О СИНТЕЗЕ СОБЫТИЙНЫХ МОДЕЛЕЙ ДИСКРЕТНЫХ СИСТЕМ

© 2006 Е. А. Бабкин

канд. техн. наук, доцент кафедры
Курского государственного университета

Предлагается метод синтеза математических и программных событийных моделей, представляемых в виде событийного графа в алгоритмической форме или событийного алгоритма. Для дискретных систем, имеющих в основе процессы обслуживания, предлагается в качестве промежуточной математической модели использовать модель в виде сети массового обслуживания, которая затем преобразуется в событийный граф. Рассмотрено также преобразование событийного графа в программную модель. Введена промежуточная модель графа: событийный граф в программно-реализуемой форме, которая является разновидностью событийного графа на уровне макрособытий.

Введение

В событийных моделях основным объектом является событие, и процесс функционирования ДС представляется в виде последовательности событий. В работе [1] рассмотрен содержательный подход к построению событийных моделей. Для представления событийных имитационных моделей дискретных систем (ДС) в работах [2, 3] предлагается использование математической модели в виде событийного графа. В работе [2] впервые предложена формализация событийной модели в виде событийного графа. В работе [3] рассмотрен другой подход к построению событийных графов, отличающийся большей детализацией и видом представления событийных графов: событийные графы имеют форму более близкую к алгоритмам и поэтому могут быть названы событийными алгоритмами. В настоящей работе рассматривается метод построения событийных моделей на базе подхода, предложенного в работе [3] и развитого в работе [4].

Этапы разработки событийных моделей

Рассмотрим метод разработки событийных моделей на базе языка высокого уровня (Delphi, C). Использование универсального языка высокого уровня ограничивает сложность разрабатываемых моделей. Однако этот недостаток можно устранить, используя иерархическую структуризацию моделей и библиотеку процедур, выполняющую набор функций системы имитационного моделирования.

В разработке модели на основе событийных графов [3] можно выделить следующие этапы:

– разработка концептуальной модели ДС;

– разработка математической модели ДС в виде сети массового обслуживания;

– разработка математической модели ДС в виде событийного графа;

– разработка программной модели ДС на языке высокого уровня (например, Паскаль), дополненного набором процедур, реализующих функции событийно-ориентированного языка.

На первом этапе разрабатывается концептуальная модель ДС на основе исследования всех существенных причинно-следственных связей и свойств процесса функционирования ДС, определяются все статистическо-временные характеристики системы.

Концептуальная модель ДС может включать структуру, временную диаграмму и содержательное описание процесса функционирования ДС.

На втором этапе производится выделение и отождествление неформализованных элементов концептуальной модели ДС и элементов системы массового обслуживания (СМО): каналов, накопителей, блокировок, разветвлений, уточняются все параметры элементов математической модели. При этом, поскольку ожидание обслуживания в СМО происходит в накопителях, каждому ожиданию на временной диаграмме сопоставляется накопитель. Поскольку обслуживание заявок в СМО производится в каналах, каждому действию обслуживания на временной диаграмме сопоставляется канал. Таким образом получается модель ДС в виде композиции СМО – сети массового обслуживания.

Разработка событийного графа

На этом этапе математическая модель ДС в виде композиции СМО преобразуется в математическую модель в виде событийного графа. Событийный граф используется для формализации процесса преобразования математической модели в программную модель.

Построение событийного графа состоит из следующих шагов:

1. Выделение статических и динамических (процессов и активностей) объектов.
2. Выделение событий.
3. Определение связей между событиями и построение событийного графа.

На первом шаге выделяются средства, очереди, активности и процессы, составляющие модель дискретной системы.

Выделение средств и очередей может производиться по математической модели ДС в виде композиции СМО: каждому элементу СМО типа "канал обслуживания" ставится в соответствие элемент функциональной модели "средство", каждому элементу СМО типа "накопитель" – элемент "очередь" с теми же параметрами.

Активности и процессы выделяются по описанию процесса функционирования ДС, прежде всего по временной диаграмме процесса

функционирования. Элементарные действия, время выполнения которых не равно нулю, представляются активностями. Связанные последовательности действий, начинающиеся с появления заявки в системе и заканчивающиеся уходом заявки из системы, объединяются в процессы.

В модель ДС включаются объекты, связанные не только с моделируемой ДС, но и с самим процессом моделирования. Поскольку за время машинного эксперимента необходимо для набора статистических данных выполнить множество процессов обслуживания, вводится дополнительный объект – процесс моделирования $\boldsymbol{П_м}$. Процесс моделирования $\boldsymbol{П_м}$ является глобальным процессом, вмещающим как элементы все моделируемые процессы.

На <u>втором шаге</u> по совокупности выделенных статических и динамических объектов определяется множество событий модели ДС. Для этого для каждого объекта определяются все возможные изменения состояний, которым сопоставляются события в модели. Средство $\boldsymbol{S_i}$ может иметь только два состояния: "занято" и "свободно". Для него возможны только два изменения состояний: с "занято" на "свободно" и наоборот. Этим изменениям состояний сопоставляются следующие события:

$\boldsymbol{e_{зsi}}$ – событие занятия средства $\boldsymbol{S_i}$,

$\boldsymbol{e_{osi}}$ – событие освобождения средства $\boldsymbol{S_i}$.

Конечная очередь $\boldsymbol{Q_h}$ с максимальной длиной $\boldsymbol{L}$ имеет $\boldsymbol{L+1}$ состояние и $\boldsymbol{2L}$ изменений состояний, которым могут быть сопоставлены события. Однако при составлении событийного графа часто удобнее выделять два состояния очереди: "занята данной заявкой" и "свободна от данной заявки". В этом случае возможны только два изменения состояния очереди, которым соответствуют два события:

$\boldsymbol{e_{зoh}}$ – событие занятия очереди $\boldsymbol{Q_h}$ заявкой,

$\boldsymbol{e_{ooh}}$ – событие освобождения очереди $\boldsymbol{Q_h}$ заявкой.

