УДК 519.876.5

Описание, компоновка и работа модели в инструментальной системе распределенного моделирования[1]

Ю.И. Бродский

В данной работе освещается вопрос, как по описаниям на языке описания комплексов и компонент конструируется распределенная модель, создается ее база данных, как осуществляется синтез модели на этапе выполнения имитационного эксперимента.

Данная работа является продолжением [1], где описывались основные принципы, на которых основе которых строится инструментальная система распределенного моделирования. Здесь мы остановимся подробнее на том, как эта система работает.

Прежде всего, на языке описания комплексов и компонент (ЯОКК) описываются компоненты модели, или же ищутся описания компонент на других рабочих станциях системы. Ниже приводится предварительное описание языка описания распределенной модели.

ЯЗЫК ОПИСАНИЯ КОМПЛЕКСОВ И КОМПОНЕНТ

В данном разделе разобран специализированный непроцедурный язык, на котором должны составляться специфика-

[1] Работа выполнена при поддержке РФФИ, грант 07-07-00071-а.

ции классов компонент модели. Отметим, что хотя аккуратнее употреблять термин "спецификация класса компонент", мы ради краткости будем пользоваться словосочетанием "спецификация компоненты". Лингвистические формулы записываются ниже в стандартной нотации, согласно которой служебные слова выделяются жирным шрифтом, разделители - двойными кавычками, а необязательные включения квадратными или фигурными скобками, причем фигурные скобки указывают на возможность повторения. Разбор языка начнем с двух нетривиальных конструкций, применяемых в спецификациях всех типов. Это - определение структуры данных и оператор коммутации.

ОПРЕДЕЛЕНИЕ ТИПОВ ЗАПИСЕЙ ФАЗОВЫХ ПЕРЕМЕННЫХ И КОНСТАНТ

Собирая модель, инструментальная система автоматически генерирует ее базу данных, руководствуясь при этом содержимым специальных параграфов спецификаций компонент. Речь идет о параграфах определений типов фазовых переменных и констант. Их синтаксис различается только открывающей ключевой конструкцией, а в остальном сводится к правилам определения структур данных. Формально определение структуры данных есть набор определений списков полей, т.е. - набор списков идентификаторов, завершающихся указаниями типов поименованных величин. При этом допускаются простые типы, структурные типы и тип массива. К простым относятся встроенные и каталогизированные типы, фиксируемые своими идентификаторами. Структурные типы - это типы обычных записей, ключей виртуальных записей и ключей виртуальных списков. Их указания выглядят как выделенные служебными словами наборы определений списков по-

лей (здесь - рекурсия !). Наконец, диапазоны индексов допускаются лишь числовые и тип элемента массива должен быть простым или структурным типом.

В стандартной нотации определение структуры данных выглядит следующим образом:

ОпрСтруктДанных: **::=** ОпрГруппыПолей { ОпрГруппыПолей }.ОпрГруппыПолей **::=** Имя { "." Имя } ":" УказаниеТипа .

УказаниеТипа **::=** ИмяПростогоТипа ";" | УказаниеСтруктурногоТипа |УказаниеТипаМассива |УказаниеТипаПеречисления.

ИмяПростогоТипа **::=** ИмяВстроенногоТипа | ИляКаталогизированногоТипа.

ИмяВстроенногоТипа **::=** BOOLEAN | CHAR | BYTE | CARDINAL |BITSET | INTEGER | LONGINT | REAL | INDEFINITE | TEXT | NAME | PICTURE |BACKGRND.

ИмяКаталогизированногоТипа **::=** Имя

УказаниеСтруктурногоТипа **::=** ИмяСтруктурногоТипа ОпрСтруктДанных END ";".

ИмяСтруктурногоТипа **::=** RECORD | LIST OF | BLOCKS OF |ADDRESS OF.

УказаниеТипаМассива **::=** ARRAY Диапазон { "," Диапазон }OF ТипЭлемента.

Диапазон :: = " [" ЦелоеЧисло ".." ЦелоеЧисло "]".

