ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С. П. КОРОЛЕВА» (САМАРСКИЙ УНИВЕРСИТЕТ)

КОНСТРУИРОВАНИЕ ДИСКРЕТНО-СОБЫТИЙНЫХ МОНИТОРОВ ДЛЯ СИСТЕМ МОДЕЛИРОВАНИЯ

Самара 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С. П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

КОНСТРУИРОВАНИЕ ДИСКРЕТНО-СОБЫТИЙНЫХ МОНИТОРОВ ДЛЯ СИСТЕМ МОДЕЛИРОВАНИЯ

Составитель *Е.В. Симонова*

С А М А Р А
Издательство Самарского университета
2017

УДК 519.876.5

ББК 22.18я73

Составитель ***Е.В. Симонова***

Рецензент: канд. техн. наук, доц. Л.С. З е л е н к о

Конструирование дискретно-событийных мониторов для систем моделирования: [Электронный ресурс]: метод. указания / *Е.В. Симонова.* – Самара: Изд-во Самарского университета, 2017. – 43 с. : ил. Электрон. текстовые и граф. дан. (Кбайт).- 1 эл. опт. диск (CD-ROM)

Методические указания содержат теоретические сведения о модельной динамике систем с дискретными событиями, методах и средствах дискретно-событийного мониторинга, а также задания для выполнения лабораторной работы. Даны рекомендации по реализации управляющих программ для систем моделирования.

Методические указания предназначены для студентов направления 09.03.01 – «Информатика и вычислительная техника» в качестве методических указаний по курсу «Моделирование информационно-вычислительных систем».

Подготовлены на кафедре информационных систем и технологий.

УДК 519.876.5
ББК 22.18я73

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В методических указаниях содержатся сведения о модельной динамике систем с дискретными событиями, а также рекомендации по разработке мониторов моделирования на основе концепций событий, состояний и процессов. Приводятся контрольные вопросы, а также индивидуальные задания для выполнения лабораторной работы.

Методические указания предназначены для студентов, обучающихся по направлению 09.03.01 – Информатика и вычислительная техника.

Содержание методических указаний соответствует разделам рабочей программы по дисциплине «Моделирование информационно-вычислительных систем» федерального компонента ГОС подготовки бакалавров по направлению 09.03.01 – Информатика и вычислительная техника.

ВВЕДЕНИЕ

Цель лабораторной работы – изучение принципов модельной динамики систем с дискретными событиями, методов разработки дискретно-событийных имитационных моделей, получение навыков реализации управляющих программ для систем моделирования на основе концепций событий, состояний и процессов.

1 КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ – ОСНОВНОЙ МЕТОД ИССЛЕДОВАНИЯ СЛОЖНЫХ СИСТЕМ

Возможности моделирования, т.е. перенос результатов, полученных в ходе построения и исследования модели, на оригинал, основаны на том, что модель в определенном смысле отображает (воспроизводит, моделирует, описывает, имитирует) некоторые интересующие исследователя черты объекта. Моделирование как форма отражения действительности широко распространено, и достаточно полная классификация возможных видов моделирования крайне затруднительна, хотя бы в силу многозначности понятия "модель", широко используемого не только в науке и технике, но и в искусстве, и в повседневной жизни. Применительно к естественным и техническим наукам принято различать следующие виды моделирования:

- концептуальное моделирование, при котором совокупность уже известных фактов или представлений относительно исследуемого объекта или системы истолковывается с помощью некоторых специальных знаков, символов, операций над ними или с помощью естественного или искусственного языков;
- физическое моделирование, при котором модель и моделируемый объект представляют собой реальные объекты или процессы единой или различной физической природы, причем между процессами в объекте-оригинале и в модели выполняются некоторые соотношения подобия, вытекающие из схожести физических явлений;
- структурно-функциональное моделирование, при котором моделями являются схемы (блок-схемы), графики, чертежи, диаграммы, таблицы, рисунки, дополненные специальными правилами их объединения и преобразования;
- математическое (логико-математическое) моделирование, при котором моделирование, включая построение модели, осуществляется средствами математики и логики;
- имитационное (программное) моделирование, при котором логико-математическая модель исследуемого объекта представляет собой алгоритм функционирования объекта, реализованный в виде программного комплекса для компьютера.

Разумеется, перечисленные выше виды моделирования не являются взаимоисключающими и могут применяться при исследовании сложных объектов либо одновременно, либо в некоторой комбинации. Кроме того, в некотором смысле концептуальное и, скажем, структурно-функциональное моделирование неразличимы между собой, так как блок-схемы, конечно же, являются специальными знаками с установленными операциями над ними.

Традиционно под моделированием на ЭВМ понималось лишь имитационное моделирование. Можно, однако, увидеть, что и при других видах моделирования компьютер может быть весьма полезен. Например, при мате-

матическом моделировании выполнение одного из основных этапов - построение математических моделей по экспериментальным данным - в настоящее время просто немыслимо без компьютера. В последние годы, благодаря развитию графического интерфейса и графических пакетов, широкое развитие получило компьютерное структурно-функциональное моделирование. Положено начало привлечения компьютера даже к концептуальному моделированию, где он используется, например, при построении систем искусственного интеллекта.

В настоящее время под компьютерной моделью чаще всего понимают:

- условный образ объекта или некоторой системы объектов (или процессов), описанный с помощью взаимосвязанных компьютерных таблиц, блок-схем, диаграмм, графиков, рисунков, анимационных фрагментов, гипертекстов и т. д. и отображающий структуру элементов объекта и взаимосвязи между ними. Компьютерные модели такого вида называют структурно-функциональными;
- отдельную программу, совокупность программ, программный комплекс, позволяющий с помощью последовательности вычислений и графического отображения их результатов воспроизводить (имитировать) процессы функционирования объекта, системы объектов при условии воздействия на объект различных, как правило, случайных, факторов. Такие модели мы будем далее называть **имитационными.**

Компьютерное моделирование - метод решения задачи анализа или синтеза сложной системы на основе использования ее компьютерной модели. Суть компьютерного моделирования заключена в получении количественных и качественных результатов по имеющейся модели. Качественные выводы, получаемые по результатам анализа, позволяют обнаружить неизвестные ранее свойства сложной системы: ее структуру, динамику развития, устойчивость, целостность и др. Количественные выводы в основном носят характер прогноза некоторых будущих или объяснения прошлых значений переменных, характеризирующих систему.

В дальнейшем сосредоточимся на рассмотрении имитационных моделей.

Имитационное моделирование основано на применении логико-математической модели сложной системы. Построение математической модели, в отличие от структурно-функционального моделирования, требует большого объема детальной информации о системе, включая всевозможные логические и количественные соотношения. Выбор математического аппарата существенно сказывается на самой имитационной модели и на выборе инструментальных средств. Ясно, что использование излишне сложного математического аппарата или привлечение большого числа методов из различных разделов математики значительно усложнит задачу имитационного моделирования. При построении логико-математической модели всегда приходится решать проблему выбора между сложностью модели и ее точностью, удобством использования и универсальностью, - поскольку эти

критерии, как правило, противоречивы. Именно поэтому составление логико-математической модели и использование ее для имитационного моделирования было, есть и будет искусством [1].

Имитационная модель (ИМ)—совокупность описания статических свойств системы, а также динамики изменения состояния и взаимосвязей элементов системы во времени под влиянием факторов внешней среды и внутренней логики функционирования системы. ИМ задаётся совокупностью моделирующих алгоритмов, воспроизводящих (копирующих) с заданной степенью детализации развитие физических процессов, происходящих в исследуемой системе, т.е. задающих модель поведения.

Структура имитационной модели

В общем случае объектом-оригиналом может быть любая естественная или искусственная система, предназначенная для выполнения определённого вида работ или решения определённого класса задач. Она состоит из компонентов C_0, называемых также элементами или подсистемами, имеет некоторое множество параметров P_0 и характеризуется определёнными свойствами S_0. Система проявляет эти свойства под влиянием внешних воздействий X_0 и в результате физических процессов A_0, определяющих принципы её функционирования. Количественной мерой свойств системы служит множество характеристик Y_0. Элементы множества характеристик $Y_{0i} \in Y_0$ являются частными показателями эффективности системы. Показатель эффективности—это числовая характеристика системы, которая оценивает степень приспособленности системы к выполнению поставленных перед ней задач, т.е. качество её функционирования. Структуру любой ИМ-модели можно представить:

$ИМ = < C_M, H_M, X_M, Y_M, S_M, F_M, A_M, O_M, E_M, T_M >$

$C_M = \{ C_{M\,r} \}$, $r = 1..,n_{CM}$ — множество компонентов модели;

$H_M = \{ h_{M\,j} \}$, $j = 1..,n_{HM}$ — множество параметров (неуправляемых переменных);

$X_M = \{ X_{Mk} \}$, $k = 1..,n_{XM}$ — множество входных переменных (внешних воздействий);

$S_M = \{ S_{M\,l} \}$, $l = 1..,n_{SM}$ — множество переменных состояния, возникающих в результате воздействия внутренних причин (управляемые переменные);

$Y_M = \{ Y_{M\,i} \}$, $i = 1..,n_{YM}$ — множество выходных переменных;

$F_M = \{ F_{M\,z} \}$, $z = 1..,n_{FM}$ — функциональные зависимости между выходными воздействиями и переменными состояния, а также между выходными и входными воздействиями;

$A_M = \{ A_{M\,d} \}$, $d = 1..,n_{AM}$ — множество алгоритмов, задающих модель поведения;

T_M — модельное время, т.е. время, в течение которого на модель оказываются внешние воздействия $\{ X_{Mk} \} \subset X_M$ и измеряются характеристики $\{Y_{Mi}\} \subset Y_M$;

$O_M = \{ O_{M\,q} \}$, $q = 1..,n_{OM}$ — ограничения - устанавливаемые пределы изменения управляемых переменных.

$E_M = E_M(Y_M) \rightarrow opt$ — целевая функция (функция критерия)

Замещение объекта-оригинала моделью возможно, если выполняется:

$x_{OK}=\omega(x_{MK})$ $\qquad$ $s_{0L}=\psi(s_{ML})$

$H_{0j}=\xi(h_{Mj})$ $\qquad$ $T_0=mT_M$,

m - масштабный коэффициент

Тогда на определённом временном интервале для оригинала и модели характерны следующие зависимости показателя эффективности от значений параметров системы, характера воздействия внешней среды, значений переменных состояния.

$$y_{0i} = f_{OZ}(\{h_{oj}\}, \{x_{ok}\}, \{s_{ol}\}, T_0)$$

$$y_{Mi} = f_{MZ}(\{h_{Mj}\}, \{x_{Mk}\}, \{s_{Ml}\}, T_M)$$

В этом случае можно сделать вывод о том, что характеристики оригинала связаны с характеристиками модели зависимостями $y_{0i}=\varphi(y_{Mi})$. Множество выходных характеристик модели является отображением множества выходных характеристик оригинала, т.е. $\varphi: y_0 \rightarrow y_M$.

2 УПРАВЛЕНИЕ ВРЕМЕНЕМ В ДИСКРЕТНО-СОБЫТИЙНЫХ МОДЕЛЯХ

Хронологические потоки

Понятие “Система с дискретными событиями” определяет специфическую схему, которая, во-первых, может служить основой для разработки и алгоритмизации функциональных моделей систем весьма широкого класса и, во-вторых, использоваться как схема программирования таких моделей. В рамках этой схемы процесс функционирования системы во времени отождествляется с последовательностью событий, возникающих в системе в соответствии с закономерностями ее функционирования. В формальное понятие “событие” разработчик может вложить любое конкретное смысловое содержание, определяемое целями имитации системы.

В момент наступления события может изменяться состояние системы. Система находится в определенном состоянии S, если все ее компоненты находятся в состояниях, совместимых с областью значений, описывающих это состояние характеристик. Т.о., имитация — динамический “портрет” состояния системы во времени.

Т.к. в информационных потоках между событиями прежде всего устанавливаются хронологические отношения, такие потоки называются ***хроно-***

логическими. Процесс моделирования хронологических потоков можно проиллюстрировать на примере функционирования некоего абстрактного устройства моделирования хронологических потоков, представленного на рис.1.

