УДК 519.711.3

О МЕТОДАХ ДЕКОМПОЗИЦИИ СОБЫТИЙНЫХ ГРАФОВ

© 2008 Е. А. Бабкин, Е. А. Бобрышев

доцент кафедры программного обеспечения и администрирования информационных систем, канд. техн. наук, доцент, bea@kursknet.ru

аспирант кафедры программного обеспечения и администрирования информационных систем, egbobryshev@mail.ru

Курский государственный университет

Рассматриваются структурный и объектный способы декомпозиции событийного графа. Предлагается способ объектной декомпозиции событийного графа, что позволяет получить объектную модель по известному событийному графу, определить в терминах событий и макрособытий поведение объектов, преобразовать событийный граф в объектную программную модель. Рассмотрены два варианта реализации декомпозиции: реализация дуг первого типа, во-первых, с помощью механизма планирования и, во-вторых, непосредственной передачей управления между процедурами. Рассмотрена модификация событийного графа – событийный граф с дорожками, в котором в явном виде представляется взаимосвязь между событиями и объектами модели.

Ключевые слова: событие, событийные графы, макрособытие, структурный способ декомпозиции, объектный способ декомпозиции, событийный граф с дорожками.

1. Введение

При преобразовании событийной модели в программную модель необходимо выделение событийных секций – частей или процедур программной модели [Автоматизация… 1979], то есть декомпозиция модели. Наиболее наглядным и удобным средством описания при дискретно-событийном моделировании является представление модели в виде событийного графа (СГ) [Schruben 1983; Бабкин 1988]. В работах [Бабкин 2005; Бабкин 2006] рассматривается декомпозиция событийного графа, которая лежит в основе преобразования СГ в программную модель. При этом СГ разделяется на связные подграфы – макрособытия, удовлетворяющие определенным структурным ограничениям [Бабкин 2006]. Эти подграфы в программной модели реализуются в виде процедур или частей программной модели. Будем называть такую декомпозицию СГ структурной или процедурной.

Процесс представления предметной области задачи в виде совокупности объектов, обменивающихся сообщениями, называется объектной декомпозицией [Буч 2004; Леоненков 2004]. Поскольку дискретно-событийные системы могут быть представлены в виде совокупности статических и динамических объектов, возможна декомпозиция СГ, основанная на выделении подграфов по признаку принадлежности событийных и условных вершин статическим и динамическим объектам модели [Бабкин 2007; Бабкин 2007а]. Будем называть такую декомпозицию СГ объектной.

Рассмотрим эти два подхода к декомпозиции СГ.

2. Структурная декомпозиция событийного графа

Рассмотрим систему массового обслуживания (СМО) с конечным накопителем и ограничением по времени ожидания в накопителе (рис. 1).

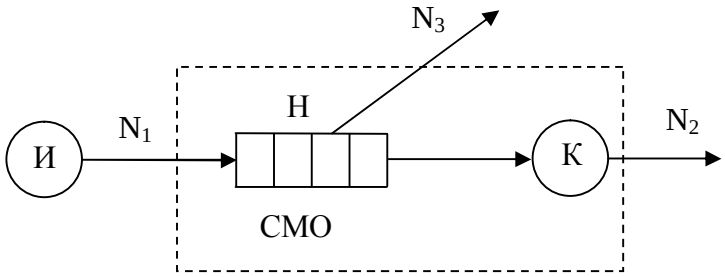

Рис. 1. Система массового обслуживания

Заявки поступают из источника ***И*** в СМО – входной поток N_1. СМО включает накопитель ***Н*** и канал ***К***. В СМО существует два выходных потока заявок: N_2 – поток заявок, обслуженных в канале ***К***, и N_3 – поток заявок получивших отказ в обслуживании в СМО либо поскольку накопитель полностью занят, либо из-за того, что время ожидания заявки превысило максимально допустимое время ожидания.

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

Соответствие элементов модели СМО и событийного графа

Модель СМО	**Событийная модель**	
		События
Каналы обслуживания	**Статические объекты**	
Канал ***К***	Средство ***К***	$e_{зК}$, $e_{оК}$
Накопители		
Накопитель ***Н***	Очередь ***Н***	$e_{зН}$, $e_{оН}$
Действия	**Динамические объекты**	
Обслуживание в ***К***	Активность ***а***	$e_{на}$, $e_{ка}$
Последовательность действий	Процессы	
Обслуживание в СМО	Процесс обслуживания ***П***	$e_{нп}$, $e_{кпN2}$, $e_{кпN3}$
	Процесс моделирования $П_м$	$e_{нпм}$, $e_{кпм}$

Здесь $e_з$ – событие занятия статического объекта;

$e_о$ – событие освобождения статического объекта;

$e_н$ – событие начала динамического объекта;

$e_к$ – событие конца динамического объекта.

Поскольку заявка освобождает очередь ***Н*** в двух случаях, необходимо различать следующие разновидности события $e_{оН}$:

$e_{о1Н}$ – освобождение очереди, когда заявка дождалась обслуживания, в этом случае заявка, стоящая в очереди первой, удаляется из очереди;

$e_{оJН}$ – освобождение очереди в связи с тем, что время ожидания в очереди достигло максимального $T_Н$, при этом заявка ***J***, для которой выполняется это условие, удаляется из очереди.

Исходный событийный граф, представляющий процесс функционирования ДС, приведен на рисунке 2. Связь во времени между событиями появления заявок в системе $e_{нп}$ отображается дугой второго типа, помеченной интервалом времени между появлениями заявок $T_{пз}$.

Окончание процесса моделирования определяется по числу ***i*** заявок, обслуженных в СМО. Поэтому событие $e_{кпм}$ происходит после события $e_{кпN2}$ при условии, что через модель прошло заданное число ***K*** заявок (***i = K***).

