ГЛАВА 4.
СИТУАЦИОННО-ИМИТАЦИОННО-ЭКСПЕРТНАЯ (SIE) МОДЕЛЬ

4.1. Принципы построения SIE-модели

Проектирование интегрированной ситуационно-имитационно-экспертной (SIE) модели требует создания единого множества элементов (понятий), используемых в имитационном, ситуационном и экспертном подходах.

$$L^{M_{SIE}} = L^{M_S} \cup L^{M_I} \cup L^{M_E},$$

где L^{M_S} *– множество элементов СМ*;

L^{M_I} *– множество элементов ИМ*;

L^{M_E} *– множество элементов ЭМ*;

$L^{M_{SIE}}$ *– множество элементов SIE – модели*.

Все рассматриваемые модели имеют ряд элементов и понятий, которые присущи только им и не используются в других. Объединение таких элементов достаточно просто.

$$L^{M_{SIE}} = (L_*^{M_S} \cup L_*^{M_I} \cup L_*^{M_E}) \cup (L_{\neg *}^{M_S} \cup L_{\neg *}^{M_I} \cup L_{\neg *}^{M_E})$$

$$(L_*^{M_S} \cap L_*^{M_I}) = (L_*^{M_I} \cap L_*^{M_E}) = (L_*^{M_S} \cap L_*^{M_E}) = \varnothing;$$

$$(L_{\neg *}^{M_S} \cap L_{\neg *}^{M_I}) \neq \varnothing; (L_{\neg *}^{M_I} \cap L_{\neg *}^{M_E}) \neq \varnothing; (L_{\neg *}^{M_S} \cap L_{\neg *}^{M_E}) \neq \varnothing,$$

где $L_*^{M_S}[L_{\neg *}^{M_S}]$ *– множество* [*не*]*уникальных элементов СМ*;

$L_*^{M_I}[L_{\neg *}^{M_I}]$ *– множество* [*не*]*уникальных элементов ИМ*;

$L_*^{M_E}[L_{\neg *}^{M_E}]$ *– множество* [*не*]*уникальных элементов ЭМ*;

Однако существуют элементы, которые используются в двух или трех моделях. Для них необходимо либо разработать новые обобщенные элементы (метаэлементы), либо использовать различное представление.

В первом случае элементы различных моделей будут частными случаями этого “метаэлемента”. Во втором случае необходимо разработать специальные процедуры преобразования между ними.

Лучше всего SIE-модель представить в виде нескольких различных уровней, соответствующих имитационному, экспертному и ситуационному представлению информации. Общие элементы должны либо дублироваться (проецироваться) на каждом уровне в различном представ-

лении, либо содержаться в одном уровне. В последнем случае уровни должны иметь возможность обратиться к соответствующему элементу.

Основное отличие ЭС от СИМ заключается в моделировании не столько физической (или иной) природы объекта, сколько механизма мышления человека применительно к этому объекту [Джексон, 2001]. Действительно, для моделирования объекта достаточно знать правила его функционирования, т.е. зависимость выходных характеристик от входных [Советов, 2001].

$$\vec{y}(t) = F_s(\vec{x}, \vec{v}, \vec{h}, t),$$

где $\vec{x}$ *– входные воздействия;*

$\vec{v}$ *– воздействия внешней среды;*

$\vec{h}$ *– собственные параметры объекта;*

$\vec{y}$ *– выходные характеристики;*

F_s *– закон функционирования*.

Первый уровень модели предназначен для описания структуры системы. Для этого каждому объекту (субъекту) сопоставляется блок. Все блоки соединяются между собой каналами взаимодействия. По этим каналам могут перемещаться динамические объекты (транзакты). Каждый блок определенное время обрабатывает транзакт и задерживает его на фиксированное время, которое рассчитывается исходя из интенсивности работы устройства. Блоки не изменяют характеристики транзактов.

Такую модель можно использовать не только для описания структуры системы и возможных треков транзактов, но и для имитации с помощью ЭВМ. Большинство блоков имеют свои аналоги в известных СИМ (GPSS, SLAM II, SIMAN) и могут быть легко в них преобразованы.

На структурном уровне нет возможности задания произвольных событий, сцепленных процессов, условных операторов и программ по изменению структуры модели. Для выполнения условных операторов используется специальный блок *селектор*. Он осуществляет обращение к базе знаний, в которой хранятся соответствующие правила.

Для описания событий, которые могут возникать в результате обработки транзактов, изменения состояний объектов и поступления внешней информации используется второй уровень SIE-модели, называемый событийным.

Если осуществить проекцию структурного уровня на событийный, то для каждого блока будут существовать, по крайней мере, два события (*начало обработки транзакта* и *окончание обработки транзакта*),

два состояния (*занят* и *не занят*) и один процесс (*обработка транзакта*).

Пространство состояний блока может быть полностью определено на структурном уровне. На событийном уровне задаются только некоторые (выделенные) состояния или макросостояния, которые называются микроситуациями. На этом уровне также задаются дополнительные события и процессы. Каждый элемент этого уровня привязывается к конкретным параметрам блоков структурного уровня. Кроме того, могут добавляться новые атрибуты. Вся информация об объектах хранится в базе знаний, поэтому в компьютерной среде уровни могут быть реализованы независимо.

Событийный уровень может быть преобразован в сеть Петри, Е-сеть, алгебру процессов или другую аналогичную модель. Возможности по ветвлению и сцеплению процессов определяются их представлением в виде последовательности состояний и событий. Для реализации условных операторов используется блок селектора, который обращается к базе знаний правил. События могут инициировать изменения на структурном уровне.

При программной реализации модели возможно последовательное или параллельное использование уровней для моделирования. Например, при возникновении типовых ситуаций, событийный уровень может прервать имитацию на структурном уровне, выполнить ее самостоятельно, и продолжить работу, изменив при этом параметры (и структуру) модели.

Третьим уровнем SIE-модели является ситуационный уровень. Он предназначен для укрупненного моделирования системы. Каждому объекту на структурном уровне сопоставляется микроситуация. Некоторые объекты объединяются в один объект более высокого уровня. Вводятся абстрактные объекты или понятия, для которых определяются возможные ситуации.

