ISSN 1684–8853

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

5/2003

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

5/2003

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Учредитель и издатель:

ФГУП «Издательство "Политехника"»

Адрес редакции: 191023, Санкт-Петербург,
Инженерная ул., д.6
Тел./факс: (812) 312-53-90
E-mail: asklab@aanet.ru

Журнал зарегистрирован в Министерстве РФ по делам печати, телерадиовещания и средств массовых коммуникаций. Свидетельство о регистрации ПИ № 77-12412 от 19 апреля 2002 г.

Журнал распространяется по подписке. Подписку можно оформить в любом отделении связи по каталогу агентства "Роспечать". Индекс 15385.

ЛР № 010292 от 18.08.98.
Сдано в набор 24.10.2003. Подписано в печать 24.11.2003. Формат 60×90 1/8.
Бумага офсетная. Гарнитура Pragmatica. Печать офсетная.
Усл. печ. л. 8,0. Уч.-изд. л. 11,3. Тираж 1000 экз. Заказ 846.

Оригинал-макет изготовлен в ФГУП «Издательство "Политехника"»,
191023, Санкт-Петербург, Инженерная ул.; д. 6
и отделе электронных публикаций и библиографии ГУАП,
190000, Санкт-Петербург, ул. Б. Морская, 67.

Отпечатано с готовых диапозитивов в ООО «Политехника-сервис».
191023, Санкт-Петербург, Инженерная ул., д. 6.

УДК 519.71

ПРИНЦИПЫ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ ДИНАМИЧЕСКИМИ ОБЪЕКТАМИ

В. В. Дубаренко,
д-р техн. наук
Институт проблем машиноведения РАН, Санкт-Петербург

Исследуются методы построения систем логического управления, обеспечивающих повышение качества нелинейных динамических объектов, и предлагаются алгоритмы их реализации.

Synthesizing methods of logic control systems are investigated. Methods provide improvement of quality of control by nonlinear dynamic objects. Realizing algorithms of these methods are offered.

Общие принципы построения моделей динамических систем как объектов управления

При управлении динамическим объектом (ДО) в реальном времени (ДО $\to T^R$) существует несколько подходов к обеспечению требуемого качества управления Ξ. Эти подходы связаны с реализацией в регуляторах законов управления $\hat{U}$, в которых для вычисления управляющих воздействий $u(t)$ либо используются поисковые процедуры $\exists\pounds^+$, либо не используются $\exists\pounds^-$. До последнего времени реализация в регуляторах $\exists\pounds^+$ наталкивалась на непреодолимые трудности из-за невозможности обеспечения вычислений в заданном темпе $\mp$. Реализация Ξ достигалась при помощи регуляторов, в которых частотными методами обеспечивался заданный уровень усиления в рабочей полосе частот. Точность управления θ гарантировалась только для замкнутого контура элементов системы, охваченных датчиками обратной связи, а значительная часть элементов, подверженных возмущениям, не контролировалась. Поэтому дальнейшее повышение $\theta \in$ ДО обусловлено прежде всего возможностью получения достоверной оценки компонент вектора состояния, недоступных для прямого измерения:

$$\tilde{\mathbf{x}}^- = [\tilde{x}_1^-, \tilde{x}_2^-, \ldots, \tilde{x}_3^-].$$

Организация оптимального управления ДО $\mathring{\boldsymbol{u}}$, описывающихся системами нелинейных дифференциальных уравнений высокого порядка (НСДУ), при ограничениях, наложенных на переменные управления $\boldsymbol{u} \in U$ и состояния $\boldsymbol{x} \in X$, еще наталкивается на значительные методические и вычислительные трудности. К их числу следует отнести разработку численных методов определения функции оптимального управления при наличии ограничений $\mathring{\boldsymbol{u}}(t) \in U$.

Проблема синтеза системы оптимального управления, как известно [1], связана с решением некоторой вариационной задачи Ψ, являющейся математической формулировкой цели управления при физических ограничениях $J = \underset{\min(\mathbf{x},\,\mathbf{u})}{J(\mathbf{x}, \mathbf{u})}$, $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t)$, $\mathbf{x} \in X$, $\mathbf{u} \in U$, $t_0 \le t \le T$. Основная трудность, возникающая при решении этой проблемы, состоит в отсутствии математического аппарата, который давал бы в общем случае решение задачи синтеза в виде оптимального оператора обратной связи $\boldsymbol{u}(t) = K(\boldsymbol{x}(t))$.

Исключения составляют задачи синтеза линейных систем управления (LCS) $\dot{\mathbf{x}}(t) = Ax(t) + Bu(t)$, $y(t) = Cx(t)$, $(A, B, C \in \Pi)$ с квадратичными критериями качества J_2, решение которых может быть получено в замкнутой форме. В этом случае отыскание оператора обратной связи $\boldsymbol{u}(t) = Q^{-1}B^tP\boldsymbol{x}(t)$, доставляющего минимум интегральной квадратичной оценке качества $J_2 = \int_{t_0}^{t_1}(\boldsymbol{y}^t(t)R\boldsymbol{y}(t) + \boldsymbol{u}^t(t)Q\boldsymbol{u}(t))dt + \boldsymbol{x}^t(t)P\boldsymbol{x}(t)$, осуществляется путем решения матричного дифференциального уравнения, известного как уравнение Риккати $\Re\ P(t) = C^tRC - P(t)BQ^{-1}B^tP(t) + P(t)A + A^tP(t)$. Однако при таком подходе серьезной проблемой становится выбор значений весовых коэффициентов Q и R в целевой функции квадратичного критерия качества J_2, обеспечивающего необходимые требования к переходному процессу с учетом ограничений L $(\boldsymbol{u} \in U, \boldsymbol{x} \in X)$.

Очевидно, что возможности применения того или иного метода решения Ψ, в первую очередь, определяются типом самой вариационной задачи, т. е. способом задания функционала J, граничных условий $\Im\ (\boldsymbol{x}(0), \boldsymbol{x}(t))$ и ограничений L.

Наибольшие достижения теории оптимального управления в области вычислительных методов решения Ψ относятся к методам отыскания

$\mathring{\boldsymbol{u}}(t, \Im, L, \Pi) \in \mathring{U}$, обеспечивающих программный перевод LCS в заданную область пространства состояний Cl^0 (внутренность кластера с заданными границами) [2].

Поэтому важнейшим этапом синтеза $\mathring{\boldsymbol{u}}$ является математическая формулировка цели управления J в виде Ψ. Как было отмечено выше, основным Ξ ДО является $\theta(t)$ отслеживания траектории его движения $x^*(t)$ в условиях действия различного рода возмущений $f_в(t)$.

Применение классических методов теории $\mathring{\boldsymbol{u}}$ для решения широкого класса Ψ позволяет сформулировать необходимые условия оптимальности [5] в виде некоторой двухточечной краевой задачи для системы дифференциальных уравнений (СДУ)

$$\aleph\left(J = \min_{(x,u)} J(x, u),\ \dot{\mathbf{x}} = f(x, u, t),\ \Im, L, t_0 \le t \le T\right).$$

Однако численное решение $\aleph$, за исключением сравнительно простых случаев, является самостоятельной проблемой. В связи с этим решение поставленной задачи с помощью известных в настоящее время методов, непосредственно использующих необходимые условия оптимальности, наталкивается на непреодолимые трудности.

Особый интерес представляет собой класс задач $\mathring{\boldsymbol{u}}$, сводящихся к линейным конечномерным аппроксимациям.

К числу таких задач прежде всего можно отнести задачу максимального быстродействия для LCS $\Delta x\ (t) \to 0$. Использующие данный подход алго-
$t=\min$
ритмы численного решения этой задачи основаны на различных способах ее сведения к конечной последовательности задач линейного программирования Σ. Алгоритмы обладают хорошей устойчивостью вычислений, но для получения точного результата требуют подсчета большого числа итераций, на каждой из которых осуществляется решение задачи ЛП большой размерности. Это связано с чрезмерными затратами машинного времени.

Трудности численного решения $\mathring{\boldsymbol{u}}$ в T^R и необходимость частого пересчета управляющего воздействия на основе текущей информации о состоянии ДО приводят к отказу от реализации строгих оптимальных решений методом ЛП.

Ограничения L в реальных системах определяются не только линейными неравенствами, а могут носить невыпуклый характер. Кроме того, сами ДО имеют явно выраженные нелинейности. При повышении требований влияние нелинейностей начинает все больше сказываться, и в конечном итоге динамические процессы приобретают нелинейный и нестационарный характер. Особенно ярко это проявляется в механических объектах со слабо демпфированными собственными колебаниями. Металлические конструкции некоторых объектов (например, радиотелескопов) на первой резонансной частоте f_1 имеют всплеск амплитуды около 20 дБ, т. е. в 10 раз превышающей амплитуду возбуждающего воздействия, с временем затухания $t_з \approx 10$ с. Очевидно, что повышение точности управления такими объектами связано с демпфированием колебаний за счет управляющих воздействий $\boldsymbol{u}(t)$. Эффективность демпфирования прямо пропорциональна значению $\sup(\boldsymbol{u}(t))$. Наибольшая эффективность демпфирования достигается тогда, когда $\boldsymbol{u}(t)$ приобретает релейный характер: $\overline{\boldsymbol{u}}(t) = \operatorname{sign}(\boldsymbol{x}(t))$ со значением $\sup(\boldsymbol{u}(t)) = U_0$. Это подтверждает известный факт из теории управления, что $\mathring{\boldsymbol{u}}(t)$ следует искать в классе кусочно-постоянных функций $\overline{\boldsymbol{u}}(t)$.

Концепция повышения качества процессов управления

В пространстве состояний рассмотрим две точки. Одна точка $\boldsymbol{x}(t)$ называется целью, другая $\boldsymbol{x}^*(t)$ — объектом. Путем квантования фазовых координат все пространство состояний разбивается на кластеры Cl^0. Основной задачей управления является перевод объекта $\boldsymbol{x}(t)$ из кластера $Cl^0(\boldsymbol{x}(t))$ в кластер цели $Cl^0(\boldsymbol{x}^*(t))$ или удержание его вблизи этого кластера в соответствии с выбранным критерием качества. Если на объект действуют случайные возмущения, то оценивать принадлежность $\boldsymbol{x}(t)$ к определенному кластеру Cl_i^0 можно только с некоторой вероятностью p_i.

При наличии нелинейностей и ограничений, наложенных на фазовые координаты, при описании ММ ДО оценки его состояния можно определить только с помощью логико-вероятностных функций. Динамические процессы рассматриваются в евклидовом пространстве E^n, и квадрат расстояния между двумя точками x_i и x_j в этом пространстве выражается квадратичной формой $r_{i,j}^2 = (x_i - x_j)^t P (x_i - x_j)$ с диагональной матрицей P.

Если из всего множества состояний ДО каждому Cl_i^0 приписывать лишь один вектор, это дает возможность уменьшить размерность задачи и служит основанием для возможности реализации $\exists £^+$ в T^R.

Таким образом, можно сделать вывод, что при изложенных допущениях существуют два подхода к синтезу законов управления сложными нелинейными ДО. Первый подход основан на линеаризации математических моделей ДО в стационарных точках, а второй — на использовании нелинейных прогнозирующих моделей ДО в поисковых процедурах принятия решений о значении управляющих воздействий.

Принципы построения систем логического управления

Проблемы логического управления динамическими объектами. При управлении нелинейными, нестационарными ДО, на фазовые координаты которого наложены произвольные ограничения типа неравенств, классические методы управления становятся непригодными.

К числу неклассических задач управления, к которым неприменимы классические методы, можно отнести такие задачи, как управление транспорт-

ными средствами при наличии препятствий, парковка транспортных средств с учетом их динамики, управление летательными аппаратами в нечетко определенной обстановке, управление электроприводами наведения крупных радиотелескопов при ограничениях на мощность, силу тока, крутящий момент и др.

Решение подобных задач связано с решением неклассических вариационных задач, необходимые условия которых для обыкновенных НСДУ формулируются в форме двухточечных краевых задач. Численное решение двухточечной краевой задачи, за исключением простых случаев, является самостоятельной проблемой. Численные методы решения вариационных задач принято разделять на две большие группы.

К первой группе относятся методы, которые направлены на отыскание управляющих функций, непосредственно удовлетворяющих необходимым и достаточным условиям оптимальности. Это различные модификации метода Ньютона [14], методы прогонки, методы «со свободным концом», И. А. Крылова и Ф. Я. Черноусько [15], методы штрафных функций [16].

Все перечисленные численные методы решения задач оптимального управления, использующие необходимые условия оптимальности, просты в описании и удобны для машинной реализации. В то же время они не пригодны для решения невыпуклых задач, какими являются задачи, перечисленные выше.

Ко второй группе относятся методы, не использующие необходимые и достаточные условия оптимальности. К их числу принадлежат, например, все методы градиентного спуска, которые также не пригодны для невыпуклых задач.

Стратегии управления ДО.

Рассмотрим уравнение движения n-мерного ДО управления в разностной форме:

$$\boldsymbol{x}(t+\Delta t) = f(\boldsymbol{x}(t), A(t), u(t)),$$

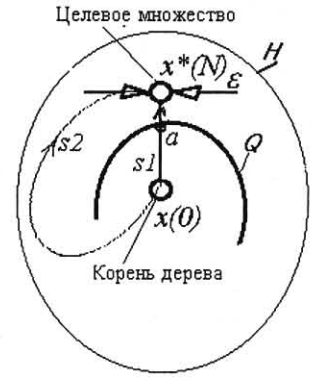
где $\boldsymbol{x}(t)$, $A(t)$, $u(t)$ — функции времени t, $t = \Delta tm$, $m = 0, 1, 2\ldots$; Δt — такт; $\boldsymbol{x}(t)$ — ВС; $A(t)$ — оператор перехода $\boldsymbol{x}(t)$ в $\boldsymbol{x}(t+\Delta t)$; $u(t)$ — функция управления.

Ограничения: $\boldsymbol{x}(t) \in \boldsymbol{H}$, $u(t) \in (U, -U)$. $u(t)$ является кусочно-постоянной функцией на интервале Δt и принимает одно из двух возможных значений: $U, -U$.

Требуется найти функцию $u(t)$, которая обеспечивает переход ДО из $\boldsymbol{x}(0)$ в $\boldsymbol{x}(T)$ за минимальное время $T = n^*\Delta t$ или минимальное число шагов n. Точность перевода ДО в конечное состояние зависит от точности, с которой известны его характеристики и возмущения.

Очевидно, что неточности математического описания ДО, наличие стохастических факторов, нестабильность его параметров и измерительных средств требуют частого пересчета (прогнозирования) управляющего воздействия на основе текущей информации о векторе состояния $\boldsymbol{x}(t)$.

Поэтому численное решение вариационной задачи в реальном масштабе времени ДО $\rightarrow T^R$ требует высокой производительности вычислительных



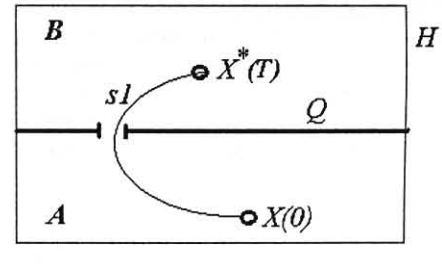
■ ***Рис. 1.*** *К выбору стратегии управления ДО при непрерывном ограничении Q*

средств, что до последнего времени сдерживало применение численных методов математического программирования в регуляторах ДО.

В области фазового пространства H (рис. 1) выделим две точки — ВС $\boldsymbol{x}(0)$ и вектора целевого множества $\boldsymbol{x}^*(T)$, а также ограничение Q — в виде непрерывной кривой.

Требуется выбрать стратегию перевода $\boldsymbol{x}(0)$ в $\boldsymbol{x}^*(T)$ и в соответствии с выбранной стратегией найти функцию управления, обеспечивающую перевод за минимальное число шагов. Если в качестве целевой функции выбрать расстояние между $\boldsymbol{x}(0)$ и $\boldsymbol{x}^*(T)$ как эвклидову норму $R = \|\boldsymbol{x}^*(T) - \boldsymbol{x}(0)\|$, то становится очевидным, что градиентные методы для этой задачи неприменимы, так как в стационарных точках, в которых вычисляется градиент, ограничение $Q(x)$ никак не учитывается. Движение по экстремали s_1, например, с максимальной скоростью уменьшения R (метод скоростного градиента) будет проходить до точки a, в которой дальнейшее уменьшение R невозможно из-за ограничения $Q(x)$. Выбор стратегии управления для построения экстремали s_2, при движении по которой ограничения соблюдаются, становится очевидным только тогда, когда «видна» вся обстановка внутри региона H, т. е. имеется информация об этой обстановке. Однако формализовать получение этой информации является проблемой.

Другой концептуальный пример трудностей формализации выбора стратегии управления поясняет рис. 2.



■ ***Рис. 2.*** *К выбору стратегии управления ДО при ограничении Q в виде щели*

Из точки $\boldsymbol{x}(0)$, находящейся в области A, требуется перейти в точку $\boldsymbol{x}^*(T)$, находящуюся в области B. В этой задаче для построения экстремали s_1 нужно найти щель в ограничении $Q(x)$ региона фазового пространства H.

