А.А. Емельянов

Технологии имитационного моделирования в системе *Pilgrim*

В последние несколько лет увеличился интерес к инструментальным средствам имитационного моделирования. Это связано с тем, что далеко не всегда исследуемые процессы можно описать математическими моделями. Имитационные модели, использующие принципы аналогового моделирования, позволяют получать интересные и практически полезные результаты. Имитационная модель представляет собой специальное программное обеспечение, позволяющее отобразить процессы, протекающие в моделируемой системе, с необходимыми для решения поставленной задачи набором параметров и уровнем детализации. В результате прогона модели автоматически собирается статистическая информация о моделируемых процессах. Обычно для создания модели не нужны громоздкие математические выражения, которые, зачастую, либо неизвестны, либо описывают идеализированные случаи.

Построение сложных имитационных моделей невозможно без применения специальных моделирующих пакетов прикладных программ. Например, программная модель простейшей системы массового обслуживания, реализованная на языке C++, содержит сотни строчек программного кода, в то время как та же модель, созданная с помощью пакета *Pilgrim*, имеет только 4 строчки. Реальная модель экономического процесса должна включать в себя десятки, а то и сотни, процессов как массового обслуживания, так и многих других, поэтому ее создание на универсальном языке программирования становится чрезвычайно сложным.

Все пакеты имитационного моделирования, независимо от удобства их использования, позволяют разработчику модели наблюдать за пространственной динамикой процесса: *GPSS World*, *Vensim*, *Pilgrim* и др. В настоящее время для моделирования экономических процессов наиболее часто используют пакет *Pilgrim*. Достоинствами этого программного пакета являются следующие возможности:

- создания не только дискретных, но и дискретно-непрерывных моделей;
- имитации не только временно́й, но и пространственной, а также финансовой динамики.

Эти возможности являются принципиальными. При моделировании работы даже простой бухгалтерии с помощью *GPSS* необходимо дополнительно создать и вставить в модель программы, реализующие бухгалтерский учет со всеми правилами проводок (т.е. создать систему бухучета). В *Pilgrim*-моделях таких проблем не возникает.

Ниже приведены основные концептуальные положения и моделирующие функции актуализированной версии пакета *Pilgrim*.

Основные объекты модели

Моделирующая система *Pilgrim* выполняет следующие основные функции:

1) предоставляет разработчику средства для формализованного описания дискретных компонентов, дисциплин выполнения различных работ, для задания структуры графа и привязки объектов модели к координатной сетке общего информационного поля;

2) осуществляет координацию событий, определение путей прохождения транзак-

тов, изменение состояния узлов и передачу управления моделям непрерывных компонентов.

Такая система позволяет передавать в базы данных экономической информационной системы результаты моделирования, используемые для принятия управленческих решений.

Существует 6 основных понятий, на которых базируется концепция моделирующей системы.

1. Граф модели. Все процессы, независимо от количества уровней структурного анализа, объединяются в виде направленного графа. Пример изображения модели в виде многослойного иерархического графа, полученного при структурном анализе процесса, представлен на рис. 1.

2. Транзакт — это формальный запрос на какое-либо обслуживание. Транзакт в отличие от обычных заявок, которые рассматриваются при анализе моделей массового обслуживания, имеет набор динамически изменяющихся особых свойств и параметров. Пути миграции транзактов по графу стохастической сети определяются логикой функционирования компонентов модели в узлах сети.

Транзакт является динамической единицей любой модели, работающей под управлением имитатора, и может выполнять следующие функции:

- порождать группы (семейства) других транзактов;
- поглощать другие транзакты конкретного семейства;
- захватывать ресурсы и использовать их некоторое время, а затем — освобождать;
- определять времена обслуживания, накапливать информацию о пройденном пути и иметь информацию о своем дальнейшем пути и о путях других транзактов.

Основные параметры транзактов:

- уникальный идентификатор транзакта;
- идентификатор (номер) семейства, к которому принадлежит транзакт;
- наборы различных ресурсов, которые транзакт может захватывать и использовать какое-то время;
- время жизни транзакта;
- приоритет — неотрицательное число; причем чем больше число, тем выше приоритет транзакта (например, в очереди);
- параметры обслуживания в каком-либо обслуживающем устройстве (включая вероятностные характеристики).

Приведем некоторые примеры транзактов:

- требование на перечисление денег;
- заказ на выполнение работ в фирме;
- телеграмма, поступающая на узел коммутации сообщений;
- сигнал о загрязнении какого-либо пункта местности;
- приказ руководства;
- покупатель в магазине;
- пассажир самолета;

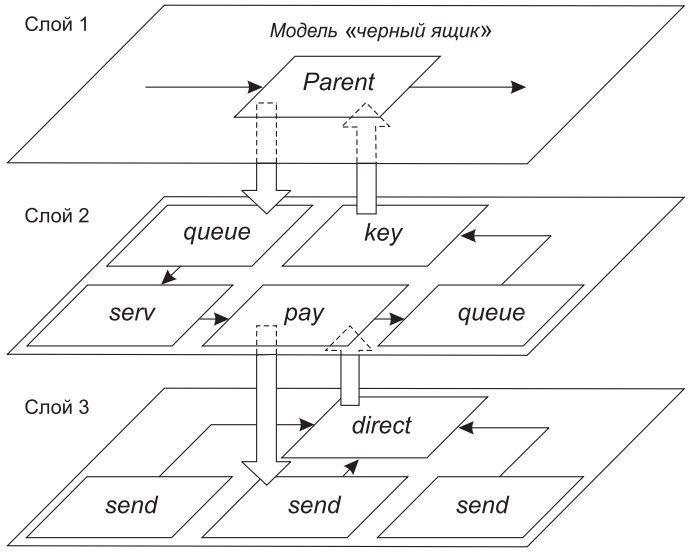

Рис. 1. Многослойный граф

• проба загрязненной почвы, ожидающая соответствующего анализа.

3. Узлы графа сети представляют собой центры обслуживания транзактов (но необязательно массового обслуживания). В узлах транзакты могут задерживаться, обслуживаться, порождать семейства новых транзактов, уничтожать другие транзакты. С точки зрения вычислительных процессов в каждом узле осуществляется независимый процесс. Вычислительные процессы выполняются параллельно и координируют друг друга. Они реализуются в едином модельном времени, в одном пространстве, учитывают временну́ю, пространственную и финансовую динамику.

Нумерация и присвоение имен узлам стохастической сети производится разработчиком модели. Транзакт всегда принадлежит одному из узлов графа и независимо от этого относится к определенной точке пространства или местности, координаты которой могут изменяться.

Примерами узлов могут быть:

• счет бухгалтерского учета;
• бухгалтерия;
• производственный (ремонтный) участок;
• генератор или размножитель транзактов;
• транспортное средство, которое перемещает ресурсы из одной точки пространства в другое;
• передвижная лаборатория;
• компьютерный центр коммутации сообщений (или пакетов сообщений);
• склад ресурсов.

4. Событием называется факт выхода из узла одного транзакта. События всегда происходят в определенные моменты времени и могут быть связаны также и с точкой пространства. Интервалы между двумя соседними событиями в модели — это, как правило, случайные величины. Предположим, что в момент времени t произошло какое-то событие, а в момент времени $t+d$ должно произойти ближайшее следующее событие, но не обязательно в этом же узле. Если в модель включены непрерывные компоненты, то передать им управление можно только на время в пределах интервала (t, $t+d$).

Разработчик модели практически не может управлять событиями вручную (например из программы), поэтому эта функция осуществляется специальной управляющей программой — координатором, автоматически внедряемым в состав модели.

5. Ресурс, независимо от его природы, в процессе моделирования может характеризоваться тремя параметрами: мощность, остаток и дефицит. *Мощность ресурса* — это максимальное число ресурсных единиц, которые можно использовать для различных целей. *Остаток ресурса* — число незанятых на данный момент единиц, которые можно использовать для удовлетворения транзактов. *Дефицит ресурса* — количество единиц ресурса, необходимое для удовлетворения потребностей запросов транзактов, стоящих в очереди к данному ресурсу.

При решении задач динамического управления ресурсами можно выделить три основных типа ресурсов: материальные, информационные и денежные.

6. Пространство бывает обычно двух видов: географическое и декартова плоскость (можно ввести и другие виды). Узлы, транзакты и ресурсы могут быть привязаны к точкам пространства и мигрировать в нем.

При построении модели используют объектно-ориентированный способ представления экономических процессов. Транзакты, узлы, события и ресурсы — основные объекты имитационной модели. Взаимодействие таких объектов показано на рис. 2, где обозначены следующие моделирующие функции: ag, key, queue, dynam, proc, term, e1 и e2. Функциональное назначение этих и других средств моделирующей системы подробно рассматривается в книге [1].

