О ФОРМАЛИЗМЕ ИЕРАРХИЧЕСКОГО СОБЫТИЙНО-АВТОМАТНОГО МОДЕЛИРОВАНИЯ

Е. А. Бабкин, Е. А. Бобрышев (Курск)

В настоящее время существует множество формальных подходов к описанию дискретно-событийных моделей: формально-логические подходы (например, формализм Лакнера) [1]; DEVS-формализм [2]; событийно-ориентированные графические методы (событийные графы [3, 4, 5], событийные алгоритмы [6–8], имитационные графы [9]); сети Петри [10]; подходы, основанные на макс-алгебре [11] и другие.

Каждый из подходов ориентирован на исследование определенных аспектов дискретно-событийной модели. Обычно при проведении модельного эксперимента исследователь использует несколько разных подходов для представления модели на разных этапах моделирования. В связи с этим существует проблема организации процесса моделирования, проблема тождественных преобразований модели из одного представления в другое.

Центральной идеей данного исследования является создание подхода, позволяющего управлять процессом имитационного моделирования от постановки задачи до анализа результатов эксперимента. В качестве основы для организации процесса моделирования был взят унифицированный процесс (RUP) [12].

Целью подхода к иерархическим событийно-автоматным моделям (ИСАМ) является разработка совокупности методов и средств проектирования, описания, разработки, верификации и валидации модели на каждом из этапов моделирования. Одной из задач создаваемого подхода является разработка инструмента формализации дискретно-событийной модели (формализм ИСАМ).

Формализм иерархического событийно-автоматного моделирования основан на концепции событийных алгоритмов, представленных в [7, 8]. Представление систем на уровне событий позволяет задавать и наблюдать динамику поведения исследуемой системы, управлять моделированием времени и дает ряд других преимуществ. Однако построение сколько-нибудь сложной модели на уровне событий затруднительно ввиду невозможности иерархичного представления модели.

В рамках формализма ИСАМ предлагается представление модели в виде подсистем двух типов, одни из которых (атомы) описывают поведение системы с использованием событийных алгоритмов, а другие (блоки) служат для объединения атомов в иерархическую систему и трансляции сообщений между ними, что позволяет снять ограничение на сложность систем, моделируемых на уровне событий.

Атом определяется как шестерка $s_a = \langle Z, X, Y, EA, T, z_0 \rangle$, где Z – алфавит состояний атома; X – входной алфавит; Y – выходной алгоритм; EA – событийный алгоритм атома; T – время; z_0 – начальное состояние.

Блок определим как $s_б = \langle S, SA_1, SA_2, X, Y \rangle$, где S – множество подсистем (блоков или атомов) блока $s_б$; SA_1, SA_2 – множества дуг, отображающих взаимодействие подсистем блока; X, Y – входные и выходные множества блока.

Элемент модели рассматривается как автомат Мили, для которого введены дополнительные ограничения:

1. переход между состояниями может произойти тогда и только тогда, когда происходит какое-либо событие внутри или вне элемента (такие события называют соответственно внутренними и внешними);
2. множество возможных последовательностей переходов ограничено событийным алгоритмом элемента;

3. добавлена одна дополнительная активная выходная переменная, значение которой $y_{out} \subset E_{ext} \cup \{0\}$, где E_{ext} – некоторое подмножество множества событий во внешней среде, порождаемых поведением элемента.

Для представления событийно-автоматной модели системы могут использоваться модифицированные таблицы переходов и выходов (табл. 1).

Таблица 1

Таблица переходов и выходов

E_v	X_v	z_v						y_v					
		z_{v-1}						z_{v-1}					
		0	1	2	…	m-1	m	0	1	2	…	m-1	m
e^1	x^1				…						…		
…	…	…						…					
e^n	x^k				…						…		

Здесь необходимо отметить, что, в отличие от автоматной модели, важен не только сам факт появления синхросигнала, но и событие, при возникновении которого система совершила переход в другое состояние. Поэтому в отличие от автоматной таблицы переходов и выходов [13] здесь на входе рассматриваются не только входные символы $x^k \in X_v$, но и события $e^n \in E_v$, вызывающие изменение состояния (табл. 1, столбец 1). Столбцы 1 и 2 таблицы должны содержать все пары из $E \times X$.

