УДК 621.856

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ТЕОРИИ ГИБКИХ ТЕХНОЛОГИЧЕСКИХ СИСТЕМ И ИХ ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Е. И. Перовская,
доктор техн. наук, профессор
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Работа направлена на решение фундаментальной проблемы создания глобальных моделей социумов для проверки управленческих решений и их последствий без экспериментирования на людях и природе. Основной задачей является построение математической теории для описания метасистемы объединения разнородных моделей компонент социумов, являющихся результатами исследований как гуманитарных, так и точных наук. Разработанная теория и формальные методы предназначены для междисциплинарных исследований, учитывающих влияние результатов различных аспектов жизнедеятельности общества и природы как на отдельные социумы, так и на цивилизацию в целом при решении проблемы жизнеспособного (устойчивого) развития. Вторая часть статьи содержит основы теории гибких дискретных технологических систем, позволяющей построить формальную модель систем Человек–Культура–Технологии–Природа.

The work is directed on a solution of a fundamental problem of creation of global community models for checking of administrative solutions and their consequences without experimenting on Human and Nature. The basic goal is development of the mathematical theory of association of diverse community components models which are researches results of the both humanitarian and exact sciences in uniform system. The developed theory and formal methods are intended for interdisciplinary researches which take into account influence of results of various aspects of functioning of a society and a nature, both on separate communities, and on a civilization as a whole at the decision of a problem of viable (sustainable) development. The second part of article contains the base of the theory of the flexible discrete technological systems, allowing to constructing formal model of Human–Culture–Technology–Nature system.

Описание целей системы и теоретико-множественной модели гибких дискретных технологических систем

Для формального описания гибких дискретных технологических систем (ГДТС) используем теоретико-множественный подход М. Месаровича и Я. Такахары [1], который базируется на теории множеств и реляционной алгебре [2]. Модель системы M может быть задана как совокупность множеств характеристик системы $M = (A, B, C, ...)$. В общем случае $M = \{A_i / I = 1...I\}$, где I – число характеристик, A_i – характеристические множества. Тогда все множество состояний системы S_M может быть представлено как декартово произведение всех этих множеств: $S = (\times A_i / i = 1...I)$. Любое состояние системы s_k может быть описано как выборка из декартова произведения этих множеств: $(a_1, a_2, a_3, ..., a_{I-1}, a_I)$.

Выбор характеристических множеств всегда целесообразно начинать с определения целей существования и жизнедеятельности описываемой системы. В первой части статьи была введена реляционная структура цели как кортеж:

$$C = \langle d, \varepsilon, \theta, \tau_н, \tau_к \rangle,$$

где d – тип целевого (потока); ε – количество объектов типа d; θ – способ выполнения изменений свойств исходного объекта до получения его целевого состояния; $(\tau_н, \tau_к)$ – период, за который необходимо достигнуть цели. Множество целей на период T задается таблицей целевых заданий:

$$C_T = \{\langle d_l, \varepsilon_l, \theta_l, \tau_{нl}, \tau_{кl} \rangle / \, l = 1, L_T\},$$

где L_T – число элементарных целей, заданных на период T. Если взять объединение всех возможных C_T, то получим множество $\boldsymbol{C}$ целевых заданий, на которые должна быть рассчитана ГДТС. Следовательно, $\boldsymbol{C}$ является подмножеством декартова произведения доменов таблицы

$$C \subset D \times E \times \Theta \times T_н \times T_к,$$

при этом D – домен, определяющий множество возможных типов целевых объектов (потоков); E – домен разнообразия количественных характеристик по различным типам целевых объектов; Θ – домен, определяющий множество техпроцессов, на которые должна быть рассчитана исполнительная подсистема (ИПС); $T_н$ – множество моментов возможных начал выполнения целевых заданий; $T_к$ – множество моментов необходимых окончаний целевых заданий.

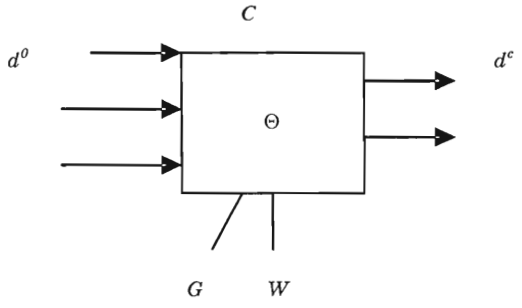
Таким образом, для описания ГДТС эти домены должны войти в число характеристических множеств ее теоретико-множественной модели:

$$M = (\boldsymbol{C}, D, \Theta, \boldsymbol{T}).$$

Технологический процесс Θ (способ изменения свойств $d^0{}_l$ для получения целевого объекта d_l) задается множеством операций: $\Theta_l = \{ O_{lj} / j = 1, J_l \}$, где O_{lj} – j-я операция l-го техпроцесса, обеспечивающая изменения каких-либо свойств d_l. Для выполнения каждой операции потребуются определенные исполнительные ресурсы r_{lj} и время t_{lj}. Множество $\{t_{lj}\}$ пополнит характеристическое множество $\boldsymbol{T}$, а домен ресурсов $\{t_{lj}\}$ из таблицы всех техпроцессов Θ составит характеристическое множество ресурсов системы R. Однако для различных систем не все выборки будут реализуемы, и чтобы определить допустимые состояния, необходимо задать множество ограничений Φ, при которых должны достигаться цели, и критериев F, по которым можно судить о качестве способов их достижения. Теперь можно написать полное теоретико-множественное представление статической модели ГДТС

$$M = (\boldsymbol{C}, D, \Theta, R, \Phi, F, \boldsymbol{T}).$$

Этот набор является инвариантом теоретико-множественного представления для систем класса ГДТС, на основе которого можно строить базы данных систем.



■ *Рис. 1. Структура компоненты ГДТС: цель C – объекты обработки (целевые потоки); d^0, d^c – технологические процессы Θ; O – характеристики операций; G – исполнительные средства; W – вспомогательные средства*

Описание формальной статической модели компоненты

Предметная область любой компоненты описывается универсальной структурой (рис. 1).

Характеристика элементарной цели уже приведена ранее:

$$C_l = \langle d_l, \varepsilon_l, \theta_l, \tau_{нl}, \tau_{кl} \rangle.$$

Характеристика объектов обработки d_l задается структурой

$$Xd_l = \langle d_l, \{FF\}_l \rangle,$$

где $\{FF\}_l$ – таблица с гибкой структурой.

Строка таблицы с гибкой структурой $FF = \langle IA, ZA, EI \rangle$ (таблицу с задаваемым именем атрибута в каждой строке назовем *гибким фреймом*); IA – имя атрибута (свойства) объекта, характеризующего d_l; ZA – значение атрибута, характеризующего d_l; EI – единицы измерения или код, определяющий метрику данного свойства, так как различные атрибуты будут иметь свои единицы измерения.

Технологический процесс (для получения целевого объекта d_l с заданными свойствами) задается множеством операций:

$$\Theta_l = (O_{lj} / j = 1, J_l),$$

где O_{lj} – j-я операция l-го техпроцесса изменения каких-либо атрибутов d_l из структуры Xd_l.

