УДК 004.415.28
О.Н. Кулешова, аспирант,
Ю.К. Апраксин, профессор, д-р техн. наук
Севастопольский национальный технический университет
ул. Университетская 33, г. Севастополь
E-mail: v_olgo4ka@inbox.ru

СПЕЦИФИКАЦИЯ РАСПРЕДЕЛЁННЫХ СИСТЕМ НА ОСНОВЕ ТАБЛИЦ СОБЫТИЙ

Рассматривается структура распределенной системы на основе таблиц событий, обеспечивающая возможности проектирования, моделирования и управления. Разрабатываются спецификации основных составляющих частей системы.

Ключевые слова: *таблица событий, спецификация, структура.*

Создание современных распределенных систем управления и моделирования требует тщательной оценки альтернатив с рассмотрением всех свойств, необходимых для эффективного продолжения работы и развития [1]. Табличные модели, значительно облегчая внешний интерфейс, дают возможность шире взглянуть на задачи проектирования систем управления и моделирования и на их решение.

Существующие табличные модели, такие как, таблицы решений [2,3], являются удобным средством для представления сложной логики программирования, но представляют собой не достаточно гибкое средство для решения задач проектирования систем.

Для эффективного использования таблиц событий [4] необходима разработка специальной программной системы, обеспечивающей возможности проектирования, управления и моделирования.

Целью статьи является разработка спецификации и структуры распределённой системы на основе таблиц событий, включая спецификации основных подсистем и связи между ними.

Таблицы событий в системе представлены в виде множества $TE=\{TE_\omega\}, \omega=\overline{1,quan}$, где *quan* — количество таблиц событий в системе.

Таблица событий в системе представляется в виде кортежа:

$$<ID, E, A, R, D, C, P, Cond, Act>.$$

Здесь $E=\{E_i\}; i=\overline{1,m}$ — множество условий, описывающих параметры выбранной предметной области;

$A=\{A_i\}; j=\overline{1,k}$ — множество действий;

$SubT=\{SubT_i\}; j=\overline{1,k}$ — множество ссылок на таблицы решений, описывающих действия;

$R=\{R_p\}; p=\overline{1,n}$ — множество решающих правил, называемых правилами решений, определяющими конкретные действия, при заданных условиях;

$D=\{D_l\}; l=\overline{1,z}$ — множество решающих правил, определяющих действие «иначе»;

$C=\{C_q\}, q=\overline{1,n}$ — множество векторов условий, где $\{C_q\}=\{c_1^q,\ c_2^q, \ldots, c_m^q\}$;

$P=\{P_v\}, v=\overline{1,n}$ — множество действий, где $\{P_v\}=\{a_1^v,\ a_2^v, \ldots, a_k^v\}$;

$Cond=\|c_{ix}\|, Act=\|a_{jx}\|$ — матрицы взаимосвязей множеств векторов данных и векторов действий.

Столбцы c_x матрицы *Cond* задают возможные разбиения множества *X* на *n* непересекающихся классов, характеризующихся определенным действием *Y*, а столбцы a_x матрицы *Act* задают отношения между множествами условий *E* и множеством действий *A*.

$R_x=(c_x, a_x)$ называется решающим правилом, а элементы матрицы *Cond* и *Act* определяются следующим образом:

$$c_{ix}=\begin{cases}1, & \text{если условие } e_i \text{ истинно;}\\ 0, & \text{если условие } e_i \text{ ожноо;}\\ \times, & \text{если условие } e_i \text{ безразлично.}\end{cases}$$

$$a_{jx}=\begin{cases}a\in(1,\ldots,k), & \text{порядок выполнения действия,}\\ & \text{если правило } R_x \text{ приводит к выбору решения } P_i;\\ 0, & \text{в противном случае.}\end{cases}$$

Таким образом, правило определяет действие, выполнение которого произойдет при конкретном сочетании результатов проверки условий.

Общий вид таблицы событий представлен в таблице 1.