Таблицу переходов можно также представить по-другому (табл. 2).

Таблица 2

Событийная таблица переходов для канала

Исходное состояние	Входной набор (события и условия)	События изменения состояния автомата (внутренние события)	Следующее состояние	Выходные события	Состояние выхода
z_0	$e_{зк}$	e_{01}	z_1		y_0
z_1	$e_{ок}$	e_{10}	z_0	$e_{ЗО}$	y_1

Динамику поведения модели можно отображать событийными алгоритмами [7]. Событийные алгоритмы и таблицы переходов/выходов полностью описывают поведение элемента системы.

Событийный алгоритм $EA = \langle V, A, \Delta \rangle$ [7], где $V = E \cup C \cup U$ – множество вершин графа; $A = A_0 \cup A_t$ – множество дуг графа; Δ – множество функций инцидентности. Здесь:

- E – множество вершин графа, отображающих события в подсистеме. $\forall e \in E \left| \left(d_o(e) \in \{0,1\} \right) \wedge \left(d_i(e) \in Z^+ \right) \right.$ ($Z^+ = N \cup \{0\}$, N – множество натуральных чисел). Под событием будем понимать мгновенное изменение состояния системы S [7].
- C – множество вершин графа, отображающих выбор. Каждой вершине $c \in C$ сопоставим некоторую функцию $con : I \times X^{(n)} \to \Lambda \subseteq R$, представляющую собой условие

выбора планируемых событий. $\forall c \in C \left| \left(d_o(e) \in N\right) \wedge \left(d_i(e) \in N\right)\right.$. Каждой инцидентной по выходу дуге для $c \in C$ ставится в соответствие некоторое подмножество множества Λ, определяющее выбор пути. В случае, если некоторый выход должен быть безусловным, поставим ему в соответствие множество Λ.

- U – множество объединяющих вершин. Каждой вершине $u \in U$ сопоставим некоторую функцию $un: I \times X^{(n)} \times H \to \mathrm{Y} \subseteq R$, где H – история – множество, состоящее из упорядоченных наборов событий. $\forall u \in U \left| \left(d_o(e) \in N\right) \wedge \left(d_i(e) \in N\right)\right.$.
- A_0 – множество дуг первого рода, определяемых следующим образом: $\forall a \in A_0 \left| a = (v_1, v_2), v_1, v_2 \in E \cup C \cup U\right.$.
- A_t – множество дуг второго рода, определяемых следующим образом: $\forall a \in A_t \left| a = (v_1, v_2), v_1 \in E \cup C \cup U; v_2 \in E \cup U\right.$. Каждой дуге второго рода ставится в соответствие некоторая функция $t: \Xi \to T = R^+$, где T – множество интервалов времени; Ξ – некоторое множество, определяемое предметной областью. В частности, это может быть множество T либо множество $X^{(n)}$.

Пример событийного алгоритма приведен на рис. 1.

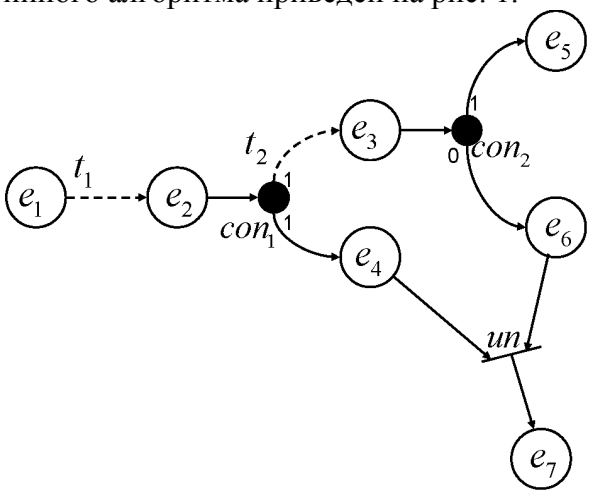

Рис. 1. Пример событийного алгоритма [7]

Событийный алгоритм *EA* определяет поведение атома в виде потока событий.

Определим поток событий, исходя из нотации событийных алгоритмов.