Активности и процессы имеют по два состояния: "выполняется" и "не выполняется". Для них возможны два изменения состояний: с "не выполняется" на "выполняется" и наоборот. Этим изменениям сопоставляются следующие события:

$\boldsymbol{e_{наj}}$ – событие начала активности $\boldsymbol{a_j}$,

$\boldsymbol{e_{каj}}$ – событие конца активности $\boldsymbol{a_j}$,

$\boldsymbol{e_{нпи}}$ – событие начала процесса $\boldsymbol{П_и}$,

$\boldsymbol{e_{кпи}}$ – событие конца процесса $\boldsymbol{П_и}$.

Процессу моделирования $\boldsymbol{П_м}$ также соответствуют два события:

$\boldsymbol{e_{нпм}}$ – событие начала процесса моделирования $\boldsymbol{П_м}$,

$\boldsymbol{e_{кпм}}$ – событие конца процесса моделирования $\boldsymbol{П_м}$.

Кроме того, поскольку все планируемые события $\boldsymbol{e_{каj}}$ могут быть отменены при вмешательстве процесса с более высоким приоритетом, каждой активности может быть поставлено в соответствие третье событие: $\boldsymbol{e_{окаj}}$ – событие отмены события $\boldsymbol{e_{каj}}$. Таким образом, если нас не интересует

дифференциация событий окончания активностей, то мы вводим два события. Если интересует, то три. То же самое для процесса: либо два события $(e_{нпи}, e_{кпи})$, либо больше, по числу различаемых исследователем окончаний процесса $(e_{нпи}, e_{кпи}, e_{кпиN1}, e_{кпиN2}, ...)$.

На третьем шаге на основе содержательного анализа причинно-следственных связей между изменениями состояний моделируемой ДС совокупность вершин, соответствующих выделенным событиям, соединяется дугами и получается событийный граф.

На рис.1. приведен пример простейшей модели ДС в виде СМО, состоящей из конечного накопителя ***Н*** и канала ***К***. Источник ***И*** генерирует входной поток заявок на обслуживание N_1, N_2 – выходной поток обслуженных заявок, N_3 – выходной поток заявок получивших отказ в обслуживании в связи с конечностью очереди.

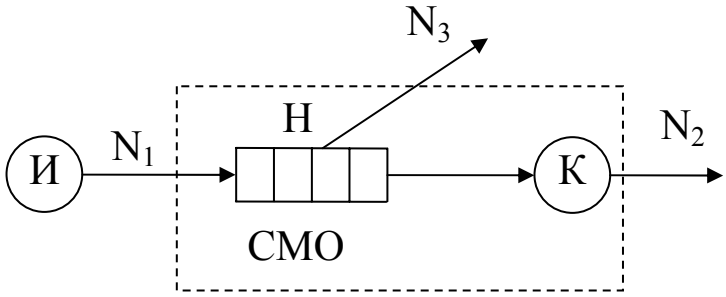

Рис.1. Система массового обслуживания

Соответствие элементов модели в виде СМО и в виде событийного графа для рассматриваемого примера представлено в таблице 1.

Таблица 1

Модель СМО	**Событийная модель**	
		События
Каналы обслуживания	**Статические объекты**	
Канал ***К***	Средство ***Sr***	$e_{зS}$, $e_{оS}$
Накопители		
Накопитель ***Н***	Очередь ***Q***	$e_{зQ}$, $e_{оQ}$
Действия	**Динамические объекты**	
Обслуживание в ***К***	Активность ***а***	$e_{на}$, $e_{ка}$
Последовательности действий	Процессы	
Обслуживание в СМО	Процесс обслуживания ***П***	$e_{нп}$, $e_{кпN2}$, $e_{кпN3}$
	Процесс моделирования $П_м$	$e_{нпм}$, $e_{кпм}$

Исходный событийный граф, представляющий процесс функционирования ДС, приведен на рис. 2.