Таблица 1 – Общий вид таблицы событий

ID			R_1	R_2	…	R_x	…	R_n	*D*					
			C_1	C_2	…	C_x	…	C_n	D_1	D_2	…	D_l	...	D_z
E	e_1		c_{11}	c_{12}	…	c_{1x}	…	c_{1n}	c_{1n+1}	c_{1n+2}	…	c_{1n+l}	…	c_{1n+z}
	e_2		c_{21}	c_{22}	…	c_{2x}	…	c_{2n}	c_{2n+1}	c_{2n+2}	…	c_{2n+l}	…	c_{2n+z}
	e_3		c_{31}	c_{32}	…	c_{3x}	…	c_{3n}	c_{3n+1}	c_{3n+2}	…	c_{3n+l}	…	c_{3n+z}
	…		…	…	…	…	…	…	…	…	…	…	…	…
	e_i		c_{i1}	c_{i2}		c_{ix}		c_{in}	c_{in+1}	c_{in+2}	…	c_{in+l}	…	c_{in+z}
	…		…	…	…	…	…	…	…	…	…	…	…	…
	e_m		c_{m1}	c_{m2}	…	c_{mx}	…	c_{mn}	c_{mn+1}	c_{mn+2}	…	c_{mn+l}	…	c_{mn+z}
			P_1	P_2	…	P_x	…	P_m	P_{n+1}					
A	A_1	$SubT_1$	a_{11}	a_{12}	…	a_{1x}	…	a_{1n}	a_{1n+1}					
	A_2	$SubT_2$	a_{21}	a_{22}	…	a_{2x}	…	a_{2n}	a_{2n+1}					
	A_3	$SubT_3$	a_{31}	a_{32}	…	a_{3x}	…	a_{3n}	a_{3n+1}					
	…	…	…	…	…	…	…	…	…					
	A_i	$SubT_i$	a_{i1}	a_{i2}	…	a_{ix}	…	a_{in}	a_{in+1}					
	…	…	…	…	…	…	…	…	…					
	A_k	$SubT_k$	a_{k1}	a_{k2}	…	a_{kx}	…	a_{kn}	a_{kn+1}					

Взаимодействие таблиц событий в системе может быть описано в виде графа (рисунок 1), где вершины графа представляют таблицы событий, а дуги — связи между таблицами.

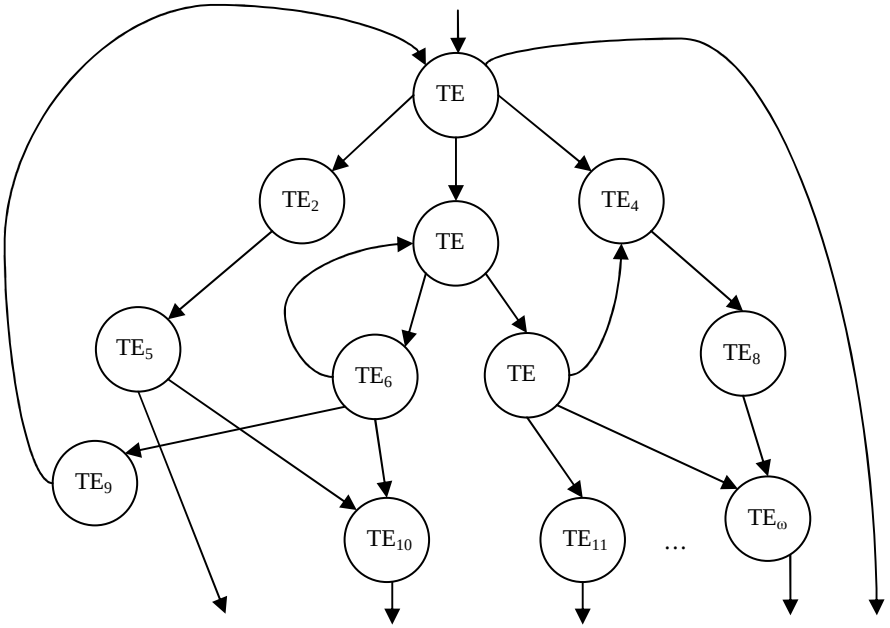

Рисунок 1 — Граф взаимодействия таблиц событий

Данные в системе представлены в виде множества $Data = \{Data_\lambda\}, \lambda = \overline{1, num}$.