Формальным потоком событий назовем такое индексированное множество событий $Tr = \left\langle e_i \left| i = \overline{1,n} \right.\right\rangle$, что $\forall e_i \in Tr \setminus \{e_0\} \left(e_{i-1} \wedge c \overset{t}{\to} e_i\right)$ (здесь c – некоторая логическая функция), причем $\neg \exists e_k \in E \left(e_k \wedge e_{i-1} \overset{t}{\to} e_i\right)$.

Два формальных потока событий $Tr_1 = \left\langle e_i \left| i = \overline{1,n} \right.\right\rangle$ и $Tr_2 = \left\langle e_j \left| j = \overline{1,m} \right.\right\rangle$ назовем параллельными, если

$$\forall e_i \in Tr_1 \wedge \forall e_j \in Tr_2 \left(e_i \neq e_j\right) \wedge \exists e_i \in Tr_1 \wedge \exists e_{j_1}, e_{j_2} \in Tr_2 \left(t(e_1) \in \left[t\left(e_{j_1}\right); t\left(e_{j_2}\right)\right]\right).$$

Два формальных потока событий $Tr_1 = \left\langle e_i \left| i = \overline{1,n} \right.\right\rangle$ и $Tr_2 = \left\langle e_j \left| j = \overline{1,m} \right.\right\rangle$ назовем пересекающимися, если $\exists e_i \in Tr_1 \wedge \exists e_j \in Tr_2 \left(\left(e_i = e_j\right) \wedge \left(e_i \overset{t}{\to} \overline{e_j}\right)\right)$.

Будем говорить, что два параллельных потока событий $Tr_1 = \left\langle e_i \middle| i = \overline{1,n} \right\rangle$ и $Tr_2 = \left\langle e_j \middle| j = \overline{1,m} \right\rangle$ сходятся к событию e_k, если $e_m \wedge c_m \wedge e_n \wedge c_n \xrightarrow{\Delta t} e_k$, где c_m, c_n – некоторые логические функции.

В случае описания потоков событий допускается последовательная нотация их следования, например $e_1 \xrightarrow{\Delta t_1} e_2 \wedge c_1 \xrightarrow{0} e_3 \wedge c_2 \xrightarrow{\Delta t_2} e_4$.

Фактическим потоком событий назовем множество уже произошедших в системе событий, представляющих собой цепь по отношению планирования и упорядоченных по времени.

Блок модели является композицией элементов и блоков более низкого уровня. Его состояния являются комбинацией состояний составляющих его блоков и элементов. Множество его входных/выходных переменных является подмножеством объединения множеств входных/выходных переменных его составляющих.

Для описания блока используется диаграмма взаимодействия подсистем – граф $SG = \langle S, SA_1, SA_2 \rangle$ [6], где $S = S_a \cup S_b$ – множество вершин графа, отображающих блоки и атомы (подсистемы); $SA_1 = \left\{ sa_{i,j} = \left(s_i, s_j \right) \right\}$ – множество дуг первого рода, отображающих взаимодействие двух подсистем через события (некоторое событие из s_i планирует одно из входных событий s_j). Причем дуге $sa_{i,j}$ ставится в соответствие идентификатор планируемого события из s_j и порождающего из s_i; $SA_2 = \left\{ sa_{i,j} = \left(s_i, s_j \right) \right\}$ – множество дуг второго рода, отображающих взаимодействие двух подсистем через пассивные переменные (на вход s_j поступает с выхода s_i некоторый символ из входного алфавита s_j). Каждой дуге второго рода ставится в соответствие пассивная выходная переменная из s_i и соответствующая ей входная переменная из s_j. На рис. 2 показаны примеры диаграмм.