В различных моделирующих системах имеются разные способы представления узлов графа, что связано с отличительными свойствами систем. Например, в системе *GPSS* узлы называются блоками, причем имеется более 100 различных типов блоков, что затрудняет восприятие графа модели. В системе *Pilgrim* имеется всего 15 типов узлов, которые функционально перекрывают все возможности блоков *GPSS* и предоставляют дополнительные средства, которые в *GPSS* отсутствуют:

- возможность работы с непрерывными процессами;
- моделирование пространственной динамики;
- работу с ресурсами, представляющими собой деньги, материальные и нематериальные ценности, счета бухгалтерского учета и банковские счета.

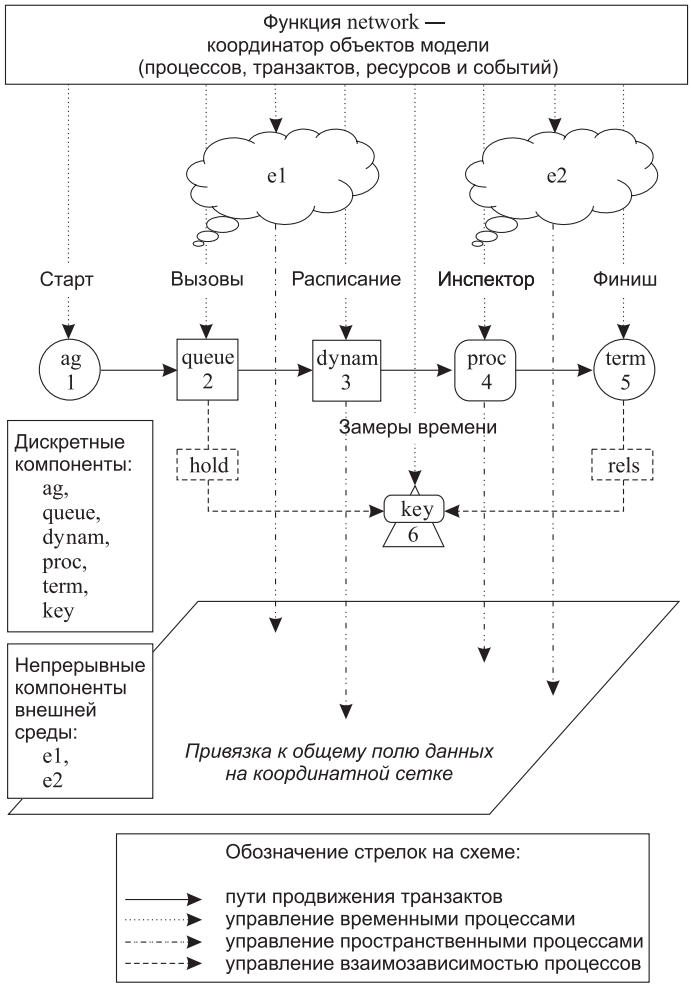

Рис. 2. Пример взаимодействия объектов имитационной модели

Языковые средства

Все узлы имитационных моделей являются процессами в системе *Pilgrim*. Стохастическая сеть, в виде которой представляется модель, не является вычислительным алгоритмом. Представление имитационной модели в виде набора алгоритмов приводит к написанию больших (и сложных) моделирующих программ. Такой подход называется алгоритмическим моделированием и он не всегда доступен экономисту, даже имеющему подготовку в области программирования. Например, запрограммированная на языке *Visual Basic* алгоритмическая модель, состоящая из очереди и обслуживающего прибора (всего два узла), занимает несколько страниц программного текста.

Реальная модель экономического процесса может состоять из десятков, а то и сотен узлов. Поэтому нужны *особые языковые средства*, не являющиеся языком программирования, которые позволили бы в лаконичном (по сравнению с текстом компьютерной программы) виде описать модель. Такие средства должны учитывать особенности узлов, жизненных циклов транзактов, условий прохождения транзактов по дугам, их размножения и гибели, а также функциональные особенности взаимодействия процессов, связанных с финансовыми, материальными и информационными ресурсами.

Ниже рассмотрен набор языковых взаимосвязанных средств, предназначенных

для описания имитационных моделей экономических процессов. Эти средства подразделяются на пять взаимосвязанных групп:

- формализация запуска имитационной модели: инициализация объектов и структур данных;
- описание узлов с помощью общих операторов управления транзактами, событиями и узлами модели;
- функциональное описание процессов управления материальными и денежными ресурсами;
- формализация структурного анализа — управление переходами между слоями модели при многоуровневой декомпозиции;
- описание сигнальных управляющих функций.

Данные средства представлены в виде функций, через параметры которых реализуются синтаксические связи между объектами (узлами, транзактами, ресурсами и событиями) имитационной модели. Форма записи различных условий и сопоставленных им действий соответствует языку C++.

Рассмотрение моделирующих функций имитатора, их аргументов или параметров при реализации конкретной модели на ЭВМ требует указывать типы соответствующих переменных. Эти типы интересны профессиональным математикам-программистам (экономистам, впервые начавшим заниматься имитационным моделированием, их знать необязательно):

- int — целое значение (обычно 16 разрядов);
- long — длинное целое значение (обычно 32 разряда);
- float — переменная с плавающей точкой (32 разряда);
- double — переменная с плавающей точкой двойной точности (64 разряда);
- char — символьная переменная или строка символов (1 байт).

Инициализация объектов и структур данных для запуска имитационной модели

Использование концепции имитационного моделирования *Pilgrim* позволяет так же, как и в теории стохастических сетей, разрабатывать два типа моделей: разомкнутые и замкнутые. Разомкнутые модели дают возможность сравнительно легко реализовать исследование внутренних процессов в фирме, но они не учитывают взаимосвязи с объектами внешней среды (рынок, госбюджет, население и др.). Граф стохастической сети замкнутых моделей выглядит сложнее, но такие модели позволяют учесть связи с внешней средой и исследовать взаимное влияние моделируемого объекта экономики с другими объектами, финансово-кредитными учреждениями, рынком.

Модель состоит из двух характерных частей: секции инициализации и блока описания стохастической сети. Все рассматриваемые ниже операторы — это программные функции, управляемые соответствующими аргументами.

Рассмотрим подробно структуру секции инициализации (рис. 3). Она состоит из 14 компонентов, причем необязательные компоненты заключены в квадратные скобки.

Секция начинается с макрооперации #include <Pilgrim.h>, которая подключает моделирующую среду имитатора к модели. Далее следует описание глобальных переменных и дополнительных функций, используемых в модели. Например, это могут быть функции, описывающие непрерывные компоненты, или вызовы этих функций.

Собственно модель начинается с оператора forward. За ним, если это необходимо, указываютя локальные переменные разработчика модели.

Оператор первоначальной настройки. Первоначальную настройку моделирующих программ и инициализацию (раз-

метку) графа модели в памяти ЭВМ осуществляет оператор modbeg (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9). Это одна из первых операций в основном блоке модели (после forward).

Аргументы этой функции имеют следующий смысл:

- параметр p_1 — символическое имя модели (char): строка длиной не более 14 символов;
- параметр p_2 — максимальный номер узла модели (int), причем справедливо соотношение $2 \le p_2 \le m_{max}$, где m_{max} — некоторое граничное значение, которое задается при установке имитатора на ЭВМ (обычно $m_{max} = 1024$);
- параметр p_3 — модельное время, в течение которого необходимо производить моделирование (типа float);
- параметр p_4 — произвольное целое число, используемое для настройки датчиков псевдослучайных величин (long). В каждом узле есть свой независимый датчик. В качестве значения p_4 полезно использовать значение таймера ЭВМ, обращение к которому имеет следующий вид: p_4 = (long) time (NULL). В этом случае результаты прогонов модели будут разными, имеющими случайные отклонения. При отладке лучше использовать постоянную комбинацию цифр, например p_4 = (long) 2013456789;
- параметр p_5 — это признак режима пространственной имитации (int):
  - earth — поверхность Земли, т. е. сферические географические координаты, широта и долгота;
  - plane — декартова плоскость, т. е. прямоугольная система координат;
  - cosmos — произвольное пространство (ответственность за правильность его представления возлагается на разработчика моделей);
  - none — если пространственная имитация в экономической модели не используется;
- параметр p_6 — номер одной из очередей (узла типа queue, attach или send), которую необходимо контролировать во времени для анализа динамики задержек с графическим отображением результатов (int). Например, если указать число 3, и в модели действительно имеется очередь с таким номером, то можно получать изображение динамики этой очереди непосредственно в процессе моделирования;
- параметр p_7 — номер (int) одного из процессов (узла типа proc), который необходимо контролировать как в пространстве, так и во времени с графическим отображением результатов. Пространственная имитация будет рассмотрена ниже. Если нет необходимости в графической интерпретации пространственной имитации, то указывается none;
- параметр p_8 — номер (int) терминатора (узла типа term), на входе которого необхо-