Ситуации и некоторые микроситуации определяются на основании правил, заложенных в ЭС, и они связаны друг с другом, т.е. зависят от общих (пересекающихся) атрибутов. Ситуационное моделирование заключается в задании некоторых характеристик, отдельных ситуаций и определение с помощью ЭС оказываемых влияний.

Ситуационный уровень может также служить укрупненным отображением процессов имитации на нижних уровнях. Переход от этого уровня к событийному или структурному неоднозначен, поэтому он может использоваться только для ограничения количества имитаций и выбора отдельных имитационных компонент в системах с большой размерностью.

Особняком стоит экспертный уровень, который представляет собой базу знаний, хранящую всю информацию об остальных уровнях и дополнительные знания ЭС. Более подробно элементы SIE-модели рассматриваются в следующих параграфах, где также приводится их формальное описание.

4.2. Описание SIE-модели

Основные понятия SIE-модели. Для описания предметной области воспользуемся понятием *Предмета*. Предмет есть реальный физический объект, процесс или явление, которое обозначается неким знаком. Все множество предметов разделим на две части:

— множество атрибутов A;

— множество значений Z, которые могут принимать эти атрибуты.

Если атрибут может принимать значение, то это обозначается следующим образом:

$Z(a_i)=z_i$

Атрибут A это свойство, характеризующее объект как самостоятельную и неделимую сущность, т.е. как целое. Теоретически любой объект можно разделить на части, при этом он становится системой. Для дальнейших рассуждений примем, что на определенном этапе деление объекта уже не осуществляется, и он рассматривается как целое. Уровень дробления может быть разным для отдельных объектов или в различных ситуациях.

Под переменной *Var* будем понимать упорядоченную пару, состоящую из атрибута и множества значений, которые он может принимать.

$Var = <A, \{z_i\}>$

Под константой будем понимать упорядоченную пару, состоящую из атрибута и его значения.

$Const = <A, z_i>$

Определим понятие объекта O как совокупность (множество) атрибутов.

$O = \{a_i\}$

Среди всего множества выделим особый объект, называемый связью R. Условием существования связи между объектами является их непустое пересечение. Связь между атрибутами может быть выражена только через объекты, к которым они принадлежат.

$R = <A^r, A^{self}>, A^r = o_1 \cap o_2 \cap ... \cap o_n \neq \varnothing$

где, A^{self} — собственные параметры связи (объекты, с которыми она связана, весовая функция и др.)

Если один объект принадлежит другому, то он является вложенным. Вложенность атрибутов, объектов и других элементов модели обозначается точкой.

$\forall(o_1 \supset o_2) \Rightarrow o_2.o_1$

Множество объектов, имеющих одинаковые атрибуты, называется классом *Cl*. Если объекты, принадлежащие одному классу, в каждый момент времени имеют одинаковые значения атрибутов, то они эквивалентны.

$Cl = \{o\}_n,\ \forall(o_i,o_j, i,j=[1,n]\ )\ o_i=o_j$

$\forall(o_1.A_i,\ o_2.A_i,)\ \forall(t)\ Z(o_1.A_i)= Z(o_2.A_i) \Rightarrow o_i \equiv o_j$

Под системой *Sys* понимается совокупность объектов, связей между ними и собственных атрибутов. В системе все объекты должны быть связаны, и эти связи должны быть постоянными. Если связь удаляется, то нужно говорить о новой системе. Однако в системе может изменяться значение параметров связи.

$Sys = <O, R, A^{self}>$

При описании системы создается ее модель *M*, в которой пренебрегают некоторыми связями, параметрами и объектами, поэтому для множества различных систем может существовать одна модель. Понятие модели будет отличаться от понятия системы набором условий *Usl,* которые определяют совокупность объектов и связей как систему.

$M=<O, R, A^{self}, Usl>$

Введем понятие математического формализма *H*, задающего характер взаимоотношений между объектами (правил преобразований). Возможны следующие варианты: продукционные правила; теоремы; математические механизмы, используемые при рассуждениях с неточными знаниями; формулы; сценарии и т.д.

Математическая формула или формализм есть слово, состоящее из последовательности объектов и отношений между ними:

$H \equiv O_1 R_= O_2 R_1 O_3 \ldots R_n O_{n+1}$

где $R_=$ — отношение эквивалентности.

Функция *Func* является видом математического формализма. В основном функции будут обозначаться греческими буквами.

Событийный и ситуационный уровни. Рассмотрение начнем с событийного уровня, так как в нем определяются такие понятия как состояние объекта и модели, процесс, событие.

Состояние объекта. Состояние объекта определяется совокупностью всех значений его атрибутов:

$$C^{obj} = < \mathrm{A}, \mathrm{Z}^c >,$$

$$|A| = |Z| = n;\ n \geq 1; \bigcup_{i=1}^{n} \{z^i\} = Z; \bigcup_{i=1}^{\Theta} Z^c = Z;$$

$$(\forall a_i \in A)\ \exists \{z^i\};\ i = \overrightarrow{1,n};\ \{z^i\} = m_i; \Theta = \prod_{i=1}^{n} m_i$$

где A – *множество атрибутов объекта*;

Z^c – *множество значений атрибутов в зафиксированный момент времени*;

n – *количество атрибутов объекта*;

Θ – *количество возможных состояний*;

Z – *множество допустимых значений*;

$\{z^i\}$ – *множество допустимых значений атрибута*;

m_i – *количество допустимых значений атрибута.*

Состояние модели определяется как совокупность состояний объектов, из которых состоит модель, и наличие связей между ними. В модели согласно условиям *Usl* могут меняться не только весовые функции связей, но и появляться новые отношения между объектами, поэтому в состояние модели входит ее структура на данный момент времени.