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

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

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

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

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

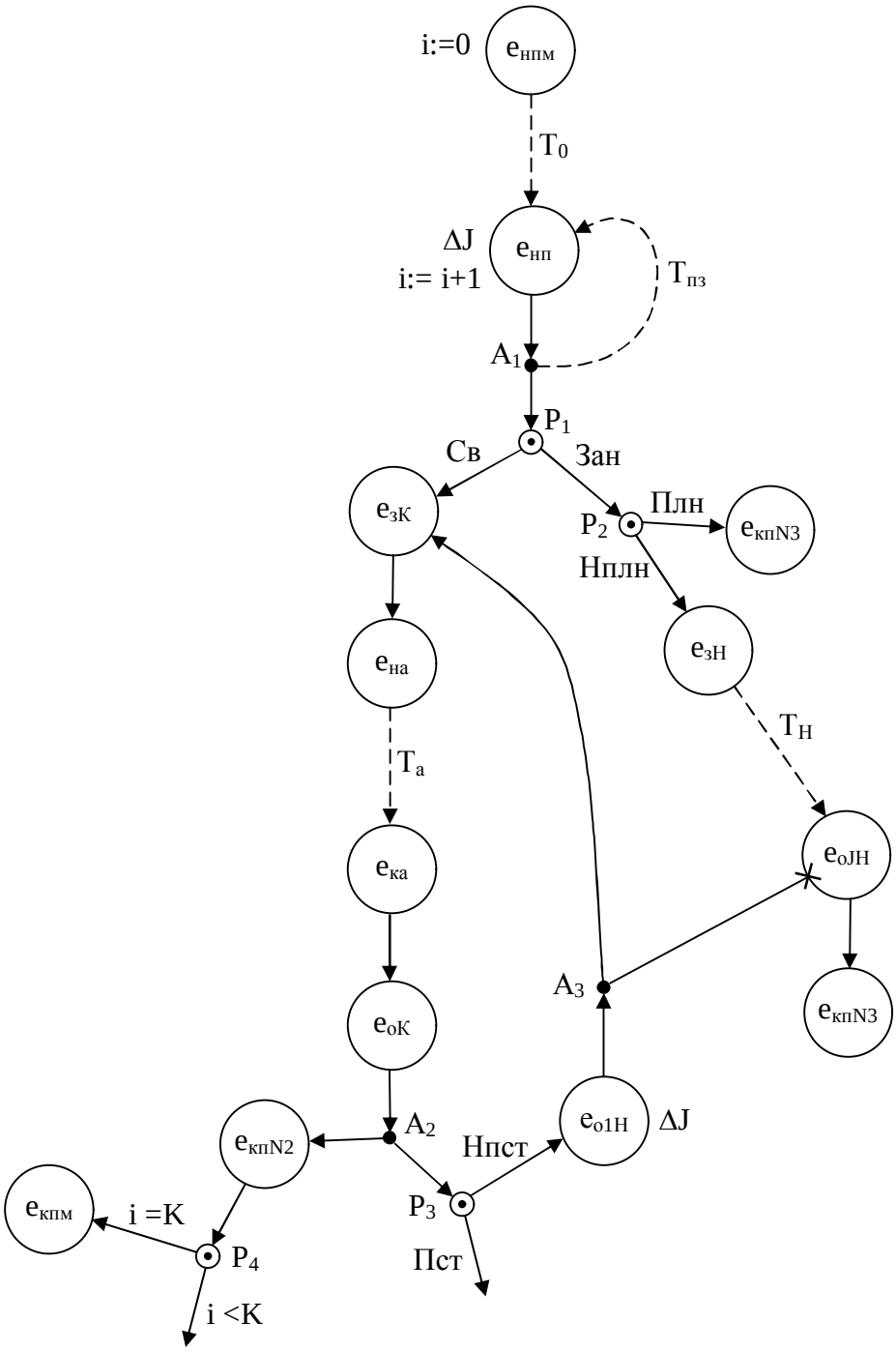

Рис. 2. Исходный событийный граф процесса функционирования СМО с ограничением по времени ожидания в очереди Q

Вершины A_1, A_2, A_3 представляют распараллеливание процесса.

Дуги второго типа помечены величинами задержки:

T_0 – время поступления первой заявки после начала моделирования,

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

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

T_H – максимальное время ожидания заявки в очереди.

Событийные вершины, в которых происходит смена активного процесса (активизация процесса J), помечены ΔJ.

Для перехода к программной модели в работе [Бабкин 2006] предложено разделять событийный граф на связные подграфы, удовлетворяющие 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.

В работе [Бабкин 2005] совокупность событий такого подграфа называется макрособытием.

Третье условие является обязательным при условии реализации переходов между макрособытиями только через механизм планирования.

В результате структурной декомпозиции на основе первых двух условий для рассматриваемого примера будут получены следующие макрособытия (рис. 3):

E_0 – начало моделирования;

E_1 – поступление заявки в СМО;

E_2 – начало обслуживания заявки в канале;

E_3 – окончание обслуживания заявки в канале и СМО;

E_4 – окончание обслуживания заявки в СМО в связи с тем, что время ожидания заявки в очереди достигло максимального T_Q ;

E_5 – окончание моделирования.

Событийный граф, состоящий из макрособытий E_0 – E_5, приведен на рисунке 4. Дуги помечаются парой ***<условие> / <время>***, прочерк означает безусловный переход.

В исходном графе вершины A и e_{oJH} связаны дугой отмены события. Эти вершины при декомпозиции попадают в различные подграфы макрособытий E_3 и E_4. Поскольку отмена начального события e_{oJH} подграфа ведет к отмене всех следующих за ним событий, то и макрособытия E_3 и E_4 также связаны дугой отмены.

Полученный граф макрособытий реализуется в виде набора процедур, соответствующих макрособытиям. Поскольку макрособытия связаны дугами первого и второго типа, то возможны два варианта программной реализации модели:

1) все дуги, связывающие макрособытия, реализуются через механизм планирования [Бабкин 2005; Бабкин 2006] (рис. 5);

2) дуги первого типа реализуются передачей управления между соответствующими процедурами, а дуги второго типа – через механизм планирования (рис. 6).

В обоих случаях вводится процедура ***Управление*** и информационная структура ***Список событий*** для реализации механизма планирования. На рисунках 5 и 6:

Вызов – программный вызов процедур макрособытий E_0 – E_5 из процедуры ***Управление***;

План. (E_i) – занесение (планирование) макрособытия E_i в список событий;

Отмена (E_i) – удаление макрособытия E_i из списка событий;

Удал. – удаление макрособытия, стоящего первым в списке событий, по его имени (номеру) процедура ***Управление*** вызывает (***Вызов***) соответствующую процедуру

этого макрособытия и устанавливает текущее время ***T*** равным запланированному времени этого макрособытия.

Второй вариант реализации предпочтительнее с точки зрения машинного времени выполнения моделирования. Однако событие E_2 не будет появляться в списке событий, что вносит некоторые трудности в процесс трассировки и отладки модели. Событие E_2 в этом случае является ненаблюдаемым при трассировке модели.