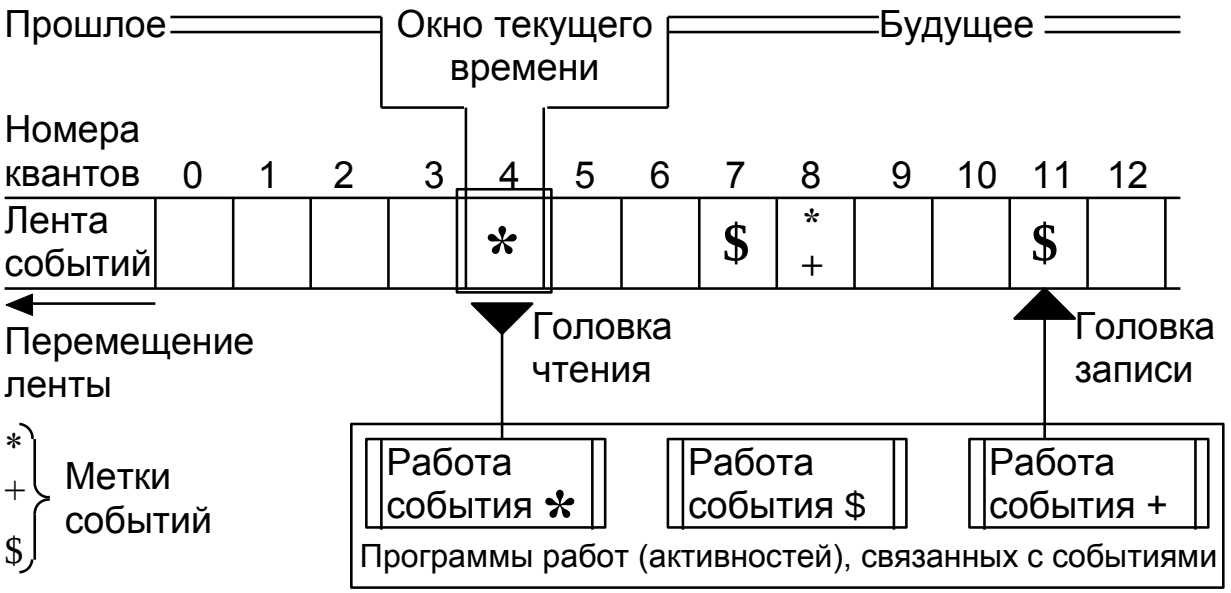


Рис. 1. Устройство моделирования хронологических потоков

Основным компонентом этого устройства моделирования является лента, организованная в виде последовательности фреймов, соответствующих временным квантам. Каждый такой фрейм может содержать метку одного или нескольких событий. Номер каждого кванта определяет момент времени в единицах, выбранных пользователем (сек, мин и т.д.). Головка чтения (ГЧ) всегда установлена на текущий момент времени и своим местоположением определяет границу между “прошлым” и “будущим”. Ход времени в модели ассоциируется с движением ГЧ, которая может перемещаться только вправо, соответственно движение ленты происходит только из “будущего” в “прошлое”. Доступ ГЧ к фреймам производится через окно текущего времени. В ответ на наступление события в текущий момент начинается выполнение программы работ (активностей), связанных с этим событием. Событие не имеет длительности во времени, а активность характеризуется моментами начала и окончания, т.е. продолжительностью во времени. Программы активностей могут записывать новые метки событий во фреймы ленты с помощью головки записи (ГЗ). Головка записи меток событий управляется планировщиком и может перемещаться как вправо, так и влево, но не левее ГЧ. После выполнения программы активности, соответствующей определенному событию, метка этого события уничтожается, так что часть ленты, соответствующая “прошлому”, всегда чиста.

Три вида времени в имитационном моделировании

Динамический процесс, определяемый последовательностью событий, протекает в так называемом системном времени. Вообще же существуют три вида времени, которые используются при определении имитационного процесса: реальное, системное и компьютерное. ***Реальное время*** — это время, в котором функционирует исследуемая система, время, определяемое ходом обычных часов. ***Системное время*** — это некоторый идентификатор программы моделирования (системный атрибут), имитирующий ход часов реального времени. Например, если имитируется процесс функционирования системы с момента ее включения $t=t_1$ до момента реального времени $t=t_2$, идентификатор системного времени STIME также определен в области значений $[t_1,t_2]$: $t_1<=STIME<=t_2$. При выполнении в процессе имитации условия $STIME>t_2$ процесс моделирования заканчивается. Понятие ***компьютерного времени*** связано лишь с процедурой выполнения имитационной программы. Длительность выполнения программы на компьютере определяется лишь тем, насколько эффективна компьютерная реализация программы имитации, и никак не связана с системным или реальным временем. Отношения между различными типами времени в моделировании иллюстрируются рис. 2.

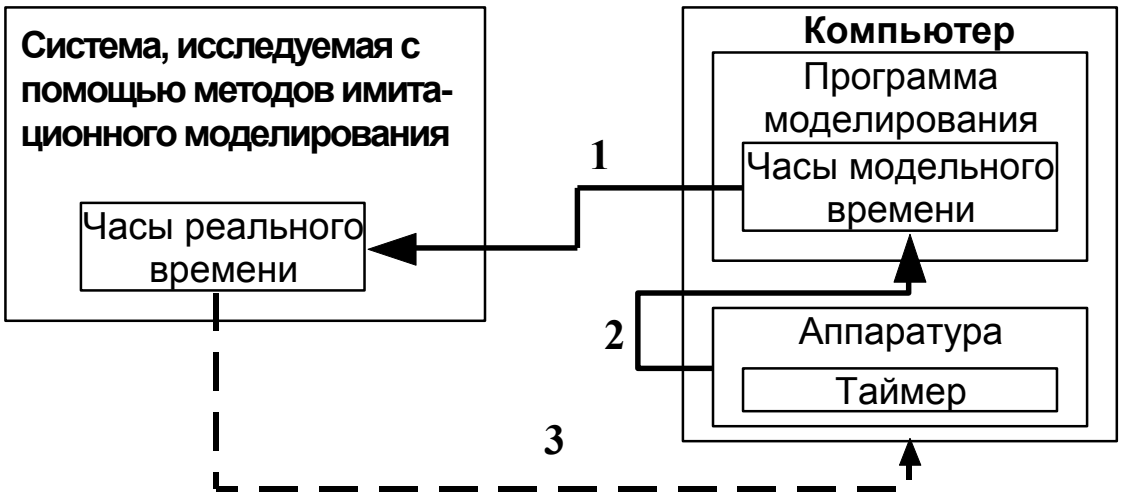


Рис. 2. Отношения между различными типами времени в моделировании

Связь (1) определяет отношение копирования: модельное время — копия реального времени; (2) определяет отношение управления: таймер управляет увеличением модельного времени; (3) определяет отношение синхронизации: таймер и часы реального времени синхронизируются. Если отношение (3) присутствует, программа моделирования называется моделью реального времени. Если (3) отсутствует, программа моделирования называется просто имитационной моделью. Такие модели имитируют системы реального времени путем копирования, без связи в реальном времени между имитационной моделью и реальной системой, исследуемой с помощью моделирования. В дальнейшем мы будем рассматривать имитационные модели без отношения

(3). Имитационная модель системы, функционирующей в реальном времени, например, в течение нескольких часов, в компьютерном времени может функционировать несколько минут или секунд и наоборот. Идентификатор системного времени в компьютерном может только возрастать, имитируя ход часов реального времени (см. рис. 3). “Эффект скачущих часов” (пунктирная кривая на рисунке) свидетельствует об ошибках в алгоритме управления моделированием. Величина же τ в общем случае никак не связана с величиной (t_2-t_1) реального времени функционирования системы.

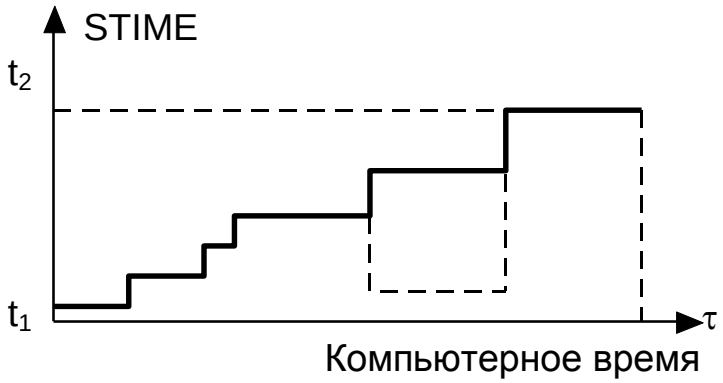


Рис. 3. Соотношение между системным и компьютерным временем

Методы пересчета модельного времени
Модели с непрерывным и дискретным изменением состояний

Пересчет системного времени в модели может быть осуществлен двумя методами:

- с использованием некоторого фиксированного шага □t (метод квантования, метод “□t”);
- с использованием переменного шага до следующего события (метод “особых состояний”), при этом считается, что в промежутке времени между событиями в модели системы никаких изменений не происходит.

В отношении пересчета системного времени имитационные модели классифицируют по двум основным категориям:

* модели с непрерывным изменением состояния;
* модели с дискретным изменением состояния.

Метод квантования используется в моделях систем с непрерывным изменением состояния или в моделях систем с высокоинтенсивными потоками событий. В моделях систем с дискретным изменением состояния используется метод “особых состояний”. Особым состоянием считается появление метки события в окне текущего времени. Алгоритм пересчета времени определяется следующим образом:

1) двигать ленту фреймов влево до появления следующего непустого кванта в окне текущего времени;

2) установить значение модельного времени равным номеру этого кванта.

Метод “особых состояний” требует меньших затрат компьютерного времени на отслеживание событий и корректировку часов в процессе имитации системы.

Рис. 4 демонстрирует два метода пересчета системного времени.

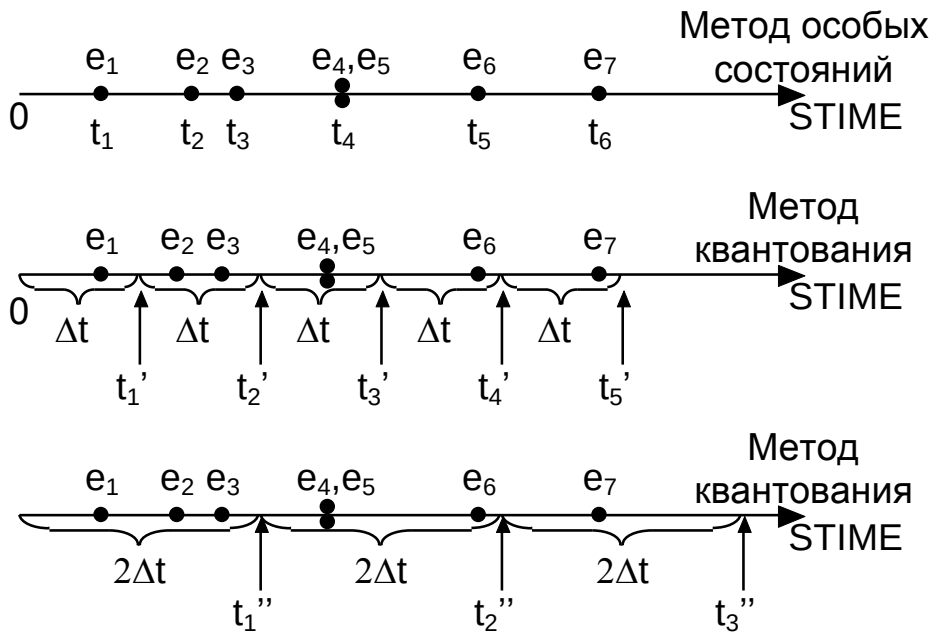


Рис. 4. Два метода пересчета системного времени

По оси системного времени отложена одна и та же последовательность событий e_i. Стрелки указывают на точки, в которых происходит приращение времени на один квант, и моменты наступления очередных событий. При использовании метода квантования последовательность значений системного времени такова:

$STIME=te_1=t_1$, $STIME=te_2=t_2$, $STIME=te_3=t_3$,
$STIME=te_4=te_5=t_4$, $STIME=te_6=t_5$, $STIME=te_7=t_6$.