№ п/п	Оператор	Необходимость применения в модели
1	#include <Pilgrim.h>	Всегда необходим
2	[Глобальные переменные и функции]	Только в сложных моделях
3	forward	Всегда необходим
4	{	Всегда необходим
5	[Локальные переменные модели]	Часто необходимы
6	modbeg(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9);	Всегда необходим
8	ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8); ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8); … ag(p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8);	Всегда необходимы
7	[Сигнальные функции]	В ресурсных моделях
9	network(p_1,p_2)	Всегда необходим
10	{	Всегда необходим
Вставляется в секцию	Блок описания узлов модели (стохастическая многоуровневая сеть)	Всегда необходим
11	}	Всегда необходим
12	modend(p_1,p_2,p_3,p_4);	Всегда необходим
13	return 0;	Всегда необходим
14	}	Всегда необходим

Рис. 3. Структура секции инициализации модели

димо наблюдать интенсивность потока транзактов во время моделирования. Если такой необходимости нет, то указывается none;

• параметр p_9 — точность (int). Данный параметр может принимать следующие значения:

- если $p_9 = 1..6$, то имитатор будет использовать от 1 до 6 знаков после десятичной точки при формировании замеренных результатов (временны́х интервалов);
- если $p_9 =$ none, то имитатор будет представлять результаты округленными до целых значений.

За оператором modbeg следует описание узлов типа ag (генераторов транзактов). Это единственный тип узла, который должен быть описан в секции инициализации. Эти генераторы не зависят от состояния других узлов сети и должны работать до того, как узлы других типов будут приведены в рабочее состояние.

Генератор транзактов. Генератор с бесконечной емкостью — это функция ag (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8). Каждый генератор задается одним оператором ag. Обычно все ag записываются подряд (если их несколько) после modbeg. Аргументы ag имеют следующий смысл:

• параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — номер узла-генератора (int);

• параметр p_3 — приоритет (int), назначаемый каждому сгенерированному транзакту: число в диапазоне 1–32 767; если приоритет не нужен, то $p_3 =$ none;

• параметр p_4 — тип функции распределения интервала времени между двумя последовательно сгенерированными транзактами (int), имеющий значения:

- norm — нормальное распределение;
- unif — равномерное распределение;
- expo — экспоненциальное распределение;
- erln — обобщенное распределение Эрланга;
- beta — треугольное распределение;
- none — если интервал между транзактами является детерминированной величиной.

Полная информация об этих распределениях и соответствующих программных датчиках, используемых в модели, содержится в книге [1];

• параметр p_5 — величина, зависимая от типа функции распределения (float):

- математическое ожидание интервала времени между двумя последовательно сгенерированными транзактами ($p_4 =$ norm, unif, expo);
- математическое ожидание одного слагаемого этого интервала ($p_4 =$ erln);
- минимальное значение интервала ($p_4 =$ beta);
- постоянная величина этого интервала ($p_4 =$ none);

• параметр p_6 — величина, зависимая от типа функции распределения (float):

- среднеквадратичное отклонение ($p_4 =$ norm);
- максимальное отклонение от среднего ($p_4 =$ unif);
- значение zero ($p_4 =$ expo, none);
- число слагаемых, входящих в случайный интервал и распределенных по экспоненциальному закону (если $p_4 =$ erln, то $p_6 > 0$);
- наиболее вероятное значение интервала времени между двумя последовательно сгенерированными транзактами ($p_4 =$ beta);

• параметр p_7 — величина, также зависимая от типа функции распределения (float):

- максимально возможное значение интервала времени между двумя последовательно сгенерированными транзактами ($p_4 =$ beta);

- значение zero (p_4 = norm, unif, expo, erln, none);

• параметр p_8 — номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

Координатор сети процессов. Координатор network (p_1, p_2) осуществляет диспетчеризацию транзактов в узлах (процессах) модели, планирует события в едином модельном времени и активизирует дискретные или непрерывные компоненты модели, имитирующие внешнюю среду. Координатор выполняет всю модель либо до первой ошибки, либо пока не истечет время, заданное аргументом p_3 в функции modbeg.

Относительно синтаксиса языка программирования эта функция является структурным оператором, за которым следует фигурная скобка {, открывающая блок описания стохастической сети. Этот блок заканчивается закрывающей скобкой }. Обращение к этой функции делается только после того, как выполнены оператор modbeg и все операторы ag.

Аргументы p_1 и p_2 — имена (адреса) соответствующих программных функций моделирования внешней среды, производящих интегрирование, решение разностных уравнений, вычисление по формулам и другие математические действия. Эти функции возвращают значение типа float. Координатор передает им единственный аргумент: float d — интервал времени, в течение которого они получают управление. Понятие внешней среды может относиться не только к области экономики, например:

• в экологии p_1 (d) и p_2 (d) могут быть произвольными функциями моделирования процессов переноса загрязняющих веществ в природной среде (в атмосфере — p_1 (d), а в воде — p_2(d));

• в химии или энергетике p_1 (d) — уравнения, описывающие процессы, происходящие в реакторе, причем дискретные компоненты модели реактора представляют собой систему управления, включающую управляющую ЭВМ, систему опроса датчиков и другие компоненты (в данном случае функция p_2 (d) не нужна);

• при моделировании вычислительной системы дискретные компоненты модели — это очереди в операционной системе, процессор, канал, диски и др., а функции p_1 и p_2 не нужны (если только эта вычислительная система не управляет, например, процессом плавки в доменной печи).

Аргумент d в функциях p_1 и p_2 — это глобальная переменная, содержащая интервал времени между двумя ближайшими событиями. Разработчик модели лишен возможности самостоятельно (т. е. напрямую из модели) вызывать p_1 и p_2 во избежание непредсказуемых или ложных результатов, а переменная d может изменяться только координатором network.

Функции float p_1 (d) и float p_2 (d), если они необходимы, пишутся пользователем. Они должны производить интегрирование, либо вычисления по формулам, либо решение разностных уравнений на отрезке времени d, который каждый раз передается в качестве параметра типа float. В общем случае, кроме C++, для этого может использоваться язык Паскаль (однако на C++ вычисления выполняются быстрее) или пакет математических программ.

Выбор очередного узла, в который необходимо «продвинуть» транзакт, осуществляется с помощью вызова network (p_1, p_2), записываемого только один раз перед блоком описания узлов стохастической сети. Если процессы p_1 (d) или p_2 (d) не моделируются, то в качестве p_1 и p_2 необходимо соответственно задать значение dummy.

Оператор завершения моделирования. Оператор завершения модели modend (p_1, p_2, p_3, p_4) должен выполняться после того, как будет нарушено внутреннее условие noerr, определяющее, что в модели либо зафиксирована грубая ошибка, либо истекло время моделирования, ука-

занное в операторе modbeg. Он удаляет все созданные в процессе моделирования управляющие структуры из памяти ЭВМ — как образованные заранее с помощью функции modbeg блоки kcb, так и блоки типа ecb и tcb из модели или цепочек свободных блоков, образуемых после их логического уничтожения в процессе моделирования. Далее этот оператор позволяет просмотреть на экране монитора графические результаты и выводит итоговые табличные результаты в файл-отчет на жестком диске. Значения аргументов функции modbeg следующие:

- параметр p_1 — символическое имя специального файла-отчета, в который записываются стандартные характеристики модели, полученные и замеренные в процессе ее выполнения. Это имя (char) является строкой и оформляется по системным соглашениям (например, имя Results1.doc);
- параметр p_2 — номер первой страницы отчета (int);
- параметр p_3 — число строк (int) на каждой странице;
- параметр p_4 — имеет два значения (int): либо page, если в файле-отчете необходимо проставить символ перевода страницы, либо none.

Общие функции управления узлами, транзактами и событиями в модели

Описание узлов графа, условий прохождения транзактов и моделирование дискретных компонент производится с помощью независимых программных ветвей, активностью которых управляет координатор network. Каждый узел имеет следующую типовую структуру, показанную на рис. 4. Узел состоит из 6 типовых компонентов; необязательные компоненты заключены в квадратные скобки.

С точки зрения топологии выделяют два типа структурных схем моделей: разомкнутые и замкнутые. Примеры таких моделей рассмотрены в книге [1]. Разомкнутая модель «Эффективность компьютеров в АРМ бухгалтерии» содержит ветвление с использованием параметров транзактов, а простая, но эффективная замкнутая модель «Минимизация затрат производства» демонстрирует схему зарядки модели.