$C^M = <m, Z^M>$

*Событие***.** Под простейшим событием будем понимать момент времени, в который произошло изменение значения какого-либо атрибута модели.

$$e^s = < \tau, a, z_1^a, z_2^a >$$

$$z_1^a \neq z_2^a;$$

где τ – *момент времени*;

a – *атрибут*;

z_1^a – *значение атрибута до события*;

z_2^a – *значение атрибута после события.*

В один момент времени может произойти несколько различных событий.

$$\exists\,(e_1^s, e_2^s \in E; e_1^s \neq e_2^s)\ e_1^s.\tau = e_2^s.\tau$$

Некоторые события могут быть связаны с изменениями нескольких атрибутов. Если изменения произошли не одновременно, то моментом наступления события считается время последнего изменения. Событие представляет собой множество элементарных событий, каждое из которых изменяет только один параметр. Последовательность этих элементарных событий есть процесс, приводящий к возникновению события, связанного с несколькими параметрами. Фактически речь идет о некотором макрособытии.

$$e = < \tau, \{a\}, \{< z_1^a, z_2^a >\} >$$

$$\left|\{a\}\right| = \left|\{< z_1^a, z_2^a >\}\right|; z_1^a \neq z_2^a,$$

где τ – *момент времени*;

$\{a\}$ – *множество атрибутов*;

z_1^a – *значение атрибута до события*;

z_2^a – *значение атрибута после события.*

Процесс. Под процессом будем понимать последовательную смену состояний объекта или модели, происходящую за конечный промежуток времени. Если существуют атрибуты, не связанные с объектами, то процессом следует считать последовательную смену значений. Далее будем предполагать, что все атрибуты связаны либо с объектом или моделью. Если это невозможно, то атрибут должен рассматриваться как объект.

Процесс можно охарактеризовать двумя событиями (начало и конец процесса) или последовательностью состояний, называемой треком *TR*.

$$p = < e_1, e_2, A^{self} > = < C, \beta, A^{self} > = < TR, A^{self} >$$

$$e_1.\tau < e_2.\tau;$$

где e_1 – *событие начала процесса*;

e_2 – *событие конца процесса*;

A^{self} – *собственные параметры процесса*;

C – *множество состояний объекта или модели*;

β – *функция, задающая порядок смены состояний*;

TR – *трек, упорядоченное во времени множество состояний.*

Действие. Любое действие может быть выполнено с помощью процесса за конечный промежуток времени. В отличие от процесса действие зависит только от начального и конечного состояний объек-

та или модели, т.е. действие приводит к изменению состоянию в течение произвольного, но конечного времени. Для действия не важен трек процесса.

$d = < \tau_1, \tau_2, \{a\}, Z_1^a, Z_2^a >$

$\tau_1 < \tau_2$

где τ_1 *– время начала действия;*

τ_2 *– время окончания действия;*

$\{a\}$ *– множество атрибутов, которое изменяет действие*;

Z_1^a *– значения атрибутов до начала действия*;

Z_2^a *– значения атрибутов полсе окончания действия.*

Микроситуация. Под микроситуацией будем понимать выделенные состояния объекта, которые описываются не значениями каждого атрибута, а символичной записью в виде одного слова или небольшого фрагмента естественно-языкового описания.

Микроситуация может соответствовать нескольким состояниям объекта, т.е. являться особым макросостоянием. Особенность микроситуации заключается в том, что она может зависеть не от всего вектора параметров объекта, а от его части (проекции). В общем случае микроситуация определяется с помощью рассуждений (правил) ЭС.

$mis = < desc, \{a\}, \{Z^a\}, h(\{a\}, \{Z^a\}) >,$

где $desc$ *– описание микроситуации*;

$\{a\}$ *– множество атрибутов объекта, определяющие микроситуацию*;

Z^a *– диапазон значений атрибутов, допустимых для заданного типа микроситуации*;

$h(\{a\}, \{Z^a\})$ *– правила определения микроситуации.*

Далее рассматриваются два элемента ситуационного уровня.

Ситуация. Под ситуацией будем понимать совокупность микроситуаций и (или) значений отдельных свойств объектов и некоторых произошедших событий. Включение событий обусловлено тем, что ситуация может складываться (накапливаться, нагнетаться) в течение времени, она также может сохраняться в течение какого-то времени. Эта временная зависимость выражается в произошедших событиях.

Правила, определяющие ситуацию, могут зависеть от не произошедших событий. Например, если ни в одном объекте не возникло экстремальных микроситуаций, то в целом ситуация стабильна.

$sit = < desc, \{mis\}, \{a\}, \{e\}, h(\{mis\}, \{a\}, \{e\}) >,$

где $desc$ – *описание ситуации*;

$\{mis\}$ – *множество микроситуаций*;

$\{a\}$ – *множество атрибутов объектов*;

$\{e\}$ – *множество событий*;

$h(\{mis\},\{a\},\{e\})$ – *правила определения ситуации.*

***Макроситуация*.** Под макроситуацией будем понимать совокупность всех ситуаций, микроситуаций и произошедших событий в модели. Макроситуация необходима для оценки системы (модели) в целом.

$$mas = < desc, \{mis\}, \{sit\}, \{e\},\ h(\{mis\},\{sit\},\{e\}) >,$$

где $desc$ – *описание макроситуации*;

$\{mis\}$ – *множество микроситуаций*;

$\{sit\}$ – *множество ситуаций*;

$\{e\}$ – *множество событий*;

$h(\{mis\},\{sit\},\{e\})$ – *правила определения макроситуации.*

В системах с большой размерностью количество ситуаций и микроситуаций может быть велико. Иногда для упрощения вводят понятия уровня абстракции, т.е. выделяются области системы, для них определяются отдельно макроситуации и на следующем уровне они рассматриваются как микроситуации или ситуации.

В SIE-модели уже введено понятие вложенности для объектов и моделей, поэтому необходимость в переопределении макроситуаций отпадает. Можно ввести сколь угодно сложные объекты. Для уменьшения размерности можно рассматривать объекты более высокого уровня (содержащие большее количество вложенных объектов).

Для рассмотрения части модели вводится понятие среза *Dec*. При декомпозиции системы на срезы возникают проблемы потери интегральной целостности. Рассмотрение этих вопросов выходит за рамки данной работы, и они подробно рассмотрены в работе [Исаев, 1994].

Структурный уровень. На этом уровне SIE-модели составляется структурная схема системы, которая представляет собой совокупность связанных блоков. Под блоком понимается сложный объект (модель), в котором связи между внутренними и внешними атрибутами задаются с помощью формализмов. Каждый блок имеет условное графическое обозначение. Структурный уровень содержит следующие типовые блоки:

Устройство; Канал; Поток; Персонал (исполнитель работ); Распаковщик; Сборщик; Транзакт; Очередь; Селектор.