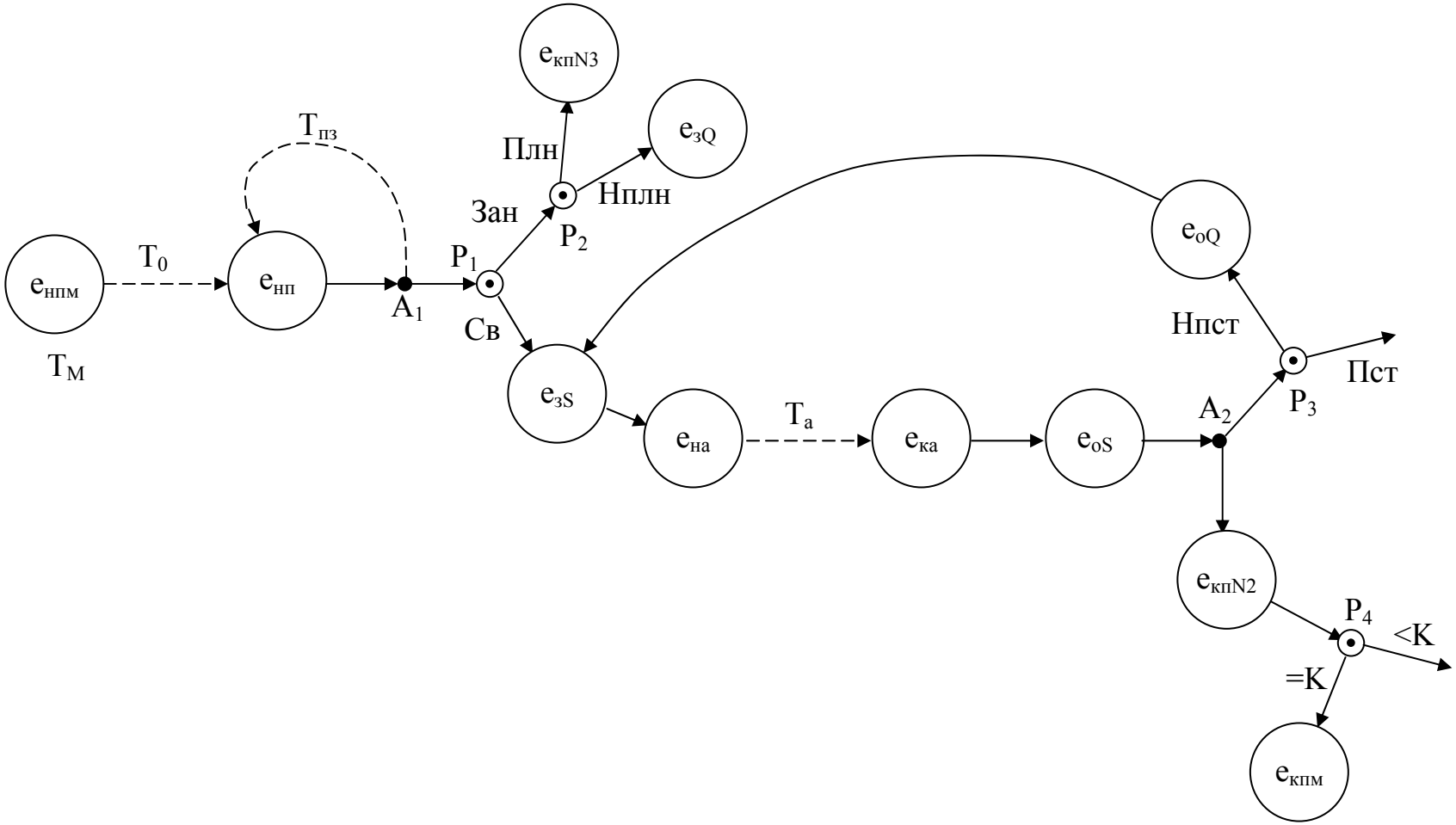

Рис. 2. Исходный событийный граф процесса функционирования СМО

Окончание процесса моделирования определяется по числу обслуженных заявок ***K***. Связь во времени между событиями появления заявок в системе $e_{нп}$ отображается дугой второго типа, помеченной интервалом времени между появлениями заявок $T_{пз}$.

В событийном графе используются вершины второго типа, которым соответствуют следующие условия:

P_1 – анализ состояния средства ***Sr***, значения: ***Зан*** – "занято" (0), ***Св*** – "свободно" (1);

P_2 – анализ состояния очереди ***Q***, значения: ***Плн*** – "полна" (0), ***Нплн*** – "неполна" (1);

P_3 – анализ состояния очереди ***Q***, значения: ***Пст*** – "пуста" (0), ***Нпст*** – "непуста" (1);

P_4 – анализ числа обслуженных заявок, значения: ***<K*** – меньше заданного числа ***K*** (0), ***=K*** – равно заданному числу ***K*** (1).

Вершины A_1, A_2 – это вершины распараллеливания процесса.

Дуги второго типа помечаются величинами задержки:

T_0 – время поступления первой заявки после начала моделирования,

$T_{пз}$ – интервал времени между поступлениями заявок,

T_a – время обслуживания заявки в канале (длительность активности).

Поскольку конец моделирования определяется в модели по числу заявок прошедших через модель, $e_{кпм}$ происходит после события $e_{кпN2}$ при условии, что через модель прошло заданное число заявок ***K*** ($P_4 = K$).

Разработка программной модели

На следующем этапе событийный граф ДС преобразуется в программную модель на языке высокого уровня. Это осуществляется в два шага:

1) преобразование событийного графа в программно-реализуемую форму,
2) программирование модели путем интерпретации элементов событийного графа.

Преобразование событийного графа в программно-реализуемую форму

Программная модель на событийно-ориентированном языке представляет собой совокупность инициирующей, управляющей и событийных СЕКЦИЙ (рис. 3) [1].

Связь между событийными секциями осуществляется через механизм планирования системы моделирования. Порядок выполнения событийных секций определяется управляющей секцией по СПИСКУ СОБЫТИЙ. Список событий включает будущие события, то есть события, которые должны произойти. Для каждого события в списке событий

хранится номер события $\boldsymbol{E_i}$, номер процесса $\boldsymbol{J}$, которому принадлежит событие, и время события $\boldsymbol{T_{Ei}}$.