Описание узла начинается с особой «метки» — функции top (i), где i — номер этого узла, а заканчивается оператором place. После top (i) ставится двоеточие. Начиная с метки top (i) транзакт проходит все операторы узла i без задержек (в смысле модельного времени) и попадает в оператор place, где находится до тех пор, пока не появятся условия для перехода в следующий узел.

В некоторых узлах одновременно может находиться несколько транзактов. Все они содержатся в одном и том же операторе place. Поэтому не следует путать поток транзактов в модели с потоком управлений в обычной программе. Задержка транзакта в place — это и есть время пребывания в узле.

Операторы анализа условий. После метки top (i) можно анализировать условия продвижения транзактов по графу модели при этом используются операторы if или switch. Для изменения направления путей транзактов, законов распределения, значения времени обслуживания и других параметров можно использовать операцию присваивания. Например, имеются локальные переменные int a, b. Если в узле 3 мы хотим изменить значение переменной b в зависимости от значения a, то должны записать:

```
top (3): if (a > 0)
b =4;
else
b =3*a+5;
key («Переключатель», b);
place;
```

где key — оператор, определяющий узел типа «клапан».

При анализе условий запрещено использовать рекурсию типа x = f (x), x = x+1 и т. п., а также нежелательно выполнять

ресурсоемкие вычисления, требующие существенных затрат времени процессора ЭВМ.

Узловой оператор. Основной обязательный оператор после двоеточия метки-функции top (i) — это оператор определения типа узла (далее — узловой оператор). Этот оператор имеет условное наименование, совпадающее с типом узла. Причем могут использоваться все типы узлов, кроме ag и parent. Функции queue, serv, term, creat, delet, key, dynam, proc, send, direct, attach, manage, pay, rent и down являются узловыми операторами. В описании каждого узла должен быть только один вызов любой из этих функций.

Сигнальные управляющие функции и блок операторов C++ используются не во всех узлах и будут рассмотрены ниже.

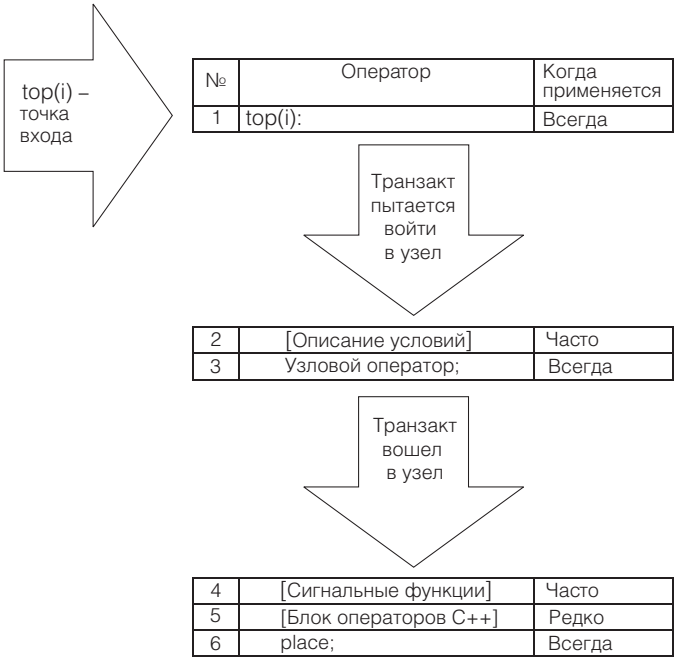

Рис. 4. Типовая структура описания узла

Оператор place — последний оператор узла. Бывает, что описание узла состоит только из метки-функции top (i), узлового оператора и оператора place. Например:

top (i): key («Переключатель», j);
place;

В конце блока описания стохастической сети полезно записать функцию fault (err), которая будет активирована, если из-за невнимательности пользователя произойдет попытка перехода транзакта в несуществующий узел. После такой попытки моделирование прекратится и будет завершено с кодом возврата err, где err — произвольно задаваемое пользователем число, обычно в пределах от 101 до 32 767.

Типы узлов модели

Полный перечень основных процессов, образующих узлы модели, представлен в табл. 1.

Рассмотрим их более подробно:

1. Очередь (с приоритетами или без приоритетов). Функция queue (p_1, p_2, p_3) определяет узел, моделирующий очередь транзактов. Эта очередь строится по одному из двух правил: либо транзакты упорядочены в порядке поступления, либо вновь поступающие транзакты помещаются в конец своей приоритетной группы (более приоритетные транзакты находятся ближе к началу очереди, а менее приоритетные — к концу). Чем больше численное значение приоритета транзакта, тем он приоритетнее. Аргументы (параметры) имеют следующие значения:

- параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);
- параметр p_2 — тип организации очереди (int): либо p_2 = prty, если очередь с приоритетами, либо p_2 = none, если очередь без приоритетов;
- параметр p_3 — номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

График динамики изменения очереди получается автоматически, если в modbeg подставить в качестве параметра p_6 номер контролируемой очереди.

Таблица 1

Перечень основных узлов в моделях *Pilgrim*

№	Узел	Функциональное назначение
		Общего назначения
1	queue	Очередь (с приоритетами или без приоритетов)
2	serv	Узел обслуживания с несколькими параллельными каналами
3	term	Терминатор, убирающий транзакты из модели
4	creat	Траназактно-управляемый генератор
5	delete	Траназактно-управляемый терминатор
6	key	Управляемый клапан на пути транзактов
7	proc	Транзактно-управляемый процесс
8	dynam	Очередь с пространственно-зависимыми приоритетами
		Управление материальными и денежными ресурсами
9	attach	Функция запроса ресурсов со склада
10	manage	Функция имитации менеджера ресурсов
11	send	Функция имитации бухгалтерской проводки
12	direct	Имитация работы бухгалтера
		Структурный анализ: управление переходами между слоями модели при многоуровневой декомпозиции
13	parent	Виртуальный структурный узел
14	pay	Функция имитации перечисления денежной суммы
15	rent	Функция имитации получения ресурса со склада
16	down	Функция перехода на нижерасположенный слой

2. Узел обслуживания с несколькими параллельными каналами. Моделирующая функция serv (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) описывает узел, осуществляющий какое-либо обслуживание транзактов в течение модельного времени, отличного от нуля. В данном случае узел представляет собой одно- или многоканальный обслуживающий прибор, работающий либо по правилам абсолютных приоритетов, либо без них и имеющий стек для прерванных транзактов (правило относительных приоритетов реализуется в узле типа queue — очередь). Аргументы, используемые в функции serv, имеют следующий смысл:

- параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);
- параметр p_2 — число обслуживающих каналов (int), причем $1 \le p_1 \le 32\,767$;
- параметр p_3 — дисциплина обслуживания:
  - если p_3 = abs, то используется приоритетная дисциплина, с прерыванием обслуживания менее приоритетного транзакта более приоритетным (будет рассмотрена ниже);
  - если p_3 = none, то используется бесприоритетная дисциплина;
- параметр p_4 — тип функции распределения интервала времени обслуживания транзакта в канале узла serv (int), имеющий значения: norm — нормальное распределение; unif — равномерное распределение; expo — экспоненциальное распределение; erln — обобщенное рас-

пределение Эрланга; beta — треугольное распределение; none — если интервал обслуживания является детерминированной величиной;

• параметр p_5 — величина, зависящая от типа функции распределения (float):
  - при p_4 = norm, unif, expo — математическое ожидание интервала времени обслуживания транзакта;
  - при p_4 = erln — математическое ожидание одного слагаемого этого интервала;
  - при p_4 = beta — минимальное значение интервала;
  - при p_4 = none — постоянная величина этого интервала;

• параметр p_6 — величина, зависящая от типа функции распределения (float):
  - при p_4 = norm — среднеквадратичное отклонение времени обслуживания;
  - при p_4 = unif — максимальное отклонение от среднего времени обслуживания;
  - при p_4 = expo, none — значение zero;
  - если p_4 = erln, то $p_6 < 0$ — число слагаемых, распределенных по экспоненциальному закону и входящих в случайный интервал обслуживания;
  - при p_4 = beta — наиболее вероятное значение интервала времени обслуживания транзакта;

• параметр p_7 — величина, также зависящая от типа функции распределения (float):
  - p_4 = beta — максимально возможное значение интервала времени обслуживания транзакта;
  - p_4 = norm, unif, expo, erln, none — значение zero;

• параметр p_8 — номер узла (int), в который передается обслуженный транзакт (узел-приемник).

Рассмотрим подробнее прерывания обслуживания неприоритетных транзактов. Если задать p_3 = abs, то имеются два варианта работы с прерванными неприоритетными транзактами:

• дообслуживание после ухода приоритетного транзакта, причем сохраняется чистое заданное время обслуживания, без учета времени нахождения прерванного транзакта в специальном стеке имитатора;

• обслуживание заново, из-за характера прерывания обслуживания транзакта.