Современные методы управления нестационарными ДО с ограничениями на ВС основываются на методах, вычислительной основой которых является математическое программирование. Один из них — метод бинарных деревьев (МБД) [18].

Метод бинарных деревьев. Суть МБД заключается в следующем. Управляющие воздействия ограничиваются классом кусочно-постоянных функций в виде положительных и отрицательных импульсов, порождающих в ПС бинарные деревья (БД). Динамический процесс интерпретируется как рост БД. По мере роста БД его узлы попадают в различные области ПС (кластеры). Ограничения, накладываемые на фазовые координаты ДО, образуют запретные области для узлов БД. В узлах, которые попали в запретные области, влючая их границы, эволюция БД заканчивается. С течением времени в одни и те же кластеры могут попадать новые узлы дерева и образовывать циклы. Для исключения зацикливаний в этих узлах эволюция БД также завершается. Целью управления является попадание, в процессе роста БД, одного или нескольких узлов в заданный кластер (целевое множество). Другими словами, целью управления является отыскание оптимальной управляющей функции, переводящей ДО из начального состояния в некоторую точку целевого множества за минимальное время и удерживание ДО в этом целевом множестве путем периодического прогнозирования решения на заданном интервале времени прогнозирования.

Реализация алгоритмов регуляторов, основанных на МБД, предназначенных для работы в реальном времени, вызывает значительные трудности из-за сложности вычислительного процесса. В работе [17] рассматривается метод поиска оптимального управления при адаптации к воздействиям внешней среды. Он назван логической оптимизацией управления по методу бинарных деревьев. Оптимизационная процедура отыскания управляющего воздействия осуществляется посредством построения БД состояний ДО. Для этого строится прогнозирующая модель, с помощью которой для заданного времени прогнозирования и шага прогнозирования в каждом узле БД вычисляются оценка вектора состояния (ВС) ДО и его евклидова норма, которая запоминается и хранится в памяти ЭВМ в течение всего цикла прогнозирования. Норма ВС рассматривается как мера расстояния от узла до начала координат. Управляющее воздействие определяется комбинаторным методом как логическая функция упорядоченного множества узлов БД. Блок-схема алгоритма управления по методу БД представлена на рис. 3.

■ ***Рис. 3.*** *Блок-схема алгоритма управления по методу БД*

Применение математического программирования для задач управления нелинейными ДО привело к большому числу различных алгоритмов, основными чертами которых являются поисковые переборные процедуры возможных решений. Практическое применение таких алгоритмов сдерживается из-за их большой вычислительной трудоемкости (сложности). Теоретическое обоснование принципиальной возможности уменьшения вычислительной сложности представляет самостоятельную проблему, которой посвящено значительное число работ. Общепринятая методика исследования таких задач предполагает разложение или сведение исходной задачи к некоторому числу задач, для которых вычислительная сложность является известной. Обычно оценка сложности связывается с размерностью объектов, входящих в задачу. При этом возможны следующие пути решения задач:

— решение в один этап на мелкой сетке;

— последовательность решений в несколько этапов на крупной сетке.

Решение за один этап на мелкой сетке связано с большой размерностью задачи, но с малой комбинаторной сложностью.

Последовательность решений в несколько этапов на крупной сетке связана с малой размерностью задачи, но со сравнительно большой комбинаторной сложностью. Очевидно, возможен компромисс при выборе числа этапов и размеров ячеек сетки.

Логическое управление ДО в кластерном пространстве состояний является характерным примером указанного подхода. Структурная схема системы логического управления приведена на рис. 4.

Предлагается следующая *стратегия управления:*

1. В ПС выделяется регион, ограниченный максимально допустимыми значениями компонент вектора ВС.

2. Если ДО находится вне региона, то для возвращения его в регион применяется Стратегия 1, согласно которой:

2.1. Посредством прогнозирующей модели (ПМ) ДО вычисляются два ВС при заданном значении времени прогнозирования для двух постоянных, противоположных по значению управляющих воздействий BC^+ и BC^-.

2.2. ВС нормируются путем деления их компонент на соответствующие максимально допустимые значения.

2.3. В нормированном пространстве определяются расстояния между целевой точкой BC_g, точками BC^+ и BC^-.

2.4. Выбирается тот знак управляющего воздействия, которому соответствует меньшее расстояние до целевой точки. Это управляющее воздействие подается на ДО.

2.5. Алгоритм Стратегии 1 повторяется до тех пор, пока ДО не войдет в заданный регион.

3. Если ДО и цель находятся в заданном регионе ПС, применяется Стратегия 2. Суть ее заключается в следующем:

3.1. Регион разбивается на кластеры.

3.2. Определяются номера кластеров, которым принадлежат цель и ДО.

■ ***Рис. 4.*** *Структурная схема системы логического управления*

3.3. Задается время запаздывания, необходимое вычислительному устройству (ЭВМ) для решения задачи перевода ПМ ДО в целевую точку.

3.4. ПМ ДО и модель цели продвигаются (путем интегрирования уравнений движения) в точки, соответствующие времени запаздывания.

3.5. Из указанных точек строятся или одно, или два дерева навстречу друг другу. Дерево из точки ПМ ДО строится с прямым оператором перехода, а из точки нахождения модели цели — с обратным оператором перехода. Алгоритм построения деревьев описан в литературе.

3.6. Решением считается событие, состоящее в том, что одному кластеру принадлежат какие-либо ВО двух деревьев, или, что ВС дерева ПМ ДО и ВС модели цели принадлежат одному кластеру.

3.7. Размеры региона уменьшаются до размеров кластера, которому принадлежат ВС ДО и цели.

3.8. Регион снова разбивается на кластеры, но меньших размеров, и осуществляется переход к п. 3.2.

4. Если ДО и цель находятся в одном кластере и размеры кластера минимально допустимы, т. е. кластер является целевым множеством, то применяется Стратегия 3, согласно которой:

4.1. Задается время запаздывания, необходимое вычислительному устройству (ЭВМ) для решения задачи перевода ПМ ДО в целевую точку.

4.2. ПМ ДО и модель цели продвигаются (путем интегрирования уравнений движения) в точки, соответствующие времени запаздывания. Управляющее воздействие на DO остается таким, каким оно было на предыдущем цикле вычислений.

4.3. Задается время прогнозирования.

4.4. Из точки состояния ПМ ДО прогнозируются два ВС при заданном значении времени прогнозирования для двух постоянных, противоположных по значению управляющих воздействий $ВС^+$ и $ВС^-$.

4.5. ВС нормируются путем деления их компонент на соответствующие максимально допустимые значения.

4.6. В нормированном пространстве определяются расстояния между целевой точкой $ВС_g$, точками $ВС^+$ и $ВС^-$.

4.7. Выбирается тот знак управляющего воздействия, которому соответствует меньшее расстояние до целевой точки. Это управляющее воздействие подается на ДО.

Для принятия решения выбирается максимум вероятности попадания случайной величины в заданный интервал.

Оценка эффективности методов логического управления ДО. Эффективность методов управления ДО обычно связывается с возможностью реализации этих методов на вычислительных устройствах (ВУ) в форме алгоритмов для управляющих функций (воздействий). Сложность реализации алгоритмов обусловлена жестким требованием к времени их выполнения в ВУ, характерным для систем реального времени. Другими словами, алгоритмы должны выполняться в ВУ за строго определенное время.

Для сравнительной оценки эффективности методов управления нужно прежде всего уметь определять сложность вычисления управляющих воздействий.

Сложность вычислений составляет одну из важнейших проблем современной науки о вычислениях [20] и представляет быстроразвивающуюся область математики. Было выработано много различных подходов к проблеме сложности вычислений [21]. Большинство из этих подходов связано с отысканием для логических схем и описывающих их булевых функций таких форм символьного или комбинаторного представления, из которых легко вычисляются длины формул и количество вычислительной работы [22]. При этом для изучения свойств алгоритмов требуется некая общая для всех форма представления. Такой общепринятой формой является машина Тьюринга, которая, согласно выдвинутому А. Чёрчем тезису, обладает свойством, заключающимся в том, что любая изобретенная человеком процедура может быть реализована машиной Тьюринга. Машина Тьюринга является простым по своей идее устройством, предназначенным для механической реализации вычислительных процедур, но как средство для описания вычисляемых функций малопригодна, так как не имеет развитого языка, в котором можно явно обращаться к функциям.

Следуя [21], рассмотрим общепринятый или, более точно, известный подход к оценкам сложности.

Назовем f булевой (логической) функцией (ЛФ) с областью определения $\{0, 1\}^n$ и множеством значений $\{0, 1\}$, где n — целое число, а $\{0, 1\}^n$ обозначает n-кратное декартово произведение множества $\{0, 1\}$ на себя, т. е. множество всех двоичных наборов длины n. Функции будем записывать в виде f: $\{0, 1\}^n \to \{0, 1\}$. Наборы из m ЛФ $f_1, f_2, \ldots, f_m$ будем записывать в виде f: $\{0, 1\}^n \to \{0, 1\}^m$, или $f(x_1, x_2, \ldots, x_n)$, где $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$ обозначено базисное множество логических переменных из области определения $\{0, 1\}^n$.

Базисом Ω назовем минимально возможное множество логических операторов (связок), достаточных для порождения любой произвольной ЛФ. Примерами базисов могут служить $\{\vee, \&, \bar{}\}$, $\{\oplus, \&, 1\}$. Часто для записи формул ЛФ используются смешанные базисы, например, $\{\vee, \&, \bar{}, \oplus\}$. Строго говоря, для записи ЛФ в смешанном базисе необходимо применение еще одного оператора, который должен включаться в базис, а именно оператор «( )» — скобки. Он определяет приоритет операций. Однако в литературе по исследованию логических систем этот оператор часто опускается, а приоритет операций как бы подразумевается, что, в конечном итоге, приводит к неоднозначностям и недоразумениям.

Такое же замечание можно сделать относительно использования оператора «=». Один из крупнейших специалистов в области математической логики С. К. Клини [23] замечает, что «равенство» помимо основных свойств: рефлексивности, симметричности и транзитивности — обладает свойством замены, а неучитывание этого последнего

свойства приводит к подмене «равенства» «эквивалентностью», что так же порождает недоразумения и ошибки. Поэтому при описании ЛФ «равенство» допустимо применять только в базисах {⊕, &, 1}, {⊕, 1} и недопустимо в базисе {∨, &, ¯}.

Одни и те же ЛФ f в различных базисах могут иметь различную длину, которая прямо связывается с понятием комбинационной сложности. Стремление получить минимальную комбинационную сложность ЛФ f определяется их реализацией в логических схемах ЭВМ, т. е. синтезом этих схем. Оценки сложности при таком подходе носят утилитарный, прагматический характер и не во всех случаях могут использоваться для сравнения алгоритмов. Для такого сравнения ЛФ должны быть представлены в одном базисе. В работе [27] обсуждаются достоинства базиса {⊕, 1}, характерного для линейных последовательностных машин (ЛПМ) [28]. Любая ЛФ в этом базисе представляется в виде полинома Жегалкина единственным образом [29], а конечный автомат — ЛПМ.

При синтезе законов управления линейных или линеаризованных ДО относительно квадратичных критериев качества для стационарных условий или градиентных критериев, для локальных областей фазового пространства, сами законы, полученные в результате синтеза, представляют собой тривиальные вычислительные алгоритмы векторно-матричных операций. Длина этих алгоритмов и вычислительная сложность могут быть определены по известным формулам в зависимости от размерности вектора состояния

Характерной особенностью нетривиальных вычислительных алгоритмов является то, что они могут быть представлены в форме двух частей (рис. 5): арифметической части, в которой последовательно выполняются вычислительные процедуры, вычислительная сложность которых известна, и логической части, которая использует результаты расчета, для того чтобы сделать один или несколько логических шагов для вычисления вектора логических переменных (ЛП). Атрибуты ЛП [25] в качестве начальных условий (или обратной связи) подаются в арифметическую часть, чтобы осуществить следующую итерацию. Процесс продолжается до тех пор, пока значения атрибутов вектора ЛП не удовлетворят заданным условиям.

Поисковые процедуры алгоритмов могут быть представлены в форме логической комбинаторной схемы или конечного автомата Милли (КА) [24]. ЛПМ — это математическая модель вычислительной машины с памятью в базисе {⊕, 1}. Следовательно, для эквивалентного преобразования КА в ЛПМ за счет расширения вектора состояния КА необходимо исключить из базиса {⊕, &, 1} оператор &.

Используя понятие фундаментального вектора [9], задачу приведения произвольной системы ЛУ к форме ЛПМ [30] можно сформулировать следующим образом.

Пусть $\mathbf{S}(t)$ = фундаментальный вектор, построенный из базисного вектора КА. Тогда его математическая модель в базисе {⊕, 1} может быть записана в форме:

$$G\mathbf{S}(t+1) = H\mathbf{S}(t) \oplus B\mathbf{u}(t);$$

$$\mathbf{y}(t) = C\mathbf{S}(t) \oplus D\mathbf{u}(t);$$

$\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}(t)$ — двоичные [0, 1] векторы; B, G, C, D, H — двоичные [0, 1] матрицы; $t = 1, 2, 3, .., T$ — параметр целого типа.

В работе [27] показано, что если базисный вектор дополнить вектором $\mathbf{u}(t)$, то фундаментальный вектор $\mathbf{S}(t)$ расширится до $\mathbf{S}_p(t)$ и КА Милли преобразуется в КА Мура:

$$\mathbf{S}_p(t+1) = A\mathbf{S}_p(t);$$

$$\mathbf{y}(t) = C_p\mathbf{S}_p(t),$$

где A — квадратная матрица [0,1].

■ ***Рис. 5.*** *Схема вычислительного алгоритма*

В итоге КА может быть представлен в форме сдвигового регистра [28] или системы сдвиговых регистров, комбинационная сложность которых известна [21]. Сравнение алгоритмов управления в форме сдвиговых регистров имеет существенное преимущество по сравнению с другими формами их представления, так как для сравнения существует небольшой набор характеристик и параметров. К таким характеристикам относятся: ранг матрицы сдвигового регистра и его характеристический полином.

Приведение произвольного КА к форме ЛПМ не исчерпывает проблем вычислительной сложности задач управления ДО. ЛПМ служит лишь удобной математической моделью для исследования динамических процессов. Одной из основных задач для ЛПМ является ее перевод из одного произвольного состояния в другое за минимальное число шагов. Не сложно показать, что эта задача представляет собой задачу булева линейного программирования, которая относится к классу NP-полных задач, для которых не гарантируется решение за полиномиальное время и не исключен полный перебор. Для этих задач все известные полиномиальные по времени приближенные алгоритмы дают результаты, которые не могут сколь угодно точно приближать точные решения [24].

Приложение

Метод бинарных деревьев был применен для повышения динамической точности системы наведения по азимуту крупного наземного полноповоротного радиотелескопа с азимут-угломестной монтировкой РТ-70. Диаметр главного зеркала радиотелескопа составляет 70 м. Радиотелескоп предназначен для приема радиоволн от удаленных космических объектов в сантиметровом диапазоне. К системе наведения радиотелескопа предъявляется требование сопровождения по угловым координатам космических объектов в следящем режиме с точностью 25 угл. с. При этом предполагается, что на систему наведения радиотелескопа действуют мешающие точному наведению ветровые, весовые и инерционные нагрузки, нагрузки от трения в опорных узлах. Зеркальная система и опорно-поворотное устройство рассматриваются как система твердых тел с упругими связями и низкими (примерно от 1 до 3 Гц) слабодемпфированными частотами собственных колебаний.

Путем имитационного моделирования в вычислительной среде MatLab Simfor было проведено сравнение динамических процессов управления СН РТ, полученных с использованием регуляторов, реализующих в реальном времени линейные законы управления относительно квадратичных критериев качества, и регуляторов, реализующих алгоритмы логического управления.

Результаты численных исследований показывают (рис. 6), что точность наведения при логическом управлении может быть увеличена почти вдвое.