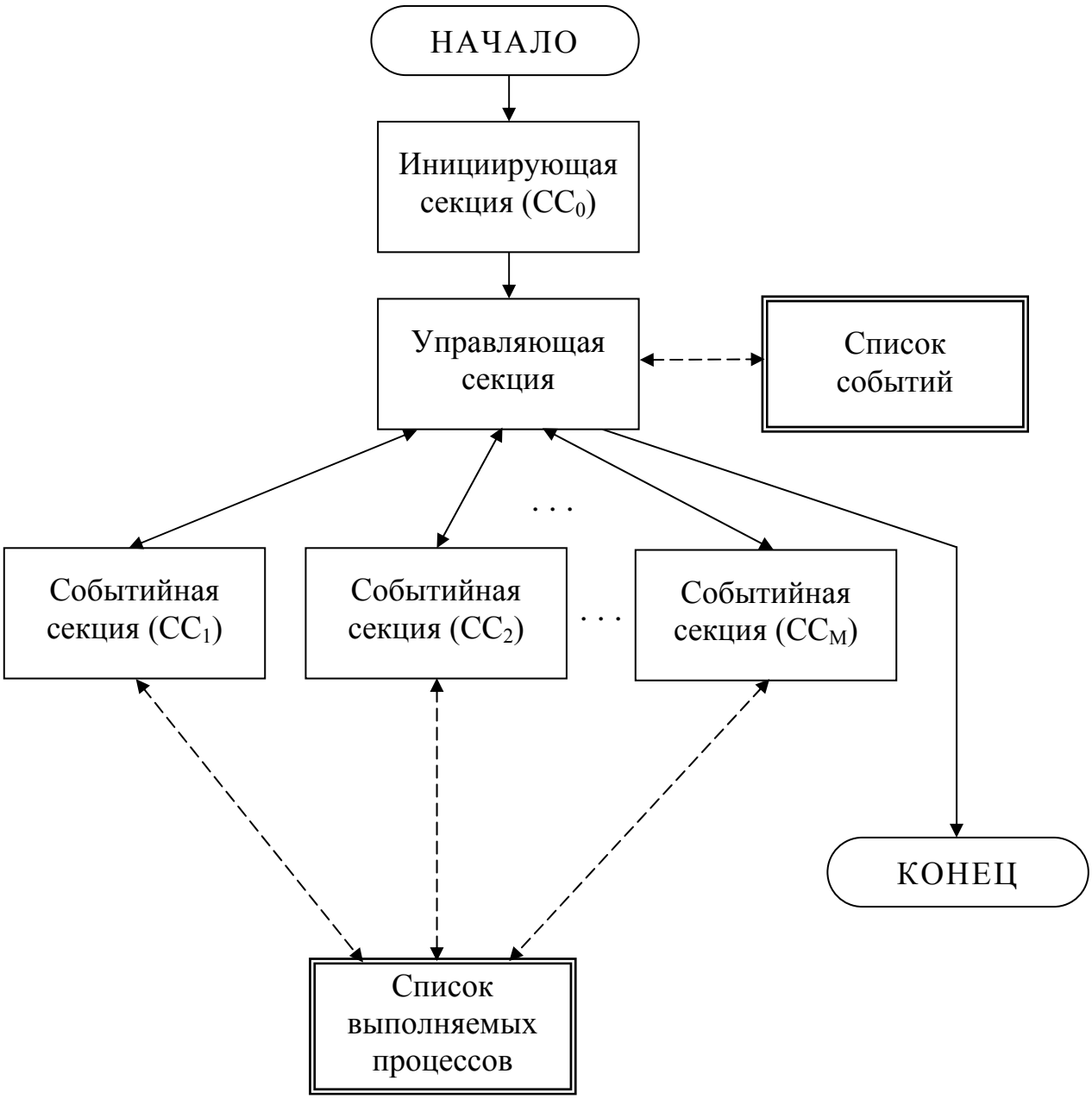

Рис. 3. Структура программной модели

Для хранения параметров и накапливаемых характеристик процессов служит СПИСОК ВЫПОЛНЯЕМЫХ ПРОЦЕССОВ.

В связи с такой организацией программной модели событийный граф, полученный на предыдущем этапе, необходимо преобразовать в программно-реализуемую форму в виде совокупности событийных секций.

Событийным секциям соответствуют подграфы событийного графа, которые удовлетворяют следующим условиям:

1) подграф имеет не более одной точки входа;

2) все дуги, соединяющие вершины внутри подграфов, являются дугами первого типа;

3) все дуги, соединяющие подграф с остальными подграфами событийного графа, являются дугами второго типа.

Первое требование возникает в связи с тем, что подграф реализуется в виде процедуры, имеющей только один вход (начало). В случае нескольких входов необходимо при обращении к процедуре задавать номер входа, а в теле процедуры вводить условия для передачи управления в соответствующие точки тела процедуры.

ТОЧКОЙ ВХОДА ***I*** называется вершина подграфа, в которую входит одна или несколько дуг второго типа из других подграфов событийного графа.

ТОЧКОЙ ВЫХОДА ***O*** подграфа называют вершину, из которой выходит дуга второго типа в другой подграф событийного графа. У подграфа может быть несколько точек выхода или ни одной.

Второе требование возникает в связи с тем, что дуги первого типа реализуются обычными программными средствами: расположением программных операторов в соответствии со связями вершин событийного графа и использованием операторов передачи управления.

Третье требование возникает в связи с тем, что дуги второго типа реализуются через механизм планирования системы моделирования.

Преобразование событийного графа в программно-реализуемую форму производится следующим образом.

Определяются точки входа и выхода подграфов. Далее выделяются связанные подмножества вершин, соединяющиеся дугами первого типа, например, путем удаления дуг второго типа. Если для выделенного подграфа выполняется первое условие, то есть он имеет не более одной точки входа, то данный подграф может быть реализован в виде программной событийной секции. Таким образом, событийная секция объединяет группу событий, то есть является сложным событием или МАКРОСОБЫТИЕМ.

Если подграф имеет несколько точек входа и параллельные участки, то он должен быть преобразован в программно-реализуемую форму. Основным требованием при преобразовании событийных графов является сохранение адекватности модели, то есть сохранение последовательностей существенно важных с точки зрения цели моделирования событий.

Преобразование подграфа с целью уменьшения числа точек входа может быть произведено следующими способами:

- перенесением концов дуг подграфов;
- заменой дуг первого типа дугами второго типа;

– дублированием вершин и соединяющих их дуг;
– перестановкой вершин подграфа.

При первом способе концы дуг второго типа перемещаются из вершины в вершину при условии сохранения адекватности модели.

При втором способе некоторые дуги первого типа заменяются дугами второго типа с задержкой $T = 0$, которые реализуются через механизм планирования таким образом, чтобы подграф распадался на подграфы, удовлетворяющие условиям.

Выделим подграфы и соответствующие им событийные секции (макрособытия) для рассматриваемого примера (см. рис. 2). Для этого (рис. 4):

– заменим дугу первого типа $A_1 - P_1$ дугой второго типа с задержкой 0;

– переместим конец дуги $e_{oQ} - e_{зS}$ в вершину P_1, это возможно, поскольку перед событием e_{oQ} выполняется событие e_{oS} освобождения средства S, задержки между событиями равны нулю, следовательно, значение условия P_1 будет Св (свободно) и произойдет переход как и в исходном графе к событиям $e_{зS}$ и $e_{на}$;

– заменим измененную дугу первого типа $e_{oQ} - P_1$ дугой второго типа с задержкой 0;

– для выделения события конца моделирования заменим дугу первого типа $P_4 - e_{кпм}$ дугой второго типа с задержкой 0.