Эти моменты времени в точности соответствуют моментам появления событий e_i. При использовании метода квантования значения системного времени $STIME=t_1'=\Delta t$, $STIME=t_2'=2\Delta t$,..., $STIME=t_5'=5\Delta t$,... никак не связаны с моментами появления событий e_i. Согласно принципу особых состояний, обработка событий идет последовательно, значение системного времени всякий раз смещается вперед на момент времени, соответствующий следующему событию, каждое из которых обслуживается по очереди. Согласно принципу Δt обработка событий осуществляется по группам, пакетам или множествам событий. Пусть задан некоторый момент $STIME=t_k'$, тогда обработка всех событий, наступивших в моменты t_p, t_q, t_r ..., такие, что $t'_{k-1}<t_p, t_q, t_r ... <= t'_k$ производится перед тем, как модельное время получает очередное приращение до t'_{k+1}. При изменении величины Δt точность моделирования может

уменьшиться, т.к. все события будут появляться в точке, соответствующей верхней границе интервала.

У модели, функционирующей по принципу особых состояний, имеется то преимущество, что в ней события рассматриваются и обслуживаются как одновременные только в том случае, если они маркированы одинаковым временем появления. Большинство систем моделирования используют для пересчета системного времени принцип “особых состояний”.

Проблема одновременности событий

Согласно рис.1, несколько меток событий могут появиться в одном фрейме. Порядок выполнения активностей, соответствующих этим меткам, определяется приоритетами событий. Чем выше приоритет, тем раньше будет выполнена активность. Если несколько меток событий имеют одинаковый приоритет, то порядок выполнения соответствующих активностей определяется порядком записи меток событий во фрейм.

Единица модельного времени, шкала модельного времени, таймер

Единица модельного времени — это абстракция, номер кванта на ленте фреймов, но обычно единица модельного времени определяется как эквивалент единице реального времени - сек, мин и т.д.

Шкала модельного времени — это упорядоченная совокупность единиц размерности времени (частей шкалы), например, сутки: часы: минуты или мин: сек: мсек: мксек. Шкала модельного времени определяется длиной шкалы (количеством используемых частей) и модулями пересчета из одной размерности (части) в другую.

Таймер — это объект системы моделирования, который используется в роли часов модельного времени. Показания таймера - мгновения — моменты времени, в которые возможны изменения состояния исследуемой системы. Каждый таймер имеет свою собственную шкалу и управляет хронологической последовательностью меток событий.

Если осуществляется перевод стрелок таймера вперед, часть меток событий оказывается в прошлом и теряется. Если стрелки таймера переводятся назад, метки событий, записанные во фреймах ленты, перемещаются в будущее.

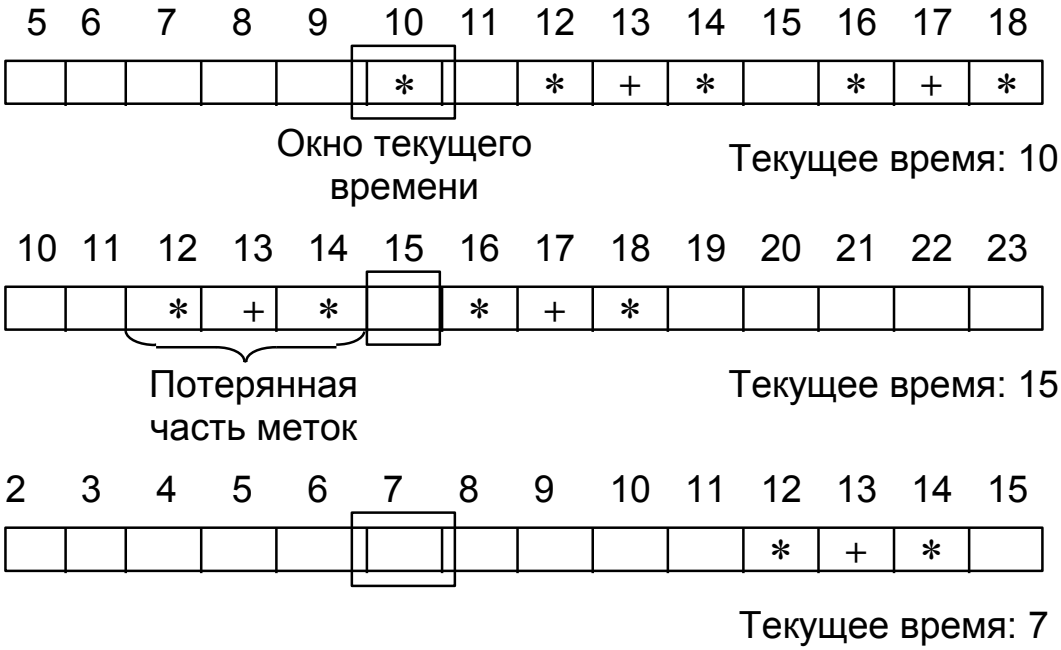


Рис. 5. Эффект перевода стрелок таймера

3 КОНЦЕПЦИЯ СОБЫТИЙ

Существует несколько схем дискретно-событийного моделирования, определяющих различные подходы к декомпозиции динамических процессов: схемы событий, состояний и параллельных процессов (транзактов). Такое деление во многом условно и концепция параллельных процессов впитывает в себя как событийную основу, так и декомпозицию по состояниям.

Структура календаря событий

При использовании схемы событий необходимо описать систему множеством типов событий {EV1, EV2, ... EVK}, установить закономерности (в том числе и вероятностные) возникновения каждого типа событий и в дальнейшем работать с этим множеством, используя специальные операции: генерировать событие, уничтожить событие и т.п. С формальной точки зрения имитация процесса функционирования системы во времени связана с генерацией определенных событий в соответствии с закономерностями функционирования системы и выполнением работ, т.е. установлением реакции системы на эти события. Процесс возникновения событий имеет в общем случае стохастический характер, моменты времени возникновения событий образуют хронологические потоки.

Функции управления динамикой работы моделируемой системы выполняет специальная управляющая программа, называемая монитором моделирования (симулятором). Монитор выполняет упорядочение событий по времени их возникновения в системе, пересчет системного времени, слежение за выполнением условий активации работ, соответствующих событиям, и т.д.

Для реализации и поддержки хронологических отношений в большинстве мониторов моделирования используется информационная структура списка — ***календарь*** — хронологически упорядоченное расписание, определяющее план развития модели во времени. Календарь состоит из меток. Каждая метка L события EV состоит из уведомления о событии EV_k, (k - тип события), момента времени, на которой запланировано наступление события EVT_k, приоритета события EVP_k (необязательно) и некоторой дополнительной информации (необязательно). Так, метка, стоящая в начале календаря, представленного на рис. 6, определяет событие десятого типа.

L(HEAD) = < EV10, EVT10, EVP10, дополнительная информация >

Программная реализация календаря связана с использованием линейной циклической списковой структуры.

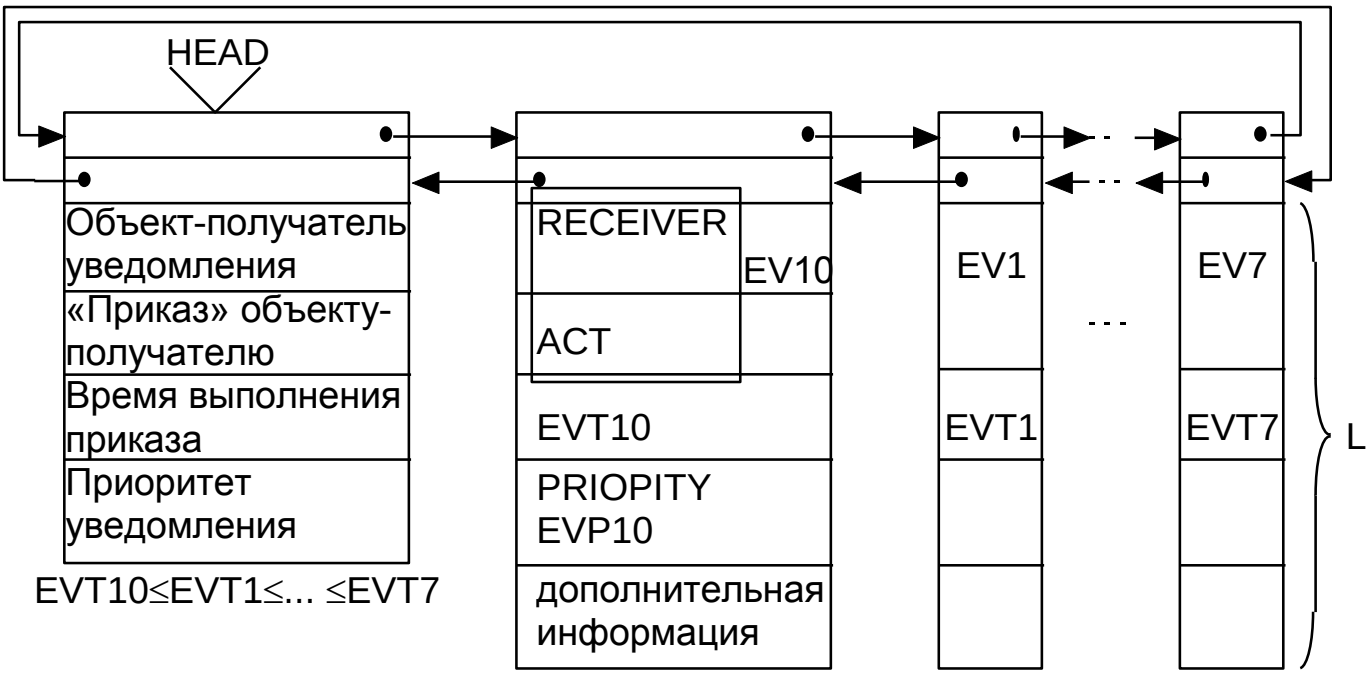


Рис. 6. Структура календаря событий

Монитор моделирования всегда выбирает из календаря событие с наименьшим временем, т.е. то, которое должно наступить в “ближайшем будущем”. В процессе имитации календарь постоянно обновляется: в него вносятся метки новых событий и исключаются события с минимальным временем (метки которых находятся по указателю HEAD) или те события, которые в соответствии с логикой системы необходимо отменить. Хронологическая упорядоченность списка при этом строго сохраняется. Интерпретацию уведомления, стоящего в голове календаря, можно определить как наступление события (мгновенный акт в системном времени), которое заключается в посылке “приказа” объекту-получателю. В ответ на получение этого приказа

объект-получатель совершает работу, соответствующую данному типу события.

Две разновидности управления в дискретно-событийных моделях

В дискретно-событийных моделях используется два вида событий: условные и безусловные. Безусловные события рассматривались в моделях хронологических потоков. Безусловное событие должно выполняться, когда запланированное время его наступления равно текущему, т.е. когда фрейм с соответствующей событию меткой появляется в окне текущего времени. Наступление условного события может зависеть, кроме времени, от дополнительных условий.

В соответствии с терминологией У. Дала [2] применительно к схеме событий различают две разновидности управления: ***императивное*** (повелительное, указывающее) и ***интеррогативное*** (вопросительное). Императивное управление состоит в управлении движением ленты фреймов и проверке содержимого фрейма текущего времени. Императивное управление связано с планированием некоторых событий на будущее, включением их меток в календарь (список будущих событий) или удалением из календаря. Используя операторы императивного управления, разработчик модели задает монитору моделирования те временные точки (критические времена), в которые в системе может что-то произойти, и указывает, что именно может произойти:

- “Запланировать событие EV на момент T”;
- “Запланировать событие EV с задержкой на t единиц относительно текущего времени”.

Однако императивное управление не обязательно всегда связано с явным указанием критического времени. Оно может реализовывать и механизм приоритетов, например, “Включить событие EV5 в календарь перед(после) события EV9”. Во всех языках имитационного моделирования императивное управление связано с изменением содержимого календаря, и таким образом (прямо или косвенно) операторы императивного управления связаны функционально с ведением “часов модельного времени”.

Интеррогативное управление определяет процесс функционирования системы при достижении определенного статуса, при выполнении некоторого набора логических условий. Обычно интеррогативное управление обеспечивает постоянное слежение со стороны управляющей программы за статусом системы и реакцию на все изменения последнего. Интеррогативное управление, в отличие от императивного, используется, когда заранее невозможно предсказать наступление какого-либо события не только детерминированным образом, но и с использованием стохастических алгоритмов. Схема интеррогативного управления представлена на рис. 7.