Параметры ММ системы управления:

$ce = 1{,}4В^*$ — коэффициент противоЭДС эдектродвигателя (ЭД) (с/рад);

$cm = 0{,}1$ — коэффициент момента ЭД (кг · м/А);

$tj = 0{,}1$ — постоянная времени якорной цепи ЭД (с);

$rj = 0$ — сопротивление якорной цепи ЭД (Ом);

$mu = 0{,}01$ — коэффициент вязкого трения ЭД (кг · м × × с/рад);

$k = 50$ — коэффициент усиления электронного усилителя (В/угл. с);

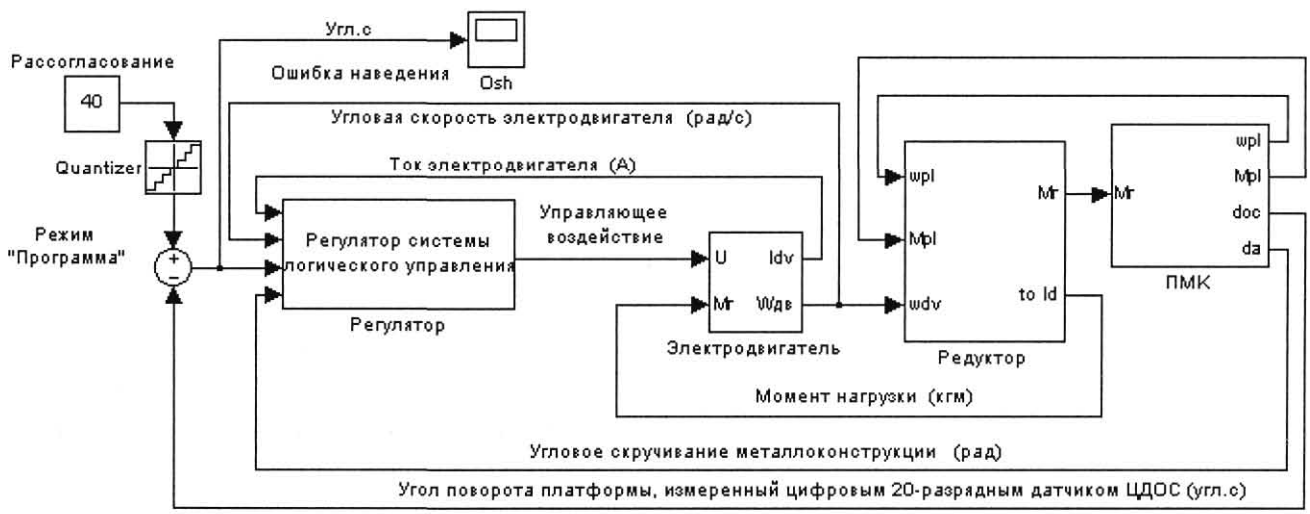
$Kw = k \cdot kr \cdot cm/(ir(rj \cdot mu + ce \cdot cm))$ — добротность по скорости (% рад/с);

$Jd = 0{,}03$ — массовый момент инерции ЭД (кг · м × × $с^2$)/рад);

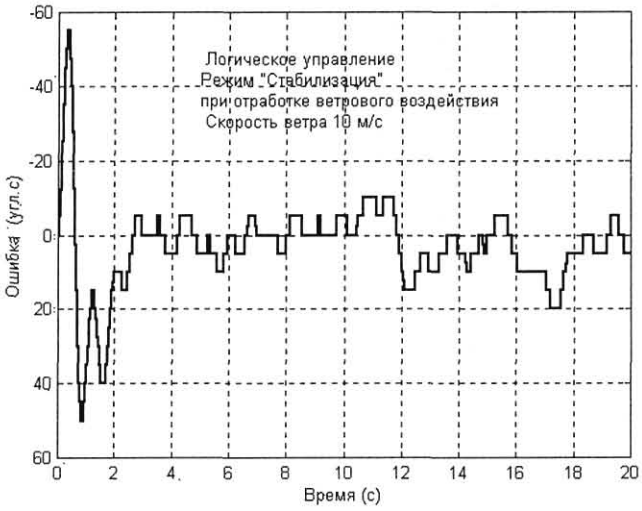
$h = 0{,}01$ — период квантования (с);

$Cr = 2{,}1 \cdot 10^{10}$ — жесткость редуктора (кг · м/рад);

$mjur = Cr \cdot 0{,}08$ — коэффициент демпфирования редуктора (кг · м/(рад/с));



■ ***Рис. 6.*** *Имитационная модель системы логического управления радиотелескопом РТ-70*



■ ***Рис. 7.*** *Ошибка системы логического управления радиотелескопом при отработке ветрового воздействия*

$C(1) = 2{,}3 \cdot 10^9$ — эквивалентная жесткость зеркальной системы (кг · м/рад);

$mju(2) = 0{,}013 \cdot C(1)$ — эквивалентное демпфирование зеркальной системы (кг · м/(рад/с));

$J(2) = 2{,}1 \cdot 10^7$ — эквивалентный момент инерции платформы (кг · м · $с^2$)/рад);

$J(3) = 3{,}74 \cdot 10^7$ — эквивалентный момент инерции зеркальной системы (кг · м · $с^2$)/рад).

Параметры эквивалентного синусного режима $A\sin(wt)$:

$V = 4$ угл. мин/с — максимальная скорость слежения;

$\theta = 25$ угл. с — точность;

$E = 0{,}005$ град/$с^2$ — максимальное ускорение слежения;

$\omega_к = E/V = 0{,}005 \cdot 60/4 = 0{,}0750$ 1/с — максимальная угловая частота управляющего воздействия (контрольная точка ЖЛАХ);

$H\omega_к = 20 \lg(V^2/(E\theta)) = 20 \log((4^2)/(0{,}005 \cdot 25)) = 97{,}0406$ Дб — логарифмический коэффициент усиления системы в контрольной точке;

$D\omega_к = \mathrm{sqrt}(2) \cdot V/\theta = \mathrm{sqrt}(2) \cdot 4 \cdot 60/25 = 13{,}5765$ (рад/с)/угл. с. — добротность по скорости.

Литература

1. **Моисеев Н. Н.** Численные методы в теории оптимальных систем. — М.: Наука, 1971.
2. **Табак Д., Куо Б.** Оптимальное управление и математическое программирование. — М.: Наука, 1975.
3. **Андреев Ю. Н.** Управление конечномерными линейными объектами. — М.: Наука, 1976. — 183 с.
4. **Квакернаак Х., Сиван Р.** Линейные оптимальные системы управления. — М.: Мир, 1977.
5. **Беллман Р., Калаба Р.** Квазилинеаризация и нелинейные краевые задачи. — М.: Мир, 1968.
6. **Марчук Г. И.** Методы вычислительной математики. — М.: Мир, 1977.
7. **Андриевский Б. Р., Фрадков А. Л.** Избранные главы теории автоматического управления с примерами на языке MATLAB. — СПб.: Наука, 1999.
8. **Фрадков А. Л.** Адаптивное управление в сложных системах. — М.: Наука, 1990.
9. **Городецкий А. Е., Дубаренко В. В.** Комбинаторный метод вычисления вероятностей сложных логических функций//ЖВМ и МФ. — 1999. — № 7. — С. 1201–1203.
10. **Gorodetsky A. E, Dubarenko V. V.** Method of approached calculation of probability of complex logic function used in logic-probabilistic description of reliability of technical systems. Mathematical Methods in Reliability//Proceedings o f the First International Conferense MMR'97. Part 1. — Bucharest, Romania, 1997.
11. **Городецкий А. Е., Дубаренко В. В.** Логическое управление в кластерном пространстве состояний динамических объектов//Тр. междунар. конф. «Интеллектуальные системы и информационные технологии управления, ИСИТУ-2000-IS&ITC». — Псков, 2000.
12. **Городецкий А. Е., Дубаренко В. В.** Метрологические характеристики кластерного пространства систем управления динамическими объектами // Физическая метрология. — СПб.: СПбГТУ, 2000.
13. **Городецкий А. Е., Дубаренко В. В.** Проблемы логико-лингвистического управления динамическими объектами//Тр. второй междунар. конф. «Логико-лингвистическое управление динамическими объектами DOLLC'99». — СПб., 2000.
14. **Исаев В. К., Сонин В. В.** Об одной модификация метода Ньютона численного решения краевых задач//ЖВМ и МФ. — 1963. — № 3, 6.
15. **Крылов И. А., Черноусько Ф. Л.** О методе последовательных приближений для решения задач оптимального управления//ЖВМ и МФ 2, 6, 1965.
16. **Шатровский Л. И.** Об одном численном методе решения задач оптимального управления//ЖВМ и МФ 2, 2, 1962.
17. **Городецкий А. Е., Дубаренко В. В., Курбанов В. Г.** Метод поиска оптимальных управляющих воздействий на динамические объекты с адаптацией к изменениям внешней среды//6-й Санкт-Петербургский симпозиум по теории адаптивных систем SPAS'99. — СПб., 1999.
18. **Gorodetsky A. E., Dubarenko V. V.** Binary trees in state space of dynamic control objects//First International Conferense on Problems of Dynamic Objects Logic-Linguistic Control DOLLC'97. — SPb., 1997.
19. **Колмыков С. А., Шокин Ю. И., Юлдашев З. Х.** Методы интервального анализа. — Новосибирск: Наука, 1986.
20. **Cook S. A.** The complexity of theorem proving procedures//Proceedings of the 3rd ACM Simposium on Theory of Computing (1971). — P. 151–158. [Русский перевод: Кук С. А. Сложность процедур вывода теорем//Кибернетический сборник. Новая серия. — Вып. 12. — С. 5–15].

21. **Сэвидж Джон Э.** Сложность вычислений: Пер. с англ.; Под ред. О. М. Касим-Заде. — М.: Факториал, 1998.
22. **Лузин С. Ю., Лутченков Л. С.** Анализ и разработка алгоритмов логического синтеза. — СПб.: ГУТ им. проф. М. А. Бонч-Бруевича, 1996.
23. **Kleene S. C.** Mathematical logic. JOHN WILEY & SONS, INC. — New York; London; Sydney, 1967.
24. **Рейнгольд Э., Нивергельт Ю., Део Н.** Комбинаторные алгоритмы. Теория и практика. — М.: Мир. 1980.
25. **Городецкий А. Е., Дубаренко В. В.** Алгебраические методы решения систем алгебраических уравнений//Управление в условиях неопределенности. — СПб.: СПбГТУ, 2002. — С. 97–132.
26. **Дубаренко В. В.** Применение метода символьных преобразований к исследованию дискретных логических динамических систем//Аппаратные и программные средства интеллектуальных автоматизированных систем. Сб. статей. — Вып. 2. — СПб.: ИПМАШ РАН, 1994.
27. **Городецкий А. Е., Дубаренко В. В., Ерофеев А. А.** Алгебраический подход к решению задач логического управления//А и Т. — 2000. — № 2. — С. 127–138.
28. **Гилл А.** Линейные последовательностные машины. — М.: Наука, 1974.
29. **Жегалкин И. И.** Арифметизация символической логики//Математический сборник. — Т. 35. — Вып. 3–4. — 1928. — С. 335.
30. **Дубаренко В. В.** О приведении систем логических уравнений к форме линейных последовательностных машин//Проблемы физической метрологии. Сб. докл. — СПб.: Изд-во KN, 1996. — С. 126–138.

УДК 629.7.062.5(075)

ПРИМЕНЕНИЕ НЕЧЕТКОЙ ЛОГИКИ В АВИАЦИОННЫХ СИСТЕМАХ АНТИЮЗОВОЙ АВТОМАТИКИ

А. С. Коновалов,
д-р техн. наук
П. Е. Шумилов,
канд. техн. наук
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Нечеткая логика уже получила широкое признание в задачах управления сложными системами. В настоящей статье рассматриваются возможности использования нечеткой логики для синтеза регулятора систем антиюзовой автоматики, характеризующихся наличием существенных нелинейностей и высоким порядком математической модели объекта управления и представляющих существенные сложности для классических методов синтеза регулятора, обеспечивающего эффективное торможение в различных условиях.

Fuzzy logic has been already widely recognized in complex system control. Aircraft antilock braking system (ABS) is a typical task for fuzzy logic, combining a number of significant non-linearities with high order of mathematical model of control object. For classical synthesis methods it is difficult to design an ABS regulator, wich would have provided effective control in various braking conditions.

Введение

Проблема эффективного торможения существует практически для всех колесных транспортных средств, которые должны иметь возможность остановиться за определенное время или на определенном отрезке пути. Особое значение эта проблема имеет для самолетов и автомобилей, используемых для перевозки людей.

Остановка осуществляется посредством создания тормозной силы, препятствующей движению и преобразующей кинетическую энергию самолета или автомобиля в другие виды энергии: в тепло, рассеиваемое колесными тормозами и шинами, или, например, в электроэнергию при рекуперативном торможении электромобилей.

В большинстве случаев колесные тормоза являются основным источником тормозной силы. Для автоматического управления торможением колес предназначены системы антиюзовой автоматики (САА), задачей которых является обеспечение эффективного и безопасного торможения с предупреждением блокирования колес и поддержанием коэффициента сцепления колес с опорной поверхностью (взлетно-посадочной полосой или дорожным покрытием) на максимально возможном уровне.

Начало активных исследований по созданию САА приходится на 1940-е годы, когда существенно возросшие взлетно-посадочные скорости самолетов потребовали изменений в конструкции шасси и более интенсивного использования колесных тормозов. Проблемы надежности конструкции шасси при переходе на трехопорную схему шасси с поворотным носовым колесом были успешно решены М. В. Келдышем в 1945 г. в ставшей классической работе «Шимми носового колеса трехопорного шасси» [1]. Проблема же создания высокоэффективных систем торможения не теряет своей актуальности и сегодня.

Необходимость автоматизации управления процессом торможения особенно актуальна в авиации, поскольку летчику во время посадки, находясь в кабине на значительном расстоянии от тормозных колес, практически невозможно определить состояние колес и эффективно управлять тормозным моментом. Поэтому первые САА появились в авиации (первая механическая САА была разработана корпорацией «Boeing» в 1947 г.), хотя изучением

движения колес автомобилей ученые занимались еще с конца XIX в.

На коммерческий автомобильный рынок САА, более известные как антиблокировочные системы (АБС), вышли лишь в 1966 г. (модифицированный вариант авиационной САА Maxaret фирмы «Dunlop» был установлен на первом серийном полноприводном автомобиле Jensen FF), но из-за технической сложности и высокой стоимости эти системы получили признание и стали входить в стандартную комплектацию автомобилей только с середины 1980-х годов. На сегодняшний день более половины производимых автомобилей оборудуются САА, которые, однако, постепенно утрачивают собственное значение и все в большей степени рассматриваются как часть общей системы безопасности автомобиля, включающей также антипробуксовочную систему, систему курсовой устойчивости и другие системы.

Динамика торможения

В общем случае динамика колеса может быть описана системой нелинейных дифференциальных уравнений, отражающих его упругие и демпфирующие свойства при изменении нагрузки на колесо и проезде неровностей [2]:

$$\begin{cases} M_{сц} = [F_N + F_д][R_0 - \delta_{пн}](\mu + \Delta\varphi); \\ \Delta\ddot{\varphi} = -\dfrac{J_к + J}{J_к J}[C_\varphi \Delta\varphi + q_\varphi \Delta\dot{\varphi}] + \dfrac{1}{J} M_{сц} + \dfrac{1}{J_к} M_т; \\ \dot{\omega}_к = \dfrac{1}{J_к}[C_\varphi \Delta\varphi_к + q_\varphi \Delta\dot{\varphi} - M_т], \end{cases} \quad (1)$$

где $M_{сц}$ — момент сцепления колеса с опорной поверхностью, Н · м; F_N — нормальная реакция пневматика колеса на обжатие, Н; $F_д$ — демпфирующая составляющая реакции пневматика колеса на обжатие, Н; $\delta_{пн}$ — осредненное по длине контакта обжатие пневматика, м; R_0 — радиус необжатого пневматика колеса, м; $\Delta\varphi$ — угол закручивания наружной части пневматика относительно центральной, рад; $\omega_к$ — угловая скорость вращения центральной части пневматика и обода колеса (измеряемая датчиком скорость вращения колеса), рад/с; $J_к$ — момент инерции центральной части пневматика колеса с ободом и тормозом, Н · м · с^2; J — момент инерции наружной части пневматика колеса, Н · м · с^2; C_φ — тангенциальная жесткость пневматика колеса, Н · м; q_φ — коэффициент демпфирования пневматика колеса в тангенциальном направлении, Н · м · с; $M_т$ — тормозной момент на колесе, Н · м; μ — коэффициент сцепления пневматика колеса с опорной поверхностью.

Без приложения тормозного момента угловая скорость колеса определяется поступательной скоростью его оси, и тормозная сила практически отсутствует в силу малости коэффициента трения качения.

Если к колесу прикладывается тормозной момент, то колесо начинает двигаться с некоторым проскальзыванием относительно опорной поверхности и сцепление колеса с опорной поверхностью в данном случае можно рассматривать как трение скольжения [2]. При рассмотрении движения колеса большее практическое значение имеет не скорость колеса, а его скольжение s (относительное проскальзывание), которое при постоянной нормальной нагрузке на колесо определяет коэффициент сцепления колеса с поверхностью ВПП. Скольжение представляет собой относительную разность поступательной скорости оси колеса и тангенциальной скорости точек на наружной поверхности колеса [2]:

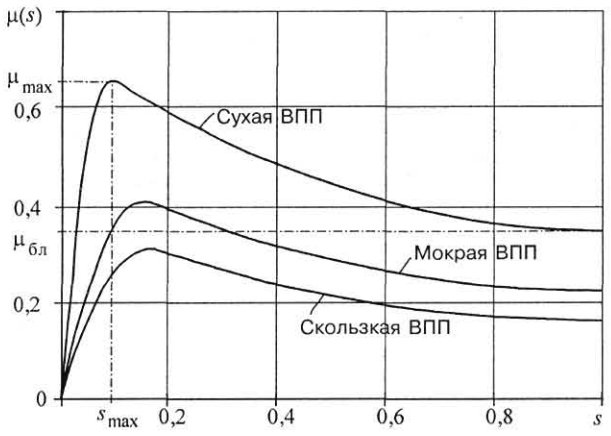
$$s = 1 - \frac{V_x}{V_\tau}, \quad (2)$$

где V_x — поступательная скорость колеса, т. е. линейная скорость оси колеса относительно опорной поверхности, м/с; V_τ — тангенциальная скорость колеса, которой обладает точка, расположенная на наружной поверхности колеса, м/с.