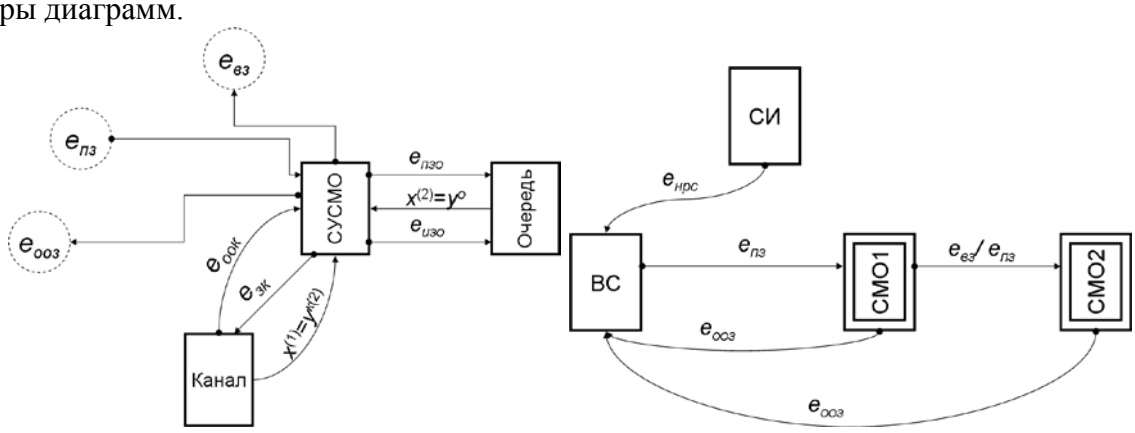

Рис. 2. Пример диаграмм взаимодействия подсистем [6]

Диаграммы взаимодействия подсистем отображают взаимодействие блоков и атомов на одном уровне. Для отображения иерархии агрегации подсистем можно использовать древовидную схему, в которой узлами будут блоки, а листьями атомы.

Предлагаемый формализм ИСАМ должен стать основой для создания методологии имитационного моделирования больших систем. Сочетая в себе модульность, иерархичность декомпозиции и преимущества событийного моделирования времени, ИСАМ позволяет создавать модели для широкого круга систем. В рамках создаваемой методологии на основе формализма ИСАМ предполагается также разработка методов декомпозиции исследуемой системы, позволяющих быстро перейти от описания системы в терминах предметной области к описанию системы в терминах формализма

ИСАМ, и библиотеки классов, предоставляющей исследователю интерфейс для создания программной модели в терминах формализма ИСАМ.

Литература

1. **Lackner M. R.** (1964). «Digital Simulation and System Theory», System Development Corporation, Santa Monica, CA.ч
2. **Zeigler B., Kim T. G., Praehofer H.** Theory of Modeling and Simulation. Second edition. Academic Press, New York, 2000.
3. **Schruben L. W.** Simulation Modeling with Event Graphs. Communications of the ACM. 26: 957–963 (1983).
4. **Schruben L. W., Yucesan E.** 1993. Modeling paradigms for discrete-event simulation. Oper. Res. Lett. 13 265–275.
5. **Ingalls R. G., Morrice D. J., Yucesan E., Whinston E. B**. Execution Conditions: A Formalization of Event Cancellation in Simulation Graphs INFORMS Journal on Computing © 2003 INFORMS. Vol. 15, No. 4, Fall 2003. P. 397–411.
6. **Бабкин Е. А. Бобрышев Е. А.** Иерархическое событийно-автоматное моделирование//Информационные технологии моделирования и управления: №1 (35) /Под ред. д.т.н. проф. О.Я. Кравца. Воронеж: Научная книга, 2007. С. 39–48.
7. **Бабкин Е. А. Бобрышев Е. А.** О событийных моделях дискретных систем//Современные проблемы информатики в моделировании и анализе сложных систем: Сб. трудов. Вып. 12/Под ред. д.т.н. проф. О. Я. Кравца. Воронеж: Научная книга, 2007. С. 141–149.
8. **Бабкин Е. А.** Методические указания по моделированию вычислительных систем на событийно-ориентированном языке. Ротапринт. Курск: КПИ, 1988.
9. **Yuecesan E.** Simulation Graphs for the Design and Analysis of Discrete Event Simulation Models, PhD Dissertation – Cornell University, Ithaca, NY. 1989.
10. **Питерсон Дж.** Теория сетей Петри и моделирование систем: Пер с англ. М.: Мир, 1984. 264 с.
11. **Пузырев В. А., Царегородский С. А.** Макс-алгебра и дискретно-событийные системы//Зарубежная радиоэлектроника: № 1. М.: Радио и связь, 1992. С. 3–31.
12. **Якобсон А., Буч Г., Рамбо Дж.** Унифицированный процесс разработки программного обеспечения. СПб.: Питер, 2002. 496 с.
13. **Гилл А.** Введение в теорию конечных автоматов. М.: Наука, 1966. 272 с.