Повторный ввод информации при отсутствии контрольных точек. Оператор ЭВМ в течение нескольких часов вводит большой массив информации, причем он не позаботился о периодическом сохранении информации в памяти на жестком диске HDD. Если произойдет случайный сбой или скачок напряжения, то информация, расположенная в оперативной памяти RAM, пропадет. В результате оператор будет вынужден возобновить многочасовую работу заново.

Однако если перед функцией serv транзакт пройдет через оператор присваивания типа t→ga = again, то он при прохождении через serv получает признак обслуживания заново. После выхода из узла serv этот признак теряется.

Следует отметить, что среднее время обслуживания транзактов в узле serv и среднее время задержки в узле — это разные времена. При прерываниях обслуживания транзакта (с его дообслуживанием) чистое время его обработки будет меньше времени задержки в узле на длительность обслуживания приоритетных транзактов.

3. Терминатор, убирающий транзакты из модели. Функция term (p_1) описывает узел-терминатор, который удаляет из модели входящий в него транзакт и фиксирует время его существования начиная с момента выхода этого транзакта из генератора. Единственный параметр p_1 — это символическое имя узла: строка длиной до 14 символов, включая пробелы (char).

Существуют средства автоматического построения графика изменения пото-

ка транзактов, поступающих на вход терминатора. График динамики потока таких транзактов получается автоматически, если номер этого терминатора задан параметром p_8 в функции modbeg. При этом время жизни каждого транзакта измеряется — с момента его генерации и до момента уничтожения.

Если пользователю нужен инструмент для анализа динамики потока транзактов по какой-то ветви графа, когда принимающий узел не является терминатором, то необходимо дополнить модель двумя узлами:

- дополнительным терминатором с указанием его номера в качестве параметра p_8 в функции modbeg;
- в ветвь перед исследуемым узлом вставить узел creat (рассмотрен ниже), который в момент прохождения каждого транзакта по ветви будет генерировать один дополнительный транзакт, направляемый в дополнительный терминатор, а основной транзакт будет входить в узел, на входе которого проводятся измерения.

4. Транзактно-управляемый генератор. Функция creat (p_1, p_2, p_3, p_4, p_5, p_6) предназначена для создания нового семейства транзактов. Следует отметить, что все транзакты принадлежат какому-то семейству. Транзакты, выходящие из обычного генератора ag, принадлежат к семейству с номером 0. Узел creat в отличие от ag — это управляемый генератор. Его назначение самое разное. В замкнутых моделях он применяется для схемы зарядки. Аргументы этого оператора следующие:

- параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);
- параметр p_2 — идентификатор (int) порождаемого семейства транзактов;
- параметр p_3 — число порождаемых транзактов (int);
- параметр p_4 — имеет следующие значения (int):

  либо copy — для тиражирования параметров порождающего транзакта каждому порожденному (включая время жизни),

  либо none — для присвоения каждому порожденному транзакту в качестве параметров нулевых значений;
- параметр p_5 — номер узла (int), в который направляются порожденные транзакты.
- параметр p_6 — номер узла (int), в который направляется порождающий транзакт.

Логика функционирования узла creat такова:

1). через узел проходит порождающий транзакт, который принадлежит семейству f_1, и поступает в узел p_6;

2). одновременно с этим в узле генерируются p_3 новых транзактов, принадлежащих семейству с номером $f_2 = p_2$, которые будут направлены в узел p_5. В общем случае p_5 и p_6 — это любые узлы (кроме ag), в частности, это может быть один и тот же узел. Номера семейств f_1 и f_2 в общем случае могут совпадать;

3). после прохождения порождающего транзакта узел creat получает его координаты, т. е. перемещается;

4). семейству f_2 могут передаваться или не передаваться все свойства (параметры) порождающего транзакта.

Для решения проблем, связанных с регулированием численности семейств транзактов или группировки нескольких транзактов в один, используется узел delet, функционально противоположный узлу creat.

5. Транзактно-управляемый терминатор. Узловой оператор delet (p_1, p_2, p_3, p_4, p_5, p_6) предназначен для уничтожения группы транзактов семейств, номера которых относятся к диапазону, задаваемому параметрами p_2 и p_3. В отличие от терминатора term он управляется специальным транзактом, который называется поглощающим. Если, например, обозначить номер семейства транзакта как Number, то транзакт будет уничтожен при входе в узел delet при выполнении двух условий:

• в узел зашел поглощающий транзакт, принадлежащий семейству p_4;

• выполнено соотношение $p_2 \le Number \le p_3$.

Рассмотрим параметры узла:

• параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — начало (int) диапазона номеров семейств уничтожаемых транзактов;

• параметр p_3 — конец (int) диапазона номеров семейств уничтожаемых транзактов;

• параметр p_4 — идентификатор (int) семейства, к которому принадлежит поглощающий транзакт;

• параметр p_5 — число уничтожаемых транзактов семейства p_2 (int);

• параметр p_6 — номер узла (int), в который направляется уничтожающий транзакт.

Принцип функционирования этого узла заключается в том, что в узел входит транзакт семейства p_4 и находится там до тех пор, пока в него не поступят p_5 транзактов семейства $p_2 \le Number \le p_3$, которые он должен мгновенно уничтожить (поглотить). Время существования этих транзактов фиксируется в узле delet. Узел получает координаты каждого уничтожаемого транзакта, т. е. он перемещается по координатной сетке. В общем случае номера семейств p_1 и p_2 могут совпадать с номером p_3 (с методической точки зрения такие совпадения нежелательны).

Среднее время задержки в данном случае — это среднее время ожидания всех уничтожаемых транзактов или части таких транзактов, если из состояния ожидания узел delet выводится принудительно с помощью функции freed. Обслуживание транзактов в таком узле — это их уничтожение.

Отметим, что если в такой узел так и не поступят p_5 транзактов, то уничтожающий транзакт будет все время находиться в нем, блокируя поступление в него других уничтожающих транзактов. Поэтому существует функция freed (i) для *изгнания* уничтожающего транзакта из узла delet. Эта функция является сигнальной. Она будет рассмотрена ниже.

В данном случае имеется возможность автоматического подсчета:

1) среднего времени жизни уничтожаемых транзактов (или части транзактов, если из состояния ожидания узел delet выводится принудительно — выполнением функции freed);

2) числа уничтоженных транзактов.

6. Управляемый клапан на пути транзактов. Функция key (p_1, p_2) описывает прохождение транзакта через некий клапан. Когда клапан закрыт, транзакт не может в него войти из другого узла. Если же клапан открыт, то транзакт проходит через него в узел *n* без задержки. Среднее время пребывания такого узла в закрытом состоянии подсчитывается автоматически. Для управления этим клапаном или ключом существуют вспомогательные функции hold и rels.

Среднее время задержки — это среднее время пребывания ключа в закрытом состоянии. Число обслуженных транзактов — это число переключений ключа из закрытого состояния в открытое.

Рассмотрим параметры функции key:

• параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

7. Транзактно-управляемый процесс. Специальная суперфункция моделирования транзактно-управляемого непрерывного процесса proc (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8), которая объединяет в себе следующие возможности:

1) обслуживание в узле подобно serv с одним каналом;

2) перемещение узла по общему полю данных на координатной сетке;

3) запуск на время активности функции типа float p_2 (*d*), где d — элементарный интервал активности (float). Этот интервал определяется системой *Pilgrim* в процессе моделирования и зависит от параметров p_3, p_4, p_5, p_6.

Интервал требуемого обслуживания транзакта может быть меньше времени пребывания транзакта в этом узле. Связано это с тем, что процесс может быть переведен в состояние «пассивен» или «активен» каким-либо транзактом из другого узла с помощью функций activ и passiv.

Если процесс пассивен, то обслуживание транзакта приостанавливается, а выполнение функции p_2 прерывается до тех пор, пока процесс не будет переведен в активное состояние.

Аргументы данной функции имеют следующий смысл:

- параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);
- параметр p_2 — имя (адрес) программы, написанной пользователем, или значение «dummy», если такой программы нет. Эта программа может моделировать процесс с помощью формул, дифференциального уравнения и т. п.;
- параметр p_3 — номер исходной точки, в которую устанавливается узел proc перед началом моделирования (int);
- параметр p_4 — это либо тип функции распределения интервала активности процесса, либо возможность работы с координатным пространством (int). Вид функции распределения задается следующим образом:

  norm — нормальное распределение,

  unif — равномерное,

  expo — экспоненциальное,

  erln — обобщенное распределение Эрланга,

  beta — треугольное распределение,

  none — интервал, являющийся детерминированной величиной.