Элемент данных представляется в виде $Data_\lambda = \{UsedTE^\lambda, FF^\lambda, Type^\lambda, Value^\lambda\}$.

$UsedTE^\lambda = \{UsedTE_\sigma^\lambda\}, \sigma = \overline{1, count}; count \le quan$ — множество ссылок на таблицы событий, использующие этот элемент данных.

Ссылка на таблицу событий $UsedTE_\sigma$ может быть представлена в виде кортежа:

<*UsedIDTE, UsedE, UsedA*>.

Здесь *UsedIDTE* – ID таблицы событий использующей элемент данных;

$UsedE=\{UsedE_\xi\}, \xi=\overline{1,countE}$ — множество ссылок на условия, использующих элемент данных;

$UsedA=\{UsedA_\varsigma\}, \varsigma=\overline{1,countA}$ — множество ссылок на действия, использующих элемент данных.

Ссылка на действие $UsedA_\varsigma$ может быть представлена кортежем:

<*UsedAID, RW*>,

где *UsedAID* — ID действия, связанного с элементом данных;

RW — флаг типа действия над элементом данных:

$$RW=\begin{cases}1, & \text{если инициируется процесс записи;}\\ 0, & \text{если инициируется процесс чтения;}\end{cases}$$

$FF^\lambda=\{FF_\varepsilon^\lambda\}, \varepsilon=\overline{1, funct}$; $funct\le\mu+\eta$ — множество ссылок на функции ввода и вывода, использующие этот элемент данных;

$Type^\lambda$ — тип элемента данных;

$Value^\lambda$ — значение элемента данных.

Функции проверки и коррекции таблиц событий представлены в системе в виде множества $CF=\{CF_\varphi\}, \varphi=\overline{1,\tau}$. Функции должны обеспечивать выполнение требований, которым должна удовлетворять система таблиц событий.

Таблица событий должна удовлетворять следующим требованиям:

1) для условий — непротиворечивость, неизбыточность и полнота,

2) отсутствие неоднозначности по взаимодействию с данными (чтение, запись), отсутствие неиспользуемых действий.

Таким образом, для корректно построенной таблицы событий должны выполняться следующие правила:

1) $\forall i, j\left[i\ne j\to C_i\cap C_j\right]=\varnothing$ — для обеспечения непротиворечивости (ортогональности);

2) $\forall i, j\left[i\ne j\to\left(\left(C_i\not\subset C_j\right)\wedge\left(C_j\not\subset C_j\right)\right)\right]$, $\forall i, j\left[i\ne j\to C_i\ne C_j\right]$,

$\forall i, j\left[i\ne j\to(P_i=P_j)\to(C_i\cup C_j)\right]$ — для обеспечения неизбыточности;

3) $\forall S, i, j(\exists R_i\vee\exists D_j)((S\to R_i)(S\to D_j))$ — для обеспечения полноты, где вектор $S=\{s_i\}$, $i=\overline{1,m}$, подающийся на вход таблицы событий;

4) $$\forall i, j, k\begin{bmatrix} i\ne j,((A_i(UsedA(RW)=1)\wedge(A_j(UsedA(RW)=0)\vee\\ (A_i(UsedA(RW)=0)\wedge(A_j(UsedA(RW)=1)\vee\\ (A_i(UsedA(RW)=1)\wedge(A_j(UsedA(RW)=1)\to\\ a_{ki}\ne a_{kj}\end{bmatrix}$$ — обеспечение отсутствия неоднозначности по взаимодействию с данными (чтение, запись);

5) $\forall i\left[A_i\ne 0\right]\ne\varnothing$ — обеспечение отсутствия неиспользуемых действий.

Внутренние переменные системы *GVar* — переменные необходимые для функционирования системы, могут быть представлены в виде кортежа:

<*TStack, AScript, Log, IVar*>,

где *TStack* — дерево стеков, хранящее точки возврата, при выполнении сложных сценариев, затрагивающих более одной таблицы событий. Схема *TStack* представлена на рисунке 2.