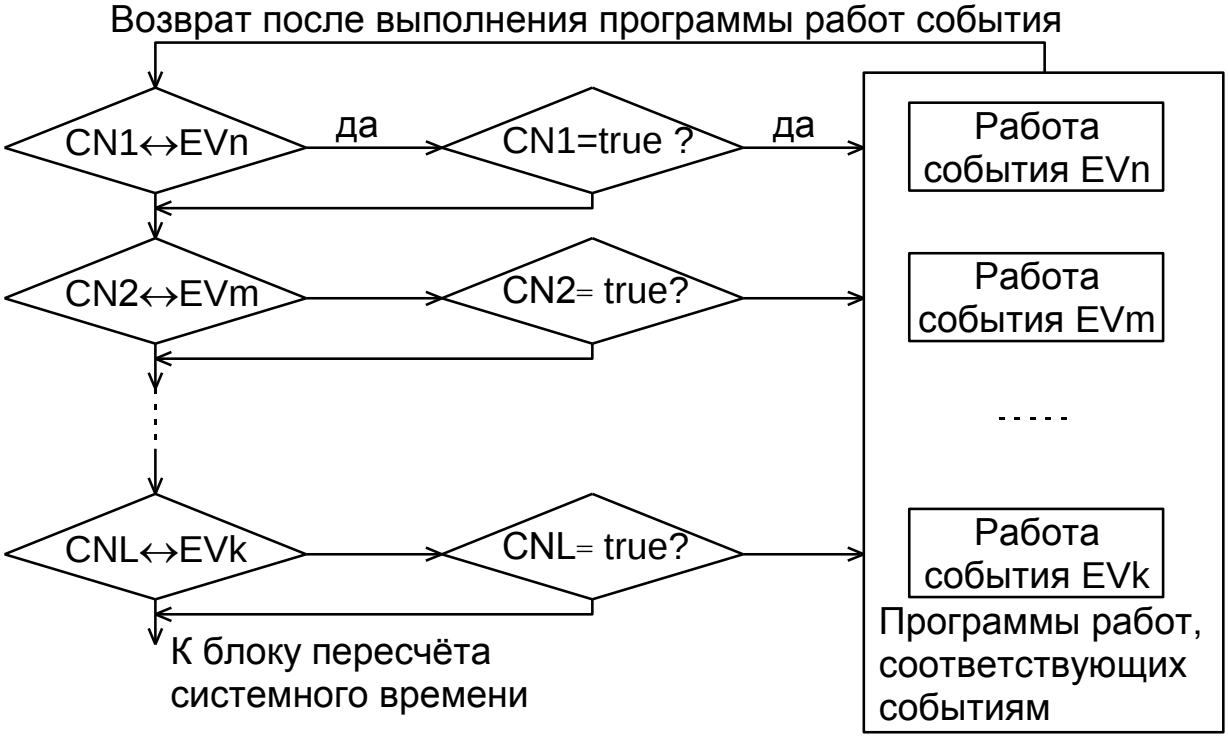


Рис. 7. Схема интеррогативного управления

Между множеством типов событий {EV1, EV2,..., EVN} и множеством условий {CN1, CN2,..., CNL} устанавливается соответствие либо перед началом имитации, либо в процессе моделирования. Условие — булевская функция, определяемая набором некоторых логических условий, при выполнении которых моделируемая система изменяет свой статус (в системе происходит событие). Если условия выполнены, соответствующая функция CN активна. В процессе имитации по окончании выполнения программы работ очередного события управление передается монитору моделирования, который проверяет выполнение условий: (CN <— >EV), т.е. “должно ли привести выполнение логических условий, определяющих функцию CN, к возникновению события EV”? При положительном ответе на этот вопрос проверяется активность функции CN: (CN=true?) и, если она активна, монитор передает управление программе события, управляемого данной функцией. Если в множестве функций нет ни одной активной, происходит пересчет системного времени и все проверки выполняются вновь. Один тип события может быть связан с несколькими условиями, с другой стороны, выполнение одного условия может вызывать наступление событий нескольких типов.

Программы работ условных событий и булевские функции условий могут использовать разделяемые переменные, определяющие статус системы. Функции условий проверяют состояние системы, а программы работ событий изменяют его (рис. 8).

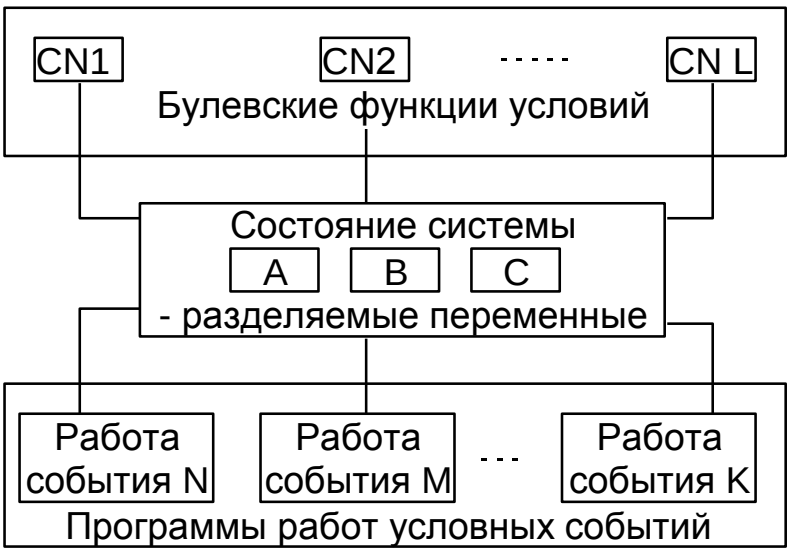


Рис. 8. Использование разделяемых переменных в схеме интеррогативного управления

Основной оператор интеррогативного управления ”Запланировать наступление события Ev_k в связи с выполнением функции условий CN_L”. Этот оператор устанавливает связь между элементами множеств CN и EV и включает интеррогативное управление для CN_L — функции. Если CN_L — новое условие и в множестве {CN} нет соответствующего элемента, монитор моделирования должен включить в множество {CN} новый элемент. Другой оператор “Разрушить связь между событием Ev_k и условием CN_L” выключает интеррогативное управление для функции CN_L. Если это условие не связано ни с какими другими событиями, оно исключается из множества {CN}. Т.о., схема интеррогативного управления может быть описана следующей циклической структурой.

WHILE <множество {CN} содержит хотя бы один активный элемент, т.е. существует функция CN_L, значение которой равно TRUE в текущий момент>
DO выполнить все программы работ всех событий, связанных со всеми активными условиями
END

Этот цикл может выполняться длительное время, т.к. программы работ событий изменяют состояние системы и, следовательно, структуру всего множества условий {CN}.

Операторы интеррогативного управления не изменяют содержимого календаря и не участвуют в поддержании правильного хода часов системного времени. При интеррогративном управлении системное время останавливается, при этом имитируются процессы и явления, происходящие в системе мгновенно, при достижении системой определенного статуса (выполнении определенных условий).

Интеррогативное управление реализуется с использованием списковой структуры следующего вида (рис. 9).

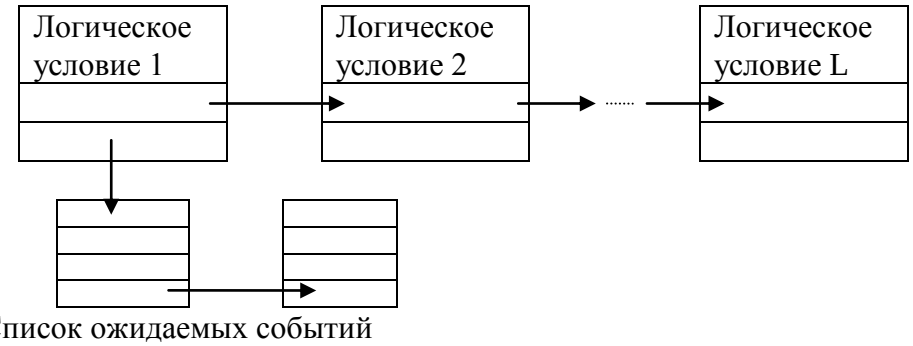


Рис. 9. Структура для реализации интеррогативного управления

Схема универсального дискретно-событийного симулятора

Схема универсального дискретно-событийного симулятора, осуществляющего императивное и интеррогативное управление, реализуется следующим алгоритмом.

```
WHILE < имитационный эксперимент продолжается? > DO
  WHILE          (*  Интеррогативное управление *)
  < имеются ли в множестве {CN} активные элементы? > DO
    ВЫПОЛНИТЬ программы работ событий, связанных с активными
              условиями
  КОНЕЦ;
  (* императивное управление *)
  ПЕРЕМЕСТИТЬ ленту фреймов влево так, чтобы непустой фрейм
              попал в окно текущего времени;
  ПОСМОТРЕТЬ  через окно текущего времени;
  ВЫБРАТЬ  метку события для выполнения программы работ собы-
           тия;
  ВЫПОЛНИТЬ программу работ события;
КОНЕЦ.
```

Т.о., пересчет модельного времени происходит, если в текущий момент нет активных условий, т.е. состояние системы не изменилось.

Как правило, в программах имитации присутствуют обе разновидности управления, хотя можно ограничиться и каким-либо одним методом. Однако их совместное использование делает программы имитации более содержательными и представляет большие возможности.

4 КОНЦЕПЦИЯ СОСТОЯНИЙ

Статической основой схемы состояний является ориентированный и размеченный граф переходов системы из состояния в состояние. Подобная формализация широко используется в сетях Петри, конечных автоматах, системах массового обслуживания. Однако для имитации динамических процессов, развивающихся во времени, требуется разработка некоторого механизма пересчета системного времени, адаптированного к схеме состояний.

Понятие схемы состояний

Концепция состояний базируется на предположении, что в каждый конкретный момент системного времени объект моделирования может находиться лишь в одном из состояний множества $S = \{s_1, s_2,...,s_n\}$, состояния несовместны. Каждое текущее состояние системы может быть охарактеризовано совокупностью атрибутов, которые в процессе имитации модифицируются в соответствии с алгоритмами, свойственными конкретной моделируемой системе. Множество атрибутов системы $A=\{a_1,a_2,...,a_m\}$. В состоянии S_i в модели выполняются действия f_i, в результате чего система переходит в состояние S_j (рис. 10).

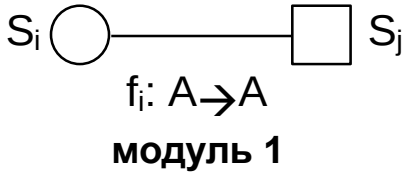


Рис. 10. Отображение *f* действий, выполняемых системой в состоянии *S*

Переходы системы из состояния в состояние связаны с возникновением событий. С каждым состоянием S_i путем введения ***графа переходов*** $G_s(S,E)$ связывается определенное подмножество E_i множества событий

$E_i \in E=\{e_1, e_2,...,e_k\}$.

Поскольку одно и то же событие в общем случае может определять возможности разных переходов, то $\bigcap_{i=1}^{n} E_i \neq \{0\}$.

Среди элементов множества E (как и среди элементов любого из подмножеств E_i) в любой момент системного времени можно выделить активные события, наступление которых ожидается в будущем, и пассивные (незапланированные на определенный момент времени) события. Причем для этих двух категорий (активные, пассивные) всегда справедливо

$$\forall t \quad \left[\left(\bigcup_{i=1}^{n} E_i^{act} = E^{act}\right)\&\left(\bigcup_{i=1}^{n} E_i^{pas} = E^{pas}\right)\&\left(E^{act} \cup E^{pas} = E\right)\right]$$

Механизм пересчета системного времени и развития процесса имитации при этом приобретает смысл порождения (генерации) элементов E^{act} во времени и их уничтожения. Обе эти функции выполняет монитор моделирования.

Планирование события e_k (k - тип события) (перевод его в активное состояние e_k^{pas} —> e_k^{act}) формально определяется как выбор пары (e_k,t), принадлежащей множеству

$E \times T = \{ (e_k,t) : (e_k \in E) \& (t \in T) \}$, причем

$T = \{ t : (STIME < t <= TSIM) \}$,

где STIME — текущее значение системного времени, TSIM — верхняя граница интервала времени, в течение которого имитируется поведение системы. Пара (e_k,t) — метка события e_k. Организация планирования событий связана с введением множества логических условий или предикатов P, определенных на элементах множества A (и в общем случае m - местных):

$P = \{P_1, P_2,...,P_l\}$, $P_i = P_i(a_1, a_2,...,a_m)$, i=1..,l.