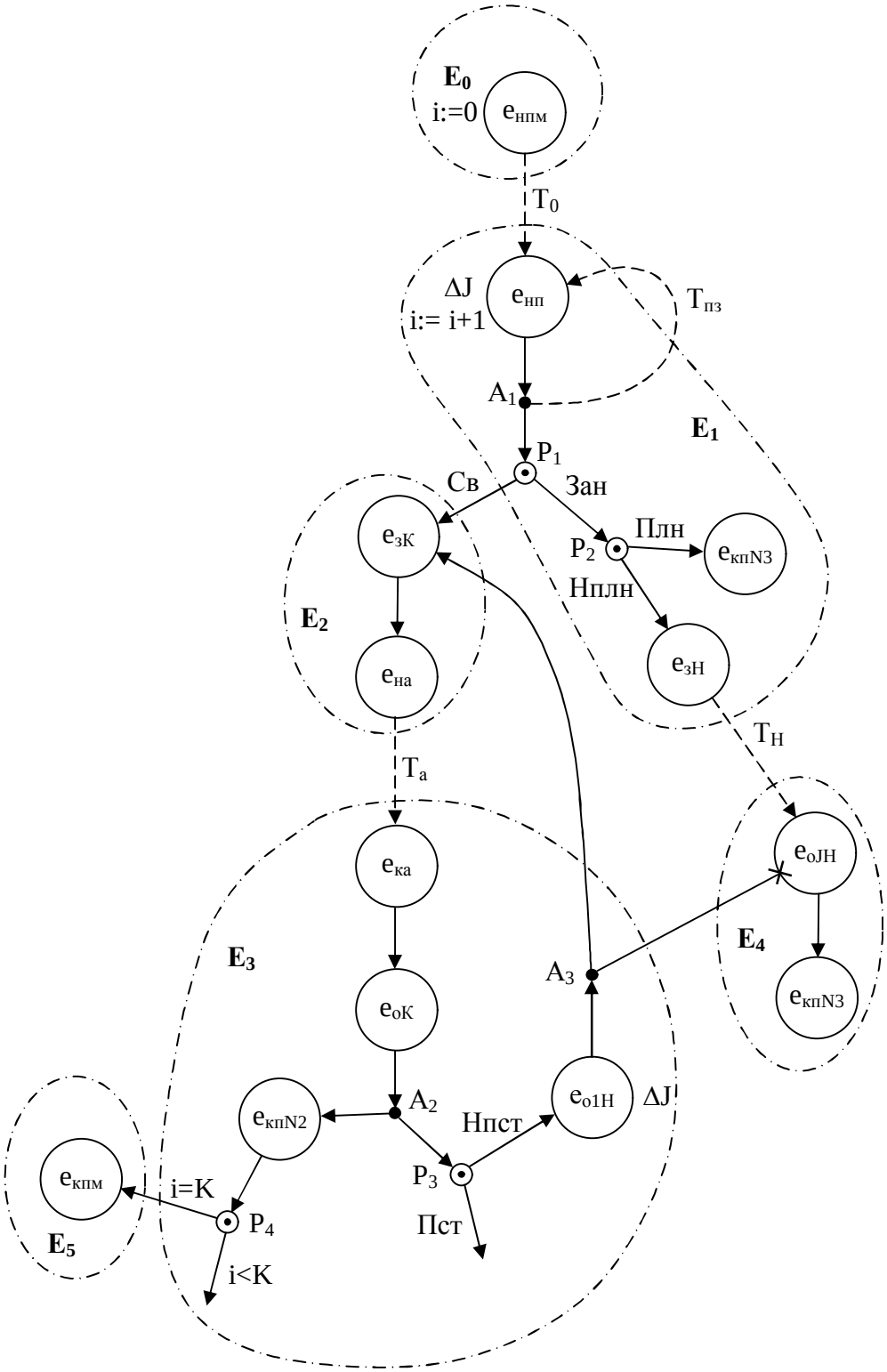

Рис. 3. Структурная декомпозиция событийного графа

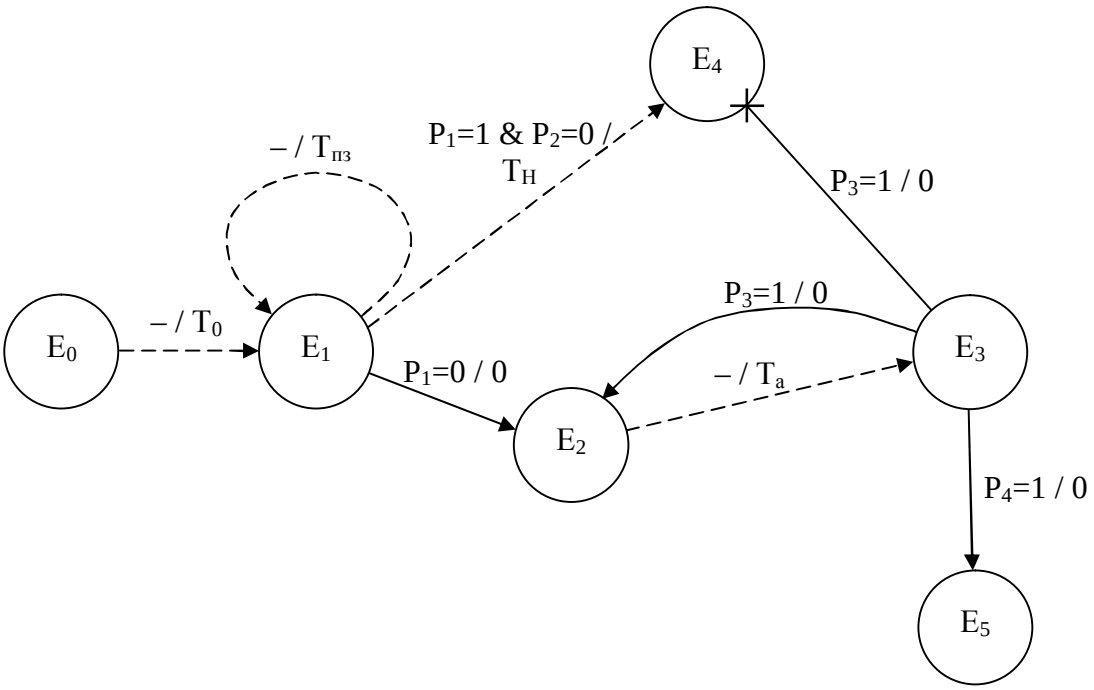

Рис. 4. Граф макрособытий процесса функционирования СМО

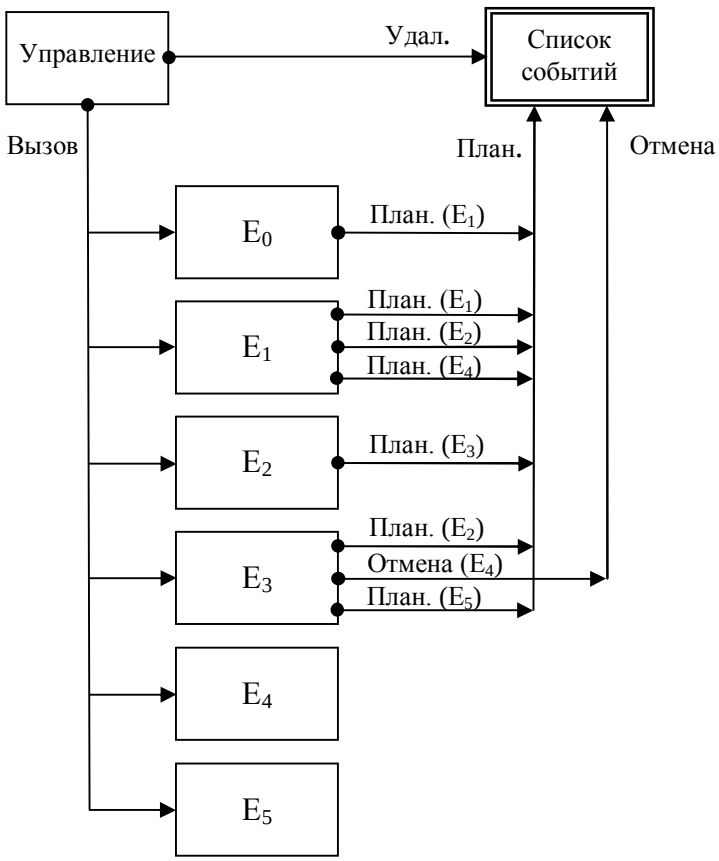

Рис. 5. Структура программной модели (первый вариант реализации)

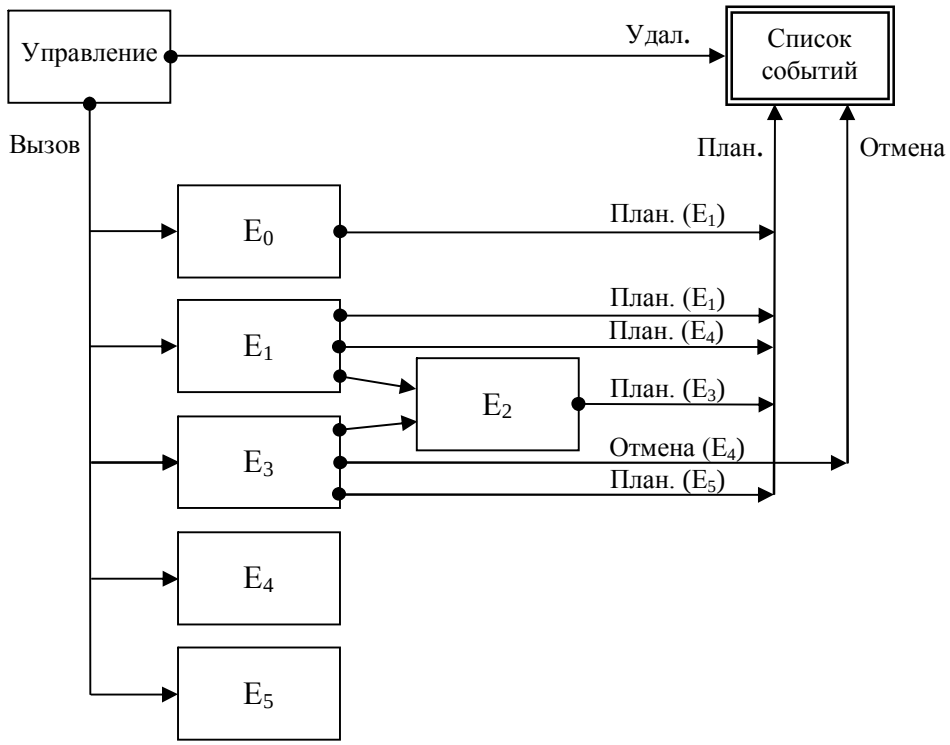

Рис. 6. Структура программной модели (второй вариант реализации)

3. Объектная декомпозиция событийного графа

Другой подход к декомпозиции СГ основан на выделении подграфов по признаку принадлежности событийных и условных вершин статическим и динамическим объектам модели.