Элемент *TStack* имеет следующий вид:

< ID_{TE}, *NumA, NumS* >,

где ID_{TE} — ID событий, к которой будет произведён возврат;

NumA — номер действия в сценарии, к которому будет произведён возврат;

NumS — номер сценария в массиве сценариев, к которому будет произведён возврат.

Элемент $AScript=\{AScript_\psi\}, \psi=\overline{1,nscript}$ — массив выполняемых сценариев системы, где $AScript_\psi$ имеет вид < ID_{TE} , *Script*>, где ID_{TE} — ID таблицы событий, откуда был получен сценарий, *Script* – сценарий выполнения действий, полученный из таблицы событий.

Элемент *Log* — журнал функционирования системы.

Элемент *IVar* — внутренние переменные системы.

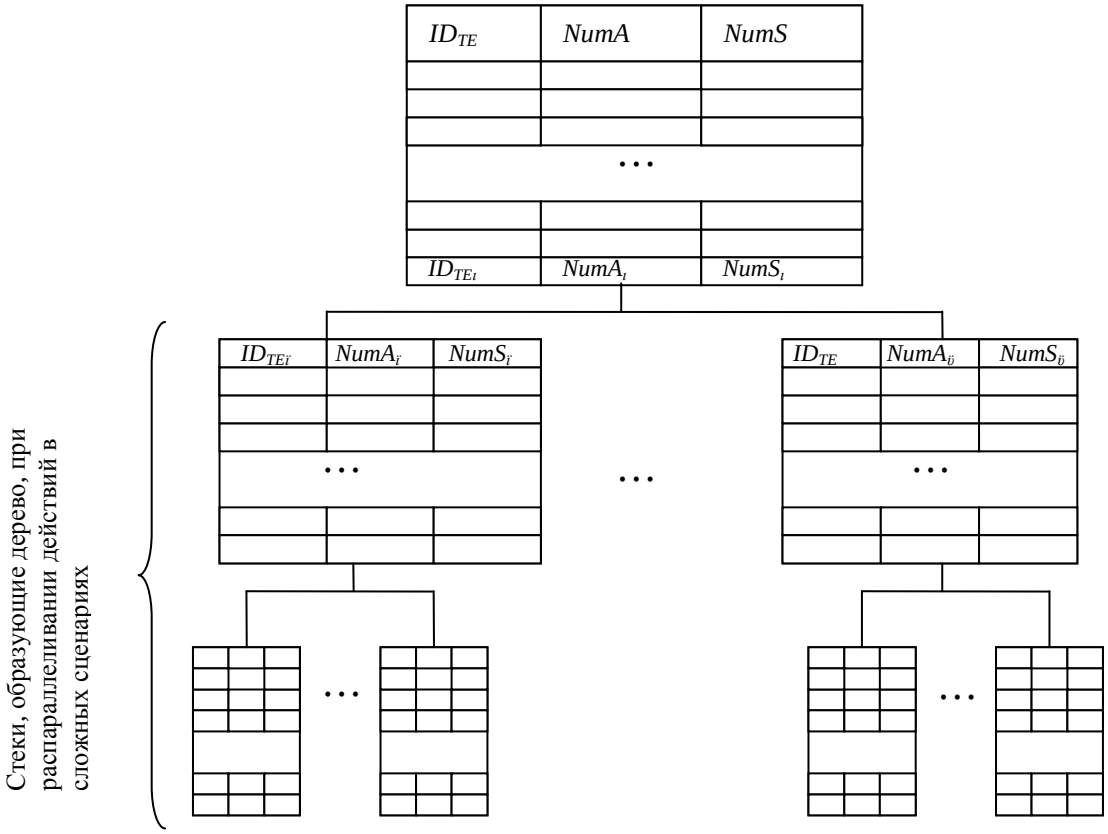

Рисунок 2 — Схема *TStack*

Функции взаимодействия $IntF = \{Int_\theta\}, \theta = \overline{1, ifunct}$ — обеспечивающие работу системы в режиме управления или моделирования, т.е. инициацию загрузки данных, поступление на вход векторов условий, выбор сценария по текущему условию и запись его в массив сценариев, обеспечение переходов по таблицам решений в системе, формирование дерева стеков точек возврата, распараллеливание или поочерёдное выполнение сценариев, обращение к функциям описанным на мета-языке, выборку и выполнение сценариев из массива сценариев по дереву стеков точек возврата, ведение логов, выполнение прерываний.