Зависимость коэффициента сцепления μ от скольжения колеса s для различных состояний твердой опорной поверхности приведена на рис. 1 [2, 3, 4]. Свободному качению колеса соответствует нулевое скольжение, а полному блокированию (юзу) соответствует $s = 1$. Эта зависимость является основной нелинейностью в контуре управления САА и во многом определяет специфику систем данного типа. Максимальный коэффициент сцепления обеспечивается при значении тормозного момента, равном проворачивающему моменту (моменту сцепления колеса с опорной поверхностью) и зависит от многих факторов, включая материал и состояние опорной поверхности, давление в шине и скорость колеса. Движение колеса на правом склоне характеристики сцепления неустойчиво, и при достижении максимального коэффициента сцепления μ_{max} начинается быстрое увеличение скольжения колеса и уменьшение коэффициента сцепления. Вопреки распространенному мнению, торможение с блокированными колесами (юзом) не является наиболее эффективным торможением. При юзе уменьшается не только продольный коэффициент сцепления (до значения $\mu_{бл}$), но и поперечный коэффициент сцепления, что помимо увеличения тормозного пути приводит также к резкому снижению устойчивости и заносу самолета или автомобиля.

Зависимость коэффициента сцепления от скорости колеса определяется экспериментально, при этом параметры этой зависимости непостоянны и зависят от типа шины и давления в ней, от состояния протектора пневматика и скорости колеса. Обычно при моделировании процесса торможения учитывается влияние на характерные точки зависимости $\mu(s)$ (μ_{max}, $\mu_{бл}$, s_{max}) нормированной нагрузки на колесо и путевой скорости движения оси колеса [2].

При рассмотрении систем торможения и оценке эффективности САА помимо описания динамики



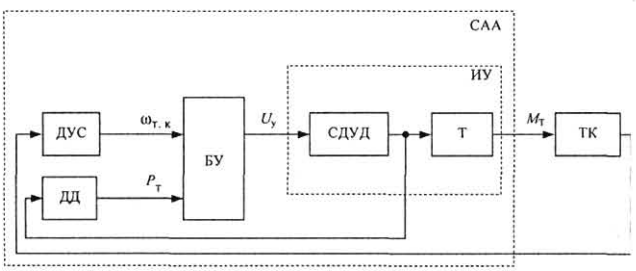
■ ***Рис. 1.*** *Зависимость коэффициента сцепления от скольжения для различных состояний твердой опорной поверхности (взлетно-посадочной полосы)*

колеса необходимо также учитывать динамику и нелинейности исполнительного устройства (гидравлического или пневматического), нестационарность параметров системы торможения, наличие случайных возмущений в виде неровностей опорной поверхности и флуктуаций коэффициента сцепления.

Динамика исполнительного устройства при моделировании автомобильных САА обычно аппроксимируется апериодическим звеном первого порядка с чистым запаздыванием. Точность такой аппроксимации достаточна, поскольку длина трубопроводов и инерционность тормозных дисков автомобиля невелики.

В отличие от автомобильных, авиационные тормозные системы (рис. 2) имеют гораздо более мощные гидроусилители, длинные гидролинии и массивные тормоза, см. 3-ю сторону обложки. Поэтому математическая модель исполнительного устройства авиационных САА должна включать описание динамики движения жидкости в гидролиниях, переменную жесткость силовых тормозных цилиндров и гистерезис многодискового фрикционного тормоза.

Кроме того, при анализе эффективности авиационной САА следует учитывать взаимное влияние динамических параметров контура управления САА и летательного аппарата: при работе тормозной системы всегда существует опасность возникновения резонансных раскачиваний самолета по тангажу и автоколебаний стойки шасси и колесной тележки, что сопровождается резким снижением эффективности торможения и приводит к разрушению элементов конструкции самолета.



■ ***Рис. 3.*** *Обобщенная структурная схема системы антиюзовой автоматики*

Несмотря на многообразие подходов и технических решений, можно составить обобщенную структуру современной САА, которая состоит из перечисленных ниже основных структурных элементов (рис. 3) [2].

• Датчики первичной информации поставляют в электронный блок управления *БУ* информацию о параметрах движения объекта управления; обычно в качестве датчиков первичной информации используются датчики угловой скорости *ДУС*, измеряющие угловую скорость тормозных колес $\omega_{т.к}$, и датчики тормозного давления *ДД*, измеряющие тормозное давление $P_т$.

• Для вычисления скольжения тормозного колеса необходима информация о продольной скорости движения оси колеса или угловой скорости нетормозного колеса («свободной скорости»). В большинстве случаев эта информация отсутствует, и для ее оценки используются специальные алгоритмы, определяющие свободную скорость исходя из угловой скорости и ускорения тормозных колес. Также возможны варианты оценки свободной скорости с помощью датчиков угловой скорости нетормозного колеса на носовой стойке шасси самолета или акселерометра, измеряющего горизонтальное ускорение самолета или автомобиля (такое решение довольно широко применяется в автомобильных САА).

• На основе текущей информации о динамике колес и тормозном давлении блок управления в соответствии с принятым законом управления вырабатывает сигнал управления $U_у = f(U_{\omega_{т.к.}}, U_{P_т})$, управляющий работой исполнительного устройства *ИУ*.

• Исполнительное устройство формирует тормозной момент $M_т$, прикладываемый к тормозному колесу, в соответствии с сигналом управления $U_у$; *ИУ* обычно состоит из системы дистанционного управления давлением *СДУД* и фрикционного тормоза *Т*.

Специфика САА

При разработке регуляторов САА возникают определенные сложности, не типичные для обычных систем регулирования и ограничивающие возможности классических методов синтеза регулятора.

САА отличается от классических следящих систем, систем стабилизации и систем программного управления невозможностью непосредственного и точного формирования заданного значения регулируемой координаты вследствие невозможности для регулятора САА определить параметры характеристики сцепления и измерить текущее

значение коэффициента сцепления колеса с опорной поверхностью. Для формирования заданного значения скольжения тормозного колеса необходимо иметь информацию о свободной скорости, которая в большинстве случаев недоступна и может быть оценена лишь приближенно.

САА должна работать в области максимума характеристики сцепления с минимизацией времени нахождения тормозного колеса на правом склоне кривой $\mu(s)$ для обеспечения поперечной устойчивости колес и снижения износа шин.

Поведение колеса на правом склоне характеристики сцепления неустойчиво, соответственно для предотвращения блокирования колеса требуется высокое быстродействие регулятора САА.

Система функционирует в условиях высокого уровня внешних возмущений, имеющих случайный характер. Параметры объекта управления САА (включая нагрузку на колесо, аэродинамические характеристики, реализуемый тормозами тормозной момент) нестационарны и меняются в процессе торможения.

Процесс торможения скоротечен (обычно не более 30 с при посадке самолета), а следующее торможение часто происходит при других параметрах объекта управления и характеристиках опорной поверхности.

В большинстве случаев регулятору САА доступны для измерения только две координаты: угловая скорость тормозного колеса и давление в гидросистеме. При этом высокий уровень случайных возмущений ограничивает возможности использования производных угловой скорости второго и более высоких порядков. В автомобильных САА в связи с широким распространением других систем автоматики (таких как круиз-контроль, автоматическое управление подвеской и двигателем) могут быть доступны и другие координаты, повышающие точность оценки коэффициента сцепления и скольжения. В авиастроении, однако, обоснованием для установки дополнительных датчиков может быть только очень существенное повышение эффективности САА при их использовании.

Синтез регулятора САА на основе нечеткой логики

САА относятся к системам управления, для которых можно создать в достаточной степени адекватную математическую модель объекта управления, но сложность этой модели, а также недостаточность и неопределенность доступной регулятору информации не позволяют использовать классические аналитические методы синтеза регулятора. В то же время в последние десятилетия активно развиваются методы интеллектуального управления, включающие теорию нечетких множеств и нечеткую логику, теорию возможностей, искусственные нейронные сети и генетические алгоритмы.

Особенно удачно в задачах управления сложными и нелинейными системами в условиях неопределенности зарекомендовало себя управление с использованием нечеткой логики, позволяющее реализовать простые, но робастные решения, покрывающие широкий диапазон возможных изменений параметров объекта и справляющиеся со значительными возмущениями [7].

Одним из основополагающих принципов, приведших к появлению интеллектуальных технологий управления, и в частности нечеткого управления, является «принцип несовместимости», согласно которому «...сложность системы и точность, с которой ее можно анализировать, обратно пропорциональны в первом приближении», и, следовательно, «... по мере того, как сложность возрастает, точные утверждения теряют значимость, а значимые утверждения теряют точность» [5, 6].

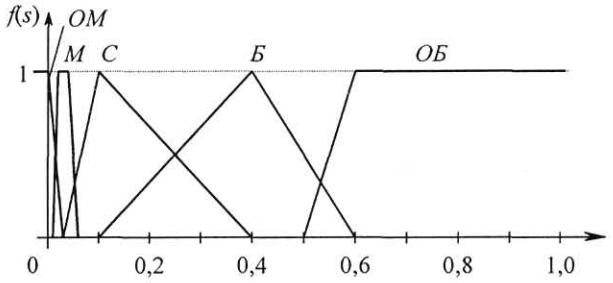
Алгоритмы управления большинства САА построены в основном на основе экспериментального опыта и результатов моделирования и испытаний. Лингвистическое представление закона управления в этом случае наиболее эффективно, делая его прозрачным для эксперта и позволяя легко изменять и подстраивать алгоритм работы системы. В отличие от других систем управления, основанных на знаниях, нечеткой логике, для того чтобы охарактеризовать определенное состояние объекта, требуется меньшее количество правил управления, так как сам алгоритм нечеткого вывода производит интерполяцию для каждой конкретной ситуации.

В случае авиационных САА, учитывая динамические характеристики исполнительного устройства, целесообразно использовать двухконтурную схему регулятора САА, в которой быстродействующий контур резко сбрасывает тормозное давление при большом скольжении колеса, а медленнодействующий контур подстройки тормозного давления представляет собой нечеткий регулятор, сигнал управления на выходе которого определяет скорость изменения тормозного давления.

Для формирования закона управления входные координаты регулятора САА (обычно это скольжение и угловое ускорение тормозного колеса) разбиваются на несколько нечетких переменных, различные сочетания которых характеризуют текущее состояние объекта управления, а также позволяют судить о соответствующем изменении сигнала управления.

Параметры функций принадлежности нечетких переменных вследствие сложности объекта управления определяются с привлечением знаний эксперта и по результатам моделирования процесса торможения. Дальнейшая оптимизация функций принадлежности может осуществляться с помощью методов численной оптимизации, причем, принимая во внимание сложность объекта управления и наличие существенных нелинейностей в контуре управления, для этой цели наиболее подходящим представляется использование генетических алгоритмов.

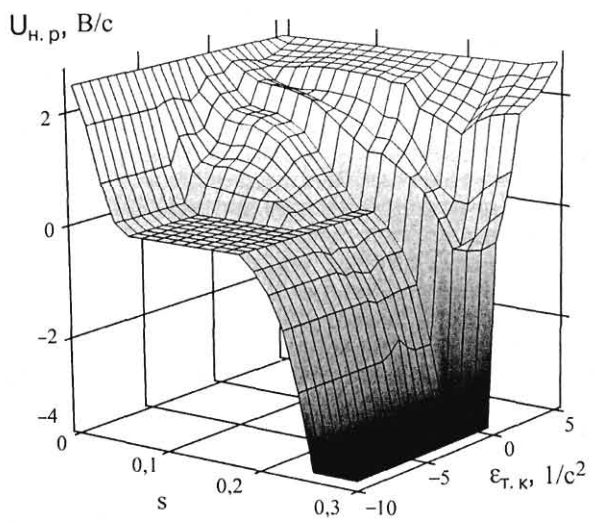
Таким образом, лингвистическая переменная скольжения может быть разбита на пять нечетких переменных: «очень малое», «малое», «среднее», «большое», «очень большое». Очень малое скольже-



■ ***Рис. 4.*** *Функции принадлежности нечетких переменных скольжения (ОМ — очень малое; М — малое; С — среднее; Б — большое; ОБ — очень большое)*

ние ($s \approx 0$) соответствует качению колеса без тормозного момента. Малое скольжение соответствует диапазону, в котором находится оптимум скольжения при высоком коэффициенте сцепления ($s \approx \approx 0{,}1$), а среднее скольжение ($s \approx 0{,}18$) — оптимуму при малых коэффициентах сцепления. Большое скольжение соответствует диапазону ($s \approx 0{,}3$), в котором следует сбрасывать давление, чтобы вернуть скольжение к оптимальному значению. Очень большое скольжение ($s > 0{,}35$) свидетельствует о юзе колеса и требует максимального сброса давления. Функции принадлежности значений нечетких переменных скольжения тормозного колеса представлены на рис. 4.

Аналогичным образом определяются нечеткие переменные углового ускорения колеса и сигнала управления регулятора. База правил регулятора составляется экспертом и описывает различные ситуации, в которых может находиться тормозное колесо, и необходимый при этом сигнал управления, который будет выводить колесо на максимальное значение коэффициента сцепления. Полученная в результате зависимость сигнала управления



■ ***Рис. 5.*** *Сигнал управления нечеткого регулятора САА $U_{н.р}(s, \varepsilon_{т.к})$*

■ ***Таблица***

Состояние ВПП	Максимальное значение коэффициента сцепления	Реализуемое значение коэффициента сцепления	
		Среднестатистическая оценка [10]	Моделирование САА с нечетким регулятором
Сухо	0,65–0,80	0,2–0,3	0,35
Мокро	0,4–0,5	0,07–0,1	0,26–0,32
Слякоть, мокрый снег	0,30–0,35	0,05–0,07	0,21–0,24

нечеткого регулятора от скольжения и углового ускорения тормозного колеса показана на рис. 5.

Как показывает практика, во многих случаях алгоритмы управления, построенные на основе нечеткой логики, достаточно робастны и обеспечивают стабильное качество регулирования при значительных изменениях параметра объекта управления и внешних воздействиях. Однако при функционировании САА характеристики сцепления меняются очень существенно, и для расширения диапазона эффективной работы регулятора САА необходимо использовать механизмы адаптации к изменению коэффициента сцепления. Наиболее подходящим в данном случае может быть применение метода декомпозиции закона управления [8, 9], при котором для существенно различных состояний поверхности ВПП (сухой, мокрой, скользкой) синтезируется свой нечеткий регулятор или своя база правил управления. Для вычисления результирующего сигнала управления также используется система нечеткого вывода, которая в зависимости от состояния ВПП отдает приоритет тому или иному нечеткому регулятору. Косвенная и приближенная оценка состояния ВПП может выполняться по величине сигнала управления регулятора САА, осредненного за предшествующий отрезок времени. Благодаря использованию нечеткой системы для выбора наиболее подходящего нечеткого регулятора обеспечивается плавное и постепенное переключение с одного регулятора на другой, а регулятор САА сохраняет высокую эффективность в различных условиях торможения.

Как показывают результаты моделирования, регулятор САА с использованием нечеткой логики позволяет не только повысить средний реализуемый коэффициент сцепления, но и существенно снизить количество юзов колес даже при торможении в плохих условиях сцепления, в том числе и на ВПП с переменным коэффициентом сцепления (на «миксте»).

Для иллюстрации эффективности нечеткого регулятора САА в таблице приведены значения среднего за время послепосадочного пробега коэффициента сцепления, реализуемого САА, используемые в работе [10] для приблизительных расчетов длины пробега различных моделей существующих самолетов и полученные в результате моделирования процесса торможения с нечетким регулятором для самолета типа Ил-86.

Выводы

Применение нечеткой логики в авиационных системах антиюзовой автоматики повышает эффективность системы торможения самолета, обеспечивая высокий коэффициент сцепления и снижая число юзов колес в различных условиях торможения. Тем самым сокращается тормозной путь, повышается устойчивость самолета на пробеге и уменьшается износ пневматиков тормозных колес, благодаря чему повышается безопасность посадки самолета, а также увеличивается ресурс колес шасси и тормозной системы в целом, что позволяет говорить и о повышении экономической эффективности полетов.