Разделим событийный подграф на подграфы, состоящие из вершин, которые имеют отношение только к одному статическому или динамическому объекту модели: процессу моделирования $\boldsymbol{П_м}$, процессу обслуживания $\boldsymbol{П}$, каналу $\boldsymbol{К}$, накопителю $\boldsymbol{Н}$, активности $\boldsymbol{а}$. Поскольку в рассматриваемом примере каналу соответствует только одна активность $\boldsymbol{а}$ (взаимнооднозначное соответствие), подграф для активности выделять не имеет смысла.

Разделим графически событийный граф на вертикальные полосы – дорожки (рис. 7) по аналогии с диаграммами деятельности в языке UML [Буч 2004; Леоненков 2004]. Каждая дорожка соответствует определенному статическому или динамическому объекту модели:

1 – процесс моделирования $\boldsymbol{П_м}$;

2 – процесс обслуживания $\boldsymbol{П}$, который является в то же время моделью внешней среды;

3 – канал $\boldsymbol{К}$ вместе с активностью $\boldsymbol{а}$, выполняемой в нем;

4 – накопитель $\boldsymbol{Н}$.

Можно предложить следующую процедуру включения вершин событийного графа в дорожки (подграфы дорожек).

Первый шаг. Размещение событийных вершин по дорожкам.

Вершина-событие (событийного графа) $\boldsymbol{e_i}$ включается в дорожку объекта $\boldsymbol{O_u}$, если событие $\boldsymbol{e_i}$ является событием изменения состояния этого объекта.

Второй шаг. Размещение условных вершин по дорожкам.

Условные вершины $\boldsymbol{P_j}$ можно размещать с учетом того,

1) состояние какого объекта O_u анализируется в условной вершине P_j (по анализируемому объекту) и
2) какова степень связи условной вершины P_j с другими вершинами (по связности с вершинами графа).

По первому критерию условную вершину P_1 – анализ состояния канала ***К*** необходимо включить в дорожку канала ***К***. По второму критерию вершина P_1 имеет одинаковую степень связи равную единице с вершинами объектов ***П***, ***К*** и ***Н***: $C_{P1\text{-}П} = C_{P1\text{-}К} = C_{P1\text{-}Н} = 1$. Итак, по первому критерию следует отдать предпочтение включению вершины P_1 в дорожку канала ***К***. Если вершину P_1 включить в дорожки ***П*** или ***Н***, то кроме связей по управлению между объектом ***К*** и соответствующим объектом появится дополнительная информационная связь, предоставляющая состояние канала ***К***. Это ведет к увеличению числа связей между объектами и усложнению структуры модели.

Аналогичны рассуждения для вершин P_2, P_3 и P_4.

Третий шаг. Размещение вершин начала и окончания распараллеливания процесса.

Поскольку в этих вершинах нет изменения и анализа состояния объектов, то их можно размещать с учетом только критерия связности с другими вершинами событийного графа.

Вершина A_1 имеет степень связи $C_{A1\text{-}П} = 2$ с дорожкой объекта ***П*** и $C_{A1\text{-}К} = 1$ с дорожкой объекта ***К***. Поэтому размещаем A_1 в подграфе объекта ***П***.

Вершина A_2 имеет степень связи $C_{A2\text{-}П} = 1$ с дорожкой объекта ***П***, $C_{A2\text{-}К} = 1$ с дорожкой объекта ***К*** и $C_{A2\text{-}Н} = 1$ с дорожкой объекта ***Н***. Поэтому являются равноценными варианты размещения A_2 в дорожках ***П***, ***К*** и ***Н***. Однако естественным является размещение A_2 в дорожке ***К***, поскольку входная дуга вершины A_2 связывает ее с вершиной e_{oS} дорожки ***К***.

Вершина A_3 имеет степень связи $C_{A3\text{-}Н} = 2$ с дорожкой объекта ***Н*** и $C_{A3\text{-}П} = 1$ с дорожкой ***П***. Поэтому размещаем A_1 в подграфе объекта ***Н***.

Заменяя дорожки объектами, которым они соответствуют, получим объектную событийную модель (рис. 8). В этой модели связи между объектами представляются дугами, которые помечаются именами вершин событийного графа (см. рис. 7): начальной и конечной, и временем задержки для дуг второго типа. Для рассматриваемого примера только одна дуга $e_{нпм} \rightarrow e_{нп}$ является дугой второго типа с задержкой T_0.

Полученная модель отражает взаимодействие между объектами в терминах событий. Поскольку в рассматриваемой декомпозиции событийные и условные вершины отнесены к тем объектам, состояния которых в них изменяются или анализируются, все объекты взаимодействуют только через события (активное взаимодействие) и нет информационных связей по состояниям объектов (пассивное взаимодействие). Таким образом, полученная модель является моделью со скрытыми (недоступными) состояниями объектов.

Поскольку дуги второго типа реализуются в программной модели через механизм планирования, необходимо полученные событийные подграфы (см. рис. 7) разделить на подграфы макрособытий, то есть выполнить их структурную декомпозицию. На рисунке 9 представлена структурная декомпозиция событийного графа с дорожками, при этом выделены следующие макрособытия $E_0, E_1, \ldots, E_{10}$:

E_0 – начало моделирования;
E_1 – поступление заявки в СМО;
E_2 – начало обслуживания заявки в канале, если он свободен;
E_3 – начало обслуживания заявки в канале;