ТипЭлемента **::=** ИмяПростогоТипа ";" |
УказаниеСтруктурногоТипа.

УкаэаниеТипаПеречисления :
: = " (" Имя { "," Имя } ");".

В этих формулах термин Имя означает любую последовательность символов без пробелов длиной не более 20, не содержащую использованных в контексте разделителей.

Понятие каталогизированного типа и специфичные встроенные типы INDEFINITE, TEXT, NAME, PICTURE, BACKGRND, LIST OF, BLOCKS OF, ADDRESS OF будут разобраны ниже. Прочие встроенные типы определяются следующей таблицей соответствий:

Тип в инструментальной системе	Тип в МОДУЛЕ-2	Тип в Языке С	Длина в байтах
BOOLEAN	BOOLEAN	Char	1
CHAR	CHAR	Char	1
BYTE	BYTE	Char	1
перечисление	перечисление		1
BITSET	BITSET	Unsigned	2
CARDINAL	CARDINAL	Unsigned.	2
INTEGER	INTEGER	int	2
LONGINT	LONGINT	long	4
REAL	LONGREAL	double	8

Чтобы была возможность использовать в разных спецификациях одни и те же структурные типы данных, введен аппарат каталогизированных типов: в библиотеке системы, наряду со спецификациями групп, объектов и приборов, можно заводить

в специально выделенной секции спецификации структурных типов данных, тем самым каталогизируя их. Синтаксис спецификации каталогизированного типа таков:

ОписательКаталогТипа. :: = **TYPE** Имя ";"
ОпрСтруктДанных **END ";".**

Смысл употребленных в правой части терминов тот же, что в предыдущей серии определений, причем Имя - и есть то имя, под которым данный тип будет зарегистрирован в библиотеке, и под которым его можно упоминать в других спецификациях. Отметим, что приведенное определение рекурсивно, т.е. допускает использование в спецификации каталогизированного типа имен иных типов того же сорта. Циклов в этой рекурсии быть не должно, и система не допустит их, отказываясь компилировать спецификацию, если еще не заведены или не откомпилированы спецификации всех упоминаемых в ней каталогизированных типов.

Примеры спецификаций каталогизированных типов:

```
TYPE VECTOR;
        X, Y, Z : REAL;
     END;
TYPE MATRIX;
        Rows : ARRAY [1..3] OF VECTOR;
     END;
```

В номенклатуре встроенных типов есть два графических. Это - PICTURE и BACKGRND. Оба определяют ключи неких картинок, формально являющихся динамическими записями особой структуры. Первый тип относится к маленьким картинкам-пиктограммам. Второй - тип ключа полномерной картинки. Переменные обоих типов предназначаются для графической анимализации моделей. Использование их в про-

граммах имитации в качестве фактических параметров соответствующих инструментальных процедур позволяет создавать "мультфильмы" эволюции воспроизводимых процессов. Инициализация этих переменных, включающая и создание ассоциируемых с ними изображений, обычно должна осуществляется через соответствующие графические редакторы, но может выполняться и программно.

Следует упомянуть и три еще не разобранных динамических типа, это типы BLOCKS OF и LIST OF относятся к ключам блокированных и неблокированных списков из записей одной длины, а тип ADDRESS OF - к ключам одиночных динамических записей. Структура записей всегда фиксируется текстом, следующим за именем типа.

При инициализации все динамические структуры размещаются в копии базы данных, создаваемой системой на время сеанса в виртуальной памяти. По завершении сеанса эту копию можно сохранить, и тогда сохранятся все заведенные данные; тем самым будет обеспечена возможность их использования в последующих сеансах. Пример определения структуры данных:

```
numb :  CARDINAL;
vect :  VECTOR;
cards : LIST OF
          card :  RECORD
                    name : (knave, queen, king);
                      pict : PICTURE;
        END;
        color : (spades, leaves, diamonds, harts);
        trump : BOOLEAN;
        END;
```

ОПЕРАТОРЫ КОММУТАЦИИ

Согласно принятой концепции часть параметров компонент комплекса может явно моделироваться в других его компонентах, т. е., являться частью фазовых переменных этих компонент. Комплексу также разрешено иметь собственные параметры и фазовые переменные. Для описания возможных связей параметров с фазовыми переменными используются операторы коммутации. Реально стыковка параметров и фазовых переменных, а также проверка возможности такой стыковки осуществляется во время сборки модели.