Характеристика операции задается структурой

$$XO_j = \langle O_j, G_m, W_p, U_{jmp}, t_{jmn} \rangle,$$

где G_m – группа однородных исполнительных средств, способных выполнить O_j; W_p – группа однородных вспомогательных средств, необходимых для исполнительных средств группы G_m при выполнении операции O_j; U_{jmp} – способ выполнения операции O_j, т. е. алгоритм изменения значений атрибутов из структуры Xd_l исполнительным средством из группы G_m с помощью вспомогательного средства из группы W_p; t_{jmn} – время, требуемое на выполнение операции O_j исполнительным средством из группы G_m с помощью вспомогательного средства из группы W_p.

Технологический процесс можно задать прямо в виде таблицы характеристик операций:

$$\Theta_l = \{\langle O_{lj}, G_m, W_p, U_{jmp}, t_{jmn} \rangle\} j = 1, J_l,$$

где J_l – число операций l-го техпроцесса.

Характеристика исполнительных средств задается реляционно-иерархической структурой

$$XG_m = \langle G_m, N_m, \{\langle g_{mn}, k_н, k_{пр}, k_{mn} \rangle\} \rangle,$$

где N_m – количество однородных исполнительных средств в группе G_m; g_{mn} – имя конкретного исполнительного средства из G_m; $k_н$ – коэффициент на-

дежности конкретного исполнительного средства g_{mn}; $k_{пр}$ – коэффициент производительности конкретного исполнительного средства g_{mn}; k_{mn} – координата конкретного исполнительного средства из G_m.

Характеристика вспомогательных средств задается реляционно-иерархической структурой

$$XW_p = \langle W_p, Q_p, \{\langle w_{pq}, \{FF\}\rangle\}\rangle,$$

где Q_p – количество вспомогательных средств в группе W_p; $\{FF\}$ – таблица, описывающая свойства вспомогательного средства w_{pq} из группы W_p с помощью гибкого фрейма FF. Использование гибкого фрейма вызвано многообразием имен атрибутов и их количеством при описании вспомогательных средств, необходимых основному средству для выполнения операции.

Таким образом, статическую модель любой компоненты или подсистемы ГДТС можно описать инвариантом из 6 реляционно-иерархических структур, которые легко сводятся к 9 реляционным, т. е. девятью типами таблиц. Все многообразие различий в свойствах и операциях техпроцессов (законов функционирования) разнообразных разнородных компонент, которые описываются профессионалами различных областей знаний в своих специфических моделях, заключено в атрибуте U_{jmp} *характеристики операции* O_j, *т. е. алгоритме изменения значений атрибутов ZA из структуры Xdl исполнительным средством из группы* G_m *с помощью вспомогательного средства из группы* W_p. При создании компьютерной имитационной модели эти алгоритмы разрабатываются независимо для каждой операции каждого техпроцесса каждой компоненты специалистами (технологами) предметной и проблемной области этой компоненты и системного аналитика-программиста и закладываются в динамические библиотеки, организованные по типам компонент и задач. Следовательно, разработка имитационной модели может вестись независимо параллельно большим коллективом различных специалистов-экспертов и аналитиков-программистов, что чрезвычайно важно при создании моделей систем класса Человек–Культура–Технологии–Природа (ЧКТП), таких как крупные фирмы, корпорации, производственные системы, территориальные образования, города, государства, экосистемы. При этом, только если все компоненты будут разрабатываться на едином принципе инварианта описания компонент, все они легко поймут друг друга, потому что не важно, каким образом изменялись атрибуты потоков, которыми обмениваются подсистемы и компоненты, важно, чтобы все понимали, в каких атрибутах каких структур в какой форме надо читать требующуюся информацию. Любая компонента становится прозрачной по структуре информации для всех других компонент любой подсистемы на любом уровне иерархии.

Формальное описание динамической модели компоненты и системы

Принимая решения в социумах или экосистемах, трудно предвидеть последствия принимаемых решений в процессе их реализации при функционировании системы. Имитационные модели предназначены для экспериментального исследования процесса функционирования систем в тех случаях, когда экспериментировать на системе невозможно или очень опасно. Для представления динамических процессов, протекающих в моделируемых системах, необходимо создание инвариантного описания динамической модели компонент и их взаимодействия в процессе функционирования всей системы, чтобы иметь возможность при создании имитационной модели объединять единым способом разрабатываемые независимыми экспертами компоненты и подсистемы.

Динамические параметры предметной области могут быть описаны следующими структурами:

таблицами сигналов об изменении состояний (о событии) и управляющих сигналов;

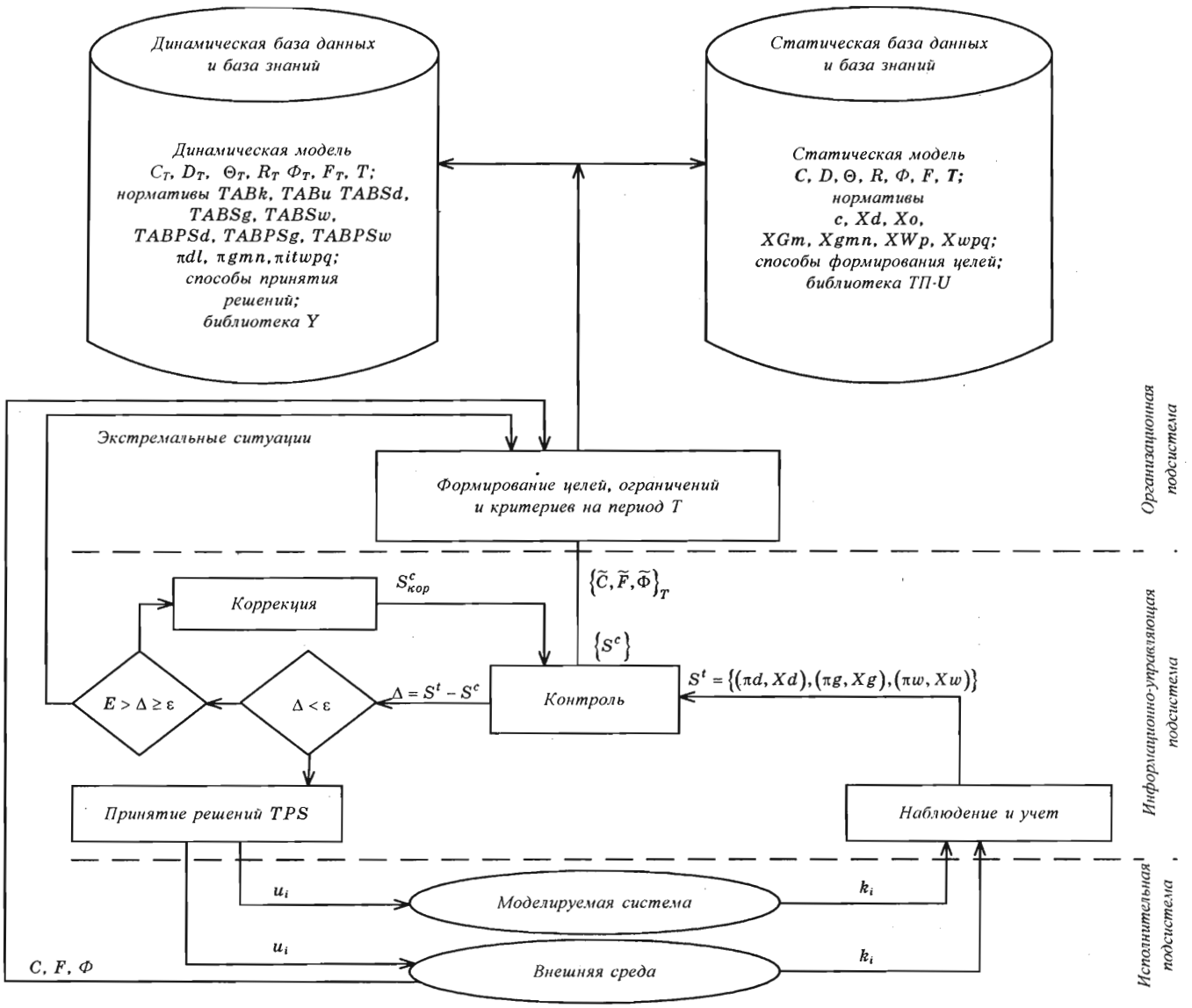
таблицами возможных состояний объектов обработки, исполнительных и вспомогательных средств;