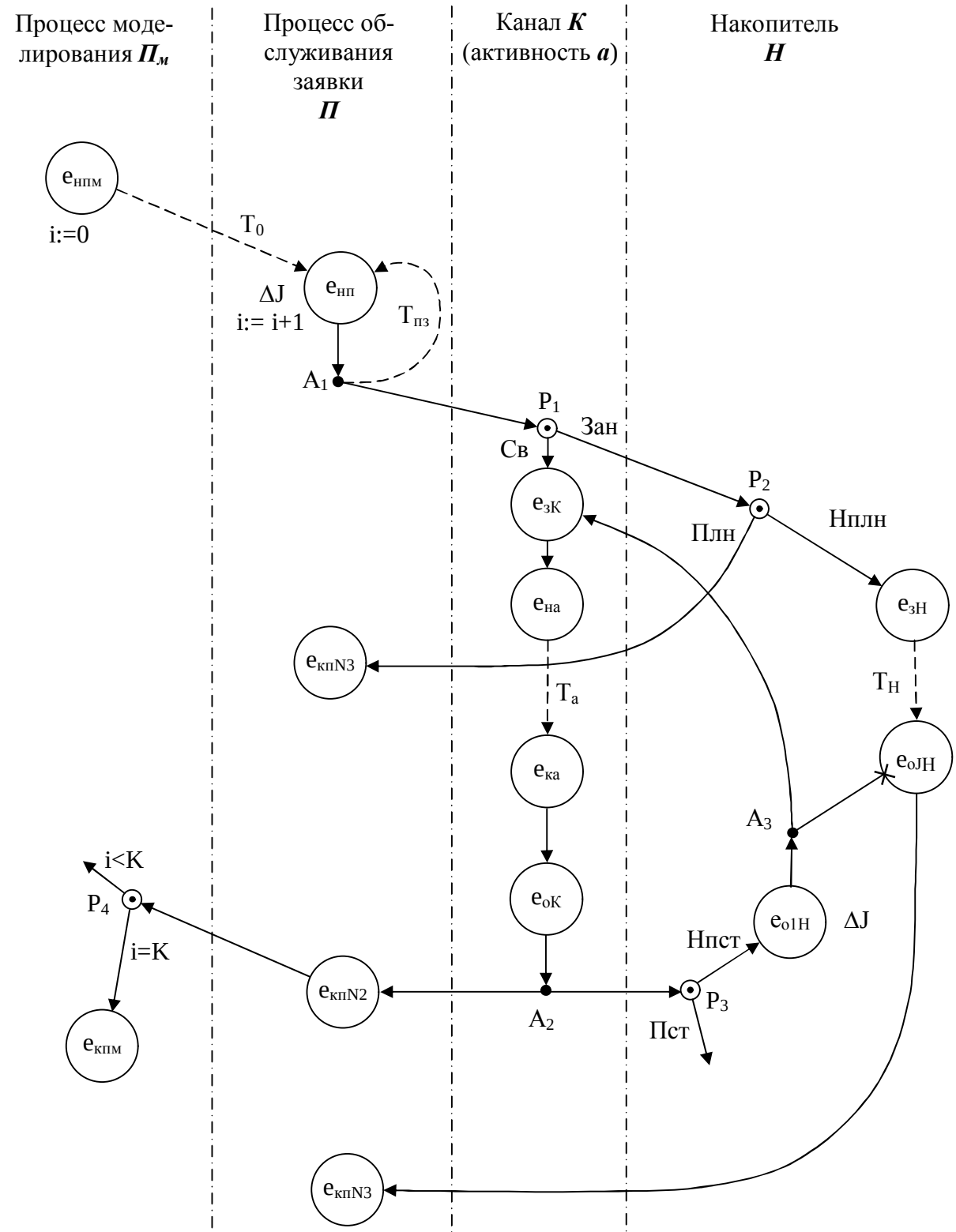

Рис. 7. Имитационный событийный граф процесса функционирования СМО с выделенными дорожками – событийными подграфами объектов модели

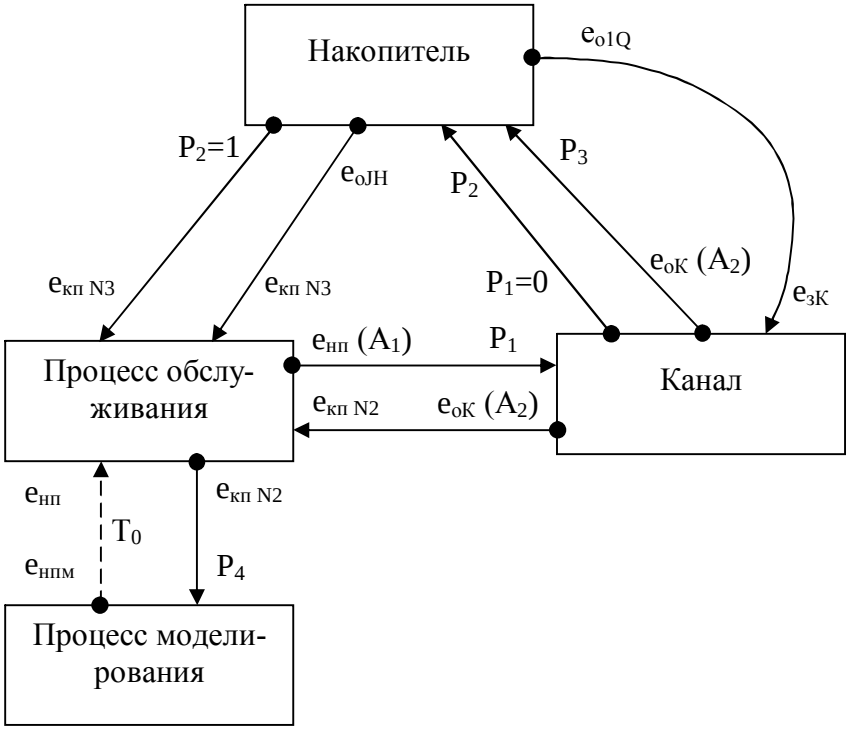

Рис. 8. Объектная модель на уровне событий

E₄ – окончание обслуживания заявки в канале;

E₅ – занятие очереди, если она неполна;

E₆ – освобождение очереди в связи с тем, что время ожидания в очереди достигло максимального T_Q;

E₇ – освобождение очереди, если она непуста;

E₈ – окончание обслуживания заявки в СМО;

E₉ – окончание обслуживания заявки в СМО в связи с отказом в обслуживании, очередь полна или время ожидания закончилось;

E₁₀ – окончание моделирования, если количество обслуженных заявок равно ***K***.

Таким образом событийный граф разделяется на событийные подграфы объектов. Эти подграфы описывают поведение объектов моделируемой системы.

На рисунке 10а приведен событийный подграф, определяющий поведение канала. Здесь:

e_{I1K} – событие на входе ***I1*** канала – требование занятия канала;

e_{I2K} – событие на входе ***I2*** канала – занятие канала;

e_{O1K} – событие на выходе ***O1*** канала – канал нельзя занять;

e_{O2K} – событие на выходе ***O2*** канала – освобождение канала.

Подграфы макрособытий ***E₂*** и ***E₃*** разделены, и событийные вершины продублированы, поскольку каждому макрособытию соответствует отдельная операция и процедура.

Событие e_{I1K} на входе ***I1*** канала вызывает выполнение внутреннего макрособытия ***E₂*** канала – начало обслуживания заявки в канале, если канал свободен, то есть начало обслуживания с условием.

Событие e_{I2K} на входе ***I2*** канала вызывает выполнение внутреннего макрособытия ***E₃*** канала – начало обслуживания заявки в канале; это безусловное начало обслуживания.