Устройство. Под устройством понимается специализированное до-

$$Ustr = < \{a^{in}\},\{a^{out}\},\{a^{self}\},\{t\},\ \phi(\tau,\{a^{in}\},\{a^{self}\}),\psi(\tau,\{a^{in}\},\{a_0^{self}\}) >$$

$$\{a^{in}\},\{a^{out}\},\{a^{self}\} \subseteq A; \{a^{in}\} \cap \{a^{out}\} = \varnothing; \{a^{in}\} \cap \{a^{self}\} = \varnothing; \{a^{out}\} \cap \{a^{self}\} = \varnothing;$$

$$\left|\{type\}\right| = n;\ n \geq 1; n \in V^I;\ (\forall type_i, type_j \in \bigcup_1^n type_i = T;\ i,j = \overrightarrow{1,n})\ type_i \neq type_j$$

печатное оборудование, персональная ЭВМ, оргтехника или сложный

$$\{a^{out}\} = \phi(\tau, \{a^{in}\}, \{a^{self}\}) = func(\tau, \{a^{in}\}, \{a^{self}\});$$

$$\{a^{self}\} = \psi(\tau, \{a^{in}\}, \{a_0^{self}\}) = func(\tau, \{a^{in}\}, \{a_0^{self}\});$$

объект, который можно рассматривать как “абстрактное” устройство.

где n – *количество различных типов транзактов*;

$\{type\}$ – *множество типов транзактов, которые может обрабатывать устройство*;

$\{a^{in}\}$ *и* – *входные характеристики устройства*;

$\{a^{out}\}$ – *выходные характеристики устройства*;

$\{a^{self}\}$ – *собственные характеристики устройства*;

$\phi(\tau, \{a^{in}\}, \{a^{self}\})$ – *функция, определяющая работу устройства*;

$\psi(\tau, \{a^{in}\}, \{a_0^{self}\})$ – *функция, определяющая состояние устройства*;

τ – *модельное время системы*;

$\{a_0^{self}\}$ – *множество собственных характеристик до начала моделирования*.

Поток. Под потоком понимается распределенное во времени и в пространстве множество транзактов. Обычно потоки распространяются внутри каналов. Если часть транзактов расположены вне системы или они формируются с помощью генераторов, то поток можно определить как последовательность событий появления транзактов, происходящих одно за другим в случайные моменты времени.

$$fl = < \{t\}, \phi(\tau, \{\alpha\}),\ \theta >$$

$$\left|\{t\}\right| = n;\ n \geq 1; n \in V^I;\ (\forall t_i, t_j \in \bigcup_1^n t_i = T;\ i, j = \overrightarrow{1,n})\ t_i.type = t_j.type$$

$$fl.\lambda = \phi(\tau, \{\alpha\}) = func(\tau, \{\alpha\}); (\tau = \overrightarrow{0, \theta}) | (\tau = \overrightarrow{t_0, t_0 + \theta}); t_0, \tau, \theta \geq 0$$

$$\left|\{\alpha\}\right| = k;\ k \geq 0; k \in V^I;\ (\forall \alpha_r; r = \overrightarrow{1,k})\ \alpha_r \in \bigcup_1^k a_r = A;$$

где $\{t\}$ *– множество однотипных транзактов в потоке*;

n *– количество транзактов в потоке*;

$fl.\lambda$ *– интенсивность потока*;

$\phi(\tau, \{\alpha\})$ *– распределение интенсивности в потоке*;

θ *– время существования потока*;

τ *– модельное время системы*;

$\{\alpha\}$ *– множество признаков для неоднородных потоков*;

$(\forall k = 0)\ \{\alpha\} = \varnothing \Rightarrow fl.\lambda = func(\tau)$.

Потоки могут быть однотипными и смешанными, однородными и неоднородными. Ниже приведено описание потоков как упорядоченной тройки, состоящей из транзактов (или простейших потоков), функции распределения интенсивности потока и времени существования потока (источника транзактов).

$$fl^{\Sigma} = < \{fl\}, \psi(\tau, \{\beta\}),\ \theta^{\Sigma} >$$

$$\left|\{fl\}\right| = k;\ k > 1; k \in V^I;\ (\forall fl_i, fl_j \in \bigcup_1^n fl_i;\ i, j = \overrightarrow{1,k})\ fl_i.T.type \neq fl_j.T.type$$

$$\psi(\tau, \{\beta\}) = fl^{\Sigma}.\lambda = func(\tau, \{\beta\}) = \sum_{i=1}^{k} \sigma_i^T * \phi_i(\tau, \{\alpha\}); \{\beta\} = \bigcup_{i=1}^{k} \{\alpha\}^i$$

$$(\tau = \overrightarrow{0, \theta^{\Sigma}}) | (\tau = \overrightarrow{t_0, t_0 + \theta^{\Sigma}}); t_0, \tau, \theta^{\Sigma} \geq 0,$$

где $\{fl\}$ *– множество* (k) *различных потоков*;

$fl^{\Sigma}.\lambda$ *– интенсивность смешанного потока*;

σ_i^T *– коэффициент, определяющий "размер" транзакта.*

Канал. Под каналом понимается линия связи, которая служит для передачи данных между устройствами по компьютерной сети, или абстрактная линия передачи информации и других объектов между субъектами и объектами.

$$kan = < \{type\}, \varphi(\tau, \{\alpha\}), \theta, O^{in}, O^{out} >$$

$$\left|\{type\}\right| = n;\ n \geq 1; n \in V^1;\ (\forall type_i, type_j \in \bigcup_1^n type_i = T;\ i, j = \overrightarrow{1,n})\ type_i \neq type_j$$

$$O^{in} \neq \varnothing;\ O^{out} \neq \varnothing; \left|O^{out}\right| = \left|O^{in}\right| = 1;$$

$$kan.\lambda = \varphi(\tau, \{\alpha\}) = func(\tau, \{\alpha\}); \theta \geq 0;$$

$$\left|\{\alpha\}\right| = k;\ k \geq 0; k \in V^1;\ (\forall \alpha_r; r = \overrightarrow{1,k})\ \alpha_r \in \bigcup_1^k a_r = A;$$