Общий вид оператора коммутации таков:

ОператорКоммутации :: = {ОпределительДиапазона}
АдресПараметра "=" АдресФазовойПеременной ";".

Один оператор определяет один фрагмент связи, причем четко фиксируется направление связи - от контакта, указанного за равенством, к контакту, указанному перед ним. Синтаксис адресов контактов в операторах коммутации определяется формулами вида:

АдресПараметра, :: = {АдресКомпоненты.}ИмяПараметра
{ИндексПараметра}

АдресКомпоненты, :: = ИмяКомпоненты " ("
ИндексКомпоненты ")"

АдресФазовойПеременной, :: = {АдресКомпоненты"." }
ИмяФазовойПеременной
{ИндексФазовойПеременной}

Если адрес контакта сводится к его индексу, то это значит, что контакт принадлежит внешнему разъему описываемой компоненты модели. ИмяПодгруппы, ИмяОбъекта и Имя-

Прибора - суть идентификаторы соответствующих классов компонент. Служебное слово INT помечает внутренние разъемы приборов. Термины ИндексПараметра, ИндексФазовойПеременной формально расшифровываются одинаково:

ИндексПараметра,
ИндексФазовойПеременной **::=** "(" ПростоеЦелоеВыражение ")".
ПростоеЦелоеВыражение **::=** ЦелоеБезЗнака |
[Целое Знак] "i*" ЦелоеБезЗнака |[Целое Знак] ЦелоеБезЗнака "*i" |[Знак] "i*" ЦелоеБезЗнака Знак ЦелоеБезЗнака |[Знак] ЦелоеБезЗнака "*i" Знак ЦелоеБезЗнака.
Знак ::= "+" | "-".

Во второй формуле все варианты, кроме первого, относятся к случаю, когда записи оператора коммутации предшествует

Определитель диапазона **::=** "i=" Целое ".." Целое ":".

Это - конструкция, позволяющая одной записью оператора коммутации определить целую группу фрагментов каналов связи, а каких именно - ясно из контекста. Осталось только подчеркнуть, что указываемые в квадратных скобках номера подгрупп и объектов или приборов являются порядковыми во внутренней индексации той группы или объекта, в чей описатель войдет оператор. Примеры операторов коммутации:

```
i = 1..10 : ОбъектПервогоТипа (i).МассивПараметров(2*i-1) =
                ОбъектВторогоТипа (10-i).МассивФазовых-
                Переменных(2*i);
```

A1=Спутник.X;
Объект(0).x=x;

СИНТАКСИС ОПИСАТЕЛЕЙ КОМПЛЕКСОВ

Двух предыдущих подразделов достаточно, чтобы без подробных дополнительных разъяснений определить принятые правила формирования спецификаций комплексов и компонент. Формальный синтаксис спецификации комплексов таков:

СпецификацияГруппы **::= COMPLEX** ИмяКомплекса**";"**
[ПараграфКомпонент]
[ПараграфФазовыхПеременных]
[ПараграфКонстант]
[ПараграфПараметров]
[ПараграфКоммутации].

ПараграфКомпонент ::= **COMPONENTS**
ПереченьСоставляющих; { ПереченьСоставляющих }
{END";"}

ПараграфФазовыхПеременных **::= PHASE**
ОпрСтруктДанных
{**END ";"}**
ПараграфПараметров **::= PARAMETERS**
ОпрСтруктДанных
{**END ";"}**
ПараграфКонстант **::= CONST**
ОпрСтруктДанных
{**END ";"}.**
ПараграфКоммутации **::= COMMUTATION**

ГруппаКоммутации { ГруппаКоммутации. } **{END ";"}.**

Во всех параграфах, кроме последнего в комплексе **END'ы** необязательны. Заголовок следующего параграфа автоматически означает конец предыдущего.