При истинности выражения P_i разрешается планирование определенного подмножества E_i множества событий E на будущий момент времени t. Причем механизм образования метки (e_k,t) может иметь как детерминированную, так и стохастическую природу. С формальной точки зрения предикат выделяет планируемые элементы множества E:

$(\exists E_i) \dashv [E_i = \{ e_k : (e_k \in E) \& (P_i (a_1, a_2, ..., a_m)) \}]$

Взаимосвязи между предикатами и определяемыми с их помощью событиями и механизмами образования меток устанавливается заданием двудольного ***графа описания динамики моделируемой системы*** $G_T(E, TAU, P)$, где элементами множеств E и TAU помечены вершины, элементами множества P - ребра, TAU - множество механизмов генерации элементов t, входящих в метки (e_k,t) (рис. 11).

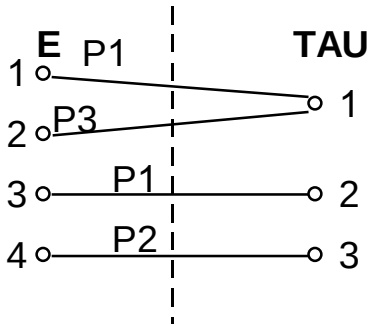


Рис. 11. Граф описания динамики моделируемой системы

Таким образом, моделируемая система может быть описана двумя графами (переходов и динамики), множеством действий, множеством предикатов и множеством временных механизмов TAU. Событийная основа выполняет при этом вспомогательные функции идентификации того или иного возможного перехода. Исследуемая система может рассматриваться как конечный автомат, функционирующий во времени.

Автоматно-событийная модель

Рис. 12 иллюстрирует ***автоматно-событийную модель***.

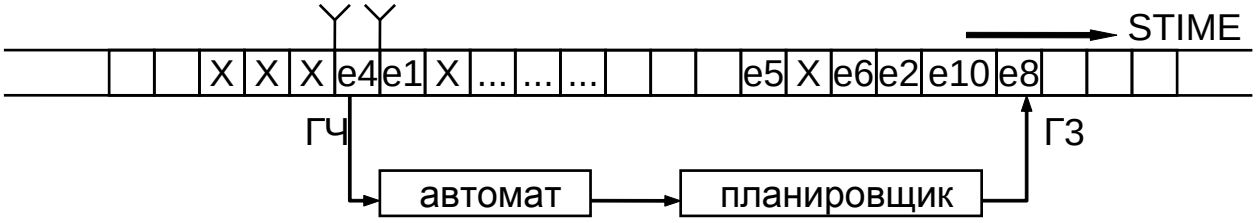


Рис. 12. Структура автоматно-событийной модели

Алгоритм функционирования системы:

Последовательность действий автомата

Сдвиг ГЧ на ближайший правый непустой квант (пересчет системного времени);

Определение нового состояния автомата S_i в соответствии с функцией переходов $ExS \rightarrow S$;

Интерпретация действий f_i, выполняемых системой в состоянии S_i;

Передача управления планировщику.

Последовательность действий планировщика:

Проверить истинность предикатов P;

Выделить элементы e_k, специфицируемые истинными предикатами;

Образовать метки $(e_k,t) \in ExT$;

Записать образованные метки на ленту через ГЗ;

Передать управление автомату.

Структура планировщика событий приведена на рис. 13.

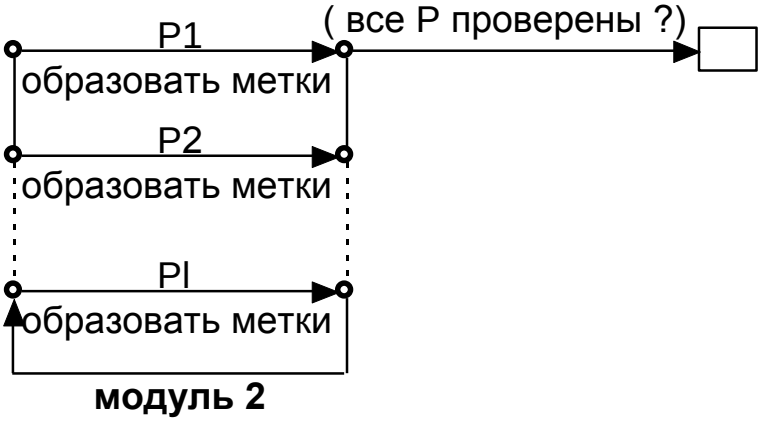


Рис. 13. Структура планировщика событий

Над стрелками записаны предикаты, под стрелками операторы планирования, причем связи между P и соответствующими e и t устанавливаются

через граф динамики. Такая диспетчеризация фактически адаптирует механизм императивного управления к схеме состояний. Интеррогативное управление при этом рассматривается как частный случай императивного управления при TAU={0}.

Алгоритм функционирования монитора, поддерживающего автоматно-событийную модель, приведен на рис. 14.

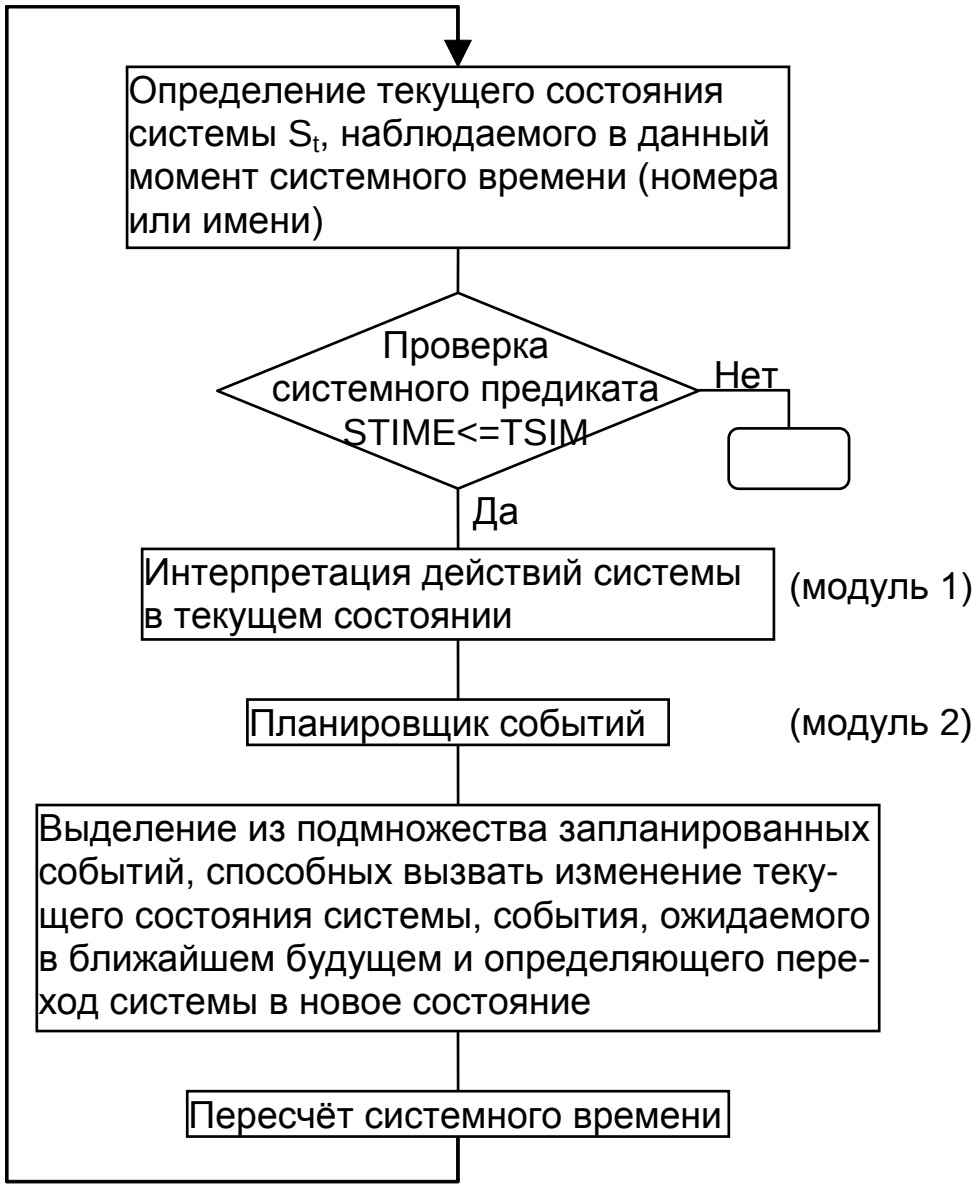


Рис. 14. Алгоритм функционирования монитора, поддерживающего автоматно-событийную модель

Формально процесс функционирования исследуемой системы ассоциируется в имитационной схеме состояний с перемещениями по графу переходов, происходящими во времени в соответствии с закономерностями изменения состояний, свойственными конкретной исследуемой системе.

Программная реализация автоматно-событийной модели

С содержательной точки зрения программа имитации, основанная на концепции состояний, содержит две части:

- декларирующая часть полностью определяется двумя графами: переходов $G_S(S, E)$ и динамики $G_T(E, TAU, P)$, которые могут представляться в виде ***объединенного графа переходов и динамики G(S, E, P, TAU)***;
- интерпретирующая часть модели состоит из двух компонент: интерпретация действий f_i и образование меток (e_k,t).

Действия f_i могут быть реализованы в виде процедур состояний, написанных на выбранном базовом языке. Предикаты P, определяющие планирование событий, могут задаваться арифметическим выражением, что означает планирование перехода через заданный этим выражением интервал модельного времени (безусловный переход), или логическим выражением, что означает переход при выполнении условия, заданного этим выражением (условный переход). Описание условия перехода может отсутствовать в автоматной части, в этом случае предикаты задаются при описании действий (процессов управления) и определяют возможный переход. TAU — это процедуры генерации информационных потоков.

Автомат и планировщик работают с информационной структурой следующего вида (рис. 15).

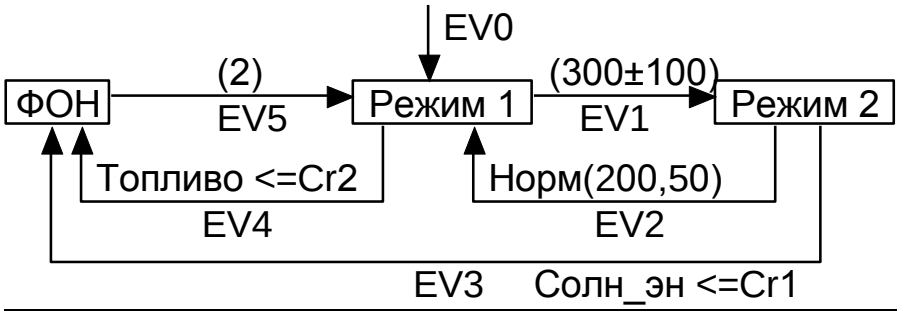
Рис. 15. Структура для реализации автоматно-событийной модели

Пример. Конечный автомат, моделирующий управление топливной системой во время полета

Ресурсы Солн_Эн, Топливо
CONST Cr1 = ... (* Критический уровень Солн_Эн *)
Cr2 = ... (* Критический уровень Топливо *)
Автомат
Фон → режим_1 / 2;
Режим_1 → Режим_2 / РАВН(300+-100), Фон;
Режим_2 → Режим_1 / НОРМ(200,50), Фон;

Процессы управления:
sit_1: напечатать “Критический уровень Солн_Эн”;
установить состояние “Фон”;
активизировать ситуацию Солн_Эн <= Cr1; Возврат;
sit_2: напечатать “Критический уровень Топливо”; Останов;
Старт: (* инициализация *)
установить состояние “Режим_1”;
активизировать sit_1 по условию Солн_Эн<=Cr1;
активизировать sit_2 по условию Топливо<=Cr2.

Объединенный граф переходов и динамики показан на рис. 16.



EV0 — “начало”;
EV1 — переход из Режим_1 в Режим_2 (безусловный);
EV2 — переход из Режим_2 в Режим_1 (безусловный);
EV3 — переход из Режим_2 в Фон (условный);
EV4 — переход из Режим_1 в Фон (условный);
EV5 — переход из Фон в Режим_1 (безусловный).