Функции ввода данных в системе представлены в виде множества $IF = \{IF_\alpha\}, \alpha = \overline{1, \mu}$.

Функции вывода данных в системе представлены в виде множества $OF = \{OF_\beta\}, \beta = \overline{1, \eta}$.

Интерфейс проектировщика в системе обозначен как DI.

Мета-язык в системе обозначен как *MetaLang* — язык для описания функций, которые бессмысленно и затратно реализовывать на основе таблиц событий (например, арифметически функции).

Таким образом, формальное описание системы на основе таблиц событий может быть представлено в виде кортежа

<TE, IF, OF, CF, Data, GVar, DI, MetaLang>.

Система на основе таблиц событий представляет собой совокупность подсистем, обеспечивающих возможности проектирования, моделирования и управления. Система должна включать в себя следующие подсистемы: интерфейс пользователя, включающий подсистемы ввода и вывода; банк спецификаций, содержащий спецификации таблиц событий; банк данных; библиотека процедур и функций; интерфейс проектировщика; подсистема проверки и коррекции таблиц событий; подсистема инициирования взаимодействия; подсистема внутренних переменных.

Для обеспечения тех или иных возможностей функционирования, системе не обязательно использовать возможности всех описанных выше подсистем.

Для обеспечения функционирования системы в режиме проектирования, должны быть включены в систему, по крайней мере, следующие подсистемы: интерфейс проектировщика; банк спецификаций, содержащий спецификации таблиц событий; банк данных; библиотека процедур и функций; подсистема проверки и коррекции таблиц событий.

Для обеспечения функционирования системы в режиме моделирования, должны быть включены в систему, по крайней мере, следующие подсистемы: интерфейс пользователя, включающий подсистемы ввода и вывода; банк спецификаций, содержащий спецификации таблиц событий; банк данных; библиотека процедур и функций; подсистема инициирования взаимодействия; подсистема внутренних переменных.

Структура системы изображена на рисунке 3.