где $\{type\}$ – *множество допустимых типов транзактов*;

O^{in}, O^{out} – *блоки, которые соединяет канал;*

n – *количество возможных типов транзактов в канале*;

$kan.\lambda$ – *пропускная способность канала*;

$\varphi(\tau, \{\alpha\})$ – *распределение* $kan.\lambda$ *во времени*;

θ – *время задержки одного транзакта в канале*;

τ – *модельное время системы*;

$\{\alpha\}$ – *множество признаков, определяющих* $kan.\lambda$;

$(\forall k = 0)\ \{\alpha\} = \varnothing \Rightarrow kan.\lambda = func(\tau)$;

Персонал. Под персоналом понимаются сотрудники фирмы или отдела, которые выполняют основные виды работы. Под неосновными работами понимаются такие виды деятельности как ремонт оборудования, уборка помещений и др. В разработанной модели они не учитываются.

Каждый сотрудник владеет определенной профессией и занимает соответствующую должность. На практике часто один сотрудник может выполнять несколько различных видов работ, поэтому каждому из них присваивается вектор специальностей, координатами которого являются коэффициенты качества и сложности конкретного вида работы.

Аналогично хранится информация о возможностях работы сотрудника с каждым типом устройства и программным пакетом.

$$pers = < \{type^T\}, \{type^{Us}\}, \{\varphi(\tau, \{\alpha\})\}, \{\theta\}, A^{self}, k^T, k^{Us} >$$

$$\left|\{type^T\}\right| = n;\ n \ge 1; n \in V^I;\ (\forall type^T{}_i, type^T{}_j \in \bigcup_1^n type^T{}_i = T;\ i, j = \overrightarrow{1,n})\ type^T{}_i \ne type^T{}_j$$

$$\left|\{type^{Us}\}\right| = m;\ m \ge 1; m \in V^I;$$

$$(\forall type^{Us}{}_i, type^{Us}{}_j \in \bigcup_1^m type^{Us}{}_i = T;\ i, j = \overrightarrow{1,m})\ type^{Us}{}_i \ne type^{Us}{}_j$$

$$pers.\mu_T^{Us} = \varphi(\tau, \{\alpha\}) = func(\tau, \{\alpha\}); \theta \ge 0;$$

$$\left|\{\alpha\}\right| = k;\ k \ge 0; k \in V^I;\ (\forall \alpha_r; r = \overrightarrow{1,k})\ \alpha_r \in \bigcup_1^k a_r = A;$$

где $\{type\}$ – *множество допустимых типов транзактов*;

n – *количество возможных типов транзактов*;

$pers.\mu_T^{Us}$ – *интенсивность обработки транзакта типа* T *на устройстве типа* $Ustr$;

$\varphi(\tau, \{\alpha\})$ – *распределение* $pers.\mu_T^{Us}$ *во времени*;

$\{\theta\}$ – *множество интервалов времени, когда субъект может работать*;

k^T – *коэффициент качества работы с заданным типом транзакта*;

k^{Us} – *коэффициент качества работы с заданным типом устройства*;

$\{\alpha\}$ – *множество признаков, определяющих* $pers.\mu_T^{Us}$.

Транзакт. Под транзактом понимается динамический объект, который может перемещаться по каналам, изменяться и обрабатываться устройствами, персоналом и другими объектами.

Для представления различных типов или состояний транзактов выделяют несколько переменных, значения которых могут изменять устройства. В системе используется два основных свойства транзакта: тип транзакта и его размер.

Транзакты могут быть составными, т.е. могут объединяться в комбинированные типы и раскладываться. Для расчетов длин очередей и времени обработки можно разбить любой транзакт на минимальные единицы, которые в системе называются простыми транзактами.

Разбиение транзактов на типы осуществляют специальные блоки — распаковщики. Для разбиения на простые транзакты необходимо воспользоваться временными распаковщиками.

$trans = < type, \{t,q\}, A^{self}, k >$

(1) $\left|\{t,q\}\right| = n;\ n \geq 0; n \in V^1;\ (\forall t_i \in \{t,q\})\ i = \overrightarrow{1,n})\ t_i \neq type \,\&\, t_i.\{t,q\} = \varnothing,$

где $\{type\}$ – *тип транзакта*;

n – *количество возможных типов транзактов*;

$\{t,q\}$ – *тип и число элементарных транзактов в сложном* $(n > 0)$.

$(\forall n = 0)\ \{t,q\} = \varnothing \Rightarrow trans = < type, A^{self}, k >$ – *простой транзакт*;

(1) – *условие отсутствия рекурсивности*;

A^{self} – *собственные параметры транзакта*;

k – *приоритет транзакта*.

Распаковщики и сборщики. Под распаковщиком понимается устройство или человек, который осуществляет разбиение транзактов на составные части. Каждая составная часть в свою очередь является транзактом.

На практике разбиение транзактов осуществляется человеком (обычно руководителем). Распространенным примером этого в полиграфии является разделение заказа на составные части. Необходимость в распаковки транзактов может возникнуть также при ошибке распределения работ, неправильной оценке размера транзакта (времени обработки), возникновении новых работ.

$rasp = < \{type^{in}\}, \{type^{out}\}, \phi(\tau, type^{in}, A^{self}), \psi(\tau, \{a\}), A^{self} >$

$\{type^{out}\} = \phi(\tau, type^{in}, A^{self}) = h(\tau, type^{in}, A^{self});$

$rasp.\mu = \psi(\tau, \{a\}) = func(\tau, \{a\});\ 0 < \left|\{type^{in}\}\right| \leq \left|\{type^{out}\}\right|$

где $\{type^{in}\}, \{type^{out}\}$ – *множество типов входных и выходных транзактов*;

$h(\tau, type^{in}, A^{self})$ – *формализм* (*отображение*), *сопоставляющий одному входному транзакту несколько выходных*;

$rasp.\mu$ – *интенсивность обработки транзактов в распаковщике*;

$\psi(\tau, \{a\})$ – *распределение интенсивности обработки транзактов*;

A^{self} – *собственные параметры распаковщика*.