В результате этого мы получим следующие макрособытия:

E_0 – безусловное макрособытие начала моделирования, точек входа нет, одна точка выхода O_1 – вершина $e_{нпм}$;

E_1 – безусловное макрособытие поступления заявки в СМО, одна точка входа I_1 – вершина $e_{нп}$, одна точка выхода O_1 – вершина A_1;

E_2 – условное макрособытие начала обслуживания заявки в СМО, одна точка входа I_1 – вершина P_1, одна точка выхода O_1 – вершина $e_{на}$;

E_3 – условное макрособытие окончания обслуживания заявки в СМО, одна точка входа I_1 – вершина $e_{ка}$, две точки выхода: O_1 – вершина e_{oQ}, и O_2 – вершина P_4;

E_4 – безусловное макрособытие окончания моделирования, одна точка входа I_1 – вершина $e_{кпм}$, точек выхода нет.

Событийный граф, состоящий из макрособытий $E_0 - E_4$, приведен на рис. 5. Макрособытия связаны только дугами второго типа. Дуги помечаются парой **<условие> / <время>**, прочерк означает безусловный переход. Если у макрособытия больше одного выхода, то они помечаются: пример – событие E_0 имеет два выхода O_1 и O_2.

Событийная модель направлена, с точки зрения экспериментатора, на глобальное, двухуровневое представление системы.