Литература

1. **Келдыш М. В.** Шимми переднего колеса трехопорного шасси // Тр. ЦАГИ им. Н.Е. Жуковского. — Вып. 564. — М.: Изд-во Бюро новой техники НКАП, 1945. — 34 с.
2. **Богачева Н. А., Жуков А. Д., Коновалов А. С.** Авиационные системы антиюзовой автоматики: Учеб. пособие. — СПб.: СПбГУАП, 1999. — 84 с.
3. **Yager T.** Aircraft and Ground Vehicle Winter Runway Friction Assessment. — NASA Technical Memorandum NASA/TM. — 1999-209142, 1999. — 12 p.
4. **Von Altrock C.** Fuzzy Logic in Automotive Engineering//Circuit Cellar. — 1997. — Issue 88. — P. 12–27.
5. **Заде Л.** Понятие лингвистической переменной и его применение к принятию приближенных решений: Пер. с англ. — М.: Мир, 1976. — 165 с.
6. **Ерофеев А. А., Поляков А. О.** Интеллектуальные системы и технологии // Матер. второй междунар. конф. «Логико-лингвистическое управление динамическими объектами DOLLC'99». 21–25 июня 1999 г. — СПб., 1999. — С. 20–25.
7. **Васильев С. Н.** От классических задач регулирования к интеллектному управлению. // Изв. РАН. Теория и системы управления. — 2001. — № 2. — С. 5–22.
8. **Бураков М. В., Коновалов А. С.** Декомпозиция закона управления // Управление в условиях неопределенности / Под ред. А. Е. Городецкого. — СПб.: СПбГТУ, 2002. — 398 с.
9. **Бураков М. В.** Механизм адаптации нечеткого регулятора // Изв. РАН. Теория и системы управления. — 1998. — № 1. — С. 84–87.
10. **Гилерсон А. Г.** Эффективность реверсивных устройств при торможении самолетов. — М.: Машиностроение, 1995. — 192 с.
11. **Layne J. R., Passino K. M., Yurkovich S.** Fuzzy Learning Control for Antiskid Braking Systems // IEEE Transactions on Control Systems Technology. — 1993. — Vol. 1. — N 2. — P. 122–129.
12. **Takahashi H., Ishikawa Y.** Antiskid Brake Control System Based on Fuzzy Inference: U. S. patent US4842342, 1989.
13. **Yen Edge C., Roan G. K., Ton J. H.** Fuzzy Controller for Anti-Skid Brake System: U.S. patent US5416709, 1995.
14. **Mertens A.** Anti Lock System for a Vehicle Electromechanical Brake System Based on a Fuzzy Controller: Patent EP0985586, 2000.
15. **Becker R., Cao C.-T., Belzner U., Moeler T.-W., Lieberoth-Leden B.** System for Controlling Brake Pressure Based on Fuzzy Logic Using Steering Angle and Yaw Speed: U.S. patent US5634698, 1997.
16. **Wiel C. T.** Fuzzy Logic Antiskid Control System for Aircraft: U.S. patent US6088646, 2000.

УДК 621(075.8)

ЛОГИЧЕСКИ ПРОЗРАЧНЫЕ СЕТИ

А. Е. Городецкий,
д-р техн. наук
И. Л. Тарасова,
канд. техн. наук
Институт проблем машиноведения РАН, Санкт-Петербург

Рассматриваются методы формирования логически прозрачных сетей. Выделяются следующие классы нейронных сетей: логико-лингвистические и логико-вероятностные. Анализируются условия и проблемы создания таких сетей.

Methods of formation of logically transparent networks are considered The following classes of neural networks are entered: logical-linguistic and logical-probabilistic. It is analyzed conditions and problems of creation of such networks.

Введение

Одним из недостатков нейронных сетей (НС), с точки зрения многих пользователей, является отсутствие объяснительного компонента получаемых результатов, т. е. из обученной НС невозможно извлечь алгоритм решаемой задачи. Конечным результатом работы алгоритма обучения является некоторый вектор весов межнейронных связей сети, в котором, в соответствии с принятым коннекционистским подходом к формализации НС, и сосредоточены все знания. Каждая компонента этого вектора представляет собой некоторое число, которое никаким образом невозможно интерпретировать.

Формирование логических нейронных сетей

Красноярской группой «НейроКомп» была сформулирована идея *логически прозрачных сетей*, т. е. сетей, на основе структуры которых можно построить вербальное описание алгоритма получения ответа [1]. Это достигается при помощи специальным образом построенной процедуры уменьшения сложности (прореживания) сети.

Другой подход к построению логически прозрачных сетей заключается в том, что в сети используются только два типа нейронов, осуществляющих логические операции отрицания и дизъюнкции или отрицания и конъюнкции, либо операции сложения и умножения по модулю «2». Такие нейроны можно назвать логическими. Входная информация в этом случае должна быть преобразована в логическую форму или набор логических переменных x_i, из которых можно создать фундаментальный вектор логической системы [2]:

$$\mathbf{F} = /x_1\, x_2 \ldots x_n\, x_1 \otimes x_2\, x_1 \otimes x_3 \ldots x_{n-1} \otimes x_n\, x_1 \otimes x_2 \otimes x_3 \ldots$$
$$\ldots x_{n-2} \otimes x_{n-1} \otimes x_n \ldots x_1 \otimes x_2 \otimes x_3 \otimes x_4 \ldots$$
$$\ldots x_{n-3} \otimes x_{n-2} \otimes x_{n-1} \otimes x_n \ldots x_1 \otimes x_2 \otimes \ldots \otimes x_{n-1} \otimes x_n /^T, \quad (1)$$

по которому можно вычислить любую логическую функцию:

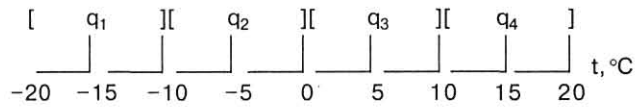
$$f_i = C_i F, \quad (2)$$

где C_i — идентификационная строка, состоящая из комбинации 0 и 1 и имеющая размерность вектора **F**, например,

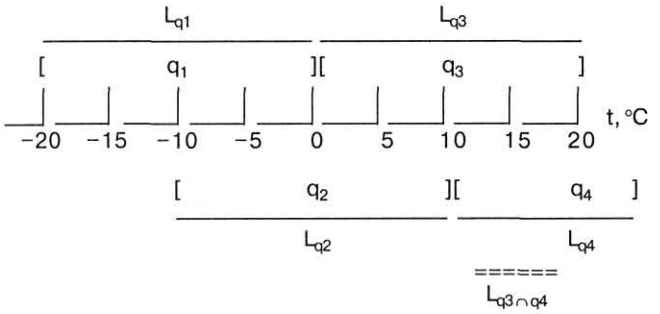
$$C_i = /0\,0\,0\,1\,1\,0 \ldots\ldots 0/. \quad (3)$$

При этом обучение полученной логической нейронной сети будет состоять в поиске идентификационных строк (или связей между нейронами).

Логические переменные x_i вектора **F** можно получить путем квантования входных величин и присвоения полученным квантам q_i имен логических переменных x_i, принимающих значения истина «1» или ложно «0». Например, если входная переменная — температура T может изменяться в пределах от −20 до +20 °С, то, введя квант в 10 °С, можно весь диапазон изменения температуры разбить на четыре кванта: $q_1 = [-20, -10]$, $q_2 = [-10, 0]$, $q_3 = [0, +10]$, $q_4 = [+10, +20]$ (рис. 1). Тогда кванту q_1 можно присвоить имя x_1 {очень холодно}, кванту q_2 присвоить имя x_2 {холодно}, кванту q_3 присвоить имя x_3 {прохладно} и кванту q_4 — x_4 {тепло}. Теперь, если, например, температура на входе $t = +5$ °С , то значения логических переменных будут следующими: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 0$. Если кванты, образующие смежные логические переменные, например «холод-



■ ***Рис. 1.*** *Получение логических переменных*



■ ***Рис. 2.*** *Получение лингвистических переменных*

но» и «прохладно», будут частично перекрываться («холодно» — [−15, +5], а «прохладно» — [−5, +15] (рис. 2), то этим квантам можно также поставить в соответствие логические переменные таких же наименований. При этом степень принадлежности входной величины x_i к тому или иному кванту (к той или иной логической переменной) характеризуется функцией принадлежности $\mu(x_i)$ подобно тому, как это делается при фазификации [1]. Полученные таким образом переменные принято называть *лингвистическими*.

Поэтому в дальнейшем нейронные сети, построенные из логических нейронов и использующие на входе лингвистические переменные, будем называть *логико-лингвистическими нейронными сетями*.

Кроме того, вместо функции принадлежности можно ввести другую характеристику степени принадлежности входной величины к тому или иному кванту. Наиболее естественно в качестве такой характеристики выбрать вероятность, вычисляемую как отношение перекрывающегося промежутка $L_{qi \cap qj}$ к протяженности кванта L_{qi} (см. рис. 2):

$$P(x_i = 1) = L_{qi}/L_{qi \cap qj}. \quad (4)$$

Полученные описанным способом переменные в дальнейшем будем называть *логико-вероятностными*, а соответствующие им нейронные сети — *логико-вероятностными нейронными сетями*.

Уменьшение избыточности логических нейронных сетей

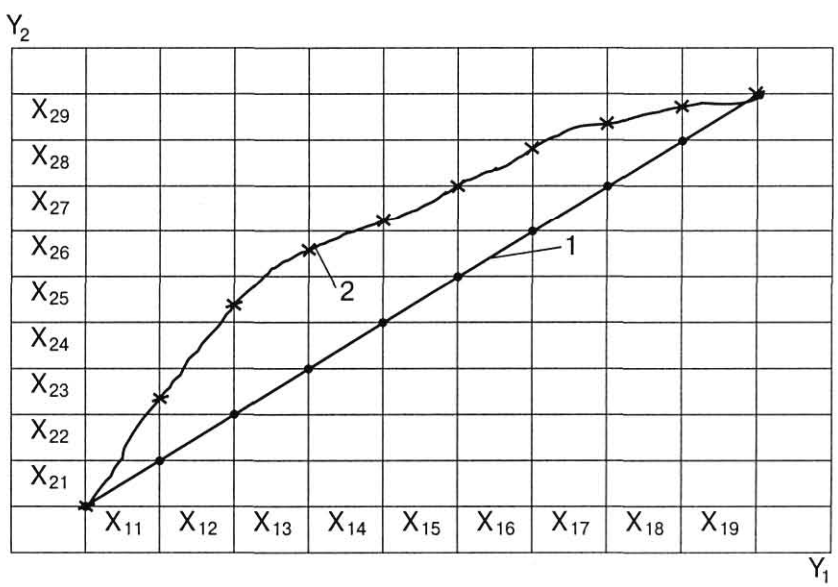
Введенные выше сети являются логически прозрачными, но имеют значительную избыточность, которая может быть уменьшена, если часть логических функций f_i известна заранее и нет необходимости поиска соответствующих им идентификационных строк C_i. Кроме того, уменьшить избыточность таких НС можно также, если известны функциональные зависимости между входными величинами.

Пусть, например, имеется функциональная зависимость входных величин $Y_2(Y_1)$, показанная на рис. 3.

Для перевода линейной зависимости *1*, представленной на рис. 3, в систему логических функций (уравнений), вначале надо, исходя из требуемой точности δ_i представления входных величин, разбить диапазоны их изменения: $D_1 = Y_{1max} - Y_{1min}$ и $D_2 = Y_{2max} - Y_{2min}$ на N_1 и N_2 квантов:

$$q_{1i} = [Y_{1min} + i\delta_i;\ Y_{1min} + (i+1)\delta_i],\ i = 0, 1, 2, \ldots, N_1; \quad (5)$$

$$q_{2j} = [Y_{2min} + j\delta_j;\ Y_{2min} + (j+1)\delta_j],\ j = 0, 1, 2, \ldots, N_2. \quad (6)$$



■ ***Рис. 3.*** *Функциональные зависимости входных величин*

■ ***Таблица 1***

	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}
X_{11}	1	0	0	0	0	0	0	0	0
X_{12}	0	1	0	0	0	0	0	0	0
X_{13}	0	0	1	0	0	0	0	0	0
X_{14}	0	0	0	1	0	0	0	0	0
X_{15}	0	0	0	0	1	0	0	0	0
X_{16}	0	0	0	0	0	1	0	0	0
X_{17}	0	0	0	0	0	0	1	0	0
X_{18}	0	0	0	0	0	0	0	1	0
X_{19}	0	0	0	0	0	0	0	0	1

■ ***Таблица 2***

	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}
X_{11}	1/1	1/1	1/0,5	0	0	0	0	0	0
X_{12}	0	0	1/0,5	1/1	1/0,2	0	0	0	0
X_{13}	0	0	0	0	1/0,8	1/0,4	0	0	0
X_{14}	0	0	0	0	0	1/0,6	1/0,5	0	0
X_{15}	0	0	0	0	0	0	1/0,6	1/0,1	0
X_{16}	0	0	0	0	0	0	0	1/0,8	0
X_{17}	0	0	0	0	0	0	0	1/0,1	1/0,5
X_{18}	0	0	0	0	0	0	0	0	1/0,7
X_{19}	0	0	0	0	0	0	0	0	1/0,1

Затем в соответствии с рис. 3 можно составить табл. 1 для вывода логических функций умножения по модулю «два» или конъюнкции между логическими переменными X_{1i} и X_{2j}, которым соответствуют кванты q_{1i} и q_{2j}:

$$Y \Rightarrow X_{1i} \otimes X_{2j}; \quad (7)$$

$$Y \Rightarrow X_{1i} \otimes X_{2j}, \quad (8)$$

либо — импликаций между X_{1i} и X_{2j}:

$$Y \Rightarrow X_{1i} \rightarrow X_{2j}; \quad (9)$$

$$Y \Rightarrow \neg X_{1i} \vee X_{2j}; \quad (10)$$

$$Y \Rightarrow X_{1i} \otimes X_{2j} \otimes X_{1j} \otimes 1. \quad (11)$$

Выбор конкретной логической функции из набора (7) — (11), отражающей приведенную на рис. 3 линейную функцию $Y_2(Y_1)$, зависит от содержательной части решаемой задачи. В большинстве случаев указанную зависимость можно представить в виде системы из логических уравнений (11).

Очевидно, что все логические уравнения типа (7)–(11) могут быть переведены в стандартную форму (2).

В случае нелинейной зависимости $Y_2(Y_1)$ (кривая *2* на рис. 3) нельзя каждому кванту q_{1i} поставить в соответствие один квант q_{2j} (см. также рис. 2). При этом какому-либо кванту q_{1i} может соответствовать: либо один полный квант q_{2j}, либо один неполный квант q_{2j}, либо два и более полных кванта q_{2j}, q_{2j+1}, либо один или несколько полных квантов q_{2j}, q_{2k} и один или несколько неполных квантов q_{2j-1}, q_{2k+1} и т. д. (табл. 2). При этом, если такие нелинейные зависимости, представленные на рис. 3, отображать, как и прежде, только системой логических уравнений, т. е. без учета степени принадлежности какого-либо кванта q_{2j} кванту q_{1i}, то будет вводиться большая погрешность отображения, зависящая от вида кривой *2*. Улучшить эту ситуацию можно путем введения либо вероятности $P\{X_{1i} \rightarrow X_{2j} = 1\}$, либо функции принадлежности $\mu(Y)$ ($Y \Leftrightarrow X_{1i} \Leftrightarrow X_{2j}$), вычисляемой подобно вероятности в уравнении (4). Вычисленные величины приведены в табл. 2. (в знаменателях). Например, $P\{X_{12} \rightarrow X_{23} = 1\} = \mu(X_{12} \rightarrow X_{23}) = 0{,}5$.

Заключение

Нами рассмотрена простейшая монотонная гладкая возрастающая зависимость. Замена более сложных функциональных зависимостей на логические функции, очевидно, будут приводить к большим погрешностям, уменьшение которых потребует дополнительных усилий. Тем не менее, приведенные примеры показывают, что имеется принципиальная возможность построения логически прозрачных сетей, позволяющих получать решения многих практических задач с достаточной точностью при некотором увеличении их размерности и интерпретировать и объяснять получаемые результаты. Однако построение таких сетей чаще всего требует большого числа экспериментов и не всегда экономически выгодно.

Литература

1. **Комарцова Л. Г., Максимов А. В.** Нейрокомпьютеры. — М.: МГТУ им. Н. Э. Баумана, 2002. — 317 с.
2. **Городецкий А. Е., Дубаренко В. В., Ерофеев А. А.** Алгебраический подход к решению задач логического управления // А и Т. — 2000. — № 2. — С. 127–138.

УДК 681.51:303.732+519.76

СТРУКТУРНО–ЦЕЛЕВОЙ АНАЛИЗ В УПРАВЛЕНИИ СИСТЕМАМИ ПРОИЗВОДСТВЕННОЙ СФЕРЫ

Л. М. Лукьянова,
кандидат технических наук
Калининградский государственный технический университет

Исследуется проблема повышения научно-технического уровня системного анализа при управлении рыбопромышленными объектами. Обсуждается возможный путь решения проблемы, основывающийся на структурно-целевой парадигме системного анализа. Рассматриваются постулаты, принципы, концепция поддержки структурно-целевого анализа и синтеза систем и ее реализация в полимодельной системе, основывающейся на логико-лингвистических формализациях.

The problem of brining scientific level of systems analysis in control for fishing industry objects up to standard is studied. Possible way of problem decision based on a structure-and-purpose paradigm of systems analysis is discussed. Postulates, principles,conception of structure-and-purpose analysis support, and its realization based on logical-and-linguistic formalizations are outlined.

Введение

При управлении сложными системами в условиях существенной неопределенности широкое применение находит системный анализ. Он способствует выяснению причин проблемного функционирования систем, неравновесности и необратимости их состояний, обоснованию путей и оценке степени решения проблем. В системах производственной сферы, в которых деятельность людей осуществляется совместно с функционированием технических подсистем, системный анализ концентрирует внимание на соотношении коллективного целеполагания, целереализации и ее результатов [1].

Научный потенциал системного анализа, относимого к «техническим системным теориям» [2] или к специфическим методологиям и методическим схемам [1, 3, 4], и опыт его применения нашли отражение в разнообразных методиках, анализ которых осуществлен в [4, 5]. Методики направлены на упорядочение анализа и снижение степени его субъективности, однако их использование не может гарантировать требуемого качества его результатов [5].