Для того чтобы проводить расчеты времени выполнения работы, средней длины очереди, количества заказов в системе и т.д., необходимо составные транзакты разбивать на элементарные, которые имеют постоянный размер, т.е. постоянное время обработки для каждого устройства. Для разбиения на элементарные транзакты используется временной распаковщик.

Сборщики выполняют сборку (объединение) транзактов. Они могут работать при неполных исходных данных, т.е. при поступлении тран-

закtов только на часть входов. В этих случаях сборщики должны иметь несколько выходов. В модели используются только простые сборщики, работающие только при полном заполнении всех входных каналов.

$$sbor = < \{type^{in}\}, \{type^{out}\}, \phi(\tau, type^{in}, A^{self}), \psi(\tau, \{a\}), A^{self} >$$

$$\{type^{out}\} = \phi(\tau, type^{in}, A^{self}) = func(\tau, type^{in}, A^{self});$$

$$sbor.\mu = \psi(\tau, \{a\}) = func(\tau, \{a\}); \ \left|\{type^{in}\}\right| \geq \left|\{type^{out}\}\right| > 0,$$

где $\{type^{in}\}$ – *множество типов входных транзактов*;

$\{type^{out}\}$ – *множество типов выходных транзактов*;

$\phi(\tau, type^{in}, A^{self})$ – *функция, сопоставляющая нескольким входным транзактам один выходной*;

$rasp.\mu$ – *интенсивность обработки транзактов в сборщике*;

$\psi(\tau, \{a\})$ – *распределение интенсивности обработки транзактов*;

A^{self} – *собственные параметры сборщика.*

Очередь. Под очередью понимается объект, в котором могут накапливаться транзакты. В системе используются различные типы очередей. В одной очереди могут находиться транзакты только одного типа.

Часто возникают случаи, когда на одном устройстве можно выполнить несколько различных задач, тогда для них создается несколько очередей. Разрешение конфликтов между очередями осуществляется с помощью приоритетов. Очередность выполнения транзакта определяется его срочностью и приоритетом очереди.

Для расчетов очереди можно объединить. Тогда в них должны находится только простые транзакты. Вставка в очередь осуществляется в соответствии с приоритетом транзакта.

На практике возникает сложность, связанная с тем, что транзакт должен вставать сразу в несколько очередей, т.е. ожидать первого освободившегося устройства. Для решения этой задачи в модели предлагается использовать технологию виртуальных транзактов, описанную в 5 главе.

$$quer = < type, \{t\}, k, m, A^{self} >$$

$$\left|\{t\}\right| = n;\ n \geq 1; n \in V^1;\ (\forall t_i, t_j \in \bigcup_1^n t_i = T;\ i, j = \overrightarrow{1,n})\ t_i \neq t_j$$

$$k \geq 0; m \geq 1;\ k, n \in V^1;$$

где $\{t\}$ *– множество допустимых типов транзактов*;

n *– количество возможных типов транзактов в очереди*;

k *– количество приоритетов*;

m *– размер очереди*;

type – тип очереди (*FIFO, FILO и др.*);

A^{self} *– внутренние параметры очереди*.

Экспертный уровень. Экспертный уровень представляет собой базу знаний, в которой хранятся описания:

1. типовых элементов (объектов, блоков) модели;
2. схем различных уровней;
3. программ-селекторов;
4. правил определения ситуаций;
5. дополнительных правил, объектов и связей ПО;
6. накопленного опыта экспертов и работы системы.

Типовые элементы. Для хранения информации о типовых элементах используется фреймовая технология. В настоящее время фреймы получили широкое распространение в виде объектно-ориентированного подхода. Для организации базы знаний фреймов будем использовать объектную СУБД и технологии объектно-ориентированного программирования (ООП). Вместо понятия объекта будем использовать понятие фрейма, так как объект был уже определен раньше.

Фрейм. Под фреймом будем понимать совокупность свойств, методов, событий, классов и идентификатора:

$Fr = < A, Me, E, Cl^{fr}, Id >$,

где A *— множество атрибутов фрейма (объекта);*
Me *— множество методов фрейма (слоты процедуры)*
E *— множество событий фрейма;*
Cl^{fr} *— множество классов, к которым принадлежит фрейм;*
Id *— идентификатор фрейма (его имя).*

Метод. Под методом понимается способ изменения свойств фрейма или операция, которая может выполняться над другими фреймами или связями. Метод — это множество программно реализованных проце-

дур, обеспечивающих выполнение правил преобразования. Каждая процедура уникальна для различных классов фреймов и типов отношений между ними. Программная процедура есть упорядоченная последовательность математических формализмов:

Me = < H1, H2, ... Hn >

Класс фрейма. В связи расширением понятия объекта необходимо изменить понятия класса. Группы фреймов можно объединить в классы. Класс фрейма — совокупность фреймов, имеющих общие свойства, методы или события. Классы могут быть простые, абстрактные, изоморфные, полиморфные и др.

Идентификатор. Для различения фреймов между собой можно использовать понятие идентификатора, который будет однозначно соответствовать одному фрейму и характеризовать его имя.

Схемы уровней. Каждая схема уровня представляет собой сеть, узлами которой являются типовые блоки. Для представления схем в базе знаний будем использовать семантические сети вида:

SemNet = <Block, Relation >

где Block — вершины сети;

Relation — дуги сети.

Для работы с сетями лучше использовать списковые структуры [Ахо, 2000], но для хранения лучше использовать объектные БД с развитым механизмом ассоциативных связей (например, СУБД Линтер + Невод) или реляционные таблицы.

Программы-селекторы. Программы-селекторы представляют собой набор правил или методов, выполнение которых инициируется обращением к блоку селектора на структурном и событийном уровнях.

В простейшем случае селектор реализует условный оператор, который в свою очередь представляет собой продукцию вида *Если-То,* для хранения которой удобно использовать реляционную таблицу. Реализация более сложных программ требует наличие развитой машины вывода ЭС.

Вывод на основе правил базы знаний требует наличия цели (решения), стратегии, фактов (аксиом), гипотез и других понятий. Надо заметить, что в данной версии SIE-модели вопросы нечеткости не рассматриваются, хотя в БЗ могут использоваться вероятностные и нечеткие правила.