■ ***Таблица Tabki***

№	Наименование	Код *ki*
1	Начат техпроцесс	*k1*
2	Завершен техпроцесс	*k2*
3	Начата подготовка	*k3*
4	Завершена подготовка	*k4*
5	Начата операция	*k5*
6	Завершена операция	*k6*
7	Начат ремонт (исправление)	*k7*
8	Завершен ремонт	*k8*
9	Сбой (отказ)	*k10*

■ ***Таблица Tabui***

№	Наименование	Код *ui*
1	Начать техпроцесс	*u1*
2	Завершить техпроцесс	*u2*
3	Начать подготовку	*u3*
4	Завершить подготовку	*u4*
5	Начать операцию	*u5*
6	Завершить операцию	*u6*
7	Начать исправление (ремонт)	*u7*
8	Завершить ремонт	*u8*



■ ***Рис. 2. Структура системы управления моделью и процессом моделирования***

таблицами переходов состояний (ТПС) объектов обработки, исполнительных и вспомогательных средств;

портретами объектов обработки, исполнительных средств, вспомогательных средств, описывающих их текущее мгновенное состояние.

Сигналы о событии в компонентах.

Для описания взаимодействия всех компонент, подсистем между собой и системы с внешней средой введем структуру сигналов о событиях. Событием в системе считается любое изменение в атрибутах d, g, w, связанное с выполнением технологического процесса или наличием сбоев и отказов. Сигнал, несущий сообщение о том, что у компоненты g_{mn} произошло событие типа k_i в момент τ, задается кортежем отношения $k = \langle g_{mn}, k_i, \tau \rangle$. Значения кодов сигналов k_i задаются таблицей *Tabki* с кортежем заголовка *Tabki* = ⟨наименование, код⟩. Аналогично с таким же кортежем задается таблица управляющих сигналов *Tabui*, вырабатываемых блоком принятия решений (рис. 2).

Таблица возможных состояний. Возможные состояния задаются относительно любой запланированной операции или техпроцесса (начата операция, идет операция, завершена операция, сбой и т. д.).

Для задания возможных состояний d относительно запланированной операции следует заполнить таблицу *TABSd*. Строка состояний *Sd* задается в виде логического слова из четырех логических параметров:

$$P1 = \begin{cases} 1, \text{ если техпроцесс начат;} \\ 0, \text{ если техпроцесс не начат;} \end{cases}$$

$$P2=\begin{cases}1, \text{ если операция выполняется;}\\ 0, \text{ если операция не выполняется;}\end{cases}$$

$$P3=\begin{cases}1, \text{ если операция завершилась браком;}\\ 0, \text{ если операция завершилась без отклонений;}\end{cases}$$

$$P4=\begin{cases}1, \text{ если техпроцесс завершен;}\\ 0, \text{ если техпроцесс не завершен.}\end{cases}$$

Строка таблицы возможных состояний для d задается кортежем

$TabSd = \langle наименование, Si, P1, P2, P3, P4\rangle$.

■ ***Таблица*** *TabSd*

i	Наименование	*Si*	*P1*	*P2*	*P3*	*P4*
0	Начальное	*S0*	0	0	0	0
1	Техпроцесс начат	*S1*	1	0	0	0
2	Операция начата (идет обработка)	*S2*	1	1	0	0
3	Операция завершена	*S3*	1	0	0	0
4	Операция завершена неудачно (брак)	*S4*	1	0	1	0
5	Техпроцесс завершен	*S5*	1	0	0	1
6	Техпроцесс завершен неудачно (с браком)	*S6*	1	0	1	1

В таблице приведено минимально необходимое число возможных состояний, однако оно зависит от решаемой задачи, и при реализации конкретной модели число строк таблицы может быть увеличено без необходимости изменения программы инвариантного ядра.

Для групп исполнительных средств G_m зададим возможные состояния логическими параметрами:

$$P1=\begin{cases}1, \text{ если исполнительное средство исправно;}\\ 0, \text{ если исполнительное средство неисправно;}\end{cases}$$

$$P2=\begin{cases}1, \text{ если исполнительное средство занято;}\\ 0, \text{ если исполнительное средство не занято;}\end{cases}$$

$$P3=\begin{cases}1, \text{ если исполнительное средство}\\ \text{готово к операции;}\\ 0, \text{ если исполнительное средство}\\ \text{не готово к операции;}\end{cases}$$

$$P4=\begin{cases}1, \text{ если исполнительное средство работает;}\\ 0, \text{ если исполнительное средство не работает.}\end{cases}$$

Таблица возможных состояний для G_m задается тем же кортежем, что и для *TabSd*.

■ ***Таблица*** *TabPSd*

i	Наименование состояния	*S*	*P1*	*P2*	*P3*	*P4*
0	Начальное	*S0*	0	0	0	0
1	Исправен	*S1*	1	0	0	0
2	Идет подготовка (наладка) к операции	*S2*	1	1	0	0
3	Подготовка завершена	*S3*	1	1	1	0
4	Идет обработка	*S4*	1	1	1	1
5	Обработка завершена	*S4* (*S3*)	1	1	1	0
6	Сбой (брак, отказ, нештатная ситуация)	*S5*	0	1	1	0
7	Идет ремонт (восстановление)	*S6*	0	1	0	0

Аналогично можно задать *TABSWp* для вспомогательных средств:

$TabSW = \langle наименование, Si, P1, P2, P3, P4\rangle$.

Законы переходов состояния. Зададим законы переходов из одного состояния в другое объектов обработки, групп исполнительных и вспомогательных средств (d, G_m, W_p) в виде таблиц переходов состояний

$$TABPS = \langle S_{old}, k_i, S_{new}, Y, u_i\rangle,$$

где S_{old} – старое состояние, в котором находился объект до наступления события, о котором сообщил сигнал k_i; S_{new} – новое возможное состояние, в которое должен перейти объект в результате события; Y – алгоритм (закон) перехода, отражающий в атрибутах и структурах изменения, произошедшие в системе в результате события; u_i – тип вырабатываемого управляющего сигнала о следующем назначаемом событии.

Например: $TABPSd = \langle S_{old}, k_i, S_{new}, Y, u_i\rangle$ для объектов обработки.