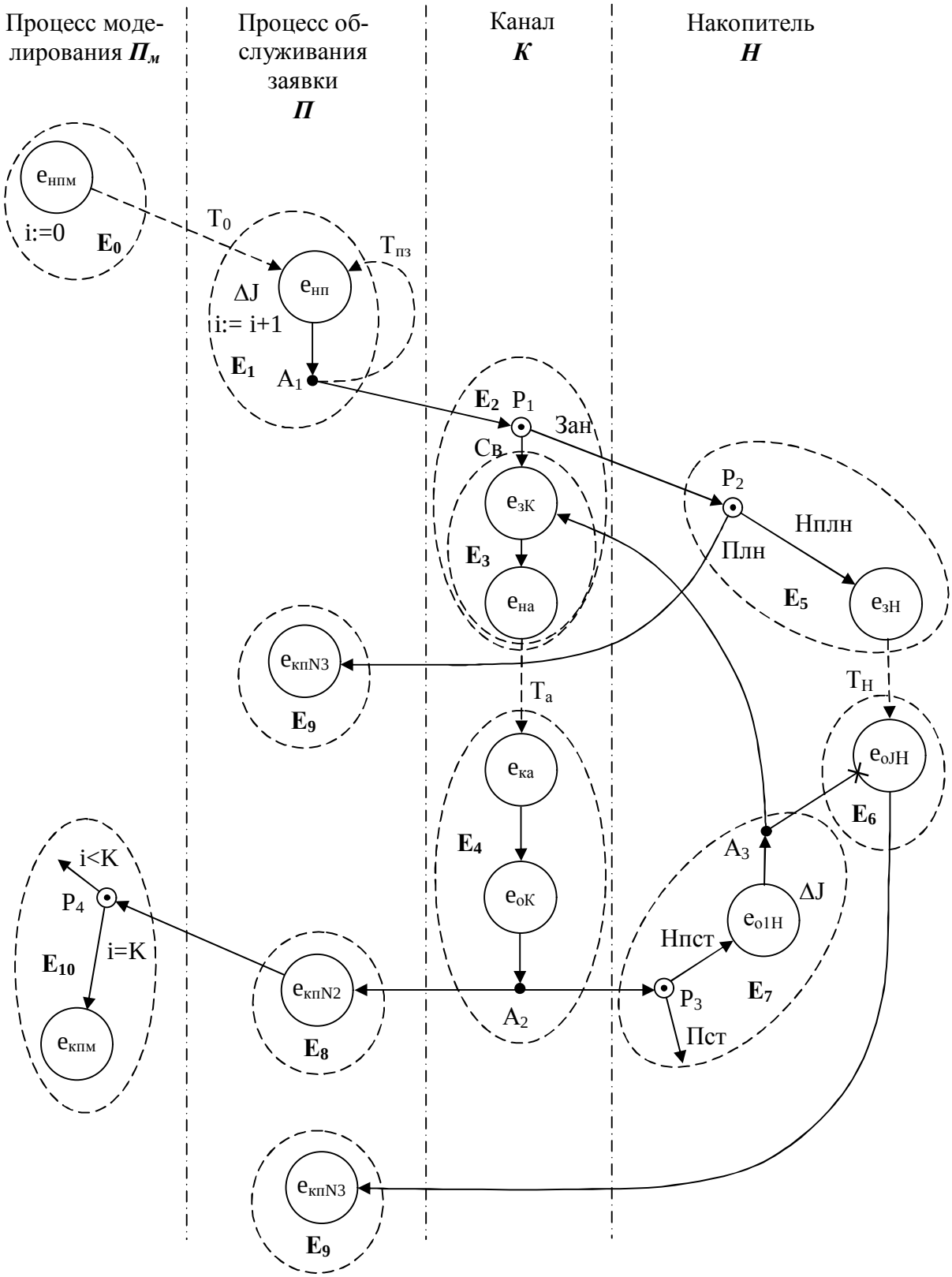

Рис. 9. Структурная декомпозиция событийного графа с дорожками

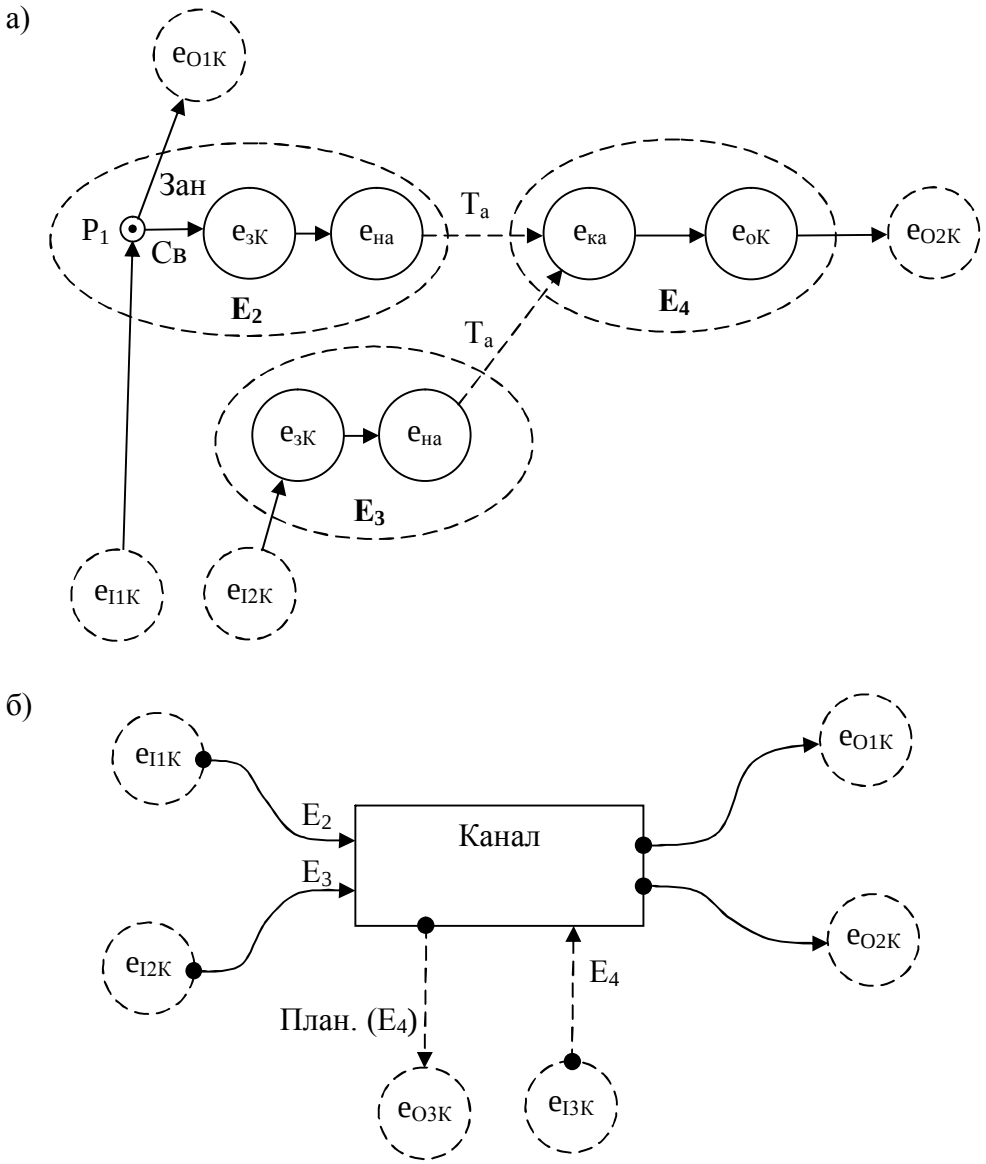

Рис. 10. Событийная модель канала

Событие e_{O1K} на выходе ***O1*** канала происходит если канал занят, заявка получает отказ в обслуживании в канале.

Событие e_{O2K} на выходе ***O2*** канала происходит если выполняется внутреннее макрособытие E_4 канала, то есть обслуживание заявки в канале завершено.

На рисунке 10б приведена событийная макромодель канала. Так как дуги второго типа реализуются с помощью механизма планирования, вместо дуги $E_2 \rightarrow E_4$ у объекта канал вводятся выход e_{O3K} и вход e_{I3K}:

e_{I3K} – вызов события E_4 – время T_a истекло;

e_{O2K} – планирование события E_4 через время T_a.

На рисунке 11а приведен событийный подграф, определяющий поведение накопителя.