  Возможность работы с пространством задается одним из следующих четырех способов:

  earth — поверхность Земли (географические координаты широта и долгота),

  plane — декартова плоскость (прямоугольная система координат),

  cosmos — произвольное пространство,

  none — если нет распределений или режим пространственной имитации в данном узле не нужен;
- параметр p_5 может характеризоваться следующими величинами:
  - при p_4 = norm, unif, expo — математическое ожидание интервала времени (float) активности,
  - при p_4 = erln — математическое ожидание одного слагаемого этого интервала,
  - при p_4 = beta — минимальное значение интервала обслуживания,
  - при p_4 = none — постоянная величина этого интервала,
  - если p_4 = earth, plane или cosmos — интервал «непрерывного» нахождения этого узла в точке на координатной сетке;
- параметр p_6 — параметр интервала активности обслуживания (float), который зависит от значения параметра p_4:
  - при p_4 = norm — среднеквадратичное отклонение,
  - при p_4 = unif — максимальное отклонение от среднего,
  - при p_4 = expo, none значение zero,
  - если p_4 = erln, то $p_6 > 0$ — число слагаемых интервала обслуживания,
  - при p_4 = beta — наиболее вероятное значение интервала времени обслуживания транзактов,
  - если p_4 = earth, plane или cosmos — скорость перемещения узла от одной точки пространства к другой;
- параметр p_7 — величина, также зависящая от типа функции распределения (float) интервала активности, либо значе-

ние zero (p_4 = norm, unif, expo, erln, none), либо максимальное значение интервала времени обслуживания (p_4 = beta);

• параметр p_8 — номер узла (int), в который передается сгенерированный транзакт (узел-приемник).

Приведенный выше набор свойств и параметров узла типа proc делает эту суперфункцию чрезвычайно удобной для имитации таких реальных процессов, как:

• запуск процесса (реакции) в реакторе и управление активностью этого процесса;

• облет на вертолете местности, подвергшейся какому-либо бедствию;

• перемещение позиционера, доступ и чтение информации с накопителя на магнитном диске в вычислительной системе;

• оперативное управление вертолетами «скорой помощи» с учетом динамически меняющейся ситуации.

Значения параметра p_4 обеспечивают дополнительные возможности имитации поведения узла proc в пространстве.

Перед входом в узел очередного транзакта координаты узла — это координаты какой-либо точки A пространства, номер которой находится в параметре этого узла k→kx.

Транзакт, который входит в узел, имеет другие координаты — координаты точки B (номер которой помещен в параметр транзакта t→tx). После выхода транзакта из узла автоматически вычисляется длина *l* отрезка AB, которая автоматически прибавляется к суммарному значению пути, пройденного этим узлом. Время обслуживания транзакта определяется в этом случае по формуле

$$t_{обсл} = 1/p_6 + p_5.$$

Значения и смысл параметров p_5 и p_6 можно пояснить на следующем примере.

Допустим, что имеется вертолет «скорой помощи», который вылетает по заявкам-радиограммам в разные населенные пункты. Во время полета на борт вертолета могут поступать новые радиограммы. Здесь радиограммы из населенных пунктов — это транзакты, а вертолет — узел proc. Время полета из одного пункта в другой $t_{полета} = 1/p_6$, а время посещения больного равно p_5. При управлении вертолетом можно оптимизировать обслуживание транзактов (уже полученных радиограмм) с помощью оптимизации маршрута и, более того, можно прервать полет к одному пункту для посещения другого, если это приведет к снижению суммарного пути.

Другими словами, полет, длительность которого равна $t_{полета}$, подвержен возможному прерыванию, а обследование больного (время p_5) — это непрерываемое нахождение в конкретной точке на координатной сетке. Здесь значения p_5 и p_6 представляют собой конкретные числа. Поэтому, если необходимо все-таки подставить случайные значения, то можно использовать датчики псевдослучайных величин. Работа с ними рассматривается в [1] в разд. 4.1.

Отметим, что в вырожденном случае узел proc эквивалентен узлу serv. Например, функции serv («Обслуживание»,1, none, expo,10.0, zero, zero, n) и proc («Обслуживание», dummy, none, expo,10.0, zero, zero, n) эквивалентны.

Они обеспечивают моделирование одноканальной системы массового обслуживания без приоритетов с временем обслуживания, распределенным по экспоненциальному закону с математическим ожиданием 10 единиц.

8. Очередь с пространственно-зависимыми приоритетами. Функция dynam (p_1, p_2) предназначена для моделирования обслуживания транзактов в очереди с динамическими пространственно-зависимыми приоритетами. Эта функция моделирует оптимально-управляемую очередь (очередь типа «скорая помощь»), которая находится на входе узла типа proc. Узел имеет два параметра:

• параметр p_1 — символическое имя узла: строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — номер узла (int) типа proc, в который передается сгенерированный транзакт (узел-приемник); proc обязательно должен использовать свой параметр p_4 = earth, plane или cosmos.

Задача оптимального расписания для обслуживания транзактов с пространственно-зависимыми приоритетами может возникнуть, например, при моделировании следующих сложных процессов и объектов:

• участка гибкого автоматизированного производства (ГАП) с роботизированными тележками, «путешествующими» по цеху;

• местности, подверженной какому-либо бедствию, в процессе ее обследования специальной командой на вертолете и др.

Рассмотрим снова процесс оперативного планирования маршрута вертолета «скорой помощи» (см. пример из п. 7). Функция proc — это вертолет, радиограммы — транзакты, имеющие координаты населенных пунктов, а функция dynam — оптимизатор курса. В модели же оператор dynam — это очередь, обслуживающая входящие транзакты по правилу динамических пространственно-зависимых приоритетов. Порядок обслуживания транзактов в этой очереди пересматривается каждый раз при поступлении в хвост очереди нового транзакта и при переходе первого транзакта из головы очереди в узел proc.

Функция dynam всегда «заглядывает» в узел proc и анализирует, не следует ли прервать обслуживание находящегося в нем транзакта. Если такое решение принято, то вычисляется местонахождение текущей точки пространства, в которой в данный момент находится узел proc. Функция dynam извлекает из этого узла транзакт обратно в свою очередь и посылает в него более выгодный относительно оптимизации транзакт.

Оригинальный алгоритм оптимизации динамического расписания — правила построения динамической очереди dynam — описан в [1] в гл. 2. Он обладает высоким быстродействием и имеет практическое применение.

Моделирование материальных и денежных ресурсов учитывает подобие их основных характеристик: остаток ресурса похож на положительное сальдо, дефицит подобен отрицательному сальдо. Есть и другие аналогии (кроме перечисления денег и бухгалтерских проводок).

9. Функция запроса ресурсов со склада. Каждый склад ресурсов описывается в имитационной модели в виде узла типа attach. В этом узле образуется очередь транзактов, которая может быть организована по приоритетному принципу: чем меньше транзакт запрашивает единиц, тем более он приоритетен. Соответствующая функция имеет вид: attach (p_1, p_2, p_3, p_4). Эта функция включает 4 параметра:

• параметр p_1 — символическое имя узла-ресурса. Это строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — требуемое число элементов ресурса (long);

• параметр p_3 — работа с приоритетами: prty или none. Если указано prty, то требования на ресурс в случае отсутствия необходимого числа элементов образуют очередь в узле attach, причем соответствующие транзакты располагаются в порядке убывания значения приоритета (ближе к голове очереди находится самая приоритетная группа транзактов). Внутри приоритетной группы транзакты расположены в следующем порядке: чем меньше элементов необходимо транзакту, тем ближе транзакт находится к голове своей приоритетной группы. Если же требования на число элементов одинаковы, то транзакты расположены в хронологическом порядке (правило FIFO): чем раньше транзакт пришел в очередь, тем раньше он будет обслужен. Когда указано значение none, работает только правило FIFO;

• параметр p_4 — номер узла-приемника (int). Таким узлом может быть только узел-менеджер (manage).

10. Функция имитации менеджера ресурсов. Обслуживанием транзактов занимается узел типа «менеджер» (manage). Обслуженный транзакт проходит узел manage и перемещается с захваченными единицами по графу модели. Транзакт может несколько раз становиться в очередь к одному и тому же ресурсу, получая дополнительные единицы. Данная функция имеет вид manage (p_1, p_2) со следующими значениями параметров:

• параметр p_1 — символическое имя узла-менеджера: строка длиной до 14 символов, учитывая пробелы (char);

• параметр p_2 — номер узла-приемника (int). Таким узлом может быть любой узел модели, кроме manage.

11. Функция имитации бухгалтерской проводки. Основные объекты системы *Pilgrim* (узел, транзакт, событие) хорошо подходят для описания финансовой динамики на счетах бухгалтерского учета предприятия. Узлом считается счет (субсчет) бухгалтерского учета. Предположим, что номер этого узла i, тогда транзакт, вошедший в узел i, — это запрос на проводку со счета i определенной суммы на какой-то другой счет. Для осуществления проводки необходимо, чтобы на счете i, т. е. в узле i, была сумма не менее требуемой. При отсутствии такой суммы транзакт ожидает момента поступления на счет i достаточных средств. Другими словами, узел с номером i, который формирует запрос на бухгалтерскую проводку, — это специальная очередь транзактов.