Качество конечных результатов системного анализа зависит прежде всего от результатов его начальных этапов — систем проблем и целей, получаемых в большой мере на основе «здравого смысла» осуществляющих его лиц [1]. В существенно неопределенных ситуациях неформальный анализ часто приводит к логическим ошибкам в системах проблем и целей. Ошибки целеполагания обусловливают синтез целереализующих систем (ЦРС), не способных решить возникшие проблемы. Цена ошибок может оказаться недопустимо высокой.

В этой связи роль качественной системно-аналитической информации в подготовке, принятии и исполнении программных, проектных, плановых решений в производственной сфере трудно переоценить и, учитывая заинтересованность руководства рыбной отрасли [5], представляется актуальным исследовать вопросы формализации системного анализа в рыбопромышленном секторе.

Постановка проблемы

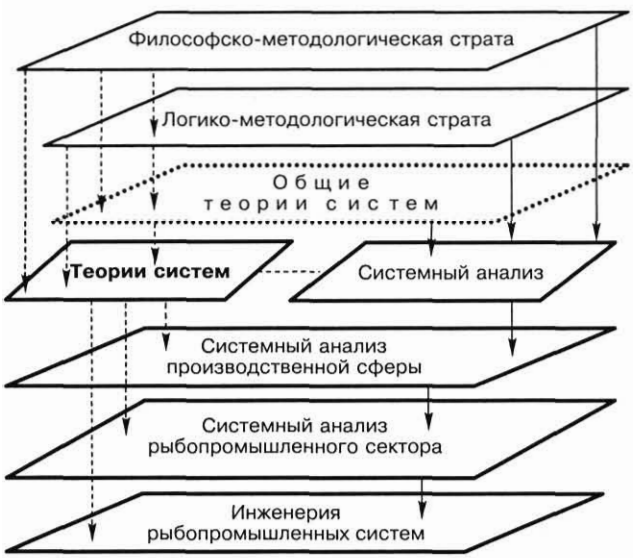
Системный анализ предполагает согласование отдельных представлений объекта в едином системном представлении путем итеративного использования неформальных и формальных методов и процедур. Однако из-за того, что он не располагает формализованными средствами получения и проверки взаимообусловленности и согласованности представлений, противоречия в одном из них, не выявленные в результате анализа, влекут противоречивость последующих. На это обращается внимание в работе [1], и это установлено практикой системного анализа [5, 6].

Однако неудовлетворительное качество результатов системного анализа обусловлено не только

слабой формализованностью, а значит неконструктивностью его специфических методов. Оно предопределено недостаточной развитостью методологических регулятивов и частных методологий системного анализа в рамках общеметодологических средств [3, 7, 8]. Это подтверждается результатами анализа систем в производственной сфере и влечет обсуждаемую проблему, в частности, в ее рыбопромышленном секторе [6].

Для постановки проблемы была проанализирована классификационная схема системных исследований из работы [2]. Предложен вариант схемы, иллюстрирующий место системного анализа рыбопромышленных объектов в среде методологических, теоретических и инженерных средств, между компонентами которой имеют место отношения конкретизации/заимствования, обозначенные сплошными/штриховыми стрелками, а системный анализ представляет собой специфическую многоуровневую надстройку над общеметодологическим и теоретическим базисами (рис. 1). Основа первого базиса — материалистическая диалектика и диалектическая логика, второго — закономерности и модели, методы моделирования и анализа систем. Число уровней непосредственно системного анализа, по нашему мнению, определяется глубиной методологических и теоретических различий, обусловленных различиями в предметных областях и классах анализируемых систем.

При системном подходе к объектам рассматривают их различные аспекты [8], а, отмечая роль целей в системном анализе организаций [1, 2, 4, 8, 10], целевой аспект не выделяют и в качестве доминирующего определяют системно-структурный [10] или системно-функциональный [9] аспект. Но цели играют ведущую роль в управлении организациями производственной сферы, и использование опирающейся на целевой аспект систем парадигмы системного анализа, дополнительно к общесистемной парадигме как «научно-философской исследовательской программе» [11], позволит повысить его конструктивность.



■ ***Рис. 1.*** *Стратификация системного исследования рыбопромышленных систем*

Системы производственной сферы создаются для достижения конечных целей, задаваемых их надсистемами, и являются средствами достижения этих целей [1]. Конечная цель в системе уточняется и конкретизируется. В случае сложной цели последняя рассматривается как система, требующая анализа. Для эффективного управления необходимо правильно анализировать сложные цели. Формируемые в ходе системного анализа системы *целей*, критериев *достижения целей*, функций *целереализующих* систем и самих *ЦРС*, структуры работ по достижению *целей* логически и семантически связаны. Логические связи определяются последовательностью этапов системного анализа, семантические — обусловлены доминированием целей, подтверждаемым их курсивным выделением.

Системы производственной сферы имеют, как правило, иерархическую структуру, вследствие чего возникающие в них сложные проблемы, выдвигаемые для решения таких проблем цели, функции ЦРС могут представляться в виде иерархических систем. Логико-семантическая формализация анализа таких систем создаст возможности определения их противоречивости, уменьшит степень субъективности их представления, число и масштаб логических ошибок в планах функционирования и программах развития систем.

Производственная сфера весьма динамична. Так, ее рыбопромышленный сектор существенно зависит от состояния сырьевой базы рыбного хозяйства, динамики складывающихся в Мировом океане ситуаций и спроса на производимую продукцию, что обусловливает ситуационную динамичность в выпуске продукции, технологическом обновлении и техническом перевооружении рыбной отрасли. Поэтому при выработке концепции решения обсуждаемой проблемы перспективна идея ситуационного управления [12], развитие и реализация которой позволит, настраиваясь на конкретную ситуацию, осуществлять моделирование слабо формализуемых начальных этапов системного анализа.

Научный аспект проблемы системного анализа в рыбопромышленном секторе производственной сферы обусловлен слишком общим характером предлагаемых системной методологией парадигм, принципов и концепций, слабой формализацией процедур моделирования отдельных системных представлений и увязки получаемых моделей в многомодельные представлениях систем, неадекватностью реальному миру логических моделей и методов формализованного анализа систем [1–11, 13–15]. Поэтому такие представления могут быть в большей степени комплексными, нежели системными, противоречивыми и неполными. Практический аспект проблемы связан с недостаточной разработанностью методических схем, общим или частным и жестким характером методик и информационно-программных средств, не обеспечивающих

эффективный анализ рыбопромышленных систем и качество его результатов [5, 6].

Сформулируем проблему системного анализа в рыбопромышленном секторе производственной сферы как *разработку повышающих его научно-технический уровень методологических, теоретических и практических средств мониторинга и анализа целеполагания и целереализации*.

Определение организационной системы производственной сферы

Понятие «система» в системном анализе является ключевым. Известно много определений системы [1, 2, 4, 8–10, 16, 17], различная степень общности которых порождает множественные интерпретации, не способствует конструктивизации анализа систем конкретных классов и может приводить к противоречивым системным представлениям.

Для формирования конструктивного определения системы производственной сферы в качестве базовых используем определения системы из работ [1, 17], дополнив их выделенными при анализе обзоров определениями системы в [2, 4], характеризующими организационные системы и необходимыми в контексте их анализа семантическими множителями: «цель», «свойство», «наблюдатель». Введя обозначения, определим систему (S_i) как *совокупность частей* ($\mathbf{Э}$), находящихся в *отношениях и связях* друг с другом (R), которая образует *определенную целостность, единство*.

Будем задавать систему как ц е л о с т н о с т ь Θ:

$$\mathbf{S_0} = <\Theta>, \qquad (1)$$

обусловливающую *закономерности системы* [17, 18] и выражающую:

— *целевую обособленность* и *единство системы со средой* $^{\text{ц-ср}}\Theta$ по соотношению «конечная_цель_системы(**Ц**)—актуальная_среда (**СР**)»;

— *целесообразные структуры системы* $^{\text{ц-стр}}\Theta$, определяемые их общеструктурными свойствами $^{\text{ц-стр}}\mathbf{C}$ по соотношению «конечная_цель_**Ц**-структура(**СТР**)» и тем самым определяющую системные свойства $^{S}\mathbf{c_j} \in {}^{S}\mathbf{C}$, j = 1, 2, ..., m_S, где m_S — число свойств, в том числе важнейшее для достижения конечной цели **Ц** *функциональное свойство* системы $^{S}\mathbf{c_1}$; или как ц е л о е:

$$\mathbf{S_1} = <\mathbf{Э}, \mathbf{R}, \mathbf{C}>, \qquad (2)$$

где $\mathbf{Э}$ — множество частей, $\mathbf{Э} = \{\mathbf{э_i}\}$, i = 2, 3, ..., $n_{сэ}$, $n_{сэ}$ — число частей, определенных на множестве свойств $\mathbf{C}$, $^{Э}\mathbf{C} = \{^{Э}\mathbf{c_j}\}$, j = 1, ..., $m_{сэ}$, $m_{сэ}$ — число свойств; $\mathbf{R}$ — множество базовых (основных и простых) отношений, k = 1, 2, ..., n_R, n_R — число отношений между $\mathbf{э_i}$, определенное на множестве свойств $^{R}\mathbf{C}$, $^{R}\mathbf{C} = \{^{R}\mathbf{c_l}\}$, $^{R}\mathbf{c_l}$ — l-е свойство отношения, l = 1, 2, ..., m_{cR}, m_{cR} — число свойств; при этом $\mathbf{R} = {}^{1}\mathbf{R} \cup {}^{2}\mathbf{R}$, ${}^{1}\mathbf{R} \cap {}^{2}\mathbf{R} = \varnothing$, ${}^{1}\mathbf{R} \neq \varnothing$, ${}^{1}\mathbf{R}$ — множество структурообразующих связей, формирующих более сложные части из менее сложных и образующих структуры системы, а ${}^{2}\mathbf{R}$ — множество неструктурных отношений.

Таким образом, в результате уточнения компонентов в выражении (2) имеем:

$$\mathbf{S_2} = <\mathbf{Э}(^{Э}\mathbf{C}); {}^{1}\mathbf{R}(^{1R}\mathbf{C}), {}^{2}\mathbf{R}(^{2R}\mathbf{C}); \; ^{\text{ц-ср}}\Theta, \; ^{\text{ц-стр}}\Theta, \; ^{S}\mathbf{C}>. \qquad (3)$$

При определении целесообразных структур системы будем учитывать, что в системе могут использоваться или создаваться различные (**i**-е) структуры, в построении которых участвуют, возможно, не все, а лишь некоторые элементарные целостности $^{i}\mathbf{Э}(^{i\text{Э}}\mathbf{C})$, $^{i}\mathbf{Э}(^{i\text{Э}}\mathbf{C}) \subseteq \mathbf{Э}(^{Э}\mathbf{C})$ и связи $^{i1}\mathbf{R}(^{i1R}\mathbf{C})$, $^{i1}\mathbf{R}(^{i1R}\mathbf{C}) \subseteq {}^{1}\mathbf{R}(^{1R}\mathbf{C})$, определенные в соответствующих пространствах свойств [в предельном случае имеет место совпадение множеств: $^{i}\mathbf{Э}(^{i\text{Э}}\mathbf{C}) = {}^{Э}(^{Э}\mathbf{C})$, $^{i1}\mathbf{R}(^{i1R}\mathbf{C}) = {}^{1}\mathbf{R}(^{1R}\mathbf{C})$]. Будем задавать **i**-ю структуру системы, реализующую цель **Ц**, ^{i}(**Ц** – **СТР**), следующей тройкой компонентов:

$$^{i}(\mathbf{Ц} - \mathbf{СТР}) = <^{i1}\mathbf{R}(^{i1R}\mathbf{C}); {}^{i}\mathbf{Э}(^{i\text{Э}}\mathbf{C}); {}^{i(\text{ц-стр})}\mathbf{C}>, \qquad (4)$$

где $^{i(\text{ц-стр})}\mathbf{C}$ — обозначает множество свойств структуры, $^{i(\text{ц-стр})}\mathbf{C} = \{^{i(\text{ц-стр})}\mathbf{c_j}\}$, $^{i(\text{ц-стр})}\mathbf{c_j}$ — **j**-е свойство, j = 1, 2, ..., f, f — количество свойств; $^{i1R}\mathbf{C}$ — обозначает множество свойств структурообразующих отношений, $^{i1R}\mathbf{C} = \{^{i1R}\mathbf{c_j}\}$, $^{i1R}\mathbf{c_j}$ — j-е свойство, j = 1, 2, ..., p, p — количество свойств.

Для выражения статического аспекта организационных систем определения (1)—(4) обладают большей конструктивностью по сравнению с определением (2). На пути формализации семантики компонентов в (1)—(4) может быть привнесена еще большая конструктивность и устранена некоторая неопределенность.

Выражения (1)—(4) полагаются объективно характеризующими систему. Аналитик (А) как наблюдатель определяет систему А-субъективно, например:

$$\mathbf{S_{13}} = <\mathbf{Э_A}, \mathbf{R_A}, \mathbf{C_A}>. \qquad (2')$$

Состав и содержание определяющих систему компонентов с течением времени могут изменяться. Во-первых, меняются сами системы. Случай изменения $\mathbf{Э}$, $\mathbf{R}$, $^{\text{ц-ср}}\Theta$, $^{\text{ц-стр}}\Theta$ при неизменных $^{S}\mathbf{C}$ свидетельствует об эволюции и самоорганизации системы, а «революционное» преобразование системы изменяет ее как целостность Θ, меняя и $^{S}\mathbf{C}$. Во-вторых, уточняется знание о системах. Целесообразно поэтому, чтобы в системных представлениях нашла отражение не обязательно синхронная, онтологическая и гносеологическая динамика систем. Это можно учесть, закрепив за ними изменения системы на основе (1)—(4) и (1′)—(4′) соответственно и введя фактор времени t.

В связи с изложенным предлагается следующая двухэтапная (I: целостность → II: целое), а на каждом этапе онтологическая ($\mathbf{S_{i2}}$) и гносеологическая ($\mathbf{S_{i3}}$) формализации системы S, статическое представление которой имеет вид:

$$\text{I. } \mathbf{S_{02}} = <^{\text{ц-ср}}\Theta, \; ^{\text{ц-стр}}\Theta, \; ^{S}\mathbf{C}>$$

$$\text{и } \mathbf{S_{03}} = <^{\text{ц-ср}}\Theta_{\mathbf{A}}, \; ^{\text{ц-стр}}\Theta_{\mathbf{A}}, \; ^{S}_{\mathbf{A}}\mathbf{C}>. \qquad (5)$$

$$\text{II. } \mathbf{S_{22}} = \langle \mathbf{Э}(^{\mathbf{Э}}\mathbf{C}); {}^{1}\mathbf{R}({}^{1\mathbf{R}}\mathbf{C}), {}^{2}\mathbf{R}({}^{2\mathbf{R}}\mathbf{C}); {}^{\text{ц-ср}}\Theta, {}^{\text{ц-стр}}\Theta, {}^{\mathbf{S}}\mathbf{C}\rangle; \quad (6)$$

$$\mathbf{S_{23}} = \langle \mathbf{Э_A}(^{\mathbf{Э}}\mathbf{C_A}); {}^{1}\mathbf{R_A}({}^{1\mathbf{R}}\mathbf{C_A}), {}^{2}\mathbf{R}({}^{2\mathbf{R}}\mathbf{C_A}); {}^{\text{ц-ср}}\Theta_{\mathbf{A}}, {}^{\text{ц-стр}}\Theta_{\mathbf{A}}, {}^{\mathbf{S}}\mathbf{C_A}\rangle.$$

На синтаксическом уровне формализации компоненты в выражениях (5)—(6) (*экстенсионально*) *обозначают* те или иные объекты, а на семантическом уровне (*интенсионально*) *выражают* их смысл. Для того чтобы в соответствии с (5)—(6) задать систему, необходимо вначале определить ее как целостность, установив по ${}^{\text{ц-ср}}\Theta$ ее конечную цель **Ц**. Затем поставить в соответствии **Ц** функциональное свойство системы $\mathbf{c}^{\mathbf{S}}_{\mathbf{1}}$, определив таким образом функцию системы **Ф**. Задав другие, необходимые и достаточные, свойства системы как целостности, доопределить ${}^{\mathbf{S}}\mathbf{C}$ (отметим сопоставимость ${}^{\mathbf{S}}\mathbf{C}$ с атрибутивным концептом [16]).

Затем следует определить систему как целое. Основываясь на понятийном базисе предметной области, задав реляционный базис системы $\mathbf{R}({}^{\mathbf{R}}\mathbf{C})$ и формализовав семантику отношений [5], надо проанализировать **Ц**, получив на этапе целеполагания системно-структурное представление целей, в том числе в критериальной форме. Проанализировав в соответствии со структурой целей $^{\mathbf{S}}\mathbf{c_1}$, определить состав функции **Ф** и, выбрав подходящий по функциональным свойствам состав $\mathbf{Э}(^{\mathbf{Э}}\mathbf{C})$ системы, по системно-функциональному представлению сформировать ЦРС. Наконец, получить системно-структурное представление целереализации с обратным по отношению к аналогичному представлению целеполагания ходом времени.