Гипотеза. Это фрейм, который определяет вероятность наступления какого-либо события или достоверность свойства (признака). Будем

считать, что гипотеза может быть привязана только к одному фрейму. Каждая гипотеза обладает по крайней мере следующими характеристиками:

B ⊆ Fr;
B.A1 — вероятность наступления события;
B.A2 — имя объекта (идентификатор) привязки;
B.Me1 — метод изменения вероятности.

Факт, Свидетельство. Это истинное утверждение, формализм, который связывает два и более объекта или свойства:

$F \Leftrightarrow O_1 R_1 \ldots R_n O_{n+1} R_= True$

Факты могут быть заданы в виде исходных данных или получены в результате эквивалентных преобразований.

Множество решений задачи. Оно определено как некоторое подмножество состояний фреймов (модели), при наличии которых решение задачи считается достигнутым *(C = Ce)*:

$S \subseteq Fr$

Стратегия. Стратегия представляет собой множество способов выполнения действий (множество процессов) для поиска решений.

$Str = < D, \{p\} >$

Остальные элементы базы знаний могут быть реализованы с помощью описанных подходов.

4.3. Графическое представление SIE-модели

Первоначально SIE-модель была ориентирована только на создание непротиворечивого и формально точного множества элементов. Впоследствии появилась необходимость графического представления элементов и связей. SIE-модель имеет 4 уровня, каждый из которых должен представляться графически отдельно (см. рис. 4.1).

Экспертный уровень

В связи с тем, что экспертный уровень представляет собой БЗ, его графическое отображение зависит от используемого ЯПЗ. Для связи с этим уровнем используется специальный блок — *Селектор*. В нем указывается номер или другое условное (сокращенное) обозначение используемого правила.

— *Селектор*

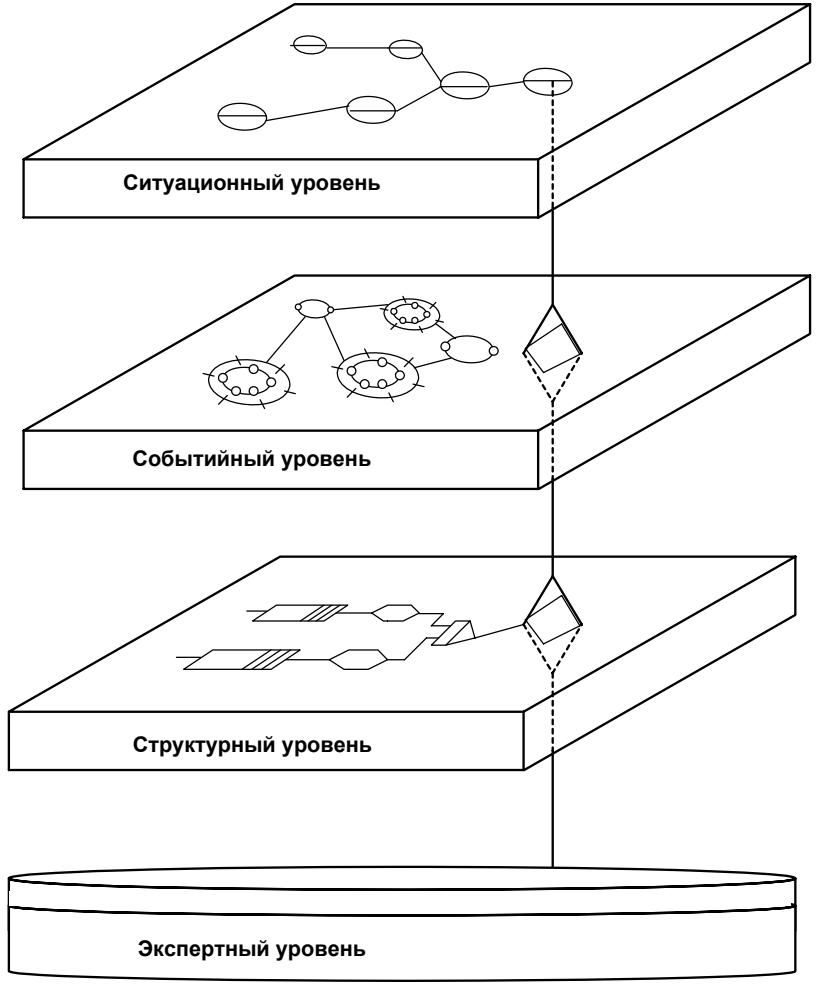


Рис. 4.1. Уровни SIE-модели.

Блок селектора может использоваться на структурном, событийном и ситуационном уровнях (С-уровнях). Все три уровня используют сетевое представление, т.е. состоят из вершин и дуг. Неориентированные

дуги без дополнительных признаков указывают на структурную связь объектов (блоков, процессов, ситуаций).

Структурный уровень

На структурном уровне используется пять типов вершин:

— *Очередь*

— *Устройство*

— *Персонал (субъект)*

— *Сборщик*

— *Распаковшик*

Данные условно-графические обозначения (УГО) могут быть заменены аналогичными, используемыми в широко известных СИМ GPSS, SLAM, SIMAN. Для связи вершин используется четыре типа дуг:

— *Неориентированная дуга.* Указывает на структурную целостность двух блоков. Например рабочим местом верстальщика (субъекта) является компьютер (см. рис. 4.2.). При моделировании переход по этой связи не осуществляется.

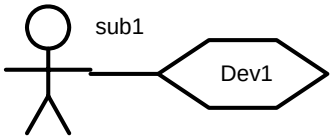
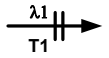


Рис. 4.2. Связь устройства и субъекта.

— *Ориентированная дуга.* Означает непосредственный переход к другому блоку без временных задержек. Например, для перехода транзакта после выхода из селектора.

— *Канал передачи данных.*



— *Поток транзактов.*

Событийный уровень

На событийном уровне используется четыре типа вершин:

— *Событие*

— *Процесс*

— *Событие* (обозначается

— *Микроситуация*

чертой и служит для
компактной записи)

Для моделирования в С-уровнях используется специальный маркер (фишка), указывающий на текущие положения транзактов. Можно использовать также цветовые выделения. Маркированные элементы событийного уровня выглядят следующим образом:

Если события, процессы или микроситуации привязаны к одному объекту, то они объединяются с помощью неориентированных дуг. Очередность указывается с помощью ориентированных дуг. В центре концентрических кругов может указываться имя объекта или его состояние (см. рис. 4.3).