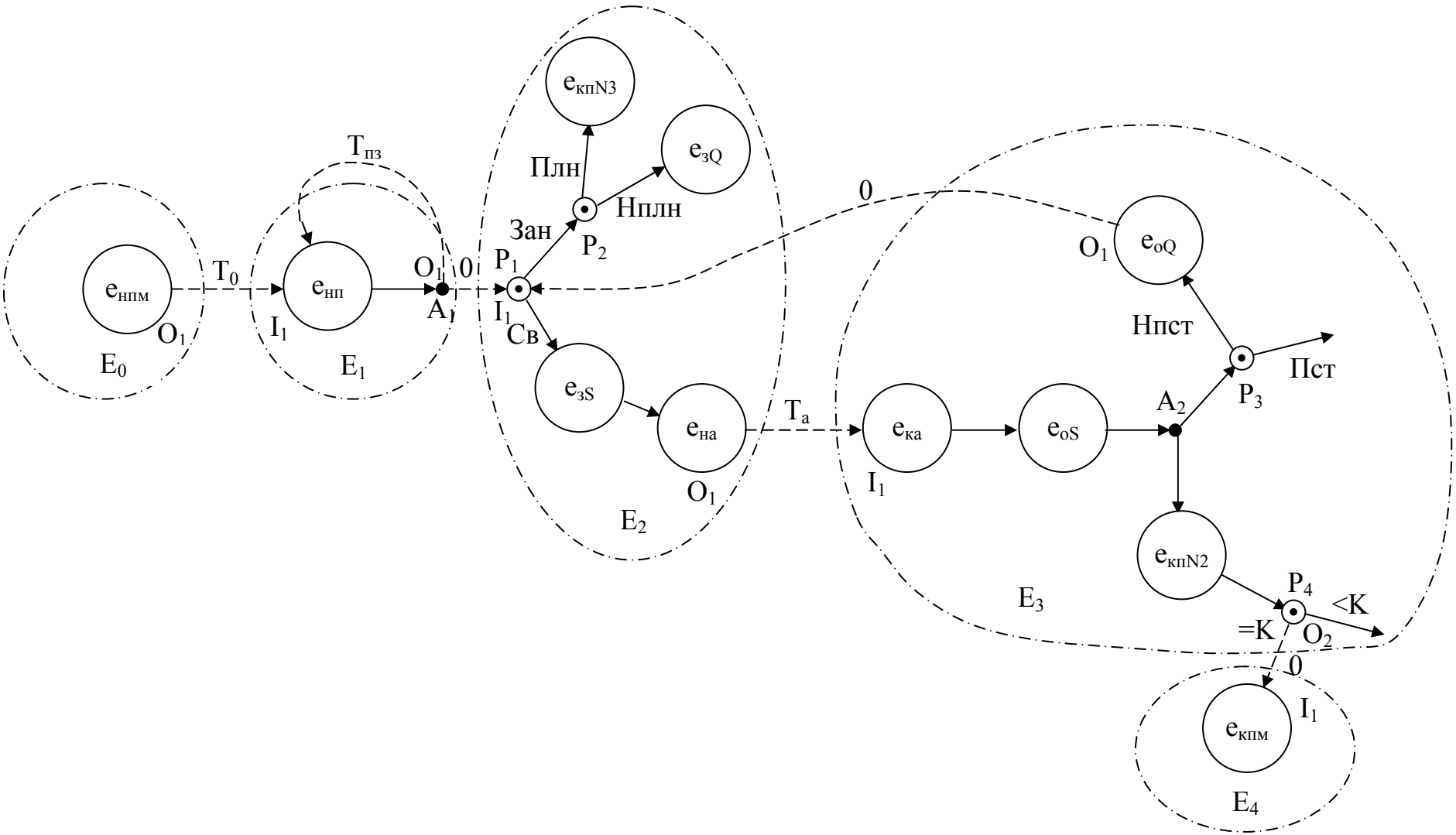

Рис. 4. Событийный граф в программно-реализуемой форме

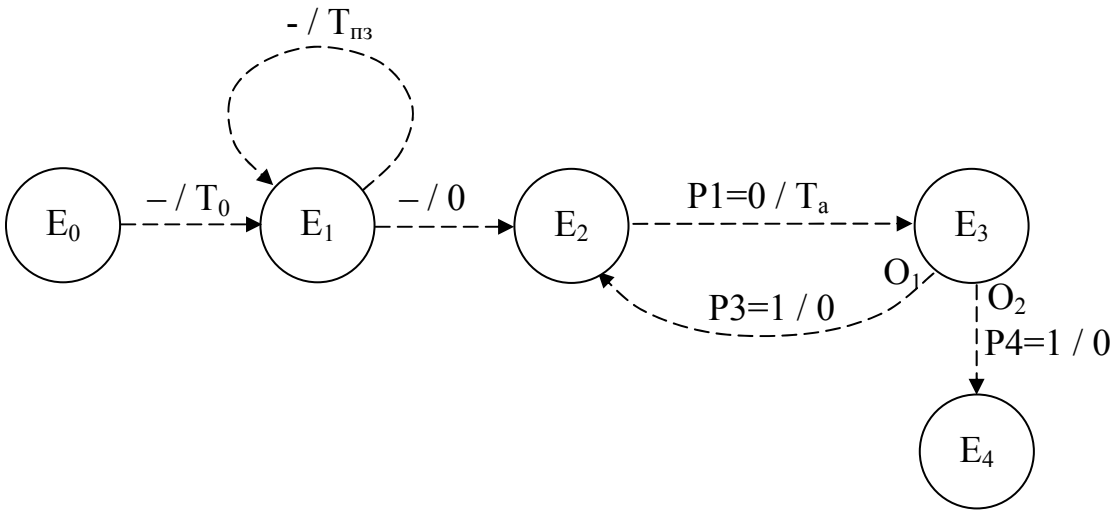

Рис. 5. Событийный граф на уровне макрособытий

В модели проявляется «тенденция к объединению в рамках одной процедуры операций, относящихся к не связанным логически активностям» и процессам [1]. В результате большая модель, описанная в терминах событий, «может потерять всякое сходство со структурой реальной системы и затруднить ее модификацию» [1]. Появление такой ситуации наиболее вероятно в том случае, если предпринимается попытка описать сложную систему, прежде всего в терминах изменения ее состояний, то есть событий. В связи с этим необходимо рассматривать функционирования системы под углом происходящих в ней процессов и соответствующим образом организовать событийный граф и программную модель [1].

Рассмотрим пример взаимодействия процессов (рис. 6). Процесс ***П1*** обслуживается (активность a_x) средством ***S1***. В это время процесс ***П2*** ждет обслуживания средством ***S1***. После окончания активности обслуживания a_x процесс ***П1*** начинает обслуживаться (активность a_{x+1}) средством ***S2***, а процесс ***П2*** начинает обслуживаться (активность a_y) средством ***S1***. Активности a_x и a_{x+1} относятся к одному и тому же процессу ***П1***, а активность a_y – к другому ***П2***, совершенно не связанному с первым. На временной диаграмме в момент ***t*** отмечены все происходящие события:

$e_{кax}$ – событие конца активности a_x процесса ***П1***;

e_{oS1} – событие освобождения средства ***S1***;

$e_{зS2}$ – событие занятия средства ***S2***;

$e_{на(x+1)}$ – событие начала активности a_{x+1} процесса ***П1***;

$e_{зS1}$ – событие занятия средства ***S1***;

$e_{нау}$ – событие начала активности a_y процесса ***П2***.

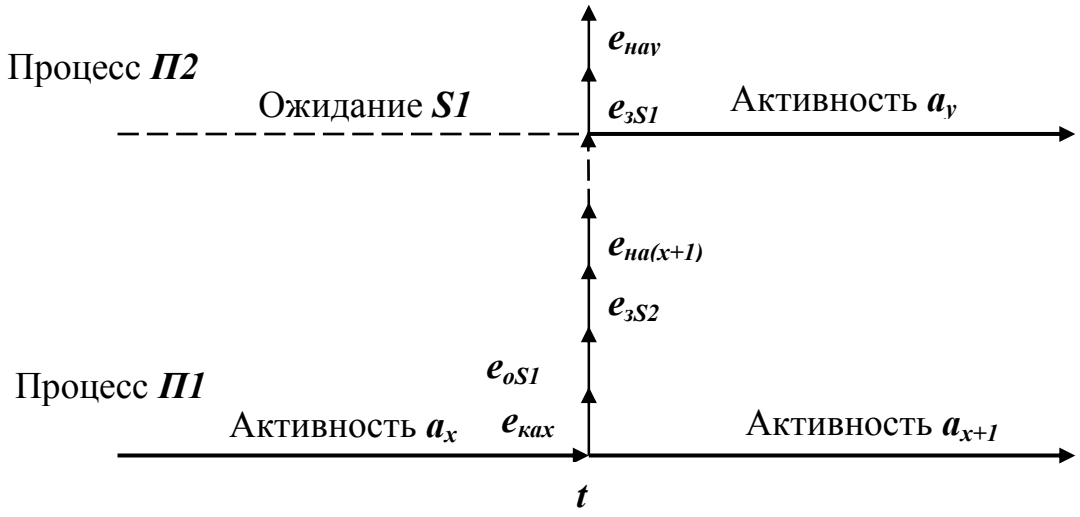

Рис. 6. Временная диаграмма взаимодействия процессов

На рис. 7 показан подграф макрособытия E_i, полученный по вышерассмотренной методике. В макрособытии E_i присутствуют события, относящиеся к различным процессам.