Интерпретируемые таким образом (5)—(6) через обусловливающие целостность Θ закономерности: *строения*, определяемые структурообразующими отношениями $^{1}\mathbf{R}(^{1\mathbf{R}}\mathbf{C})$; *функционирования*, определяемые функциональным свойством $^{\mathbf{S}}\mathbf{c_1}$; *целеполагания*, определяемые через неявно задаваемую посредством Θ конечную цель **Ц**, — все еще не обеспечивают полной конструктивности определения для системного представления и анализа. Степень конструктивности определения может быть повышена явным заданием конечной цели системы, доопределением временного периода **Т** — определения задачи по терминологии [1], а также заданием результатов системно-структурного анализа **Ц** и **ЦРС**.

Структурно-целевая парадигма системного анализа в производственной сфере

Расширительное толкование выражений (5)—(6) способствовало выработке парадигмы системного анализа в производственной сфере, основывающейся на двух определяющих концептах целостности организационных систем — цели и целесообразных структурах, при доминирующей роли первого концепта.

Действительно, при системном анализе имеют место попарные семантические трансформации: между проблемами и целями как отрицаниями проблем [5], между целями и критериями их достижения как правилами, включающими описание целей, между целями, в которых нередки ссылки на реализующие их средства с указанием функциональных свойств средств [5], и функциями ЦРС, между структурами целеполагания (ЦП), получаемыми в ходе анализа конечных целей ${}^{\text{ЦП}}$СЦ, и синтезируемыми на этапе целереализации (ЦР) структурами ${}^{\text{ЦР}}$СЦ. При этом, как правило, имеет место изоморфизм структур проблем (СП), целей, критериев, функций, на что обращается внимание в работе [10] и что подтверждается практикой системного анализа [6]. Так как в производственной сфере цели и их структуры определяют функционирование и развитие систем, программы, планы и их исполнение, использование в качестве методологического регулятива структурно-целевой парадигмы системного анализа будет способствовать повышению его корректности.

Постулаты структурно-целевого анализа систем производственной сферы

Для повышения степени объективности и избежания паралогизмов системного анализа необходима, как следует из предыдущих рассуждений, формализация логического аспекта его доминантной, целевой составляющей. При этом, как показано в работах [5, 18], целесообразно частично-формальное (лингвистическое) описание целей лицом, выдвигающим цели и генерирующим гипотезы о связях между целями в СЦ. Логическая же система должна проверять непротиворечивость гипотез, а в случае их опровержения — вырабатывать рекомендации по устранению ошибок. В ходе такого рассуждения осуществляется постепенная логико-лингвистическая формализация конечной цели, корректно уменьшается ее сложность и повышается определенность, что выражено в следующих постулатах.

Постулат 1. Моделирование логического анализа целей, основывающегося на лингвистических моделях целей, семантике целей и отношений между ними, и получаемые в результате такого моделирования структуры целей определяют соответствующие ЦРС производственной сферы.

Постулат 2. Система ситуационного логического анализа целей обеспечивает непротиворечивость рассуждений о целях и получаемых в результате таких рассуждений структур целей, а настройка системы на предметную область — проверку их полноты.

Принципы структурно-целевого анализа-синтеза организационных систем производственной сферы

Междисциплинарное исследование ЦП и ЦР, опирающееся на закономерности ЦП [5], иерархические структуры, семантику целей, логических и структурных отношений между целями в СЦ, выбранный наиболее эффективный класс методов выявления паралогизмов в структурах проблем и целей (K^{232} [20]) и постулаты 1, 2, способствовало выработке следующих принципов структурно-целевого анализа-синтеза систем.

1. Структуры проблем, целей, критериев достижения целей, ЦРС, их функций, графиков ЦР и их составляющие логически и семантически связаны.

2. Для практического рассуждения о проблемах/целях и оценки СЦ наиболее целесообразен человеко-машинный анализ-синтез проблем/целей.

3. *Структурно-целевой анализ проблем/целей.*

3.1. Логический анализ проблем/целей обеспечивается посредством формализации вывода лингвистически представленных проблем/целей.

3.1.1. Моделирование проблем/целей обеспечивается посредством лингвистической формализации формулировок проблем/целей.

3.1.2. Моделирование базовых знаний об анализе базируется на структурно-лингвистической формализации тезауруса предметной области.

3.1.3. Моделирование анализа проблем/целей базируется на логико-семантической формализации вывода проблем/целей на основе тезауруса.

3.1.3.1. Непротиворечивость получаемых в результате анализа СП/СЦ обусловлена логическим выводом.

3.1.3.2. Полнота СП/СЦ обусловлена выводом всех возможных проблем/целей в соответствии с ситуационной проекцией тезауруса.

3.1.4. Моделирование результатов структурного анализа проблем/целей обеспечивается структурно-лингвистической формализацией СП/СЦ.

3.2. Оценивание структур и систем обеспечивается посредством логико-семантической формализации данных понятий и математических методов.

4. *Структурно-целевой синтез систем.*

4.1. Синтез главной проблемы/цели системы обеспечивается посредством логико-лингвистической формализации ее проблем/целей.

4.2. Синтез целей в критериальной форме описания (критериев K) обеспечивается их формализацией в соответствии с принципом 3.1.1.

4.3. Синтез функций Ф для осуществления целей СЦ обеспечивается формализацией их соответствия СФ-свойствам целей-средств по принципу 3.1.1.

4.4. Синтез ^{ЦР}СЦ обеспечивается посредством формализации соответствия ^{ЦП}СЦ $\rightarrow$ ^{ЦР}СЦ.

4.5. Непротиворечивость графиков ^{ЦР}СЦ обеспечивается принципом 4.4.

5. Восприятие интерфейсных средств системы поддержки структурно-целевого анализа-синтеза обеспечивается учетом возможностей человека.

Концепция поддержки структурно-целевого анализа-синтеза систем основывается на предложенных выше принципах.

Трехкомпонентная семиотическая система $\mathbf{S_{СП/СЦ}}$ [20], реализующая принцип 3.1, включает формальную подсистему анализа куста проблем/целей $\mathbf{S_{кп/кц}}$, **Ш**-механизм, настраивающий $\mathbf{S_{кп/кц}}$ на текущую ситуацию на кусте проблем/целей, **О**-преобразователь лингвистических представлений проблем/целей в логико-лингвистические формулы и наоборот (принцип 3.1.3). Принцип 3.1.2 реализуется моделью **М** базовых знаний о предметной области [5, 20]. Интеллектуальный интерфейс с $\mathbf{S_{СП/СЦ}}$ реализован посредством языка описания проблем/целей $\mathbf{L_{in}}^1$ [5] (принцип 3.1.1), а входной интерфейса с базой знаний — с помощью языка $\mathbf{L_{in}}^2$ как упрощенной версии $\mathbf{L_{in}}^1$ (принцип 3.1.2). Посредством $\mathbf{L_{in}}^1$ семиотическая система получает лингвистические описания проблем/целей, преобразует их с помощью **О**-преобразователя в логические формулы и посредством формальной подсистемы $\mathbf{S_{кп/кц}}$ проверяет корректность текущего куста СП/СЦ. **Ш**-механизм настраивает $\mathbf{S_{кп/кц}}$ на анализ текущего куста, используя соответствующую проекцию **М** в качестве собственных доменов и возможных связей между ними. Если проверяемый куст противоречив или неполон, $\mathbf{S_{кп/кц}}$ идентифицирует ошибку и формирует рекомендацию по ее исправлению. Язык описания структур проблем/целей $\mathbf{L_{out}}$ реализует принцип 3.1.4 и осуществляет интерфейс с базой СП/СЦ. Он основывается на теоретико-графовой древовидной модели, узлы которой описаны в $\mathbf{L_{in}}^1$, и теоретико-множественном языке для описания семантически сложных дуг [20].

Система оценивания целей и СЦ использует методы оценки значимости целей, свойств СЦ и выбора альтернативных решений по целям [14] (принцип 3.2). Система анализа критериев K достижения целей использует СЦ — главный критерий сопоставляется главной цели, локальные — локальным целям СЦ (принцип 4.2). Система анализа функций также использует СЦ (принцип 4.3). Для каждой цели СЦ определяется функция ЦРС, полученное множество функций систематизируется — функции группируются по признакам: субъект-объект, уровень и функция управления, характер производства и «жизненный цикл» продукции и др. Затем выделяются функции управляющей и управляемой подсистем и в соответствии с общепринятыми правилами и нормами осуществляется группировка функций внутри каждой из подсистем ЦРС. Частично-формальный метод синтеза двух/трехуровневой структуры ЦРС описан в работе [20]. Структура ^{ЦР}СЦ строится как ^{ЦП}СЦ с обратным ходом времени (принцип 4.2).

Структуры целей рыбопромышленных систем

Среди определенных на целях СЦ отношений выделены две группы: структурные ($^{11}\mathbf{R} \subseteq {}^{1}\mathbf{R}$) и неструктурные ($^{21}\mathbf{R} \subseteq {}^{2}\mathbf{R}$). К структурным в соответ-

ствии с принципом иерархичности [5, 20] отнесены отношения, задающие строгий порядок на целях. При этом отношению связанности целей поставлена в соответствие связанность СЦ ($^{ц\text{-}стр}\mathbf{c_1} \in {}^{ц\text{-}стр}\mathbf{C}$). Неструктурные отношения представлены двумя подгруппами — вспомогательными, анализирующими непротиворечивость и полноту СЦ, и дополнительными, используемыми при уточнении СЦ [20].

Отношения непротиворечивости целей в соответствии с принципом 3.1.3.1 выражают выводимость целей. Отношение полноты СЦ в соответствии с принципом 3.1.3.2 выражает полноту сопоставимых подцелей кустов СЦ [20].

Дополнительное отношение значимости целей используется для анализа состава целей уже построенной непротиворечивой и полной СЦ. Такой анализ проводится в случае ограниченных ресурсов на достижение целей, для получения ресурсно уточненной $СЦ^0$, непротиворечивой, но не полной [20].

Анализ рыбохозяйственной деятельности и ее целей [5, 6] позволил уточнить состав и семантику базовых отношений между целями. В соответствии с принципами 3.1.3.1–3.1.3.2 в структурах целей рыбопромышленных систем определены базовые отношения с именами ^{o}I:

— *подчинения* ($^{11п}\mathbf{R} \subset {}^{11}\mathbf{R}$): «результат—средство» (I_1), «целое—часть» (I_2), «род—вид» (I_3), «ранг—субранг» (I_4), «система—аспект (системы)» (I_5), «элемент—стадия_ЖЦ(элемента)» (I_6), так что множество имен отношений подчинения целей $I^{п} = \{I_j\}, j = \{1, ..., 6\}$;

— *сопоставимости:* «средств» (I_7), «частей» (I_8), «видов» (I_9), «субрангов» (I_{10}), «аспектов» (I_{11}), «стадий_ЖЦ» (I_{12}), так что множество имен отношений сопоставимости целей $I^{сп} = \{I_j\}, j = \{7, ..., 12\}$;

— *полноты:* «средств» (I_{13}), «частей» (I_{14}), «видов» (I_{15}), «субрангов» (I_{16}), «аспектов» (I_{17}), «стадий_ЖЦ» (I_{18}), так что множество имен отношений полноты целей $I^{п} = \{I_j\}, j = \{13, ..., 18\}$. При этом $^{o}I = \{I^{п}, I^{сп}, I^{п}\}$.

В соответствии с базовыми тактиками структурного ЦП [20] определены базовые структурные элементы СЦ и их свойства. Расчленение сложной цели по тактике *стратифицирования* производится с помощью отношения с именем I_5 или I_6, по тактике *эшелонирования* — посредством отношения с именем I_4, а при использовании тактики *расслоения* — на основе отношения с именем I_1, I_2 или I_3. Показано, что все базовые и производные тактики структурного ЦП обеспечивают формирование строгого порядка на целях СЦ [5, 20].

В качестве дополнительных отношений ^{д}I используются: регистрирующие ошибки целеполагания отношения с именами ^{б}I, формирующие рекомендации по исправлению ошибок ЦП активные отношения с именами $I^{а}$ и отношения значимости с именами $I^{з}$, так что $^{д}I = \{^{б}I, I^{а}, I^{з}\}$ [20].

Определения отношений *подчинения* $R^{п} \subseteq {}^{11}R$, *непосредственного подчинения*, *соподчинения*, а также *сопоставимости* целей $R^{сп}$ в кусте целей СЦ, *непротиворечивой по вертикали* и *горизонтали*, *полной*, *допустимой*, *ошибочной* и *ресурсно уточненной* СЦ приведены в [20].

Поскольку семантика проблем, целей, критериев и функций ЦРС коррелированна, к структурам проблем, критериев и функций ЦРС применимы приведенные здесь результаты исследования структур целей. Для каждого сектора производственной сферы необходимо уточнить набор базовых тактик структурного ЦП и, руководствуясь принципами 3.1.3.1 и 5 и отраслевыми тезаурусами, сформировать базовые кусты целей и канон СЦ, являющийся образцом последовательного членения целей на определенный временной период. Таким образом в соответствующем секторе регламентируются семантика структурообразующих отношений и логика СЦ.

Одна из методик формирования канона СЦ выработана практикой СА рыбохозяйственной деятельности [5, 6]. Методика предполагает построение и использование стратегической структуры целей рыбной отрасли. Канон СЦ формируется, исходя из стратегической СЦ посредством систематизации и обобщения составляющих ее целей, уточнения семантики отношений подчинения, соподчинения и полноты целей. Базовые кусты СЦ, выражающие общие закономерности ЦП в секторе производственной сферы, используются при анализе функционирования и развития его систем. Каноны СЦ, являющиеся более жесткими структурами и характеризующие закономерности систем на определенном временном интервале, более полезны при анализе функционирования систем. Процедура построения канона структуры целей и пример канонической СЦ рыбной отрасли приведены в работе [5].

Языки описания проблем и целей рыбопромышленных систем

Как показал анализ, наиболее адекватным для реализации $L_{in}{}^{1}$ является фреймовый язык [5], имеющий структуру, аналогичную «матрешке», что позволяет легко представлять как отдельные, так и две или более непосредственно связанные проблемы/цели. Присущая фреймовым представлениям регулярность уменьшает разнообразие процедур их обработки по сравнению с другими языками. $L_{in}{}^{1}$ базируется на двухуровневой лингвистической модели формулировки проблемы/цели, макроописатель которой есть настраиваемый по составу ролей, выражающих функциональную формулу рыбохозяйственной деятельности, ролевой фрейм. Микроописатель — описатель замещающих роли понятий ($э_i \in Э$), определенных в пространстве свойств ($^{э}c_j \in {}^{э}C$), посредством которых детализируется и упорядочивается внутриролевое (фразовое) описание. Свойства разбиты по видам на непересекающиеся группы, состав которых определяется предметной областью и их декомпозиционными возможностями. Из-за избыточности естественно-языковых формулировок предусмотрено выделение в них проблемных/целевых частей спе-

циальными указателями. Роль, вид свойства и указатели выражают обобщенную, а термины предметной области — предметную семантику проблем/целей. Макроописатель проблемы/цели — фрейм «средства—результат» имеет вид:

<< 1: агенс > < 2: технология управления >
< 3: техника > < 4: исходный (7) объект >
< 5:технология производства > <6: место>
<7: конечный объект >>,

а в общем случае: $<< u^{ri} ... > [[< u^{rj} ... >] ...] >$, где u^{ri} — указатель r_i-й роли, $i, j = \{1, ..., n\}$, $U^r = \{u^{ri}\}$.

Фрейм настраивается на каждый вид деятельности, определяемый характерным для него результатом. Например, операция «переработка сырья и полуфабрикатов» имеет результатом роль 7. На основе (7) цель «создать оборудование для производства пищевой и технической продукции из рыбы» имеет вид: <<3 создать оборудование> <4 рыба> <7 пищевая и техническая продукция>>. Целевые фразы распознаются по отношению «быть целью», на которое обычно указывают глаголы в неопределенной форме (например, создать). Поскольку кроме функции указания на цель они идентифицируют операцию по целереализации (создание), эти функции разделены и в L_{in}^1 эксплицируются так: первая — снабжением целевой фразы указателем цели G, вторая — определением стадии жизненного цикла (ЖЦ) получения соответствующего результата. Тогда рассматриваемая цель примет вид: <<G 3 оборудование ЖЦ создание> < 4 рыба > <7 пищевая и техническая продукция>>. Соответственно «проблемные» фразы помечаются в L_{in}^1 указателем проблем H.

Упорядочивающий фразовую форму и эксплицирующий семантику фраз микроописатель основывается на термах t_1/t_2: $<t_1>/<t_2>::=<u^{ri}>$ <БЭ>/ $<u^{sj}>$<БС>, где u^{sj} — указатель s_j-го вида свойства, $j = \{1, 2, ..., m\}$, m — число видов свойств (по умолчанию $m = 4$, u^{s1} (или СХ) — указатель характеристического свойства; u^{s2} (или СФ) — функционального; u^{s3} (или СЗ) — физического; u^{s4} (или СИ) — именного [5]), $U^s = \{u^{sj}\}$. Имея набор базовых элементов(БЭ)/свойств(БС), на основе микроописателя получают описания производных элементов (ПЭ)/свойств (ПЭ) необходимой степени детализации. Приведем пример правила описания предложения—цели $ц$ в языке L_{in}^1:

<ц> ::=<G u^{ri}БЭ[u^{sj} БС[[, БС]...]...]>
[< G] u^{rj} БЭ [u^{sj} БС[[, БС]...]...]>...]>.