■ ***Таблица*** *TabPSd*

a	S_{old}	*ki*	S_{new}	*Y*	*ui*
0	*S0*	*k1*	*S1*	*Y0_1*	*u3*
1	*S1*	*k3*	*S2*	*Y1_3*	
2	*S2*	*k6*	*S3*	*Y2_4*	*u3*
3	*S3*	*k3*	*S2*	*Y3_3*	
4	*S3*	*k2*	*S5*	*Y3_2*	
5	*S2*	*k10*	*S4*	*Y2_10*	
6	*S4*	*k10*	*S6*	*Y5_10*	*u7*

Аналогичные таблицы переходов состояний должны быть заданы для G_m и W_p в соответствии с их таблицами возможных состояний *TabSGm* и *TabSWp* соответственно.

Портреты объектов обработки, исполнительных средств, вспомогательных средств.

■ *Таблица TabPSGm*

S_{old}	ki	S_{new}	Y	ui
S0	k0	S1	Y0_1	u3
S1	k3	S2	Y1_3	
S2	k4	S3	Y2_4	u5
S3	k5	S4	Y3_5	
S4	k6	S3	Y4_6	u3 (u5)
S1	k10	S5	Y1_10	u7
S2	k10	S5	Y2_10	u7
S3	k10	S5	Y3_10	u7
S4	k10	S5	Y4_10	u7
S5	k7	S6	Y5_7	u3
S6	k8	S1	Y6_8	u3 (u5)

Портреты предназначены для обеспечения возможности в любой момент времени в процессе функционирования системы наблюдать текущее состояние каждой компоненты системы.

Портрет объекта обработки πd задается кортежем

$$\pi d = \langle d_l, O_j, g_{mn}, w_{pq}, \varepsilon, \varepsilon^*, \varepsilon^{бр}, U_{mnj}, \tau_{нl}, \tau_{кl}, \tau_{нl}^{ф}, \tau_{кl}^{ф}, S\rangle,$$

где ε – число объектов обработки, заданное целевым заданием C_l; ε^* – число обработанных объектов типа d_l к текущему моменту; $\varepsilon^{бр}$ – число неудачно завершенных обработок объекта d_l(брак); U_{mnj} – способ выполнения операции (алгоритм изменения значений атрибутов из XO_j); $\tau_{нl}$, $\tau_{кl}$ – планируемые моменты начала и окончания операции O_j; $\tau_{нl}^{ф}$, $\tau_{кl}^{ф}$ – фактические моменты начала и завершения операции O_j; S – текущее состояние из списка возможных состояний объектов обработки относительно операции.

Портреты исполнительного πg и вспомогательного πw средств характеризуются соответственно кортежами:

$$\pi g = \langle g_{mn}, O_j, d_l, w_{pq}, \tau_{нl}, \tau_{кl}, \tau_{нl}^{ф}, \tau_{кl}^{ф}, S\rangle \text{ и}$$

$$\pi w = \langle w_{pq}, O_j, g_{mn}, d_l, Res, Rest, \tau_{нl}, \tau_{кl}, \tau_{нl}^{ф}, \tau_{кl}^{ф}, S\rangle,$$

где *Res*, *Rest* – полный и остаточный ресурс вспомогательного средства, если средства расходуемые.

Сигналы о событиях в моделируемой системе или внешней среде k_i поступают на блок формирования текущего состояния объектов моделирования (блок наблюдения, см. рис. 2), который производит изменения атрибутов портретов всех объектов, являющихся участниками события. Блок наблюдения обеспечивает возможность ЛПР иметь полную информацию в любой текущий момент о любом компоненте системы. Наблюдаемость системы определяется полнотой таблицы возможных сигналов относительно таблиц возможных (допустимых) состояний S всех типов объектов системы (d, g, w). Необходимость различимости состояний определяется целями, ограничениями и критериями решаемой задачи, т. е. зависит от компетентности экспертов, определяющих содержание всех таблиц.

Введенные структуры и два вида функций (U и Y) позволяют полностью задать предметную и проблемную область задачи имитационного моделирования.

Функция U_{jmp} из структуры характеристики операций $XO_{ij}=\langle O_{ij}, G_m, W_p, U_{jmp}, t_{jmp}\rangle$ отражает правила выполнения операции, т. е. способ изменения значений атрибутов в характеристике целевых объектов (потоков). Чтобы задать техпроцесс, надо сначала эксперту-технологу описать U_{jmp} для всех операций.

Правила обработки сигналов k_i по таблицам переходов состояний (Y) задают:

последовательность отображения изменений параметров состояния в портретах участников события (πd, πg, πw);

запуск процедуры U_{jmp} при сигнале окончания операции O_{ij} (фиксация изменения атрибутов объектов, исполнительных и вспомогательных средств в их гибком фрейме – *TABFF*);

поиск следующей операции и установка ее имени в портрет объекта обработки;

поиск свободных ресурсов в группах G_m и W_p, указанных в характеристике следующей операции, фиксация планируемой операции и ее исполнителей в портретах участников следующей операции;

формирование управляющего сигнала $u=\langle g_{mn}, u_i, \tau\rangle$ о необходимости исполнительному средству g_{mn} начать новую операцию.

Алгоритм обработки сигналов $k=\langle g_{mn}, k_i, \tau\rangle$ состоит из стандартной последовательности алгоритмов:

$$Y = Y_{inv} \oplus Y_{\tau} \oplus U_{jmp} \oplus F(u_i).$$

Y_{inv} – полностью инвариантная часть алгоритма, не зависящая ни от типа пришедшего сигнала k_i, ни от его источника g_{mn}, осуществляет поиск портретов всех участников операции, относительно которой произошло событие, и по ТПС фиксирует новое состояние в портретах πg_{mn}, πd_l, πw_{pq}.

Y_{τ} – инвариантная часть алгоритма, фиксирующая момент происшедшего события из кода сигнала $k.\tau$ в портреты участников πg_{mn}, πd_l, πw_{pq}.

U_{jmp} – алгоритм фиксации *результатов выполнения операции из структуры XO в структуре Xd*, если тип сигнала о событии $k.k_i$ является «завершением события».

$F(u_i)$ – алгоритм выработки управляющего сигнала u_i, выполняющий поиск следующей по техпроцессу операции для объекта предыдущей опе-

рации по результатам контроля правильности выполнения завершенной операции и наличию требуемых свободных ресурсов для нее.

Первые три алгоритма относятся к функциям блока наблюдения и учета, а четвертая выполняет функцию контроля соответствия текущего состояния целевому и принятия решений (см. рис. 2).

Запись алгоритмов в языке теории множеств и реляционной алгебры позволяет строго формализовать их до уровня общих операторов любых алгоритмических языков. В терминах операторов стандартных алгоритмических языков квантор общности $\forall$ (α) можно рассматривать как оператор цикла с заданным списком значений ($\forall\, \alpha(1...A)$ – «выполнять для всех значений списка»). Квантор существования может быть интерпретирован как условный оператор: $\exists\,(l')$:(P) – существует ли такое l', для которого выполняется предикат $P(l')$ или $\exists\,(P)(O)$ – если существует предикат P, то выполнить оператор O. Например, сформулируем несколько упрощенно алгоритм Y_{inv}.