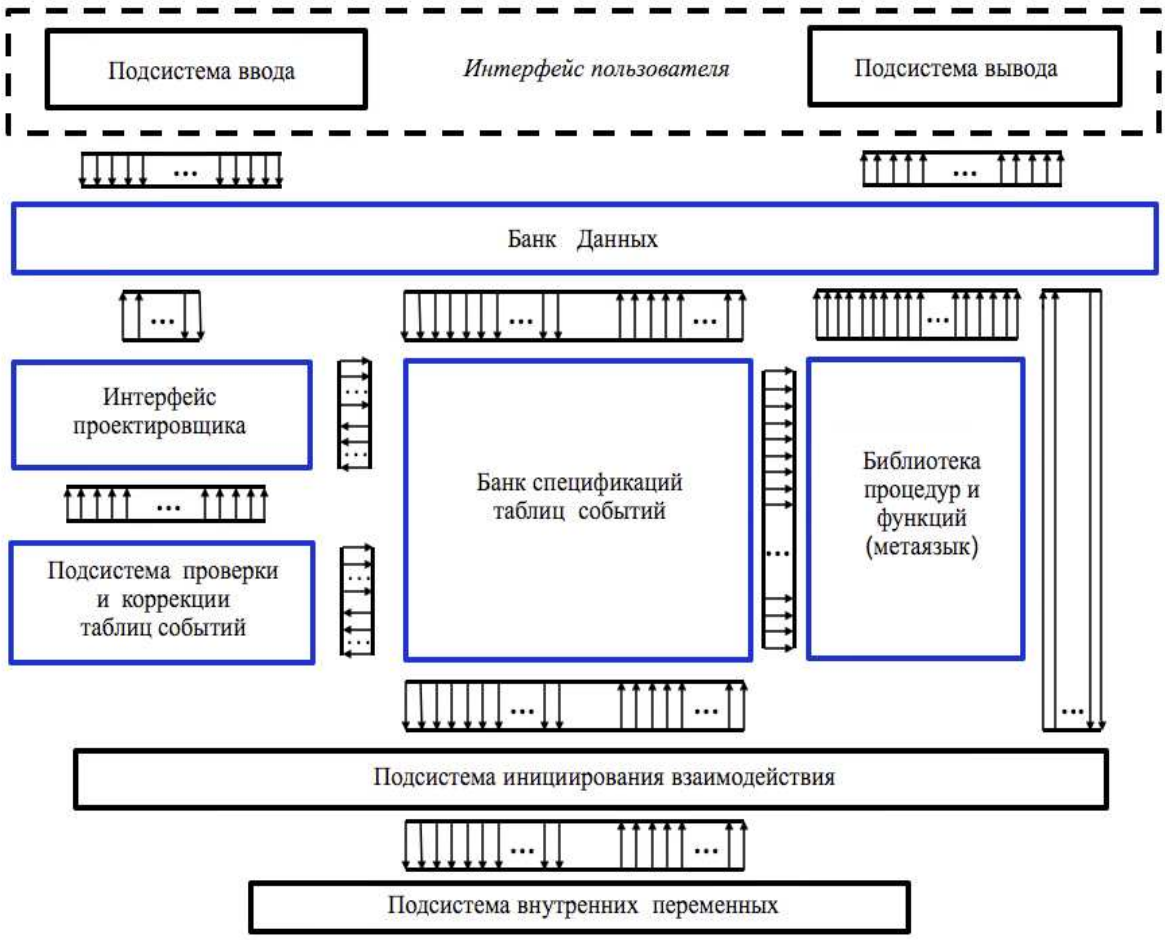

Рисунок 3 — Структура системы

Развитие систем на основе таблиц событий может дать мощный аппарат для автоматизации процессов в проектировании и моделировании. Таблицы событий могут стать неотъемлемой частью системного проектирования, так как с помощью языка таблиц событий доступно визуальное представление сложной логики программирования, простота для модификации, реализации, и автоматического преобразования в исполняемые программы.

Благодаря модульности, система на основе таблиц событий может быть легко модифицирована и легко расширена для обеспечения более широкого круга функциональных возможностей. Таким образом, разработка методов и средств спецификации, методик проверки и коррекции, методов функционирования, а также анализ средств реализации должны стать основным направлением исследований в области создания распределенных систем на основе таблиц событий.

В дальнейших исследованиях предполагается разработка методов проверки и коррекции таблиц событий, связанных с необходимостью организации интерактивного взаимодействия пользователя и системы в процессе проектирования сложных технических систем.

Библиографический список использованной литературы

1. Поспелов Д.А. Ситуационное управление: теория и практика / Д.А. Поспелов. — М.: Наука, 1986. — 288 с.

2. Хамби Э. Программирование таблиц решений / Э. Хамби. — М: Мир, 1976. — 86 с.

3. Сметанин Ю.М. Таблица решений как частный случай продукционных систем и их применение для фиксации принципа действия и ООД при обучении экономистов / Ю.М. Сметанин // Вестник Удмуртского Университета. Сер. Экономика и право: сб. науч. тр. — Ижевск, 2009. — Вып. 1. — С. 85–96.

4. Апраксин Ю.К. Спецификация сетевых протоколов средствами таблиц событий / Ю.К. Апраксин // Вестник СевГТУ. Сер. Информатика, электроника, связь: сб. науч. тр. — Севастополь, 2001. — Вып. 31. — С. 4–8.

Поступила в редакцию 10.02.2011 г.

Кулешова О.М., Апраксін Ю.К. Специфікація розподілених систем на основі таблиць подій

Розглядається структура розподіленої системи на основі таблиць подій, що забезпечує можливості проєктування, моделювання та управління. Розробляються специфікації основних частин системи.

Ключові слова: таблиця подій, специфікація, структура.

Kuleshova O.N., Apraksin U.K. Specification of distributed systems based on the event tables

The article describes the structure of the distributed system based on the event tables, providing the possibility of designing, modeling, and control. Specifications for main parts of the system are developed.

Keywords: event table, specification, structure.