Здесь e_{I1H} – событие на входе ***I1*** накопителя – требование занятия очереди;

e_{I2H} –событие на входе ***I2*** накопителя – требование освобождения очереди;

e_{O1H} – событие на выходе ***O1*** накопителя – накопитель нельзя занять, поскольку он полон;

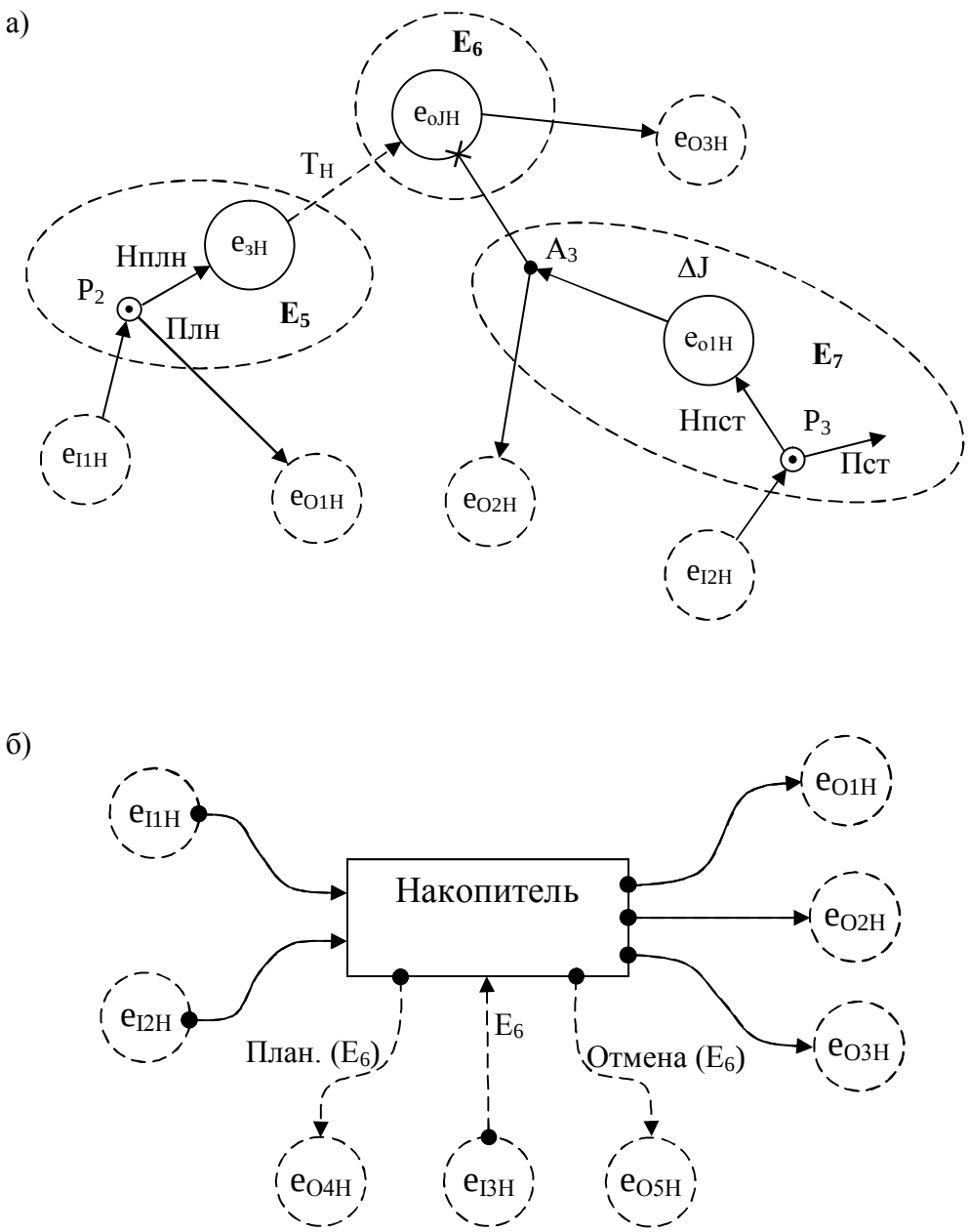

Рис. 11. Событийная модель накопителя

eO2H – событие на выходе ***O2*** накопителя – освобождение накопителя от заявки, стоящей первой в очереди;

eO3H – событие на выходе ***O3*** накопителя – освобождение накопителя от заявки, для которой закончилось время ожидания.

Событие ***eI1H*** на входе ***I1*** накопителя вызывает выполнение внутреннего макрособытия ***E5*** накопителя – занятие очереди накопителя, если накопитель не полон, то есть занятие очереди с условием.

Событие ***eI2H*** на входе ***I2*** накопителя вызывает выполнение внутреннего макрособытия ***E7*** накопителя – освобождение накопителя от заявки, стоящей первой в очереди, если очередь накопителя непуста, это освобождение накопителя с условием.

Событие ***eO1H*** на выходе ***O1*** накопителя происходит, если очередь накопителя полна, заявка получает отказ в постановке в очередь накопителя.

Событие e_{O2H} на выходе ***O2*** накопителя происходи, если выполняется внутреннее макрособытие E_7 накопителя и очередь канала не пуста.

Событие e_{O3H} на выходе ***O3*** накопителя происходит, если выполняется внутреннее макрособытие E_6 накопителя, то есть время ожидания заявки в очереди накопителя закончилось и заявка удаляется из очереди накопителя.

На рисунке 11б приведена событийная макромодель накопителя. Так как дуги второго типа и дуги отмены события реализуются с помощью механизма планирования, вместо дуги $E_5 \rightarrow E_6$ у объекта накопитель вводятся выход e_{O4H} и вход e_{I3H} и вместо дуги $E_7 \rightarrow E_6$ – выход e_{O5H}:

e_{I3H} – вызов события E_6 – время T_H истекло;

e_{O4K} – планирование события E_6 через время T_H;

e_{O5K} – отмена события E_6 – заявка дождалась обслуживания.

Поскольку для реализации дуг второго типа используется механизм планирования, в модель, кроме выделенных объектов, необходимо ввести управляющий объект ***Управление*** и объект-сущность ***Событие***. Эти объекты позволяют хранить список будущих событий, планировать событие, удалять и обрабатывать первое событие в списке и отменять ранее запланированное событие. Объект ***Управление*** удаляет первое событие из списка событий (объект-сущность ***Событие***) и вызывает выполнение операции E_i соответствующего объекта.

На рисунке 12 представлена программно-реализуемая объектная (объектно-событийная) модель. Дуги первого типа реализуются непосредственным взаимодействием объектов в соответствии с объектной декомпозицией событийного графа (см. рис. 9). Дуги второго типа $E_0 \rightarrow E_1$, $E_1 \rightarrow E_1$, $E_2 \rightarrow E_4$, $E_3 \rightarrow E_4$, $E_5 \rightarrow E_6$ реализуются с помощью механизма планирования, то есть при участии объектов ***Управление*** и ***Событие***.

Входы объектов определяют операции, выполняемые объектом. Поэтому каждому макрособытию, которое вызывается другим объектом, соответствует операция, выполняемая объектом.

Объект ***Процесс моделирования***:

E_0 – начать (инициализировать),

E_{10} – проверить окончание.

Объект ***Процесс обслуживания***:

E_1 – начать (создать),

E_8 – закончить ***N2*** (процесс обслуживания завершается нормально),

E_9 – закончить ***N3*** (процесс обслуживания прерывается по одной из причин: накопитель полон или время ожидания закончилось).

Объект ***Канал***:

E_2 – начать обслуживание, если канал свободен,

E_3 – начать обслуживание,

E_4 – закончить обслуживание.

Объект ***Накопитель***:

E_5 – занять очередь, если накопитель не полон,

E_6 – удалить из очереди процесс ***J***,

E_7 – освободить очередь, если накопитель непуст.

Объект ***Управление***:

$e_{кпм}$ – закончить моделирование.

Объект ***Событие***:

Удал. – удалить из списка событий первое по времени событие,

План. – планировать событие, то есть поместить в список событий указанное событие E_i,

Отмена – отменить указанное событие E_i, то есть удалить его из списка событий.