Рис. 16. Объединенный граф переходов и динамики

5 ПРОЦЕССНО-ОРИЕНТИРОВАННЫЙ ПОДХОД К МОДЕЛИРОВАНИЮ СИСТЕМ

Понятие процесса

Понятие процесса обычно связывается с изменением состояния системы во времени. Применительно к классу дискретных систем, процесс удобно интерпретировать как перемещение по графу состояний системы некоторого абстрактного элемента (например, указателя текущего состояния системы, в схеме транзактов — транзакта и т.д.). Если для класса аналитических моделей понятие процесса четко связано с множеством состояний системы, то понятие процесса применительно к алгоритмам имитации может быть определено следующим образом (без использования концепции состояний, например, только в рамках схемы, событий). Такой подход связан с разбиением множества событий {EV1, EV2,..., EVn}, возникающих в системе, на подмножества {E1, E2,..., Ek} (может быть пересекающиеся), каждое из которых ассоциируется с определенным процессом {PR1, PR2,..., PRk}. Возникновение события $EV_i \in E_k$ в некоторый момент времени означает активизацию процесса PR_k, т.е. в течение временного интервала между метками ближайших событий $EV_l \in E_k$ и $EV_m \in E_k$ процесс PR_k “ждет” активизации. В такой схеме внутри того или иного подмножества событий можно имитировать переход системы из одного состояния в другое за счет выполнения определенных действий над атрибутами системы.

Процесс — это ориентированная во времени последовательность событий, наступление которых приводит к изменению состояния. Понятие процесса является концептуально более емким, чем понятие события или состояния. Состояние — это моментальный снимок процесса, а событие — управляющее воздействие, заставляющее процесс изменить состояние.

Особое значение в концепции параллельных процессов приобретают вопросы синхронизации их взаимодействия. Основной аппарат такой синхронизации — события, управляемые интеррогативно.

Многие языки моделирования основаны на схеме параллельных процессов. Однако понятие процесса применительно к подобным языковым средствам приобретает некоторые специфические оттенки, обусловленные использованием определенных лексикографических средств описания процесса. Подобное описание содержит в себе набор инструкций, указывающих условия активизации процесса, синхронизации с другими процессами и с течением системного времени, альтернативы его развития в зависимости от складывающихся условий; здесь же задаются атрибуты процесса и т.д. Описание процесса определяет не конкретную его реализацию во времени, а скорее класс процессов.

Схема слабо связанных процессов. Интеррогативное управление специального вида с помощью сигналов

С помощью схемы слабо связанных процессов организуется взаимодействие квазипараллельных процессов. Каждый процесс представляет собой последовательную программу, выполнение нескольких таких программ осуществляется за счет передачи управления между ними.

Термин “слабо связанные процессы” означает, что связь между ними осуществляется через монитор, причем вместо функций условий для интеррогативного управления используются объекты специального вида — сигналы синхронизации. Объект “сигнал” может находиться в одном из двух состояний: “сигнал послан” и “сигнал не послан”. Посылка сигнала рассматривается как выполнение условия CN = (сигнал послан ?) = TRUE. На уровне реализации сигнал связан с множеством процессов, ожидающих этот сигнал (см. рис. 17).

Рис. 17. Объект “сигнал”

Все существующие в системе процессы представляются дескрипторами, которые связаны в кольцевой список (рис. 18,19).

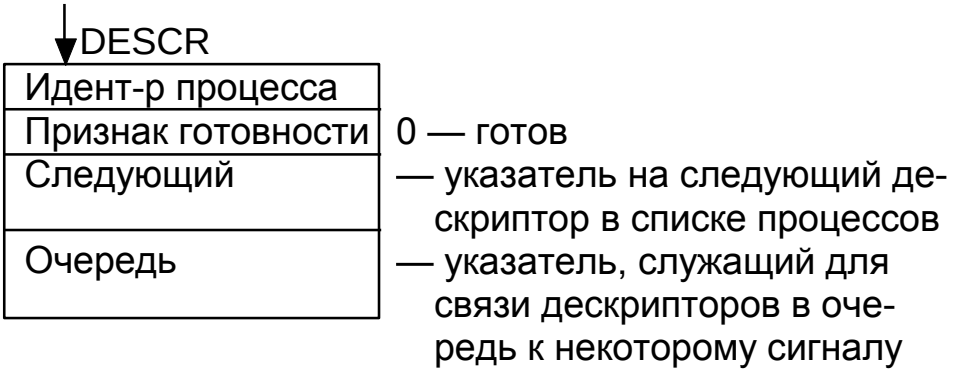


Рис. 18. Дескриптор процесса

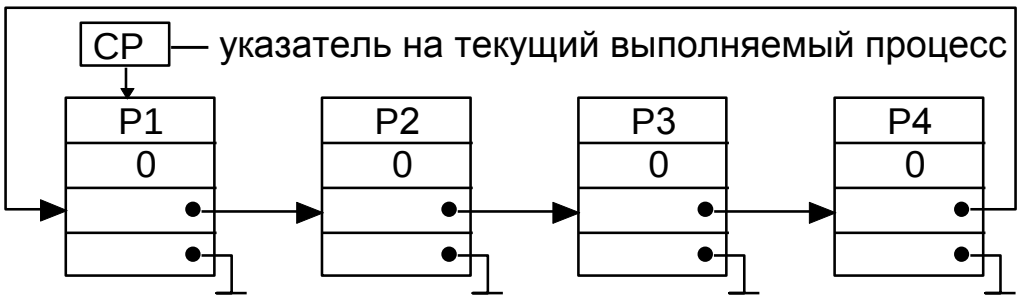


Рис. 19. Кольцо готовности процессов

Каждый процесс может посылать сигнал (оператор SEND) и ожидать посылки сигнала (оператор WAIT). Управление пересылкой сигналов от одного процесса к другому осуществляет монитор (рис. 20). Когда текущий процесс выдает команду WAIT(S), его признак готовности устанавливается в 1, он включается в конец очереди к сигналу S, текущим становится процесс, у которого признак готовности 0. Когда текущий процесс подает команду SEND(S), то если очередь к S пуста, ничего не происходит, если же очередь не пуста, из нее удаляется первый процесс, признак готовности у него устанавливается в 0 и он становится текущим. Процесс, выдавший команду SEND, перестает быть текущим, но признак готовности у него сохраняется 0. Одновременно могут существовать очереди к нескольким сигналам, но они никогда не пересекаются. Каждый процесс может ожидать посылки единственного сигнала.

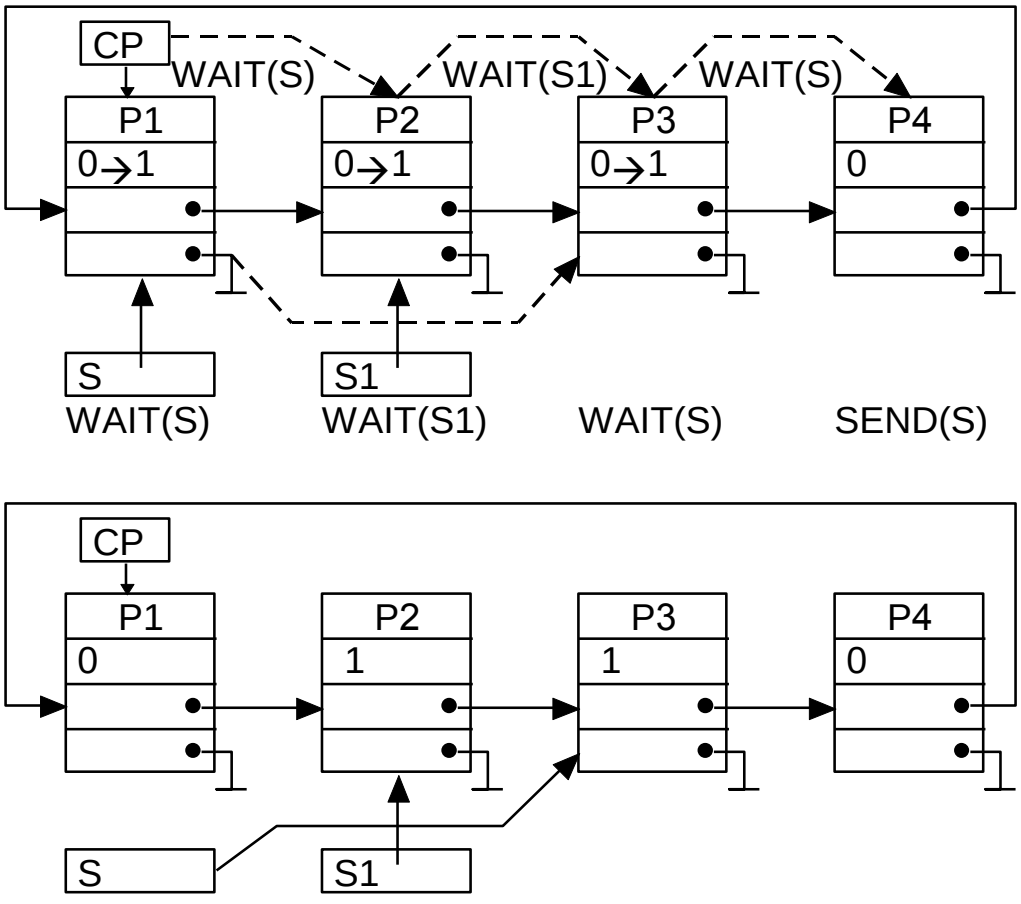


Рис. 20. Сигнальная синхронизация слабо связанных сигналов

Взаимодействие квазипараллельных процессов организуется по схеме сопрограмм, когда тело процесса выполняется не с самого начала, а с той точки, где выполнение было приостановлено (рис. 21).

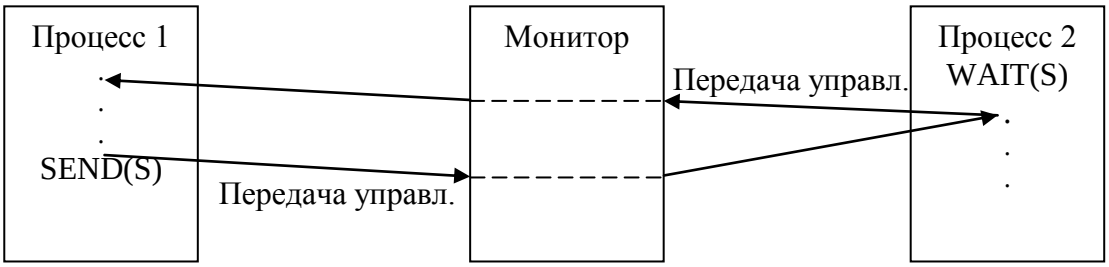


Рис. 21. Управление пересылкой сигналов между процессами

Пример. Взаимодействие двух процессов поставщик и потребитель происходит с использованием общего буфера сообщений.

```
IMPLEMENTATION MODULE BUFFER;
Const N=128; (*размер буфера*)
Var n:[0..N]; (*текущее число элементов в буфере*)
   НЕПОЛОН: SIGNAL;  (*n<N*)
   НЕПУСТ: SIGNAL; (*n>0*)
   IN, OUT: [0..N-1]; (*индексы*)
   BUF: array[0..N-1] of char; (*буфер*)

procedure ПОМЕСТИТЬ(x:char);
begin
   If n=N then WAIT (НЕПОЛОН);
   n:=n+1; BUF[IN]:=x;
   IN:=(IN+1) mod N; SEND (НЕПУСТ);
end;

procedure ИЗВЛЕЧЬ (var x:char);
begin
   If n=0 then WAIT(НЕПУСТ);
   n:=n-1; x:=BUF[OUT];
   OUT:=(OUT+1) mod N;
   SEND (НЕПОЛОН);
end;


begin
   n:=0; IN:=0; OUT:=0;
   INITSIGNAL (НЕПОЛОН);
   INITSIGNAL (НЕПУСТ);
end.
```

Концепция транзактов как частный случай схемы параллельных процессов

Одним из средств описания параллельных динамических процессов, имитируемых движением абстрактного элемента-транзакта, по графу состояний системы, является схема транзактов. Каждый транзакт движется в пространстве и во времени. Путь движения транзакта — абстракция, рассматриваемая как последовательность секций, называемых блоками. Каждый блок определяется специальными свойствами, например, “начало движения”, “останов”, “задержка на время”, “задержка до выполнения условия” и т.п. Выбирая блоки в соответствии со своими представлениями о закономерностях функционирования моделируемой системы, пользователь конструирует последовательность блоков - модель пути движения транзакта.