Пусть пришел сигнал о событии $k = \langle g_{12}, k_2, 12\rangle$. Найдем портрет источника сигнала $k.g_{12}$, для чего необходимо в наборе портретов исполнительных средств $\pi g[m, n]$ найти тот, у которого значение поля $\pi_g.g_{mn}$ совпадает с полем $k.g_{12}$:

$$\forall\, m\,(m=1...M)\,\forall\;\; n\,(n=1...N)$$
$$(\exists\,(m', n')\,(\pi g.g_{mn}[m,n] = k.g_{12}).$$

В языках программирования это соответствует вложенным циклам по m и n, а во внутреннем цикле условный оператор проверяет выполнение условия $(\pi g.g[m,n] = k.g_{12})$. При его выполнении фиксируются текущие значения m и n:$(m' := m;\ n' := m;)$.

Теперь найденные координаты (m', n') строки таблиц портретов позволят получить портрет адресата $\pi g[m', n']$ и из него найти портреты второго и третьего участника событий:

$$\forall\, l\,(l = 1...L)\,(\exists\;\,(l')\colon (\pi d.d_l[l'] = \pi g.d_l[m', n']);$$

$$\forall\, p(p=1...P)\,\forall\, q(q=1...Q)(\exists\,(p', q')\colon(\pi w.w_{pq}[p', q'] = \pi g.w_{pq}[m', n']).$$

В трех найденных портретах $\pi g[m', n']$, $\pi d[l']$ и $\pi w[p', q']$ необходимо заменить старые значения состояний на новые по таблицам переходов состояния, для чего в таблицах переходов состояния для d_l требуется найти строку, удовлетворяющую следующим требованиям:

$$(\pi d.S[l'] = TABPSd.S_{old}[\alpha] \text{ и}$$
$$k.k_i = TABPSd.k_i[\alpha],$$

где α – индекс строк таблицы переходов состояния, а β и γ – индексы строк ТПС для g_{mn} и w_{pq} соответственно.

$$\forall\, \alpha(1...A)\,(\exists\;\,(\alpha')\colon (\pi d.S[l'] = TABPSd.S_{old}[\alpha]\ \&$$
$$k.k_i = TABPSd.k_i[\alpha]);$$

$$\forall\, \beta(1...B)\,(\exists\,(\beta')\colon (\pi g.S[m', n'] = TABPSg.S_{old}[\beta]\ \&$$
$$k.k_i = TABPSg.k_i[\beta]);$$

$$\forall\, \gamma(1...\Gamma)\,(\exists\,(\gamma')\colon (\pi w.S[p', q'] = TABPSw.S_{old}[\gamma]\ \&$$
$$k.k_i = TABPSw.k_i[\gamma]).$$

Остается занести новые состояния из найденных строк ТПС в поля S портретов $\pi g[m', n']$; $\pi d[l']$ и $\pi w[p', q']$:

$$\pi d.S[l'] := TABPSd.S_{new}[\alpha'];$$

$$\pi g.S[m', n'] := TABPSg.S_{new}[\beta'];$$

$$\pi w.S[p', q'] := TABPSw.S_{newd}[\gamma'].$$

Следующий алгоритм Y_τ должен проверить принадлежность сигнала либо к сигналам типа начал (начало техпроцесса, операции, ремонта, подготовки и т. д.), либо к сигналам типа окончания и занести в портреты участников события момент прихода сигнала $k.\tau$. При задании таблиц сигналов они разбиваются на два подмножества: KU – множество начал ($k_i \in KU$ нечетные) и KE – множество окончаний ($k_i \in KE$ четные). Тогда алгоритм в той же интерпретации представляется в следующем виде:

$$\exists\;(k.k_i \in KE)(\pi d.\tau_{кl}^{\Phi}[l'] := k.\tau)\;\Lambda$$
$$\exists\;(k.k_i \in KU)(\pi d.\tau_{нl}^{\Phi}[l'] := k.\tau);$$

$$\exists\;(k.k_i \in KE)(\pi g.\tau_{кl}^{\Phi}[m', n'] := k.\tau)\;\Lambda$$
$$\exists\;(k.k_i \in KU)(\pi g.\tau_{нl}^{\Phi}[m', n'] := k.\tau);$$

$$\exists\;(k.k_i \in KE)(\pi w.\tau_{кl}^{\Phi}[p', q'] := k.\tau)\;\Lambda$$
$$\exists\;(k.k_i \in KU)(\pi dw.\tau_{нl}^{\Phi}[p', q'] := k.\tau).$$

Запись приведена только как пример возможности описывать алгоритм формально и точно до типов операторов алгоритмических языков (для простоты опущены проверки тупиков и неоднозначности выходов).

Следующая составляющая алгоритма принятия решения в ответ на сигнал о событии в системе должна зафиксировать результаты, выражающиеся в тех атрибутах, свойствах и параметрах, которые изменились в процессе выполнения завершенной операции. Все они хранятся в структурах характеристик Xd, Xg, Xw в форме таблицы гибкого фрейма $TABFF = \{\langle IA, ZA, EI\rangle\}$. Это и позволяет объединять разнородные компоненты, функционирование которых описывается различными способами: таблицами результатов экспериментальных наблюдений за процессом функционирования (графиками, диаграммами, гистограммами и т. д.); расчетами по известным формулам в момент завершения операции; решением дифференциальных или алгебраических уравнений, логических или реляционных отношений, уравнений нелинейной динамики, в частных производных и т. д. Алгоритм U_{jmp} может описывать любыми способами любые процессы выполнения операций. Для общей модели важно лишь, чтобы они были выбраны и обоснованы высокопрофессиональными экспертами в конкретной области

знаний и исходные данные и результаты были представлены в структурах единой описанной модели. Алгоритмы функционирования компонент и подсистем U_{jmp} составляют собственную библиотеку для каждой конкретной системы и задачи и не входят в инвариантное ядро модели. Это и есть та часть модели, которая позволяет настраивать инвариант модели ГДТС на любую систему этого класса. Остальная настройка осуществляется просто заполнением значениями структур данных (баз данных) описанной модели.

Следующая часть алгоритма Y обработки сигнала о событии $k - F(u_i)$ отображает процесс принятия решения по выработке управляющего сигнала для выбора следующей операции. Количество возможных способов выбора управляющего воздействия зависит от оставшихся невыполненных целей и техпроцессов, ограничений, критериев и текущих состояний всех компонент системы. Так как функция наблюдаемости и учета изменения состояний исполнительного средства g_{mn}, объекта обработки d_l и вспомогательного средства w_{pq} уже выполнена в алгоритмах $Y_{inv} \oplus Y_\tau \oplus U_{jmp}$, то для принятия решения необходимо выполнить функцию контроля возможного рассогласования между целевым и текущим состоянием системы. Текущее состояние системы отражается в наборе портретов $S^t = \{\pi g, \pi g, \pi w\}$ и множестве характеристик Xd, Xg, Xw.