Смысл использованного здесь термина ОпрСтруктДанных разобран в предыдущем разделе, а несложные формулы определений других новых терминов выглядят следующим образом:

ПеречеьСоставляющих **::=** ИмяКомпоненты " (" ЧислоЭкземпляров ")"
{"," ИмяКомпоненты "(" ЧислоЭкземпляров ")" } ";".
ГруппаКоммутации ::= [ОпределительДиапазона]
ОператорКоммутации.

В этих формулах ИмяКомплекса и ИмяКомпоненты суть имена классов компонент соответствующего типа, а формально – идентификаторы длиной не более 20 символов; ЧислоЭкземпляров – положительное целое, равное числу составляющих указанного перед скобкой класса; ОпределительДиапазона, ОператорыКоммутации - термины из предыдущего раздела.

СИНТАКСИС ОПИСАТЕЛЕЙ МЕТОДОВ

Методы – это или элементы процессов, или же методы вычисляющие наступление событий. Это те части модели, которые могут вызываться как локально, так и распределено. Считается, что в качестве параметров методу передается некая запись, тип которой описан, и она же возвращается методом. Так как тип этой записи, в особенности для удаленных методов, определяется разработчиком метода, а не разработчиком модели, возникает вопрос о коммутации полей параметров метода с полями фазовых переменных и/или параметров компоненты. Кроме того, методы реализующие элементы процессов различаются по отношению с модельному времени на

- сосредоточенные – происходящие мгновенно,
- распределенные – занимающие не менее одного модельного такта и дающие определенный результат своего выполнения в виде изменений внутренних переменных модели в конце каждого такта,
- условно-распределенные – занимающие не менее одного модельного такта, но дающие результат своего выполнения в виде изменений внутренних переменных модели лишь при полном завершении выполнения.

ОписательМетода ::= METHOD ИмяМетода **":"** ТипМетода **";"**

[ПараграфПараметровМетода]

ТипМетода ::= **FAST** | **CONV** | **SLOW** | **EVENT.**

По умолчанию тип метода, реализующего элемент процесса, считается **SLOW,** и в этом случае явное его указание разрешается опустить.

Синтаксис параграфа параметров метода такой же, как у параграфа фазовых переменных описателя комплекса. Поскольку это единственный параграф описателя, он обязан закончиться ключевым словом **END.**

СИНТАКСИС ОПИСАТЕЛЕЙ КОМПОНЕНТ

Формула спецификации компоненты такова:

ОписательКомпоненты :: = **COMPONENT** ИмяКомпоненты **";"**

[ПараграфФазовыхПеременных]

[ПараграфПараметров]

[ПараграфКонстант]

[ПараграфМетодов]

[ПараграфКоммутации]

[ПарграфСобытий]

[ПарграфПереключателей].

Как и в случае комплексов, **END** обязателен лишь в самом последнем параграфе. В параграфе методов спецификации компоненты одним или несколькими процессами перечисляются все выполняемые в данном процессе методы, начиная с корневого, с которого по умолчанию будет начинаться модельная жизнь каждого процесса. Имена методов должны быть уникальными в пределах спецификации: нельзя одинаково назвать два разнотипных метода одной компоненты, но использование одного имени в спецификациях разных компонент не

возбраняется. Формально синтаксис параграфа методов определяется так:

ПарагрефМетодов **::** = ПотокМетодов |
| ПарагрефМетодов ПотокМетодов **{END ";"}.**
ПотокМетодов **::=** ИмяМетода { "," ИмяМетода } ";".

Здесь ИмяМетода - идентификатор длиной не более 20 символов.

Параграф коммутации синтаксически не отличается от параграфа коммутации компонент комплекса. Содержательный же его смысл в том, что параметры методов достаются разработчику модели от разработчиков методов такими, какими они были удобны последним, в частности множество параметров метода может быть существенно уже множества фазовых переменных, параметров и констант компоненты, а может и совпадать или даже оказаться шире последнего. В любом случае, параграф призван указать, каким параметрам метода соответствуют те или иные фазовые переменные, параметры и константы компоненты.