Транзакт **–** это абстрактный динамический объект, который может продвигаться по программе модели, останавливаться, уничтожаться. Транзакт может быть динамически введен в модель и выведен из модели. Каждый транзакт имеет свои собственные атрибуты, например, “время рождения”, “длительность пребывания в системе” и т.п. (рис. 22).

идентификатор транзакта (номер)
время рождения
время пребывания в модели
идентификатор блока, в котором находится транзакт
параметры

Рис. 22. Структура динамического объекта ”транзакт”

Схема транзактов строится на событийной основе. В календарь собраны транзакты, управляемые императивно (список будущих событий), интеррогативно управляемые транзакты собраны в список текущих событий. Понятие “событие” остается прежним: изменение состояния системы, происходящее мгновенно во времени. Однако концепция транзакта как основного элемента динамического процесса связывает каждое событие с передвижкой транзакта по блокам модели пути, и состояние моделируемой системы в каждый момент времени характеризуется тем, где (в каком блоке) находится транзакт, и тем, какую часть модели пути он прошел. Передвижка транзакта есть пара событий: “транзакт вошел в блок” (“IN”) и “транзакт вышел из блока” (“OUT”). Движение транзакта по модели пути можно представить как поток событий “IN”, “OUT”; “IN”, “OUT”; ... Событие “OUT” и следующее событие “IN” в схеме транзактов рассматриваются как одновременные, поэтому поток событий следует видоизменить:

“OUT-IN”; “OUT-IN”; “OUT-IN”; ... ; “OUT-IN”.

OUT — событие “транзакт вышел из особого (первого) блока, являющегося блоком генерации транзактов”.

IN — событие “транзакт вошел в особый (последний) блок, являющийся блоком уничтожения транзактов”.

Транзакт не может войти в блок генерации и не может выйти из блока уничтожения, транзакт не может находиться между блоками, поэтому движение транзакта — это серии переходов от блока к блоку. Если транзакт не может немедленно войти в блок, он находится в специальной списковой структуре монитора моделирования. В модели могут присутствовать несколько транзактов одновременно, причем каждый из них инициирует свой цикл действий, свой динамический процесс. Блок генерации может генерировать регулярные или стохастические потоки транзактов. Нерегулярные стохастические потоки порождают две основные проблемы: появление очередей и появление тупиков.

Модели очередей, возникающих при задержке движения транзакта

Возникновение очередей транзактов является результатом многих причин, среди них основные — типы потоков событий, интенсивности потоков, структура модели и т.д. Каждый блок, который может задержать движение транзакта, — возможное место возникновения очереди. Основные дисциплины управления очередями — FIFO, LIFO, приоритетная, внутри одного приоритетного класса используется дисциплина FIFO. Основные характеристики очереди — средняя длина и среднее время пребывания транзакта в очереди.

Важнейшими видами очередей являются очереди ограниченной длины (рис. 23) и очереди ограниченной длины с ограниченным временем пребывания (рис. 24).

Условие ограничения времени пребывания проверяется для всех членов простой очереди, и один или несколько членов должны покинуть ограниченную очередь. Направление дальнейшего движения транзактов зависит от дополнительных условий и структуры модели. Т.о., модели ограниченных очередей имеют собственное интеррогативное управление. В системе моделирования модели очередей инкапсулированы в мониторе, так что работа с моделью очереди выполняется автоматически и пользователь такой системы может не знать о деталях реализации моделей очередей. Очередь является сложной стохастической структурой, которая может появляться и исчезать в модели исследуемой системы.

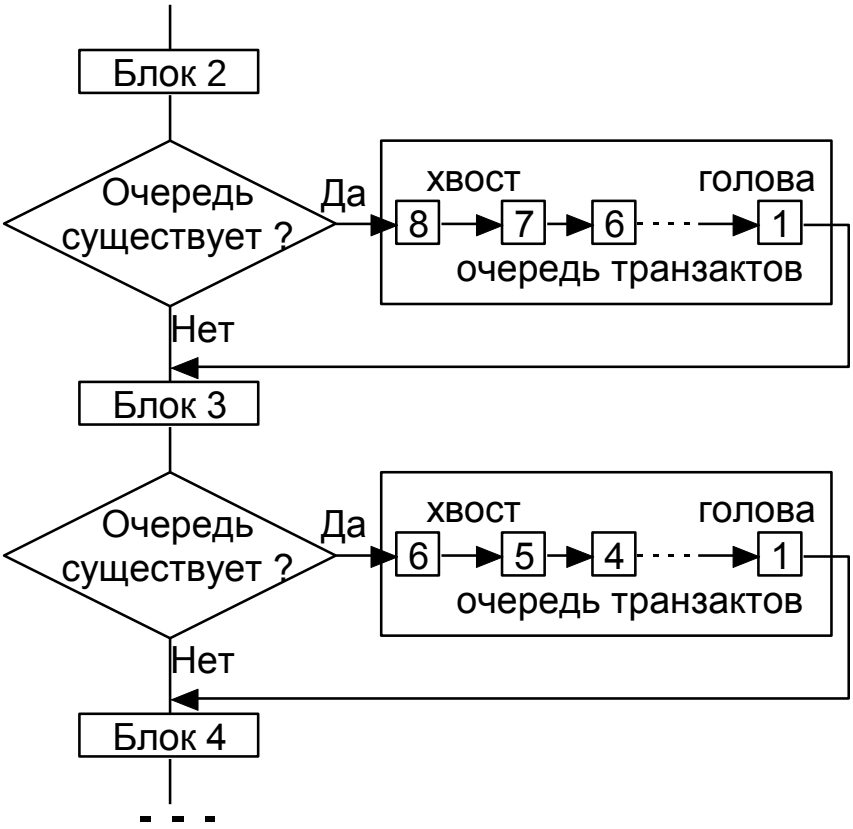


Рис. 23. Очередь ограниченной длины

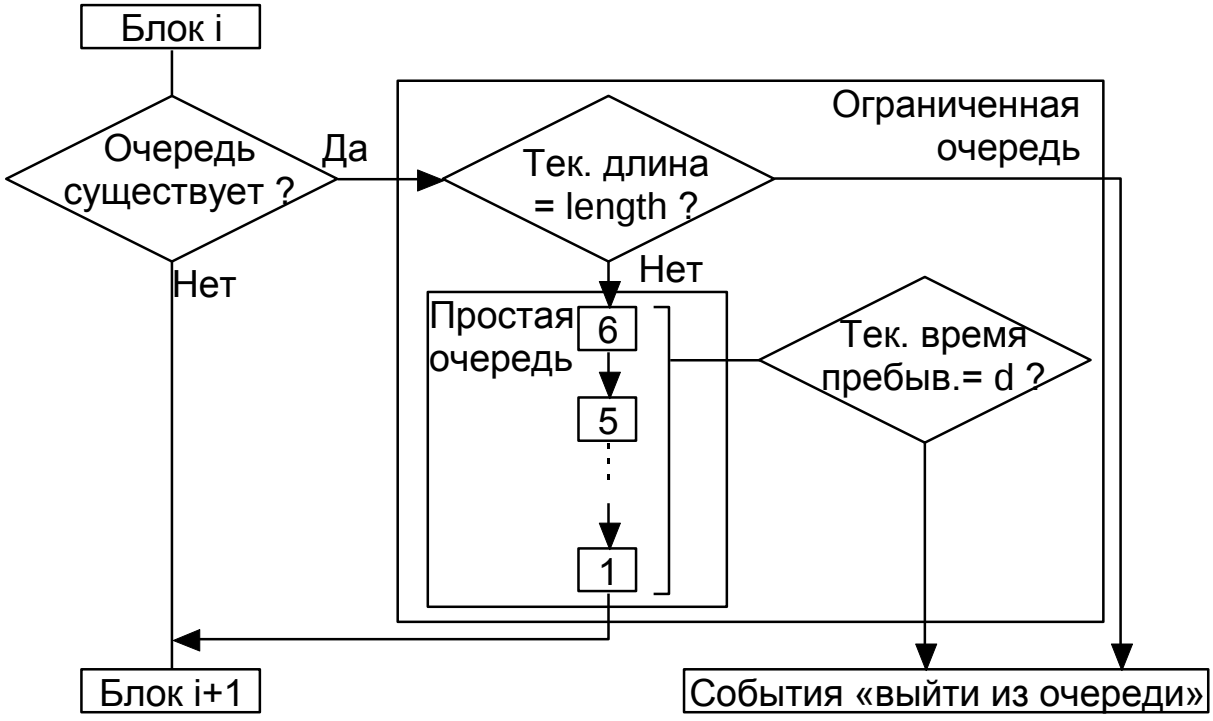


Рис. 24. Очередь ограниченной длины с ограниченным временем пребывания транзактов

Проблема тупиков

Тупики имеют стохастическую природу возникновения, как и очереди, но, в отличие от очереди, не могут сами исчезнуть. Тупик возникает в ситуации, когда каждый из взаимодействующих процессов получил в свое распоряжение часть необходимых ему ресурсов, разделяемых с другими процессами, но их не хватает ни одному процессу для завершения обработки и последующего освобождения ресурсов. Появление тупика является фатальным и все процессы в моделируемой системе, попавшие в тупик, должны быть немедленно остановлены.

Пример иллюстрирует схему программы, которая может привести к появлению тупика. A и B - абстрактные устройства, которые являются ресурсами, общими для некоторых процессов.

;Процесс i

ЗАНЯТЬ A на время t_1;
ОСВОБОДИТЬ A;
ЖДАТЬ (B не используется);
ЗАНЯТЬ B на время t_2;
ОСВОБОДИТЬ B.

;Процесс j

ЗАНЯТЬ B на время t_3;
ОСВОБОДИТЬ B;
ЖДАТЬ (A не используется);
ЗАНЯТЬ A на время t_4;
ОСВОБОДИТЬ A.

Появление тупика (взаимная блокировка) зависит от многих факторов:

- в какой момент процессы войдут в блоки, занимающие ресурсы, одновременно или нет
- каковы значения переменных t_1, t_2, t_3, t_4.

Т.о., наличие тупика в модели не означает, что система обязательно в него войдет.

Существует два метода борьбы с тупиками, их обнаружение и предотвращение. Обнаружение предусматривает запоминание трассы взаимодействия процессов, при обнаружении тупика производится откат до момента, когда процессы могут продолжать свое развитие.

Предотвращение заключается в ограничениях на конструирование самих процессов.

Методы предотвращения тупиков:

1. Линейное упорядочение типов ресурсов, необходимых процессам. Создается иерархия ресурсов R1<R2<…<Rn, n – число типов ресурсов.

Если процесс уже использует ресурс Ri, то он может запросить только такой ресурс Rj, для которого выполняется условие Ri<Rj. Ресурс Rj освобождается раньше Ri. При таком использовании ресурсов тупик возникнуть не может.

2. Применение матриц запросов и матриц используемых ресурсов. Пусть имеется n процессов P1,P2,…,Pn и m типов ресурсов R1,R2,…,Rm. Каждый ресурс типа Ri имеет количество единиц Ci. Использование ресурсов характеризуется матрицей запросов Z и матрицей используемых ресурсов U. Матрица Z=(z1, z2,…, zn), где вектор Zi(j) задает максимальное количество единиц ресурса типа j, необходимое процессу Pi. Матрица U = (u1, u2, …, un), где вектор Ui(j) задает количество единиц ресурса j, уже используемого процессом Pi. В начале вычислительного процесса U=0. Каждый процесс последовательно проходит этапы запроса ресурса, использования и освобождения ресурса. Последовательность выдачи ресурсов процессам считается допустимой, если при очередной выдаче ресурсов запрос процесса не превышает числа свободных единиц по каждому требуемому ресурсу.

6 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание на моделирование и изучить необходимые теоретические материалы.
2. Разработать GPSS-программу моделирования.
3. Выполнить программу на компьютере.
4. Оформить отчет.

7 ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Сконструировать дискретно-событийный монитор, реализующий набор операторов, соответствующих одному из перечисленных ниже методов управления. Отобразить ход часов модельного времени и проиллюстрировать динамику функционирования монитора моделирования.