Целевые состояния можно задать в различных формах, например в виде расписания функционирования подсистем или компонент. Важно, чтобы структуры описания целевого (S^c) и текущего (S^t) состояний имели одинаковую метрику для реализации функции их сравнения и контроля. При использовании расписания в качестве последовательности целевых состояний это условие выполняется, так как структура его соответствует структуре динамических портретов. Тогда рассогласование на выходе блока контроля Δ (см. рис. 2) будет иметь структуру расписания

$$\Delta = \langle \Delta d, \Delta o, \Delta g, \Delta w, \Delta\varepsilon, \Delta t \rangle,$$

где $\Delta d, \Delta o, \Delta g, \Delta w$ – результат символьного сравнения фактического и планового значения типа целевого объекта (потока), операции, исполнительного и вспомогательного средства; $\Delta\varepsilon = \pi d.\varepsilon - \pi d.\varepsilon^*$ показывает разность между плановым и фактическим количеством целевого объекта (потока), а $\Delta t = \pi d.\tau_{кl}^{ф} - \pi d.\tau_{кl}$ – рассогласование фактического и планового времени завершения операции. Остается выбрать следующую операцию из расписания для того целевого объекта, у которого успешно завершилась операция, найти портреты запланированных ресурсов, внести изменения в их портреты (в случае, если у них за это время не произошли незапланированные события) и сформировать сигнал $u5$ «начать следующую операцию». Однако создание алгоритмов составления расписаний, расчет локальных и глобальных резервов, коррекций расписаний при небольших сбоях – все это сложные комбинаторные задачи, но они тоже содержат инвариантные части решений и настраиваемые на конкретные системы и ситуации алгоритмы, определяемые экспертами в конкретных областях знаний. Важно понять, что число возможных решений для каждого типа рассогласований определяется ограничениями и критериями, которые организационная подсистема (ОПС) задает организационно-управляющей подсистеме (ИУПС). Она должна задавать правила принятия решений и допустимость ситуаций. В случае экстремальной ситуации принятие решений переходит к ОПС, потому что только она может отменить какие-либо цели или увеличить ресурсы системы.

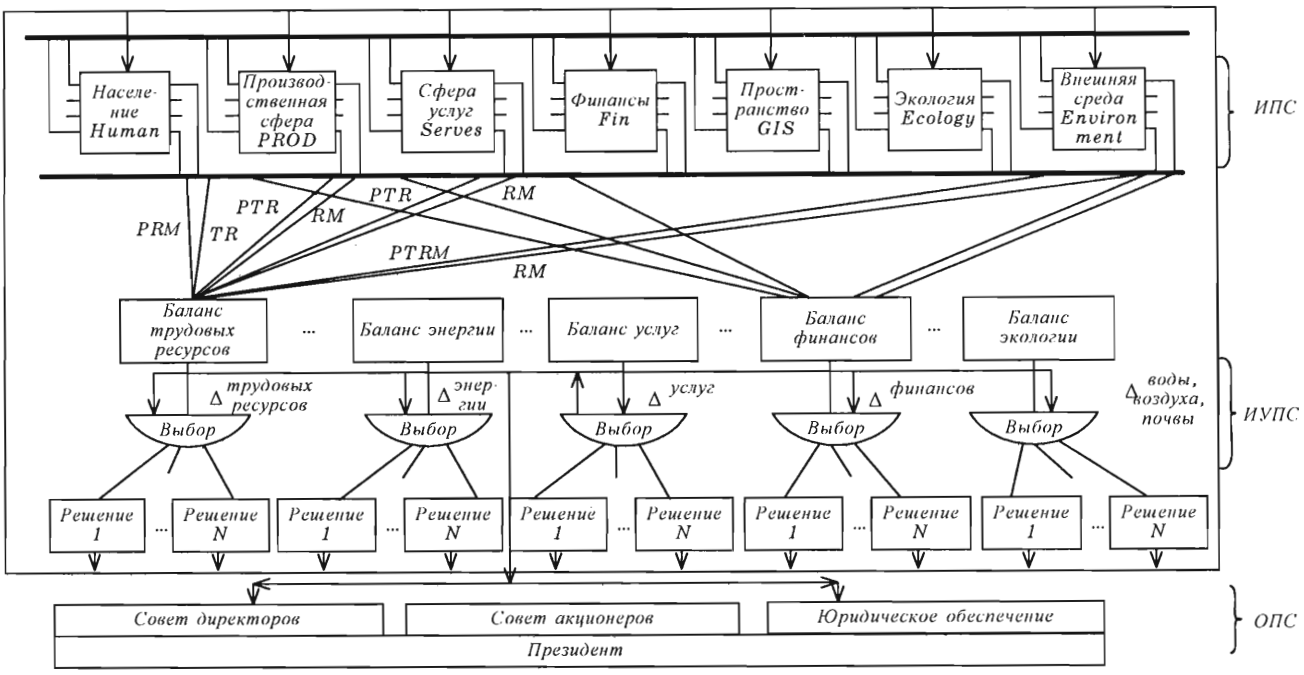
Объединение компонент в единую модель

Описание межкомпонентных связей в системе. Инвариантная структура описания исполнительной подсистемы ЧКТП содержит только самые общие подсистемы и потоки, которыми они обмениваются в процессе существования системы (рис. 3; подробнее см. рис. 5, ч. I). На самом верхнем уровне иерархии каждой подсистемы потоки могут обладать характеристиками, отражающими только самые общие их свойства. Например, подсистема «Население» (подсистема, обеспечивающая жизнедеятельность людей как социальных и биологических объектов любого социума) обладает потоком «Люди». Весь поток в целом можно характеризовать только количеством людей в потоке и общими показателями, характерными для всех людей или для всего потока. На этом уровне может решаться только задача замкнутости системы по наличию потоков из каждой подсистемы и потребностей в них других подсистем. Так как потоки различны по своей природе и имеют собственные метрики, то для проверки замкнутости системы и балансов ресурсов в процессе обмена различными потоками между подсистемами необходимо иметь общий эквивалент для сравнения. Его функцию и выполняет финансовый поток. Характеристика потоков любого уровня имеет инвариантный вид для всех потоков:

$$Xd^c = \{\langle IA, ZA, EI \rangle\} = TABFF.$$

Однако строки таблицы у всех потоков на верхнем уровне будут одинаковы. Например, *TABFF* для потока «Люди» подсистемы «Население»:

IA	*ZA*	*EI*
Имя потока *ID*	«Люди»	Строка
Количество в потоке ε	5000000	Целое число
Цена единицы потока cost		Число
Стоимость потока price		Число
Минимально необходимое количество min	4500000	Целое число
Потребность всех подсистем в этом потоке $\varepsilon^{тр}$	5100000	Целое число



■ ***Рис. 3.*** *Структура имитационной модели ЧКТП системы*

Или *TABFF* для основного целевого потока фирмы «Товары» из подсистемы «Производственная сфера»:

IA	*ZA*	*EI*
Имя потока *ID*	«Товары»	Строка
Количество в потоке ε	1000000	Целое число
Цена единицы потока cost	100	Число
Стоимость потока price	100000000	Число
Минимально необходимое количество min	500000	Целое число
Потребность всех подсистем в этом потоке $\varepsilon^{тр}$	950000	Целое число

Атрибут «Минимально необходимое количество» определяет уровень выпуска целевого потока каждой подсистемы, который еще обеспечивает выживание всей системы.