ПарагрефКоммутации **::= COMMUTATION**
ГруппаКоммутации { ГруппаКоммутации. } **{END ";"}.**
ГруппаКоммутации ::= [ОпределительДиапазона]
ОператорКоммутации.

Параграф событий определяет набор событий, связанных с данной компонентой. Определяется идентификатор события и условие его наступления.

ПараграфСобытий **::** = ОписаниеСобытия { ";" ОписаниеСобытия } ";" **{END ";"}**.

ОписаниеСобытия **::** = ИмяСобытия ":" МетодВычисляющийСобытие ";".

Параграф переключателей определяет автоматную функцию процесса. В нем для каждого элемента каждого процесса должны быть перечислены все разрешенные переходы, т.е. названы все элементы, на которые прибор может переключаться по завершении данного, и указаны события, в зависимости от которых реализуется то или иное переключение. Формат параграфа таков:

ПарграфПереключателей **::= SWITCHES** АвтоматнаяФункция **{END ";"}**

АвтоматнаяФункция **::=** ПереключенияЭлемента { ПереключенияЭлемента }.

ПереключенияЭлемента :: = ИмяЭлемента ":" СписокПереходов.

СписокПереходов **::=** { ИмяЭлемента "," ИмяСобытия "."}ИмяЭлемента ";".

Понятно, что перед двоеточием во второй формуле стоит имя того элемента, переходы с которого определяет следующая за двоеточием конструкция. В последней перечисляются возможные переключения. Они упорядочиваются по важности событий, причем завершается перечень именем элемента, при котором не стоит никакого события. Это значит, что если не реализуется ни одно из вошедших в СписокПереходов событий, то произойдет переключение на элемент с указанным именем.

КОМПИЛЯЦИЯ ОПИСАТЕЛЕЙ КОМПОНЕНТ МОДЕЛИ

После того как имеется описание модели и всех ее компонент, можно перейти к компоновке модели. Первый этап компоновки модели – компиляция описателей. Компиляция описателя – это получение из него одной или нескольких незаполненных таблиц для будущей базы данных модели. Например, в результате компиляции описания типа

```
TYPE VECTOR;
        X, Y, Z : REAL;
     END;
```

в базе данных модели в таблице «Типы_модели» появится еще одна строка вида:

Типы_модели

Номер_типа	Имя_типа	Размер_типа
. . .	. . .	. . .
n	VECTOR	24

кроме того, в таблице «Поля_типов» добавятся три новых строчки:

Поля_типов

Ключ поля типа	Нмер типа	Номер поля	Имя поля	Тип поля	К-во экзем-пляров
	. . .	. . .	. . .	. . .	
k	N	1	X	**REAL**	1

k+1	N	2	Y	**REAL**	1
k+2	N	3	Z	**REAL**	1

Таблицы «Типы_модели» и «Поля_типов» связаны отношением «один ко многим». Заметим, что поле «Размер_типа» в таблице «Типы_модели» может остаться незаполненным, если типы полей компилируемой записи сложнее, чем встроенные. В последнем случае это поле либо будет заполнено в процессе компоновки компоненты, содержащей этот тип, если удастся пройти до конца (до встроенных типов, являющихся терминальными элементами) всю цепочку взаимных ссылок типов. Либо компоновка закончится аварийно, например, если какой-то упомянутый в описании поля тип не будет найден в таблице «Типы_модели», либо если цепочка взаимных ссылок типов зациклится. Чтобы не придумывать для встроенных типов каких-то специальных механизмов узнавания, а пользоваться для всех типов одним и тем же алгоритмом, можно считать, что все встроенные типы изначально занесены в таблицу «Типы_модели» и отсутствуют в таблице «Поля_типов».