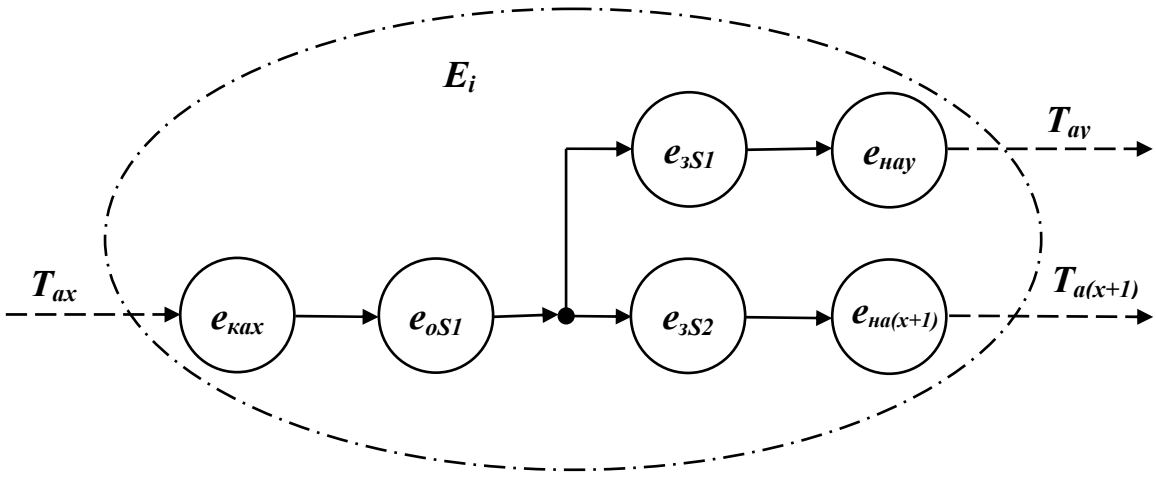

Рис. 7. Фрагмент событийного графа для двух взаимодействующих процессов

Разделение описаний процессов возможно двумя способами:

1) заменой дуг первого типа на дуги второго типа;
2) вызовом процедуры макрособытия E_j из процедуры макрособытия E_i.

На рис. 8 приведена декомпозиция макрособытия E_j на два макрособытия E_j и E_i заменой дуг первого типа на дуги второго типа.

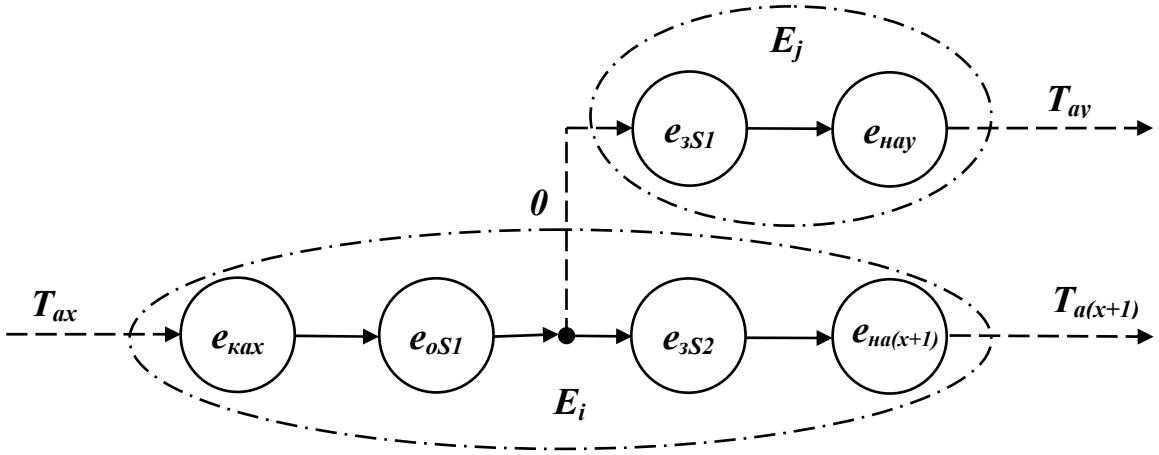

Рис. 8. Первый способ разделения описаний процессов

Здесь макрособытие E_j представляет завершение активности a_x и начало выполнения активности a_{x+1} процесса 1. После этого макрособытия без задержки безусловно следует макрособытие E_j, которое представляет в модели начало выполнения активности a_y процесса 2. Таким образом разделяются описания событий различных процессов. Событийная секция ***CCj***, программно реализующая макрособытие E_j, в этом случае является самостоятельной процедурой вызываемой через механизм планирования и выполняемой одновременно с ***CCi*** макрособытия E_i.

Событийная секция ***CCj*** моделирующей программы, реализующая макрособытие E_j, может быть вспомогательной и вызываться в качестве подпрограммы из событийной секции ***CCi*** реализующей макрособытие E_i (см. рис. 9).

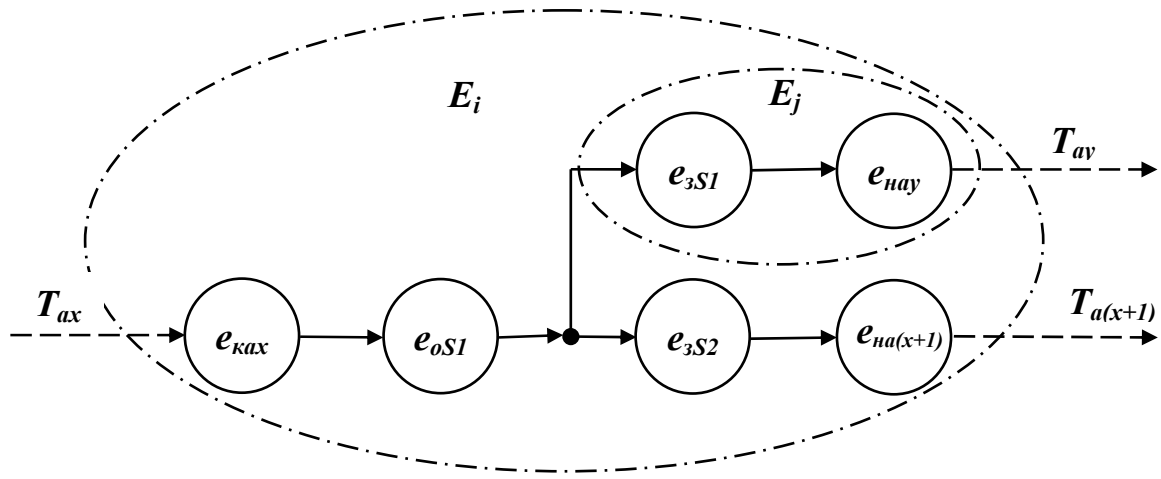

Рис. 9. Второй способ разделения описаний процессов

Первый способ может быть более удобным, так как в этом случае обеспечивается независимое следование процедур. Вызванная процедура ***CCj*** может не возвращать управление вызвавшей ее процедуре ***CCi***.

