, куда входят транзакты, запланированные на данный момент времени или заблокированные ввиду текущих условий в модели. При анализе цепи текущих событий интерпретатор просматривает каждый транзакт и продвигает его в модели по некоторой траектории до тех пор, пока не встретится блок *ADVANCE*

или TERMINATE , либо не возникнет условие блокировки, означающее, что транзакт не может войти в следующий блок. Когда транзакт прекращает движение, интерпретатор выполняет одно из двух действий:

продолжает просмотр цепи текущих событий;

без продвижения модельного времени начинает просмотр цепи текущих событий от начала.

При отсутствии в цепи текущих событий транзактов, которые могут быть продвинуты в модели на текущий момент времени моделирования, интерпретатор проверяет наличие транзактов в цепи будущих событий, продвигает модельное время к значению, запланированному для движения следующего транзакта, и этот транзакт переносится из цепи будущих в цепь текущих событий. Любые другие транзакты, движение которых можно возобновить в это новое значение модельного времени, также переносятся из цепи будущих в цепь текущих событий.

После того как перенос транзактов в цепь текущих событий завершен, интерпретатор начинает новый просмотр цепи текущих событий. Весь этот цикл повторяется до тех пор, пока не будут исчерпаны цепи текущих и будущих событий.

В цепь будущих событий транзакты помещаются с помощью операторов императивного управления GENERATE и ADVANCE . Первый оператор определяет момент времени генерации транзакта, второй – задержку в продвижении транзакта, вызванную длительностью обслуживания транзакта в обслуживающем приборе или блоке.

Интеррогативное управление в GPSS осуществляется с помощью операторов SEIZE , ENTER , а также TEST и GATE , работающих в режиме отказа. Если транзакт пытается в текущий момент времени моделирования войти в блок, определяемый этими операторами, и получает отказ на вход ввиду его занятости другим транзактом, то он помещается в цепь текущих событий. В цепи текущих событий сформированный таким образом транзакт находится до тех пор, пока требуемый им блок не освободится. Это значит, что в каждый текущий момент времени моделирования интерпретатор путем возможно неоднократного просмотра цепи текущих событий проверяет условия продвижения этого транзакта и удаляет его из этой цепи, как только эти условия будут выполнены.

В языке GPSS , также как и СИМУЛА-67, имеются специальные средства, определяющие приоритеты исполнения событий, запланированных на один и тот же момент текущего времени. Это позволяет более точно учитывать в процессе моделирования причинно-следственные связи между элементами, и тем самым правильно отображать логику функционирования модели.

4.4. Инициация и проведение имитационного моделирования

Помимо перечисленных ранее программных компонент в состав имитационной модели должны входить средства, обеспечивающие:

инициацию имитационной модели;

доступ к данным, отображающим текущие состояния элементов системы;

обработку и вывод результатов моделирования в соответствии с назначением имитационной модели.

Инициация имитационной модели заключается в задании необходимых для начала ее функционирования значений переменных состояний элементов системы и программного имитатора. При использовании универсальных языков программирования или языков моделирования эти функции в модели часто выполняет специальный программный блок. Этот блок исполняется всегда перед началом моделирования. Пользователь с помощью операторов этого блока может задать генерирование и активацию необходимых элементов, которые обеспечат дальнейшие вычисления.

Некоторые языки, например СИМУЛА-67, допускают наличие в блоке инициации операторов планирования. Соответствующая программа в таком случае становится псевдоактивной, обладающей такими же средствами реализации, как и другие элементы.

Заслуживает внимание проблема доступа к данным в процессе имитационного моделирования. В общем случае она решается традиционными средствами, характерными для любых языков программирования. Однако при программировании имитационных моделей очень часто возникает необходимость во введении так называемых неэволюционных переменных, которые используются только для отображения промежуточных состояний элементов модели. По своему характеру они напоминают переменные состояния, но отличаются от них тем, что не участвуют в реализации операторов перехода системы из одного состояния в другое.

Большим разнообразием отличаются существующие системы моделирования в организации доступа одних элементов системы к атрибутам других элементов с целью их модификации. Из средств, имеющихся в современных языках, необходимо обратить внимание на средства доступа к атрибутам в языке СИМУЛА-67. Имеющиеся в нем средства предназначены для системной проблематики и обеспечивают доступ к различным частям структур данных.

Использование существующих систем моделирования дает широкие возможности пользователю по отображению и обработке результатов

моделирования. Например, в языке GPSS полный набор промежуточных статистических данных можно получить при использовании операнда C оператора START. Выборочный набор статистических данных может быть получен при использовании оператора PRINT. Аналогичные процедуры существуют в языке СИМУЛА-67: outimage, outfrac, outfix и др. Кроме того, в распоряжении пользователя имеются программы для построения таблиц, гистограмм, различных графиков и других форм, удобных для анализа результатов моделирования.

При программировании имитационных моделей на универсальных алгоритмических языках средства обработки и отображения информации разрабатываются самостоятельно. Если используется процессная концепция формализации систем, то эти средства могут быть объединены в одном программном блоке, который обладает такими же средствами реализации, как и другие элементы, но активизируется только в момент времени окончания моделирования.

К сожалению, объем пособия не позволяет более подробно рассмотреть вопросы инициации и проведения имитационных экспериментов. Однако они в значительной степени определяются возможностями используемых систем моделирования, и читатель может познакомиться с ними самостоятельно без особых трудностей.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Киндлер И. Языки моделирования/Пер. с чешск. М.: Энергоатомиздат, 1985.

2. Кирюхин В. М. Моделирование технологических процессов при проектировании АСУ ТП : Учебное пособие. М.: МИФИ, 1984.

3. Технология системного моделирования/Аврамчук Е.Ф., Вавилов А.А. Емельянов С.В. и др. Под общ.ред. С.В. Емельянова и др. М.: Машиностроение; Берлин: Техник, 1988.

4. Достойнова Т.Е. Язык "Динамо". М.: ВНИИСИ АН СССР, 1982. Ч. II.

5. Шрайбер Т. Дж. Моделирование на GPSS. М.: Машиностроение, 1980.

6. Голованов О.В., Дуванов С.Г., Смирнов В.Н. Моделирование сложных дискретных систем на ЭВМ третьего поколения. М.: Энергия, 1978.

7. Андрианов А.Н., Бычков С.П., Хорошилов А.И. Программирование на языке СИМУЛА-67. М.: Наука, 1985.

8. Имитационное моделирование средствами системы НЕДИС и ГАСП-1У/ Т.П. Марьянович, С.С. Азаров, В.В. Гусев и др. – Кибернетика, 1980, № 3, с. 35–51.

9. Вигдорчик Г.В., Воробьев А.Ю., Ростокин Б.И. Системы имитационного моделирования дискретных и непрерывных процессов для СМ ЭВМ/ Вычислительная техника социалистических стран. Вып. 8. М.: Статистика, 1980, с. 96–103.

ОГЛАВЛЕНИЕ