В результате компиляции описателя компоненты добавляется очередная строчка в таблицу «Компоненты_модели»:

Компоненты_модели

Номер компоненты	Имя компоненты	Фаза	Параметры	Константы	Процессы начало	Число процессов
. . .		. . .	. . .	. . .	. . .	. . .
n	Имя	n_Ph	n_Par	n_Con	n_Pr_b	n_Pr

Здесь n – номер компоненты – ключевое поле в таблице компонент модели, n_Ph, n_Par, n_Con – номера типов фазовых переменных, параметров и констант модели в таблице

«Типы_модели», n_Pr_b – номер первого процесса компоненты, а n_Pr – количество процессов компоненты в таблице «Процессы_модели». При этом предполагается, что все записи процессов компоненты идут подряд в таблице «Процессы_модели». Соответственно, в таблице «Процессы_модели» появляется n_Pr новых записей, соответствующих процессам компилируемой компоненты:

Процессы_модели

Номер процесса	Корневой элемент

Также будут добавлены новые строки в таблицы «Методы_модели»,

Номер метода	Имя метода	Номер процесса	Параметры метода	Тип метода	Адрес метода

Чтобы получить, например, список имен и адресов методов m–го процесса, необходимо выполнить SQL-запрос:

SELECT [Имя элемента], [Адрес элемента] FROM [Методы_модели] WHERE [Номер процесса]=m.

Коммутация методов должна быть отражена в следующей таблице:

Коммутация_методов

Ключ коммутатора методов	Номер метода	Поле параметра метода	ФПК (фаза-парам.-конст.)	Поле компоненты

«Переключатели_модели»

Ключ переключателя	Важность переключателя	Номер процесса	Номер элемента	Номер события	Элемент перехода

Поле «Важность переключателя», вообще говоря, избыточно. Можно было бы договориться, что переключатели добавляются в таблицу «Переключатели_модели» в порядке важности, так как они фигурируют в параграфе переключатели описателя компоненты. Однако, во избежание путаницы в этом важном вопросе, представляется целесообразным добавить такое поле. Чтобы получить, например, сортированный в порядке важности список событий перехода и следующих элементов для элемента n, m-го процесса, необходимо выполнить SQL-запрос:

SELECT [Номер события], [Элемент перехода]
FROM [Переключатели_модели] WHERE
[Номер процесса]=m AND [Номер элемента]=n
ORDER BY [Важность переключателя].

Таким образом, компиляция описателя компоненты может вызвать пополнение следующих таблиц базы данных:

- Типы_модели,
- Поля_типов,
- Компоненты_модели,
- Процессы_модели,
- Методы_модели,
- Коммутация_методов,
- Переключатели_модели.

Теперь остановимся на том, как компилируется описатель комплекса. В рамках предлагаемой концепции построения инструментальной системы распределенного моделирования, понятие комплекса является вспомогательным, удобным для построения сложных иерархических моделей. При компиляции комплекс преобразуется в «комплекс как компоненту» путем объединения констант, фазовых переменных, процессов, элементов, событий и наконец, параметров. При этом из объединенных параметров исключаются те, что участвуют в коммутации компонент [1]. При исключении параметра, нужно проследить с какими методами он коммутировал, и заменить в строках таблицы коммутации методов комплекса как компоненты поля связанные с этим параметром на поля соответствующей фазовой переменной.

На первом этапе компиляции описателя комплекса, добавится строчка в таблицу «Комплексы_модели»

Комплексы_модели

Ключ комплекса	Имя комплекса	Фазовые переменные	Параметры	Константы

Перечень компонент компилируется в несколько строк таблицы «Компоненты_комплексов».

Компоненты_комплексов

Ключ компонент комплексов	Номер комплекса	Номер компоненты	Количество экземпляров

Параграф коммутации компилируется в строки таблицы коммутации компонент:

Коммутация_компонент

Ключ коммутатора	Номер комплекса	Номер вх. компоненты	Экземпляр компоненты	Поле параметра	Номер исх. компоненты	Экземпляр компоненты	Поле фазовой переменной

Далее, на основе этих таблиц, можно строить «комплекс как компоненту», объединяя с учетом количества экземпляров фазовые переменные, константы, параметры, процессы, элементы, события.