Таким образом, указанные способы позволяют разделить описания макрособытий различных процессов. На следующем уровне описания макрособытий объединяются в рамках описаний процессов. На всех уровнях описаний необходимо следить за независимостью описаний процессов. Разделение описаний позволяет уменьшить трудоемкость изменения модели, например при включении или удалении активностей. Этот подход имеет большое значение при построении сложных моделей [1].

Программирование модели

Событийный граф в программно-реализуемой форме преобразуется в программную модель путем интерпретации всех его элементов (дуг и вершин). Преобразование событийного графа в программно-реализуемую форму состоит в замене каждого элемента графа операторами языка высокого уровня.

Эта процедура может выполняться в два шага:

– преобразование событийных подграфов в алгоритмы событийных секций,

– собственно разработка программы моделирования на основе интерпретации операторных и условных вершин алгоритмов.

На первом шаге сначала осуществляется преобразование параллельных ветвей графа в последовательные, а затем элементы событийных подграфов заменяются операторами алгоритма.

Если в подграфе есть вершина третьего типа, то некоторые его вершины могут выполняться одновременно. При преобразовании подграфа в алгоритм необходимо параллельные ветви подграфа, начинающиеся с вершины третьего типа, преобразовывать в последовательные. Если ветви состоят из независимых вершин, то возможно в алгоритме располагать операторы ветвей последовательно в любом порядке. Если ветви состоят из зависимых вершин, то необходимо при преобразовании в последовательную форму рассматривать все возможные комбинации условий и взаимодействий.

Поскольку событиям в событийном графе соответствуют действия в программной модели, замена элементов событийных подграфов операторами алгоритма осуществляется следующим образом.

Событию $e_{нпм}$ соответствует группа операторов инициализации модели:

– установка в 0 переменной времени моделирования и установка в начальное состояние всех остальных переменных модели;

– установка в начальное состояние списков системы моделирования: списка выполняемых процессов, списка событий и других;

– объявление статических объектов модели: средств, ресурсов и очередей.

Событию $e_{кпм}$ соответствует завершение программы и вывод накопленной информации в виде отчета о моделировании.

Событию $e_{нп}$ соответствует группа операторов, выполняющих:

– генерацию параметров экземпляра процесса (заявки),

– запись параметров экземпляра процесса в список выполняемых процессов.

Событию $e_{кп}$ соответствует группа операторов удаления процесса из списка выполняемых процессов.

Событийные вершины типа $e_{зS}$, $e_{оS}$, $e_{зQ}$, $e_{оQ}$ заменяются операторами занятия и освобождения средств и очередей.

Событиям $e_{на}$ и $e_{ка}$ либо не соответствует ни один оператор, либо соответствуют операторы измерения временных характеристик модели, например операторы подсчета числа активностей, вычисления среднего и распределения времени между заданными событиями.

Логическим вершинам P_j событийного графа ставятся в соответствие логические вершины алгоритма. В программной модели вершинам второго типа P_j может соответствовать группа операторов, состоящая из операторов формирования значения логического условия и операторов условного перехода.

Дуги первого типа определяют порядок следования операторов в алгоритме и программе.

Дугам второго типа соответствуют операторы планирования события (помещение события в список событий), причем эти операторы располагаются в тех событийных секциях, откуда исходит дуга. Операторы планирования событий помещают планируемое событие E_i в список событий.

На рис. 10 приведен пример преобразования подграфа макрособытия E_3 в алгоритм событийной секции ***CC3*** программной модели.

В этом алгоритме две параллельные ветви реализуются последовательно, вначале ветвь окончания процесса обслуживания, а затем ветвь анализа состояния очереди и вызова на обслуживание первого процесса ожидающего в очереди.

При программировании полученных алгоритмов на языке высокого уровня необходимо использовать библиотеки готовых процедур или создавать процедуры выполняющие функции операторных и условных вершин алгоритма.

Рис. 10. Алгоритм событийной секции СС3

Заключение

Таким образом, в настоящей работе предлагается метод синтеза математических и программных событийных моделей, представляемых в виде событийного графа в алгоритмической форме или событийного алгоритма. Для дискретных систем, имеющих в основе процессы обслуживания, предлагается в качестве промежуточной математической модели использовать модель в виде сети массового обслуживания, которая затем преобразуется в событийный граф. Рассмотрено также преобразование событийного графа в программную модель. Определены ограничения при преобразовании в программную модель. Введена промежуточная модель графа: событийный граф в программно-реализуемой форме, которая является разновидностью событийного графа на уровне макрособытий. Рассмотрено преобразование параллельных подграфов макрособытий в последовательные алгоритмы, реализуемые в программной модели ДС.

Библиографический список

1. Автоматизация проектирования вычислительных систем. Языки, моделирование и базы данных / Под ред. М. Брейера. – М.: Мир, 1979. – С. 12–29, 35–44.
2. Schruben L.W. Simulation Modeling with Event Graphs. // Communications of the ACM. – 1983. – Vol. 26. – Num. 11. – P. 957–963.
3. Бабкин Е.А. Методические указания по моделированию вычислительных систем на событийно-ориентированном языке. – Курск: КПИ, 1988.
4. Бабкин Е.А. Событийные модели дискретных систем / Курск. гос. ун-т. – Курск, 2005. – 18 с. Деп. в ВИНИТИ 14.01.05, № 30–В2005.