Функция узла-счета i имеет вид send (p_1, p_2, p_3, p_4, p_5), а ее параметры имеют следующий смысл:

• параметр p_1 — символическое имя узла-ресурса: строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — узел-счет, на который необходимо перевести заданную сумму (int);

• параметр p_3 — размер заданной суммы (double), единица измерения которой может быть любой (рубли, доллары и т. д.). После точки необходимо указывать один или два разряда — доли используемых единиц измерений. Например, сумма 1 млн руб. будет иметь вид 1000000.00;

• параметр p_4 — возможность работы с приоритетами. Может принимать значения prty или none. Если указано prty, то требования на перечисление денег со счета i в случае отсутствия необходимой суммы образуют очередь в узле send. При этом соответствующие транзакты располагаются в порядке убывания значения приоритета (ближе к голове очереди находится самая приоритетная группа транзактов). Внутри приоритетной группы транзакты расположены в следующем порядке: чем меньше требуемая сумма, тем ближе транзакт находится к голове своей приоритетной группы. Транзакты с одинаковыми суммами расположены в хронологическом порядке (правило FIFO): чем раньше транзакт пришел в очередь, тем раньше он обслужен. Когда указано значение none, работает только правило FIFO;

• параметр p_5 — номер узла типа «финансовый директор» (узел direct), который осуществляет финансовый менеджмент и выполняет проводки по мере необходимости.

Событием в узле типа send является выполнение проводки со счета i на счет p_2. Момент времени такого события — это момент времени проводки, определяемый выводом транзакта из узла send.

В каждом узле типа send имеется внутренний атрибут saldo, который отражает остаток средств на счете i. Дефицит средств на счетах бухгалтерского учета содержится в атрибуте defic. Если атрибут saldo в узле i имеет нулевое значение и в этом узле имеются транзакты (один или несколько), запросившие проводки, то общий дефицит затребованных этими

транзактами сумм автоматически отражается в атрибуте defic.

12. Имитация работы бухгалтера. Обслуживание очередей типа send возможно с помощью одного или нескольких узлов типа «финансовый директор». Описание такого узла имеет вид direct (p_1, p_2). Суть параметров этой функции следующая:

- параметр p_1 — символическое имя узла-ресурса. Это строка длиной до 14 символов, включая пробелы (char);
- параметр p_2 — это узел-приемник транзакта, выполнившего проводку. Этот узел может быть любого типа, кроме direct.

При моделировании бизнес-процесса небольшого предприятия для обслуживания всех узлов типа send достаточно одного узла типа direct. Однако, можно имитировать одновременную работу нескольких бухгалтеров, каждый из которых отвечает за свою группу бухгалтерских операций.

Процесс построения графа имитационной модели сопровождается *структурным анализом исследуемого процесса*. При структурном анализе возникает задача перехода между слоями: нижние слои модели содержат декомпозицию узлов, расположенных выше. Декомпозиция представляет собой детализацию одного узла с помощью совокупности других.

Существуют четыре разновидности декомпозиции процессов:

1) общий случай декомпозиции сложного процесса с помощью узлов типа down;

2) декомпозиция процессов перечисления денег (платежей, бухгалтерских проводок и др.) с помощью узлов типа pay;

3) декомпозиция процессов выделения ресурсов с помощью узлов типа rent;

4) абстрактное объединение группы процессов в один псевдопроцесс с помощью виртуального, т.е. не существующего в реальности, узла parent без образования нового узла.

13. Виртуальный структурный узел. На применении узла parent основано управление переходами между слоями модели при многоуровневой декомпозиции, которое подробно рассмотрено в [1] в гл. 5. Этот узел — атрибут диалогового CASE-конструктора, позволяющего проводить структурный анализ и создавать модели в графическом виде.

14. Функция имитации перечисления денежной суммы. Имитация перечисления денежной суммы с помощью узла pay более наглядна, чем с использованием запутанных цепочек send⇒direct, рассмотренных выше. Однако на более низком уровне узел pay подлежит детализации именно с помощью узлов send и direct. Функция pay имеет следующий вид: pay (p_1, p_2, p_3, p_4, p_5, p_6, p_7). Аргументы данной функции имеют следующий смысл:

- параметр p_1 — символическое имя узла pay: строка длиной до 14 символов, включая пробелы (char);
- параметр p_2 — номер узла-счета типа send, на который переводится денежная сумма (int);
- параметр p_3 — значение денежной суммы (или стоимость). Это переменная типа double;
- параметр p_4 — номер узла-счета типа send, с которого переводится денежная сумма (int);
- параметр p_5 — признак работы с приоритетами (prty или none);
- параметр p_6 — номер узла-приемника на нижнем слое (int);
- параметр p_7 — номер узла возврата на данном слое модели, где расположен узел pay (int).

15. Функция имитации получения ресурса со склада. Имитация получения ресурсов со склада внешне похожа на работу с узлом pay. Это делается с помощью узла rent, который подлежит декомпозиции на более низком уровне с помощью attach и manage. Функция rent имеет следующий вид: rent (p_1, p_2, p_3, p_4, p_5, p_6). Для этой функции существуют шесть аргументов, подобных аргументам узла pay:

• параметр p_1 — символическое имя узла rent: строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — требование на число элементов ресурса (long);

• параметр p_3 — номер узла-склада ресурсов attach, с которого необходимо получить ресурсы (int);

• параметр p_4 — признак работы с приоритетами (prty или none);

• параметр p_5 — номер узла-приемника на нижнем слое модели (int);

• параметр p_6 — номер узла возврата на данном слое, где расположен узел rent (int).

16. Функция перехода на нижерасположенный слой. Иногда бывает полезно большую группу узлов специально объединить в один общий, который находится на одном из слоев модели. Затем этот узел подвергается декомпозиции на расположенных ниже слоях модели. Функция, описывающая такой узел, называется down: down (p_1, p_2, p_3). Эта функция характеризуется следующими аргументами:

• параметр p_1 — символическое имя узла down: строка длиной до 14 символов, включая пробелы (char);

• параметр p_2 — номер узла-приемника на нижнем слое модели (int);

• параметр p_3 — номер узла возврата на данном слое, где расположен узел down (int).

Сигнальные управляющие функции

Сигнальные управляющие функции используются не во всех узлах, а только там, где необходимы специальная логика или функциональные уточнения особенностей узла. Сигнальные функции (см. табл. 2) можно использовать в любом узле, кроме ag.

1. Функция interrupt для прерывания модели. Модель, которая с точки зрения пользователя выглядит отлаженной, на самом деле может таковой не являться. При попытке существенно изменить входные данные и подправить под них текст модели она может потерять быстродействие и логику рабрты. Поэтому, независимо от опыта пользователя, создавшего очень сложную модель, необходимо иметь возможность временной приостановки модели и просмотра промежуточных результатов, после чего возобновить счет. Если результаты окажутся подозрительно неправдоподобными, то следует прекратить моделирование

Таблица 2

Сигнальные управляющие функции

№	Функция	Назначение сигнальной функции
1	interrupt	Выполняет прерывание модели
2	cheg	Изменяет параметры генератора транзактов ag
3	rels	Открывает клапан key
4	hold	Закрывает клапан key
5	activ	Возобновляет активность обслуживания в узле proc
6	passiv	Приостанавливает активность обслуживания в узле proc
7	supply	Присваивает начальное значение мощности ресурса attach
8	assign	Задает начальное значение на счете бухгалтерского учета send
9	freed	Изгоняет уничтожающий транзакт из узла delet
10	sewt	Переносит транзакт в другую точку пространства
11	sewk	Переносит узел в другую точку пространства
12	geoway	Определяет расстояние между точками на поверхности Земли
13	decart	Определяет расстояние на декартовой плоскости
14	change	Заменяет узел обслуживания очереди
15	clcode	Включает блок операторов языка C++

или перейти в режим отладки средствами «трассировки».

Прерывание осуществляется двумя способами:

- в процессе диалога с моделью во время ее выполнения (клавиша Esc);
- программно с помощью функции interrupt, которую можно по какому-либо условию выполнить в любой ветви графа модели. Эта функция не имеет параметров и не возвращает никаких результатов. Она записывается следующим образом: interrupt().

После выполнения этой функции в каком-либо узле работа модели прекращается, а экспериментатор может исследовать полученные результаты.