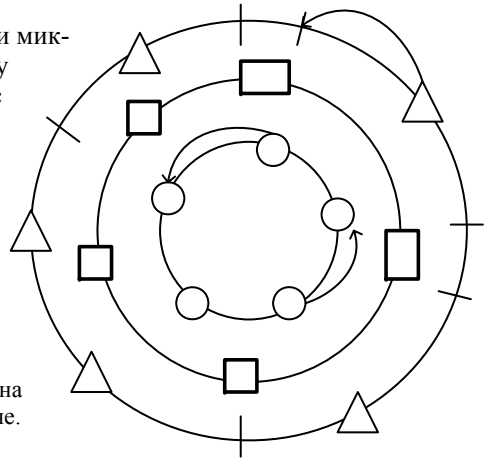
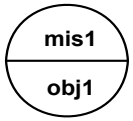


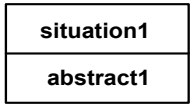
Рис. 4.3. Пример описания на событийном уровне.

Ситуационный уровень

На ситуационном уровне используется три типа вершин:



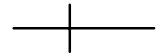
— Микроситуация, в которой находится объект.



— Ситуация, в которой находится объект или система.

— Событие. Обозначается также как на событийном уровне.

Для связи ситуаций используется неориентированная дуга, для указания последовательности — ориентированная (см. рис. 4.4). Связь объек-



тов, установленная на структурном уровне, обозначается здесь специальной дугой.

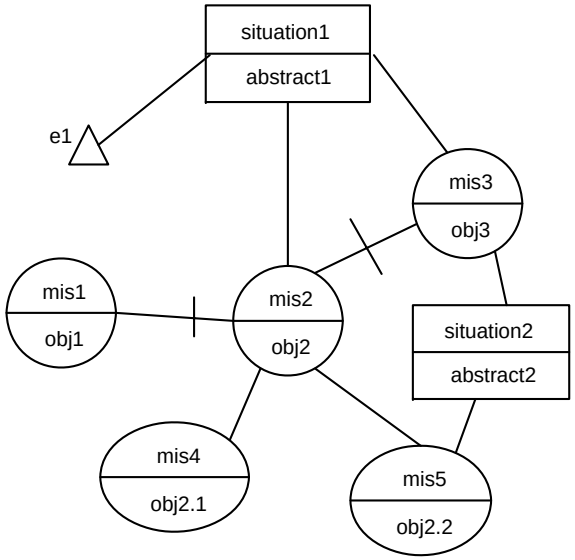


Рис. 4.4. Пример описания на ситуационном уровне.

В SIE-модели на ситуационном и других уровнях предусматривается возможность обобщенного представления микро- и макроситуаций, процессов и подпроцессов и т.д. Для указания вложенности объектов необходимо объединить их дугой (рис. 4.5).

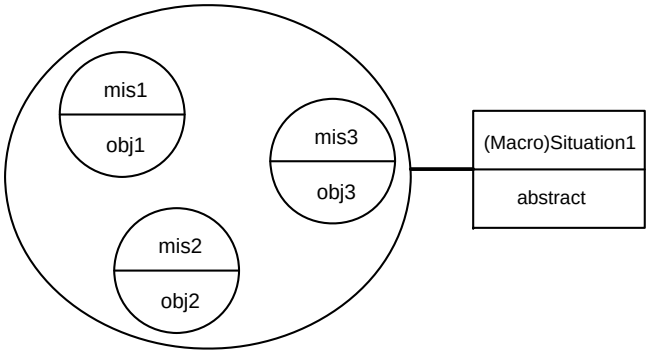


Рис. 4.5. Пример описания макроситуации.

Выводы

В главе предложена интегрированная (SIE) модель описания для имитационных, ситуационных подходов и языков представления знаний в ЭС. В рамках модели выделены четыре уровня описания: структурный, событийный, ситуационный и экспертный. Осуществлено формализованное математическое описание всех элементов системы. Выделено 53 основных понятия SIE-модели, среди них: объект, класс, атрибут, связь, событие, процесс, действие, фрейм, гипотеза, решение, микроситуация, ситуация и др.

Большинство элементов SIE-модели построены на основе общепринятых и широко известных определениях, приводимых в литературе. Формирование множества понятий осуществлялось постепенно от более простых к более сложным. Сами определения неоднократно корректировались при участии ведущих специалистов в области имитационного, ситуационного моделирования и экспертных систем: В.М. Черненького, Л.А. Соломонова, А.А. Башлыкова, Ю.Н. Филипповича. Модель неоднократно выносилась на публичное обсуждение на кафедрах ИУ5 МГТУ им. Баумана, ИТ МГУП, на конференциях и в публикациях.

Основное назначение SIE-модели заключается в интегрированном и формально-точном описании системы, адаптируя при необходимости широко распространенные подходы. В книге также приводятся правила использования SIE-модели для имитации (моделирования), которые, однако, носят обобщенный характер и требуют дополнительной проработки. Возможно, что ориентация на моделирование (симуляцию) внесет изменения в SIE-модель, которые могут иметь следующий характер: более точно определятся свойства элементов, появятся новые объекты, усложнится характер взаимодействия уровней и отдельных элементов.

В рамках процесса создания SIE-модели существует задача о формировании правил преобразования описаний, выполненных с помощью других моделей и языков. Частично эта задача решена на уровне графического языка SIE-модели. Однако утверждение о тождественности SIE-модели и другого языка моделирования требует проверки справедливости всех аксиом, теорем и правил, а также исследования их влияния на параметры элементов и связи с другими объектами.

В настоящее время осуществляется разработка системы SIE-моделирования, на основании которой можно будет более обосновано судить об эффективности и правильности SIE-модели. Необходимо еще раз подчеркнуть, что необходимость в создании интегрированной модели появилась из-за неудовлетворительных результатов взаимодействия систем на основе существующих подходов. SIE-модель строилась при

решении конкретных задач с использованием определенных систем, поэтому она имеет некую “полиграфическую специфику”.