1. Императивное управление без приоритетов уведомлений

1. Включить в календарь уведомление о событии EVi в абсолютном времени.
2. Включить в календарь уведомление о событии EVi перед событием EVj.
3. Отменить все события EVi, запланированные в календаре.
4. Определить, сколько событий запланировано на заданный момент времени.

5. В течение заданного промежутка времени выполнить действия, соответствующие событиям, запланированным в календаре.

2. Императивное управление без приоритетов уведомлений

1. Включить в календарь уведомление о событии EVi в относительном времени.
2. Включить в календарь уведомление о событии EVi после события EVj.
3. Отменить все события EVi, запланированные в календаре на заданный момент времени.
4. Определить запланированное время наступления события EVi.
5. Выполнить все действия, соответствующие событиям, запланированным в календаре.

3. Императивное управление с приоритетами уведомлений

1. Включить в календарь уведомление о событии EVi в абсолютном времени в заданном приоритетном классе.
2. Включить в календарь уведомление о событии EVi перед событием EVj в заданном приоритетном классе.
3. Отменить все события EVi, запланированные в календаре в заданном приоритетном классе.
4. Определить, сколько событий запланировано на заданный момент времени в каждом из приоритетных классов.
5. В течение заданного промежутка времени выполнить действия, соответствующие событиям, запланированным в календаре.
6. Изменить приоритет события EVi.

4. Императивное управление с приоритетами уведомлений

1. Включить в календарь уведомление о событии EVi в относительном времени в заданном приоритетном классе.
2. Включить в календарь уведомление о событии EVi после события EVj в заданном приоритетном классе.
3. Отменить все события EVi, запланированные в календаре на заданный момент времени в каждом из приоритетных классов.
4. Определить запланированное время наступления события EVi и его приоритет.
5. Выполнить все действия, соответствующие событиям, запланированным в календаре.

6. Изменить приоритет события EVi.

5. Императивное управление без приоритетов уведомлений при наличии в системе нескольких таймеров

В системе имеется один таймер с нулевым смещением показаний, называемый системным. Все остальные таймеры задаются смещением их показаний относительно текущих показаний системного таймера.

1. Включить в календарь уведомление о событии EVi под таймер Tj в абсолютном времени этого таймера.
2. Определить значение текущего времени в шкале таймера Tj.
3. Оцифровать таймер Tj абсолютным значением времени TT, т.е. определить смещение показаний таймера Tj относительно показаний системного таймера и изменить положение меток соответствующих событий в календаре.
4. Отменить все события, связанные с таймером Tj.
5. В течение заданного промежутка времени выполнить действия, соответствующие событиям, запланированным в календаре.

6. Императивное управление без приоритетов уведомлений при наличии в системе нескольких таймеров

В системе имеется один таймер с нулевым смещением показаний, называемый системным. Все остальные таймеры задаются смещением их показаний относительно текущих показаний системного таймера.

1. Включить в календарь уведомление о событии EVi под таймер Tj в относительном времени этого таймера.
2. Определить число событий под таймером Tj.
3. Оцифровать таймер Tj абсолютным значением времени TT, т.е. определить смещение показаний таймера Tj относительно показаний системного таймера и изменить положение меток соответствующих событий в календаре.
4. Отменить событие EVi, связанное с таймером Tj.
5. Выполнить все действия, соответствующие событиям, запланированным в календаре.

7. Интеррогативное управление с использованием сигнальной синхронизации

1. Связать наступление события EVi с подачей сигнала Sj (ожидание подачи сигнала Sj).
2. Подать сигнал Sj.
3. Сбросить сигнал Sj.
4. Определить статус сигнала Sj.
5. Определить, имеются ли в системе уведомления, ожидающие подачи сигнала Sj.
6. Запланировать выполнение действий, соответствующих событиям, до тех пор, пока в системе имеется хотя бы один активный сигнал, подачи которого ожидает хотя бы одно уведомление.

8. Интеррогативное управление с использованием сигнальной синхронизации в сочетании с императивным управлением без приоритетов уведомлений

1. Связать наступление события EVi с подачей сигнала Sj (ожидание подачи сигнала Sj).
2. Подать сигнал Sj.
3. Сбросить сигнал Sj.
4. Включить в календарь уведомление о событии EVi в абсолютном времени.
5. Запланировать выполнение действий, соответствующих событиям, до тех пор, пока календарь событий не пуст и в системе имеется хотя бы один активный сигнал, подачи которого ожидает хотя бы одно уведомление.

9. Интеррогативное управление с использованием ситуационного управления (управления по условию)

1. Создать ситуацию Si, определяемую логическим выражением Cj.
2. Связать наступление события EVi с ситуацией Si.
3. Определить статус ситуации Si.
4. Определить, есть ли уведомления, ожидающие активизации ситуации Si.
5. Отменить ситуацию Si, определяемую логическим выражением Cj.
6. Разрушить связь между событием EVi и ситуацией Si.
7. Запланировать выполнение действий, соответствующих событиям, до тех пор, пока в системе имеется хотя бы одна активная ситуация, с которой связано хотя бы одно уведомление.

10. Интеррогативное управление с использованием ситуационного управления (управления по условию) в сочетании с императивным управлением без приоритетов уведомлений

1. Создать ситуацию Si, определяемую логическим выражением Cj.
2. Связать наступление события EVi с ситуацией Si.
3. Определить, есть ли уведомления, ожидающие наступления ситуации Si.
4. Отменить ситуацию Si, определяемую логическим выражением Cj.
5. Включить в календарь уведомление о событии EVi в относительном времени.
6. Запланировать выполнение действий, соответствующих событиям, до тех пор, пока календарь событий не пуст и в системе имеется хотя бы одна активная ситуация, с которой связано хотя бы одно уведомление.

11. Схема состояний (конечный автомат, функционирующий в модельном времени)

1. Определить состояние Si автомата (т.е., идентификатор данного состояния, идентификаторы состояний, в которые возможны переходы из данного, и признаки переходов).
2. Определить действия Fi, которые должны выполняться в состоянии Si.
3. Определить условный переход, т.е. запланировать переход из состояния i в состояние j в связи с выполнением условия, задаваемого логическим выражением Cj.
4. Определить текущее состояние автомата.
5. Запустить автомат (автомат функционирует до тех пор, пока он не перейдет в такое состояние, из которого нет ни одного перехода).

12. Схема состояний (конечный автомат, функционирующий в модельном времени) в сочетании с императивным управлением без приоритетов уведомлений

1. Определить состояние Si автомата (т.е., идентификатор данного состояния, идентификаторы состояний, в которые возможны переходы из данного, и признаки переходов).
2. Определить действия Fi, которые должны выполняться в состоянии Si.
3. Определить безусловный переход, т.е. запланировать переход из состояния i в состояние j через промежуток времени, задаваемый значением арифметического выражения Ai.

4. Определить множество временных механизмов формирования уведомлений.
5. Запустить автомат (автомат функционирует до тех пор, пока он не перейдет в такое состояние, из которого нет ни одного перехода).

13. Схема процессов (очереди в блоках имеют ограниченную длину и ограниченное время пребывания транзактов)

1. Реализовать процесс, определяющий модель пути движения транзакта:
   - ввести транзакт в модель (сгенерировать транзакт),
   - транзакт занимает устройство,
   - задержка движения транзакта на заданное время (обслуживание в устройстве),
   - транзакт освобождает устройство,
   - вывести транзакт из модели (уничтожить транзакт).
2. Определить множество алгоритмов генерации транзактов и формирования задержек.
3. Транзакты, выводимые из очереди по причине переполнения или превышения времени пребывания, покидают модель. Определить вероятность того, что транзакт покинет модель необслуженным.
4. Сгенерировать и провести по блокам модели 100 транзактов.
5. Отобразить таблицу транзактов и таблицу блоков.

14. Схема процессов (очереди в блоках имеют неограниченную длину и неограниченное время пребывания транзактов)

1. Реализовать процесс, определяющий модель пути движения транзакта:
   - ввести транзакт в модель (сгенерировать транзакт),
   - транзакт занимает устройство,
   - задержка движения транзакта на заданное время (обслуживание в устройстве),
   - транзакт освобождает устройство,
   - вывести транзакт из модели (уничтожить транзакт).
2. Определить множество временных механизмов генерации транзактов и формирования задержек.
3. Определить среднее время пребывания транзакта в модели.
4. Сгенерировать и провести по блокам модели 100 транзактов.
5. Отобразить таблицу транзактов и таблицу блоков.

15. Императивное управление без приоритетов уведомлений

1. Включить в календарь уведомление о событии EVi в абсолютном времени.
2. Включить в календарь уведомление о событии EVi на заданный момент времени перед другими событиями с таким же значением времени (в абсолютном времени).
3. Отменить события всех типов, запланированные в календаре на заданный момент времени.
4. Определить, сколько событий каждого типа запланировано на заданный момент времени.
5. В течение заданного промежутка времени выполнить действия, соответствующие событиям, запланированным в календаре.

16. Императивное управление без приоритетов уведомлений

1. Включить в календарь уведомление о событии EVi в относительном времени.
2. Включить в календарь уведомление о событии EVi на заданный момент времени после других событий с таким же значением времени (в относительном времени).
3. Отменить все события заданного типа EVi, запланированные в календаре.
4. Определить запланированное время наступления события EVi.
5. Выполнить все действия, соответствующие событиям, запланированным в календаре.

8 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как осуществляется управление временем в дискретно-событийных моделях?
2. Какие виды времени в имитационном моделировании Вам известны?
3. Какие методы пересчета модельного времени Вам известны? Как они работают? Какие методы используются в моделях с непрерывным и дискретным изменением состояний?
4. В чем заключается концепция событий в имитационном моделировании? Опишите структуру календаря событий.
5. В чем состоят особенности императивного управления в дискретно-событийных моделях?
6. В чем состоят особенности интеррогативного управления в дискретно-событийных моделях?
7. Как функционирует универсальный дискретно-событийный симулятор?
8. В чем заключается концепция состояний в имитационном моделировании? Опишите структуру и принципы использования графа переходов моделируемой системы. Опишите структуру и принципы использования графа описания динамики моделируемой системы.

9. Опишите алгоритм функционирования автоматно-событийной модели. Опишите структуру и принципы использования объединенного графа переходов и динамики моделируемой системы.
10. Как функционирует монитор, поддерживающий автоматно-событийную модель?
11. В чем заключаются особенности процессно-ориентированного подхода к моделированию систем? Объясните понятие процесса.
12. В чем заключается концепция объектов-транзактов? Объясните понятие движущегося объекта-транзакта.
13. Как работает схема слабо связанных процессов? Что такое объект-сигнал? Как выполняется интеррогративное управление специального вида с помощью сигналов?
14. Какие модели очередей, возникающих при задержке движения транзакта, используются в процессно-ориентированном подходе к моделированию систем?
15. В чем заключается проблема тупиков в процессно-ориентированном подходе к моделированию систем? Как разрешается эта проблема?

ЗАКЛЮЧЕНИЕ

Методические указания «Конструирование дискретно-событийных мониторов для систем моделирования» посвящены рассмотрению методов и алгоритмов модельной динамики систем с дискретными событиями, разработке дискретно-событийных имитационных моделей.

Методические указания содержат рекомендации по реализации управляющих программ для систем моделирования на основе концепций событий, состояний и процессов.

Логическим завершением методических указаний являются контрольные вопросы и индивидуальные задания для самостоятельной работы студентов.

Вопросы, имеющие практическое значение для студентов при выполнении лабораторной работы, освещены с необходимой для использования полнотой.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шеннон Р. Имитационное моделирование систем — искусство и наука: Пер. с англ. - М. : Мир, 1978. – 418 с.
2. Цифровая имитация автоматизированных систем / Болтянский А.А., Витих В.А., Кораблин М.А. и др. - М. : Наука, 1983. – 215 с.
3. Лоу А., Кельтон В. Имитационное моделирование [Simulation Modeling and Analysis]. СПб.: Издательство:Питер, 2004. – 848 с.

Методические материалы

КОНСТРУИРОВАНИЕ ДИСКРЕТНО-СОБЫТИЙНЫХ МОНИТОРОВ ДЛЯ СИСТЕМ МОДЕЛИРОВАНИЯ

Методические указания

Составитель ***Симонова Елена Витальевна***

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С. П. КОРОЛЕВА»
(Самарский университет)
443086, САМАРА, МОСКОВСКОЕ ШОССЕ, 34.

Изд-во Самарского университета.
443086 Самара, Московское шоссе, 34.