Так как число строк и наименование атрибутов для всех потоков одинаковы, то выгоднее использовать для представления структуры описания свойств потока на верхнем уровне не гибкий фрейм *FF*, а жесткий, заданный кортежем $Xd^{c} = \langle ID, \varepsilon, \text{cost}, \text{price}, \text{min}, \varepsilon^{тр} \rangle$. Этих данных будет достаточно для решения задачи проверки условий жизнеспособности целостной системы в начальный момент (при создании системы) и в любой текущий момент при ее функционировании. Введение стоимостного эквивалента каждого потока позволяет решать (отображать) все экономические задачи как статического, так и динамического типа, так как изменение цены единицы потока и его стоимости зависит от способа функционирования (Θ) подсистемы-источника потока и стоимости всех входных потоков из других подсистем. Такая модель позволяет ЛПР в одной подсистеме использовать знания экспертов о законах функционирования различных подсистем, выраженных в моделях, языках и теориях, незнакомых этому ЛПР, но видимых ему в структурах, отражающих результаты их жизнедеятельности. На рис. 3 приведена схема структуры связей трех подсистем ЧКТП-систем. На верхнем уровне находится ИПС, состоящая из семи подсистем. Результатом их функционирования являются события, изменяющие значения атрибутов выходных потоков. Информация о событиях поступает на блоки наблюдения и контроля информационно-управляющей подсистемы. Так как сравниваться могут только потоки, одинаковые по метрикам, то и балансных блоков должно быть столько, сколько имеется сравнимых потоков. Например, из подсистемы «Население» может поступить информация об изменении числа работающих (массовые увольнения в городе, «демографический провал» и т. д.). На блок баланса по трудовым ресурсам приходит информация о числе рабочих мест (*RM*) и потребности в трудовых ресурсах (*PTR*) из подсистем «Производственная сфера» и «Сфера услуг». Измеряются три рассогласования: количество трудовых ресурсов и рабочих мест (*RM-TR*), количество трудовых ресурсов и потребность в трудовых ресурсах (*TR-PTR*), количество рабочих мест и потребность в рабочих местах (*RM-PRM*). Эти рассогласования и порождают задачу (цель) системе управления: выработать управляющее воздей-

■ *Таблица* $XPS_1^{вых}$

№ потока l	Имя целевого потока d_{il}	Имя подсистемы-приемника потока PS_j	Функция отчисления $Xdl.кол$, %	Примечания
1	Трудовые ресурсы	Производственная сфера	70	На текущий момент 70% используются производственной сферой
1	Трудовые ресурсы	Сфера услуг	25	На текущий момент 25% используются сферой услуг
1	Трудовые ресурсы	Внешняя среда	5	5% уволилось из системы
2	Потребность в трудовых ресурсах	Внешняя среда	6	На 0,06% отдел кадров объявил вакансии
3	Потребность в финансовом потоке	Подсистема финансов	100	Подошел срок выплаты зарплаты

■ *Таблица* $XPS_1^{вх}$

№ потока l	Имя целевого потока d_{il}	Имя подсистемы-приемника потока PS_j	Функция отчисления $Xdl.кол$, %	Примечания
1	Люди	Население	30	Переместились на рабочие места внутри системы
1	Люди	Внешняя среда	25	Пришли в поисках работы
2	Потребность в трудовых ресурсах	Производственная сфера	9	Требуется замена уволенным
2	Потребность в трудовых ресурсах	Сфера услуг	25	Набирается новый отдел

ствие для уменьшения или устранения рассогласования. Для решения этой задачи необходима дополнительная информация о свойствах (атрибутах) потоков, требующих балансировки. Это означает, что для решения поставленной задачи в характеристику потока Xd должны быть добавлены новые атрибуты. Появление новых атрибутов потребует дополнения модели новыми компонентами, определяющими техпроцессы обработки новых атрибутов.

В качестве примера рассмотрим эту схему как модель фирмы. ИПС состоит из семи подсистем. Каждая подсистема имеет свой основной целевой поток и множество потоков потребностей, адресованных другим подсистемам, необходимым для формирования своего основного целевого потока. Если при создании фирмы правила функционирования всех подсистем были заданы таким образом, чтобы все потребности были удовлетворены, то на балансных блоках все рассогласования Δ должны быть в пределах возможных допусков ε.

Первая задача, которая должна решаться на имитационной модели, – проверка всех входных и выходных потоков на замкнутость. Не должно быть «висящих потоков» каждой потребности на входе подсистем, должен существовать целевой поток из какой-либо подсистемы (включая «Внешнюю среду»), удовлетворяющий эту потребность. Так как выходные целевые потоки одной подсистемы могут быть востребованы несколькими подсистемами, то необходимо ввести еще одну структуру, характеризующую подсистемы и их потоки с точки зрения законов потребления выходов одной подсистемы несколькими другими. Зададим их в форме реляционно-иерархического отношения.

Выходные потоки i-й подсистемы $PS_i(\forall\, i(i=1,7))$:

$$XPS_i^{вых} = \left\{ \left\langle \; d_{il}^{вых}, \left\{ \left\langle \; PS_j^{пр}, f\,(Xdl.кол) \; \right\rangle \right\} \; \right\rangle_{j\in J} \right\}_{l=1,L_i},$$

где $XPS_i^{вых}$ – характеристика взаимодействия подсистемы $PS_i\,(i=1,\,7)$ со своими приемниками по всем своим целевым потокам; $d_{il}^{вых}$ – целевые потоки подсистемы PS_i; $PS_j^{пр}$ – подсистемы-приемники; $f(Xdl.кол)$ – функция отчисления количества потока d_{il} для подсистемы PS_j.

Например, таблица, характеризующая связи подсистемы «Население» с другими подсистемами по целевым потокам «Трудовые ресурсы» (TR), «Потребность в финансовом потоке» (PF), «Потребность в трудовых ресурсах» (TR), отражена в таблице $XPS_1^{вых}$.

Такая же структура задается для всех входных потоков каждой подсистемы, так как в общем случае входные потоки могут складываться из выходных потоков нескольких подсистем:

$$XPS_i^{\text{вх}} = \left\{ \langle\, d_{il}^{\text{вх}}, \left\{ \langle\, PS_j^{\text{ист}}, f(Xdl.\text{кол}) \,\rangle \right\} \,\rangle_{j \in J} \right\}_{l=1,L_i},$$

где $XPS_i^{\text{вх}}$ – характеристика взаимодействия подсистемы PS_i со своими поставщиками; $d_{il}^{\text{вх}}$ – входные потоки для PS_i, характеристики которых описаны в структурах потоков поставщиков $PS_j^{\text{ист}}$.

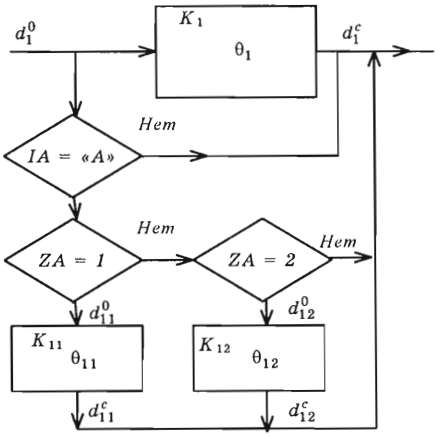
Например, из таблицы $XPS_1^{\text{вх}}$ видно, что на входе подсистемы «Население» поток «Люди» складывается из двух источников: «Внешней среды» и самой подсистемы «Население». Переход потока «Люди» с выхода подсистемы на собственный вход означает, что произошло событие, которое в результате изменило какие-то атрибуты потока (люди болеют, повышают квалификацию, перемещаются на новые рабочие места и в новом качестве снова входят в систему, изменяя свои правила функционирования).