2. Функция cheg для перенастройки генератора ag. Часто бывает необходимо перенастроить генератор транзактов ag (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8). Номер генератора — это параметр p_2. Функция cheg (p_1, p_2, p_3, p_4, p_5, p_6, p_7) предназначена для изменения параметров генератора ag. Номер перенастраиваемого генератора содержится в параметре p_1. Параметры p_2 — p_7 соответствуют параметрам p_3 — p_8 генератора ag и заменяют их значения на новые. Такая перенастройка произойдет за нулевое модельное время. Смысл аргументов p_3 — p_8 такой же, как и у параметров генератора с номером p_2.

3, 4. Функции rels и hold для управления клапаном key. Узел типа «клапан» управляется из других узлов. Функции rels (i) и hold (i) предназначены для управления клапаном с номером i из любых других узлов.

После выполнения rels клапан принимает состояние «открыт», если до этого он был закрыт. Соответственно после hold клапан перейдет в состояние «закрыт», если до этого он был в открытом состоянии. Во всех других случаях никаких действий не осуществляется. Обычно клапаны помещают на выходах каких-либо очередей, но могут быть и другие варианты их использования.

5, 6. Функции activ и passiv для управления активностью процесса proc. Функция activ (i) переводит процесс (узел типа proc) с номером i в активное состояние, если он был пассивен. При этом возобновляются обслуживание транзакта (т. е. отсчет активного времени) и выполнение модели процесса. Если процесс уже был активен, либо вообще узел i был пустым (нет ни транзакта, ни выполнения модели процесса), то никаких действий не осуществляется. Функция activ в общем случае выполняется в другом узле, номер которого не равен i.

Функция passiv (i) переводит процесс (узел типа proc) в пассивное состояние, если он был активен. При этом транзакт и соответствующее событие исключаются из списка планируемых событий и переводятся в стек прерванных транзактов. Прекращается и выполнение модели процесса. Если процесс уже был пассивен, либо узел i был пустым, то никаких действий не осуществляется.

7. Функция supply для обеспечения начальной мощности ресурса. При работе со складом ресурсов attach в какой-то момент необходимо принудительно установить начальное значение, либо его изменить на заданную величину. Это делается с помощью функции supply (p_1, p_2, p_3), имеющей три параметра:

- параметр p_1 — номер узла attach (int);
- параметр p_2 — признак add (добавить) или none (безусловно установить);
- параметр p_3 — количество единиц ресурса (long).

8. Функция assign для ассигнования на счет денежной суммы. При работе со счетами send в какой-то момент необходимо принудительно установить начальное значение остатка денежных средств, либо его изменить на заданную величину. Это делается с помощью функции, имеющей вид assign (p_1, p_2, p_3). Функция имеет следующие параметры:

- параметр p_1 — номер счета — узла send (int);

• параметр p_2 — признак add (добавить) или none (безусловно установить);

• параметр p_3 — значение денежной суммы (double).

9. Функция freed для изгнания уничтожающего транзакта из узла delet. Рассмотрим узел delet (p_1, p_2, p_3, p_4, p_5, p_6). Если в такой узел не поступят p_5 транзактов, то, как было сказано выше, уничтожающий транзакт будет все время находиться в нем, блокируя его для поступления других уничтожающих транзактов. Поэтому существует задача изгнания уничтожающего транзакта из узла delet. Для изгнания используется оператор freed (i), который работает в соответствии со следующей логикой. Допустим, что в узле delet с номером i «застрял» транзакт. Текущий транзакт, который проходит через функцию freed (i) в каком-то другом узле, с ее помощью генерирует вспомогательный транзакт и мгновенно направляет его в узел i. Этот вспомогательный транзакт выталкивает «застрявший», приводит delet в рабочее состояние, а сам уничтожается. «Застрявший» транзакт будет направлен в узел p_6, определенный в узловом операторе delet.

Факт посещения вспомогательным транзактом фиксируется в узле delet и отражается на статистических результатах, но время его жизни равно нулю.

10, 11. Функции sewt и sewk для привязки транзактов и узлов к точкам пространства. Функция sewt (x) помещает текущий транзакт в точку пространства, имеющую номер x, т. е. приписывает ему координаты этой точки путем занесения значения x в параметр транзакта t→tx. Функция sewk (x, i) помещает узел с номером i в точку пространства, имеющую номер x, т. е. приписывает узлу координаты этой точки, записывая значение x в параметр узла k→kx.

12, 13. Функции geoway и decart для определения расстояний между точками пространства. Функция geoway (latA, lonA, latB, lonB) служит для определения расстояния между точками A и B по их географическим координатам, измеряемым в радианах, где latA и lonA — широта и долгота точки A, а latB и lonB — координаты точки B.

Функция geoway используется имитатором автоматически внутри узла dynam, однако ее также можно вызывать из любой программы пользователя. Метод расчета расстояний, используемый в ней, описан в [1] в гл. 6. Расстояние между двумя пунктами определяется по основным формулам сферической тригонометрии. Радиусы Земли для разных широт вычисляются по эллипсоиду Красовского.

Функция decart (xA, yA, xB, yB) аналогична geoway. Она служит для определения расстояний на декартовой плоскости в прямоугольных координатах по теореме Пифагора. Обозначим координаты точки A как xA и yA, а координаты точки B — xB и yB. В этом случае функция decart возвращает расстояние между точками по прямой:

$$L_{AB} = \sqrt{(xA - xB)^2 + (yA - yB)^2}.$$

14. Функция change для замены узла обслуживания очереди. Данная сигнальная функция чаще всего используется при моделировании клиринговых процессов. Она имеет вид: change (p_1, p_2).

Предположим, что узел-очередь имеет номер p_1. Этот узел относится к одному из трех типов: 1) send — счет; 2) attach — склад ресурсов; 3) queue — очередь. Если состав или количество транзактов в очереди перестают удовлетворять необходимым требованиям, то все транзакты из очереди необходимо направить в узел с номером p_2 и разгрузить эту очередь. Здесь параметр p_1 — номер узла-очереди (int), а параметр p_2 — номер узла, в который необходимо перенаправить транзакты (int).

15. Функция clcode для включения блока операторов языка C++. В блоке описания узлов разработчик модели может помещать свою программу (хотя чаще всего достаточно собственных средств

имитатора). Программные блоки обращения к функциям и системным вызовам такой программы должны быть, написаны на языке C++ или Паскаль. Однако это нужно делать при соблюдении определенных правил.

Во-первых, если необходимо производить сложные или длительные по времени ресурсоемкие вычисления, использовать рекурсии с некоторыми переменными (например, $x = f(x)$ или x++), работать с системными вызовами, обращаться к ресурсоемким вычислительным функциям, то это делается между функцией определения типа узла и соответствующим оператором place, с помощью специального средства clcode.

Во-вторых, нельзя использовать операцию языка C++ типа goto не только внутри одного узла (между меткой-функцией top и соответствующим оператором place), но и для перехода в другой узел. Для принудительного завершения моделирования используют только операцию interrupt. После такой операции управление будет от координатора передано оператору, следующему за блоком описания графа.

После узлового оператора можно записать блок любых операторов языка C++, оформленный в виде:

clcode
{
<Группа операторов языка C++>
}

Если необходимо использовать блок из одного оператора, то это записывается так:

clcode
<Оператор языка C++>

Одно из назначений программных функций, рассмотренных выше, заключается в посылке сигналов (передаче и получении информации) из одних узлов в другие непосредственно, а не с помощью транзактов.

Рассмотренные средства описания структуры имитационной модели, логики ее работы и функциональных особенностей моделируемых процессов представляют собой специализированный язык. В системе *Pilgrim* имеется возможность подключения пользовательских программ на C++ в любой узел, если такая необходимость возникнет.

Узловые функции, реализующие дискретные процессы в имитационных моделях и их взаимодействие, выполняют следующие операции:

- структурную декомпозицию в модели, переходы с одного структурного слоя на другой;
- запуск и координацию процессов управления транзактами, событиями и узлами модели;
- управление материальными, информационными и денежными ресурсами.

При этом учитывается различие пути перемещения транзактов по графу модели и пути перемещения денежных ресурсов (между узлами типа «счет»).

В моделях используется специальный набор сигнальных функций. Эти функции не являются обязательными, однако без них невозможно выполнять следующие действия:

- посылать сигналы, передавать и получать информацию из одних узлов в другие непосредственно, а не с помощью транзактов;
- управлять из одних узлов другими узлами и транзактами;
- защищать модель от «клинча» (явления блокировки), когда в ней нет ошибок с точки зрения экономиста-пользователя, но сложилась случайная ситуация, вызвавшая «зависание» компьютера из-за замыкания нескольких процессов через совместно используемые ресурсы.

Литература

1. *Емельянов А.А.*, *Власова Е.А.*, *Дума Р.В.* Имитационное моделирование экономических процессов. Изд. 2-е. М.: Финансы и статистика, 2006.