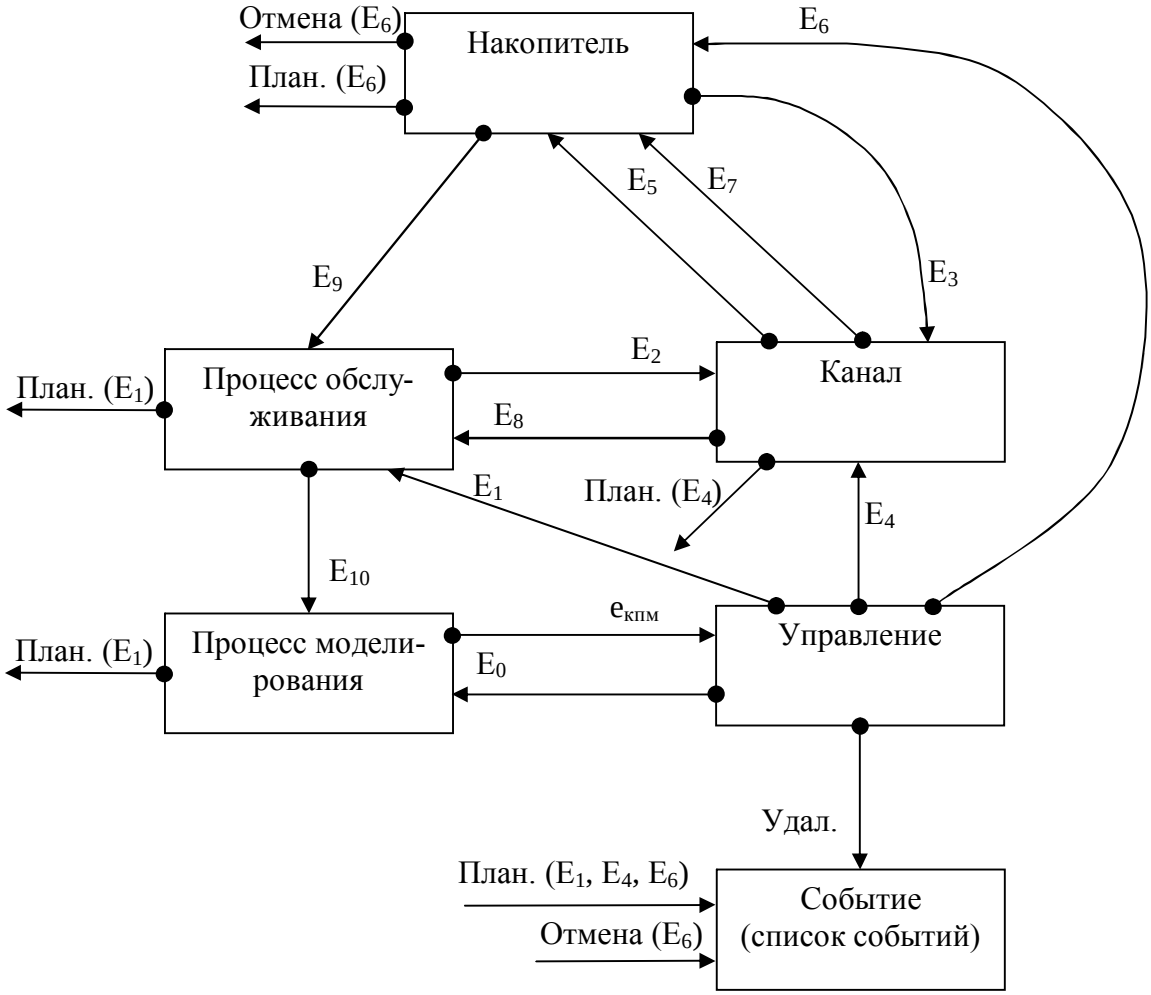

Рис. 12. Программно-реализуемая объектная модель

Атрибутами объектов являются входные параметры, переменные состояния и накапливаемые в процессе моделирования характеристики. Выделим основные атрибуты для рассматриваемого примера.

Объект ***Процесс моделирования***: ***K*** – количество процессов обслуживания для достижения целей проведения имитационного моделирования факторы эксперимента (***T_0, $T_{пз}$, T_a, T_H*** и др.).

Объект ***Процесс обслуживания***: приоритет, максимальное время ожидания в очереди ***T_Q***, время выполнения процесса (среднее время, сумма времен), дисперсия времени выполнения, количество процессов, получивших отказ в обслуживании.

Объект ***Канал***: состояние канала, процент использования, количество обслуженных заявок.

Объект ***Накопитель***: состояние накопителя (число заявок в очереди, для каждого процесса, находящегося в очереди: номер процесса, время постановки в очередь, приоритет), среднее время ожидания, дисперсия времени ожидания, средняя длина очереди, максимальная длина очереди.

Объект ***Управление*** – атрибутов не имеет.

Объект ***Событие***: номер события, номер процесса, параметры события, время события.

Также как и при структурной декомпозиции возможен другой вариант реализации событийного графа, при которой все дуги и первого и второго вида реализуются через механизм планирования. В этом случае взаимодействие по управлению между выделенными объектами отсутствует. Все взаимодействия осуществляются через планирование событий, то есть через объекты ***Событие*** и ***Управление.***

4. Заключение

Таким образом, в работе рассмотрены два способа декомпозиции событийного графа: структурный и объектный. Структурный способ рассматривался нами ранее [4, 5] и использовался для преобразования событийного графа в программную модель, в которой каждое макрособытие реализуется соответствующей процедурой. В настоящей работе, в отличие от предыдущих, рассмотрены два варианта реализации структурной декомпозиции: дуги первого типа могут реализовываться, во-первых, с помощью механизма планирования и, во-вторых, как непосредственные передачи управления между процедурами. Предложен способ объектной декомпозиции событийного графа, что позволяет:

- получить объектную модель по известному событийному графу,
- определить в терминах событий и макрособытий поведение объектов,
- преобразовать событийный граф в объектно-ориентированную программную модель.

Рассмотрена также модификация событийного графа – событийный граф с дорожками, в котором в явном виде представляется взаимосвязь между событиями и статическим и динамическими объектами модели.

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 / L. W. Schruben // 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 / Е. А. Бабкин. – Курск : КПИ, 1988.
4. Бабкин, Е. А. Событийные модели дискретных систем / Е. А. Бабкин ; Курск. гос. ун-т. – Курск, 2005. – 18 с. Деп. в ВИНИТИ 14.01.05, № 30–В2005.
5. Бабкин, Е. А. О синтезе событийных моделей дискретных систем / Е. А. Бабкин // Ученые записки : электронный научный журнал Курского государственного университета. Эл № 77-26463. – 2006. – № 1. – 17 с. http://www.scientifical-notes.ru/pdf/s15.pdf.
6. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон ; пер. с англ. А. А. Слинкина. – 2 изд., стер. – М. : ДМК Пресс; СПб. : Питер, 2004. – С. 242–247.
7. Леоненков, А. В. Самоучитель UML / А. В. Леоненков. – 2 изд. перераб. и доп. – СПб. : БХВ-Петербург, 2004. – С. 249–251.
8. Бабкин, Е. А. Иерархическое событийно-автоматное моделирование / Е. А. Бабкин, Е. А. Бобрышев // Информационные технологии моделирования и управления. Научно-технический журнал. – Воронеж : Научная книга, 2007. – № 1 (35). – С. 39–48.
9. Бабкин, Е. А. О формализме событийно-автоматного моделирования / Е. А. Бабкин, Е. А. Бобрышев // Третья всероссийская научно-практическая конференция по ими-

тационному моделированию и его применению в науке и промышленности «Имитационное моделирование. Теория и практика» ИММОД-2007 : сборник докладов. Т. I. – СПб. : ФГУП ЦНИИТС, 2007а. – С. 82–86.