Выбрать параметры, подлежащие исключению можно сделав запрос

SELECT * FROM [Коммутация_компонент] WHERE [Номер комплекса] = n

Выбрать методы, подлежащие перекоммутации, можно по запросу:

SELECT [Номер метода] FROM [Коммутация_методов] WHERE [ФПК] = “параметры” AND [Поле компоненты] = ИмяПоля.

Наконец, поля фазовых переменных на которые нужно заменить поля параметров в таблице «Коммутация_методов» также находятся в таблице первого из запросов.

КОМПОНОВКА МОДЕЛИ

На стадии компиляции описателей модели формируется, как это было показано выше база данных, описывающая состав компонент модели и связи между ними. Задача этапа компоновки – проверить полноту и непротиворечивость системы этих описаний и сгенерировать на их основе рабочую базу данных для последующего проведения имитационных экспериментов.

Исходным материалом для компоновки модели является результат компиляции компоненты или «комплекса как компоненты», а именно, таблицы

1. Типы_модели,
2. Поля_типов,
3. Компоненты_модели,
4. Процессы_модели,
5. Методы_модели,
6. Коммутация_методов,
7. Переключатели_модели.

На основе первой и четвертой из них строятся две новые таблицы: по имеющимся описаниям типов переменных выделяется реальное пространство или для всего имитационного эксперимента, или же для нулевого момента времени, с последующим выделением пространства для очередного шага имитации. Заполнение базы данных начальными условиями осуществляется вручную или же с помощью специальной программы, в данной работе мы не останавливаемся на технике заполнения базы данных модели начальными значениями, хотя

этот процесс и может оказаться нетривиальным. Также таблица «Процессы модели» разворачивается в модельном времени, и на каждом шаге моделирования в ней будет указываться текущий элемент процесса.

РАБОТА МОДЕЛИ ВО ВРЕМЯ ЭКСПЕРИМЕНТА

Предположим, что мы находимся в начале такта имитации. Задан некий стандартный шаг времени Δt .

- Просматривая таблицу процессов, выполняем текущие сосредоточенные элементы, передавая им параметры в соответствии с таблицей коммутации методов. Возвращенные значения параметров присваиваем соответствующим фазовым переменным. В соответствии с таблицей переключателей, вычисляем следующие элементы.
- Повторяем предыдущий шаг, пока имеются сосредоточенные элементы.
- Выполняем со стандартным шагом Δt распределенные и условно-распределенные элементы. Вычисляем наступление событий, связанных с очередным элементом. В частности, такими событиями могут быть и заранее запланированные с помощью специальных системных средств, как в [2], моменты окончаний элементов. Как указывалось в [1], наступление события есть обращение в ноль соответствующей функции. Методы событий реализуют именно такие функции. Смена функцией события знака на концах отрезка Δt говорит о наступлении события внутри этого отрезка. Далее, с помощью того или иного метода аппроксимации, с приемлемой точностью находится момент наступления события. Минимум из этих времен по всем элементам даст продолжи-

тельность следующего шага имитации, после которого для закончившихся элементов в соответствии с таблицей переключателей и наступившими событиями вычисляются последующие.
- Модельное время переводится на вычисленную продолжительность прошедшего шага имитации.

Мы оказываемся в начале нового такта имитации, который снова пытаемся пройти исходя из стандартного шага Δt.

Литература

1. Бродский Ю.И. К разработке концепции построения инструментальной системы распределенного моделирования. //Моделирование, декомпозиция и оптимизация сложных динамических процессов, М.: ВЦ РАН, 2007, С.14-34.
2. Бродский Ю.И., Лебедев В.Ю. Инструментальная система имитации MISS. М.: ВЦ АН СССР, 1991, 180с.
3. Павловский Ю.Н., Белотелов Н.В., Бродский Ю.И. Имитационное моделирование. М.: «Академия», 2007, 236с.