Естественно, пример не соответствует конкретной системе. Информационные потоки на входе должны быть получены от множества различных экспертов из разных подсистем. Эксперты отделов кадров зададут общие требования ко всем сотрудникам данной фирмы. Эксперты-технологи зададут требование к трудовому ресурсу, способному обеспечить *XO.U* – алгоритм изменения свойств целевого объекта, потому что при внедрении новых технологий требуются специалисты принципиально других знаний и квалификации (что знает технолог, но плохо понимает ЛПР, судя по тому, что говорят о развитии высоких технологий при все ухудшающемся институте подготовки высококвалифицированных кадров по этим технологиям, а дело это небыстрое!). Эксперты-социологи, психологи, демографы и т. д. должны сформулировать личностные и социальные требования. Если компонента (трудовой ресурс) срабатывает не на цель системы, а на собственную цель («порадеть родному человечку», «откат», работа на более щедрого конкурента, назначение тепловика на место руководителя атомной станции и т. п.), то в систему закладываются потенциальные «Чернобыли», пожары, отключения энергосетей, катастрофы и т. д. Проиграть стратегию выбора кадров на имитационных моделях менее опасная стратегия формирования ИС для систем типа ЧКТП, особенно при создании ОПС, потому что можно увидеть последствия, к которым может привести наличие или отсутствие у претендента того или иного качества.

Итак, еще две структуры позволяют проверить взаимодействие подсистем по потокам между собой и системы с внешней средой по горизонтальным связям.

Общий принцип декомпозиции подсистем и компонент

Остается решить вопрос о едином способе декомпозиции целей, структур и функций в любой



■ ***Рис. 4.*** *Принцип декомпозиции компоненты*

подсистеме. Вспомним, что на верхнем уровне иерархии (на уровне подсистем) имеются только общие атрибуты целевых потоков, и, следовательно, для решения любой конкретной задачи нет информации о необходимых атрибутах. Адекватность модели определяется целью моделирования, ограничениями на способ моделирования и получения результата и критериями оценки результатов, в описании которых и задаются те атрибуты потоков, исполнительных и вспомогательных средств, которые необходимы для решения поставленной задачи. Как только определяются необходимые атрибуты, можно заполнить *FF*-фреймы всех потоков, исполнительных и вспомогательных средств и определить способы их обработки, задав характеристики операций *XOj*. Таким образом, получается новая компонента, которая в качестве своего целевого потока *d* имеет атрибут, подлежащий обработке, в качестве его техпроцесса θ – заданный алгоритм изменения его свойств, а в качестве исполнительных средств – средства, необходимые для изменения или получения новых значений *FF*-фрейма в *Xd.ZA*.

Декомпозиция компоненты показана на рис. 4. Поток $d_1{}^0$ обрабатывается компонентой K_1 с техпроцессом θ_1, при этом обнаруживается, что у части элементов потока $d_1{}^0$, обладающих атрибутом *IA* = «*A*», требуется дополнительная обработка в зависимости от значения атрибута *ZA*. Тогда надо проверить наличие имени атрибута «*A*» в *FF*-фрейме элементов потока $Xd_l{}^0$ и, если оно имеется, то проверять возможные значения этого атрибута. При этом поток будет разделяться на число подпотоков, соответствующее числу возможных различных значений атрибутов плюс 1, если возможны элементы потоков, не обладающих атрибутом с именем «*A*». Например, если *ZA* может принять только значения 1 или 2, то придется добавить новые две компоненты K_{11} и K_{12} с различными техпроцессами θ_{11}, θ_{12} обработки возникших подпотоков $d_{11}{}^0$ и $d_{12}{}^0$. В результате выполнения тех-

процессов будут получены два различных целевых потока $d_{11}{}^c$ и $d_{12}{}^c$, которые вместе с оставшейся частью потока, не обладавшей атрибутом с именем «A», составят целевой поток $d_l{}^c$ на выходе компоненты K. Такое разделение потоков возможно на любом уровне иерархии. Оно всегда связано с определенной целью, которая выделяет часть потока верхнего уровня, требует введения дополнительного атрибута и техпроцесса и порождает компоненты следующего уровня иерархии. Этот подход позволяет получить формальный метод декомпозиции и встраивания новых компонент в иерархические структуры любых подсистем в ОПС, ИУПС и ИПС сложных систем типа ЧКТП.

Заключение

Таким образом, основы общей теории систем типа ГДТС для построения имитационных моделей базируются на следующих принципах:

модель системы рассматривается всегда как целостная, состоящая из исследуемой системы и внешней среды (см. рис. 2, ч. I);

функциональная структура системы представляется тремя подсистемами: организационной, информационно-управляющей и исполнительной (см. рис. 3, ч. I);

каждая компонента системы на любом уровне иерархии задается инвариантом из 5 атрибутов (C, D, Θ, G,W) (см. рис. 1);

для описания любой компоненты системы определена математическая статическая и динамическая модель, состоящая из инвариантной и настраиваемой на конкретный тип системы части;

введен принцип декомпозиции подсистем по цели и атрибутам потоков, позволяющий формальным способом строить иерархические структуры подсистем и включать новые компоненты (см. рис. 4);

на основе декомпозиции цели существования системы в цели поведения (жизнедеятельности) выделен инвариант структуры исполнительной подсистемы систем типа ЧКТП.

Разработанные математические основы общей теории систем типа ГДТС могут служить хорошей основой программного инструментального средства для создания имитационных моделей и процесса имитационного моделирования сложных систем.

Литература

1. **Месарович М., Такахара Я.** Общая теория систем: математические основы. М.: Мир, 1978.
2. **Кодд Е. Ф.** Реляционная модель данных для больших совместно используемых банков данных. СУБД N1, 1995.
3. **Перовская Е. И.** Основные принципы построения моделей сложных гибких дискретных технологических систем // Системный анализ в проектировании и управлении: Тр. VIII Междунар. науч.-практ. конф. СПб.: Изд-во СПбГТУ, 2004. С. 126–130.
4. **Перовская Е. И.** Создание глобальных моделей социумов для проверки управленческих решений и их последствий // Системный анализ в проектировании и управлении: Тр. VII Междунар. науч.-практ. конф. СПб.: Изд-во СПбГТУ, 2003. С.186–190.
5. **Perovskaya E. I.** Support of acceptance of the administrative decisions in systems Humanity-Culture-Technology-Nature / Instrumentation in ECOLOGY and HUMAN SAFETY 2002. St.Petersburg, 2002. C.194–196.
6. **Перовская Е. И., Фетисов В. А.** Автоматизация гибких дискретных систем. Л.: Изд-во ЛГУ, 1986. 183 с.
7. **Перовская Е. И.** Имитационные модели для поддержки принятия решений в системах Человек–Культура–Технологии–Природа // Системный анализ в проектировании и управлении: Тр. V Междунар. науч.-практич. конф. СПб.: Изд-во СПбГТУ, 2001. С. 177–181.