Макро- и микроописатель совместно с методикой перехода от ЕЯ-формулировки к L_{in}^1-описанию [5] обеспечивают удобный интеллектуальный интерфейс и возможность формализованной обработки полученных описаний. Обобщенная семантика проблемы/цели задается следующими компонентами: $(H, U^r, U^s)/(G, U^r, U^s)$, а предметная семантика определяется множеством базовых элементов $^oP^1$ и базовых свойств $^oP^2$. В [5; 20] формально определены основные типы целей/проблем производственных систем.

Структурно-целевой анализ и синтез проблем и целей

Ядром семиотической системы логико-лингвистического анализа проблем/целей и моделирования СП/СЦ $\mathbf{S_{СП/СЦ}}$ [20] является подсистема анализа куста проблем/целей $\mathbf{S_{кп/кц}}$, $\mathbf{S_{кп/кц}} = \mathbf{<V, B, A, P>}$, где **V** — «алфавит», включающий множества описанных в $\mathbf{L_{in}^1}$ и переведенных в логическую форму ролевых фраз предложений—проблем/целей и имен семантических отношений $\mathbf{I} = \{^o\mathbf{I}, ^\delta\mathbf{I}, \mathbf{I}^a\}$; **B** — множество синтаксических правил; **A** — множество аксиом, $\mathbf{A} = \mathbf{A_1} \cup \mathbf{A_2}$, $\mathbf{A_1(A_2)}$ — множество неизменяемых(изменяемых) аксиом; $\mathbf{A_1}$ включает полное множество аксиом логики утилитарных оценок А. А. Ивина [5]; $\mathbf{A_2}$ включает схемы собственных аксиом $\mathbf{S_{кп/кц}}$, выражающих утверждения об анализируемой проблеме/цели, о семантических отношениях на фразах проблема—подпроблемы/цель—подцели и допустимых комбинациях таких отношений; **P** — схемы вывода.

Полная семантика $\mathbf{S_{кп/кц}}$ построена на основе модифицированной модели Крипке, разработанной Г. С. Осиповым [5], — $\mathbf{K} = \mathbf{<K_1, K_2>}$, компонент $\mathbf{K_1}$ которой определяет постоянную (внутреннюю), а компонент $\mathbf{K_2}$ задает переменную (внешнюю) семантику системы, определяемую в соответствии с **M**.

На вход $\mathbf{S_{СП/СЦ}}$ поступают лингвистические представления, а на вход $S_{кп/кц}$ — преобразованные в логические, в общем случае импликативные, формулы, L_{in}^1 — описания предложений — проблем/целей, антецедент которых в общем случае — конъюнкция фраз f_i, $i = \{1, 2, ..., n\}$ с ролями «средство», а консеквент — фраза f_{n+1} с ролью «результат» (ею может быть, например, фраза «конечный объект»).

Исходя из предположения об истинности проблемы/цели анализируемого куста СП/СЦ, $\mathbf{S_{СП/СЦ}}$ посредством $\Psi_{\mathbf{V,B,A}}$ настраивает $\mathbf{S_{кп/кц}}$ на очередной такт функционирования, актуализируя необходимую/допустимую проекцию понятийно-реляционного базиса внешней для $\mathbf{S_{кп/кц}}$ модели **M**. $\mathbf{S_{кп/кц}}$ на основе **M** анализирует непосредственную выводимость подпроблемы/подцели куста СЦ из его проблемы/цели, формируя в качестве результата такта своего функционирования непротиворечивый и полный куст проблем/целей.

При функционировании $\mathbf{S_{СП/СЦ}}$ использует семантические отношения, задаваемые в отличие от традиционных парами $\mathbf{<I_j, R_j>}$, в которых компонент $\mathbf{I_j}$ — имя отношения, $\mathbf{I_j} \in \mathbf{I}$, а **I** — множество имен, выражающих реляционный базис предметной области, отношений как закономерности формирования кустов объектов ($\mathbf{I} \subset \mathbf{V}$). Поэтому m-местному отношению в $\mathbf{S_{СП/СЦ}}$ соответствует $(m + 1)$-местное, записываемое как $\mathbf{I_j(f_1, ..., f_m)}$ семантическое отношение, имя которого выступает в роли предметной переменной. Это дает возможность использования имен семантических отношений в формулах первой ступени как в качестве свободных, так и связанных переменных.

Такт функционирования $\mathbf{S_{СП/СЦ}}$ состоит в пошаговой работе $\mathbf{S_{кп/кц}}$, которая не изменяется во время анализа куста проблем/целей. Один шаг есть

производимый по схеме $\mathbf{p_1} | \Rightarrow \mathbf{p_2}$ вывод, где $\mathbf{p_1}$ и $\mathbf{p_2}$ — проблемы/цели, а условиями применимости правила вывода являются возможные в соответствии с **M** семантические отношения между $\mathbf{p_1}$ и $\mathbf{p_2}$. Вывод для пары $<\mathbf{p_1}, \mathbf{p_2}>$ основывается на гипотезе об импликативной связи $\mathbf{p_1} \rightarrow \mathbf{p_2}$, в которой $\mathbf{p_1}$ полагается истинной (а соответствующий результат — абсолютно ценным). Истинностное значение $\mathbf{p_1} \rightarrow \mathbf{p_2}$ анализируется по **M**, и в случае истины по правилу отделения устанавливается истинность $\mathbf{p_2}$. Ложность $\mathbf{p_1} \rightarrow \mathbf{p_2}$ означает противоречие с базовыми знаниями; в этом случае $\mathbf{S_{кп/кц}}$ синтезирует и выводит в соответствии с **M** рекомендуемую $\mathbf{p_2}'$. Вывод упрощен за счет классификации ситуаций на целях [20]. Для представления результатов функционирования $\mathbf{S_{СП/СЦ}}$ используются графо-лингвистические модели [5, 19, 20].

Заключение

Примеры структурно-целевого анализа систем приводятся в работах [5, 6, 19, 20]. Предложенные средства использовались при анализе программ развития рыбной отрасли, проблемных ситуаций в отраслевом технологическом оборудовании и в региональных рыбопромышленных системах. Качество результатов структурно-целевого анализа систем подтверждено экспертизой и практикой решения проблем рыбной отрасли.

Литература

1. **Проблемы** программно-целевого планирования и управления / Под ред. Г. С. Поспелова. — М.: Наука, 1981. — 464 с.
2. **Садовский В. Н.** Основания общей теории систем. Логико-методологический анализ. — М.: Наука, 1974. — 260 с.
3. **Гвишиани Д. М.** Материалистическая диалектика — философская основа системных исследований // Системные исследования. Методологические проблемы. — М.: Наука, 1979. — С. 13–16.
4. **Денисов А. А., Волкова В. Н.** Основы теории систем и системного анализа. — СПб.: СПБГТУ, 2001. — 512 с.
5. **Лукьянова Л. М.** Системный анализ: Структурно-целевой подход. — Калининград: КГТУ (Госкомрыболовство РФ), 2001. — 234 с.
6. **Исследование** технологий анализа систем: системный анализ рыбопромышленного комплекса // Отчет о НИР. № Гос. рег. 01.20.00 06420, инв. № 02.20.02 06264; Руков. Л. М. Лукьянова. — Калининград, КГТУ, 2002. — 87 с.
7. **Щедровицкий Г. П.** Принципы и общая схема методологической организации системно-структурных исследований и разработок // Системные исследования. Методологические проблемы. — М.: Наука, 1981. — С. 193–226.
8. **Афанасьев В. Г.** Общество: системность, познание и управление. — М.: Политиздат, 1981. — 432 с.
9. **Сетров М. И.** Основы функциональной теории организаций. — Л.: Наука, 2002. — 164 с.
10. **Перегудов Ф. И., Тарасенко Ф. П.** Введение в системный анализ. — М.: Высшая школа, 1989. — 367 с.
11. **Садовский В. Н.** Становление и развитие системной парадигмы в Советском Союзе и в России во второй половине XX века // Системные исследования. Методологические проблемы. — М.: Наука, 2001. — С. 7–35.
12. **Поспелов Д. А.** Ситуационное управление. Новый виток развития // Изв. РАН: Теория и системы управления. — № 5. — 1995. — С. 152–158.
13. **Сараева И. Н., Уемов А. И.** К проблеме взаимоотношения и интеграции системных теорий // Системные исследования. Методологические проблемы. — М.: Наука, 1986. — С. 79–96.
14. **Saaty T., Kearns K.** Analytical planning: The organization of systems. — N. J., 1991.
15. **Моросанов И. С.** Первый и второй законы теории систем // Системные исследования. Методологические проблемы. — М.: Наука, 1996. — С. 97–114.
16. **Уемов А. Н.** Логический анализ системного подхода к объектам и его место среди других методов исследования // Системные исследования. Методологические проблемы. — М.: Наука, 1969. — С. 80–95.
17. **Философский** энциклопедический словарь. — М.: Сов. Энциклопедия, 1983. — 840 с.
18. **Смирнов Г. А.** Логические аспекты проблемы целостности // Системные исследования. Методологические проблемы. — М.: Наука, 1996. — С. 108–126.
19. **Лукьянова Л. М.** Структурно-целевой анализ систем на основе логико-лингвистических формализаций // Proceedings of the X-th International Conference «Knowledge-Dialogue-Solution.» — Varna, 2003. — P. 482–490.
20. **Лукьянова Л. М.** Методология структурно-целевого анализа организационных систем производственной сферы // СПИИРАН. Вып. 1. — СПб.: 2002. — С. 297–315.

УДК 681.3.06

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД К АВТОМАТНОМУ ПРОГРАММИРОВАНИЮ

Д. Г. Шопырин,
аспирант
А. А. Шалыто,
д-р техн. наук, профессор
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

В данной работе предлагается подход к реализации объектно-ориентированных систем с явным выделением состояний. Рассмотрены вопросы повторного использования программных компонентов, параллельных вычислений, автоматического протоколирования работы системы и повышения отказоустойчивости системы.

In this work an approach to an implementation of object-oriented systems with obvious state dedication is given. Considered questions of reusing of software components, multithreading, automatic system work logging and increasing of fault tolerance of a system.

Введение

В последнее время в программировании все шире применяются конечные автоматы [1–3].

Один из подходов к совместному использованию объектной и автоматной парадигм программирования в работах [4–7] получил название «объектно-ориентированного программирования с явным выделением состояний», или «SWITCH-технологии» [8, 9]. В указанных работах подробно рассмотрены вопросы проектирования программ этого класса, однако предложенный в них метод реализации автоматов обладает следующими недостатками:

— не выделено состояние системы в целом;

— в функции, реализующей автомат, применяются два оператора switch, так как нет механизма различения действий и деятельности в состояниях; это снижает удобочитаемость кода и увеличивает вероятность ошибки;

— нет механизма обеспечения повторного использования реализованных автоматов;

— функции протоколирования вводятся в текст программы вручную, что не является «автоматическим» построением протокола;

— не предложен механизм обработки ошибок, возникающих при работе системы;

— отсутствует механизм организации параллельных вычислений.

В данной работе предлагается подход к реализации программных систем рассматриваемого класса, устраняющий указанные выше недостатки. В качестве базы для разработки «автоматной» части программ с явным выделением состояний (автоматы, входные и выходные воздействия, инфраструктура) предлагается библиотека STOOL (SWITCH-Technology Object Oriented Library). Эта библиотека реализована на языке C++ и доступна к загрузке с сайта http://is.ifmo.ru (раздел «Проекты»). Остальная часть (контекст) программы разрабатывается традиционным образом без использования этой библиотеки.

Определения

Класс автоматов — множество автоматов, реализующих один и тот же граф переходов. Например, классом автоматов является множество одинаковых автоматов, осуществляющих подсчет повторений в последовательности.

Автомат — конкретный экземпляр класса автоматов.

Входное воздействие — булева функция, характеризующая состояние внешней среды. Значение входного воздействия не зависит от порядка и количества обращений к нему. Таким образом, обращение к входному воздействию не изменяет состояния внешней среды.

Выходное воздействие — некоторая операция, изменяющая внешнюю среду.

Система автоматов — совокупность автоматов.

Состояние системы — совокупность состояний всех автоматов системы.

Системный переход — переход системы в другое состояние. Системный переход начинается с запуска некоторого автомата A_i. Этот автомат может запускать другие автоматы. После того как автомат A_i и каждый из запущенных им автоматов совершит не более одного перехода, считается, что системный переход осуществлен.

Исключительная ситуация — возникает в случае невозможности опроса входного воздействия или выполнения выходного воздействия.

Этап автомата — набор последовательных запусков, в течение которых автомат сохраняет свое состояние.

Шаг этапа — каждый запуск автомата в течение этапа.

Действие в состоянии — вызов некоторого выходного воздействия, происходящий во время первого шага этапа автомата.

Деятельность в состоянии — вызов некоторого выходного воздействия, происходящий на каждом шаге этапа.

Действие на переходе — вызов некоторого выходного воздействия, происходящий при переходе автомата из одного состояния в другое. При этом сначала выполняются действия на переходе, а затем изменяется состояние автомата.

Особенности архитектуры

Предлагаемая архитектура программных систем отличается от предлагаемой в работах [4–7]. Эти отличия состоят в следующем:

— автоматы, входные и выходные воздействия являются объектами;

— явно вводятся понятия *класс автоматов* и *экземпляр класса автоматов*;

— каждый автомат не располагает никакой информацией о других автоматах системы; при этом, во-первых, автомат не знает о существовании других автоматов, во-вторых, он не может непосредственно проверять их состояния, как это было предложено, например в [8], а в-третьих — не может непосредственно запускать другие автоматы; единственная связь автомата с «внешним миром» — это входные и выходные воздействия.

Введение понятий *класс автоматов* и *экземпляр класса автоматов* необходимо для обеспечения повторного использования автоматов. Пусть, например, в системе имеются два одинаковых автомата, отличающиеся только входными и выходными воздействиями. При этом нет необходимости каждый из них проектировать отдельно, а достаточно создать и включить в систему два *экземпляра* одного и того же *класса автоматов*, что соответствует парадигме объектно-ориентированного программирования (ООП). Разработанный класс автоматов может повторно использоваться и в других системах.

Введенные ограничения на взаимодействие автоматов повышают их модульность и тем самым увеличивают вероятность их повторного использования.

При этом, если необходимо, например, осуществить переход по номеру состояния другого автомата, то вводится входное воздействие, возвращающее значение *true*, если автомат находится в искомом состоянии. Если же необходимо осуществить запуск другого автомата, то вводится выходное воздействие, запускающее этот автомат.

Поясним теперь, почему в рамках предлагаемого подхода понятие «событие» не применяется. Это связано с переносимостью библиотеки STOOL, на базе которой предлагается реализовывать системы. В разных системах существуют различные модели возникновения и обработки событий, которые весьма трудно обобщаются в понятной и краткой форме. Введение в библиотеку какой-нибудь одной модели обработки событий сузило бы область применения библиотеки, а введение нескольких моделей усложнило бы дизайн библиотеки и простоту ее использования. Поэтому в рамках предлагаемой архитектуры реализация механизмов обработки событий возлагается на пользователя.

Пусть, например, для левой кнопки мыши введено входное воздействие x_0, возвращающее значение *true*, если кнопка нажата. При получении уведомления от пользовательского интерфейса о том, что кнопка нажата (например, событие WM_LBUTTONDOWN в ОС Windows), пользователь устанавливает соответствующий флаг и запускает автоматы, обрабатывающие это событие. При получении уведомления о том, что кнопка отпущена, пользователь сбрасывает флаг.

Обзор классов библиотеки STOOL

Предлагаемый подход базируется на библиотеке STOOL. Эта библиотека предоставляет абстрактные базовые классы для реализации автоматов, входных и выходных воздействий, а также «инфраструктуру» для организации системы в целом.

При этом для создания конкретного класса автоматов создается потомок абстрактного базового класса *Auto*.

Все классы библиотеки определены внутри пространства имен *stool*.

Опишем классы для реализации автоматов и входных и выходных воздействий:

— *Auto* — базовый класс для разработки классов автоматов; для определения класса автоматов программист должен переопределить метод *void execution* (*State & state*), реализующий граф переходов;

— *State* — класс хранит состояние автомата; каждый экземпляр класса *Auto* содержит экземпляр класса *State*; класс *State* обеспечивает протоколирование любого (даже ошибочного) изменения состояния автомата; изменять экземпляр класса *State* программист может только внутри функции, реализующей граф переходов автомата;

— *Input* — базовый класс для реализации входных воздействий;

— *Output* — базовый класс для реализации выходных воздействий;

— *Impact* — класс описывает процесс выполнения выходного воздействия; он предоставляет следующие методы: «выполнить», «откатить» и «подтвердить»; при каждом выполнении выходного воздействия создается соответствующий этому воздействию экземпляр класса *Impact*, который и осуществляет это воздействие.

Перейдем к описанию классов для создания «инфраструктуры» системы:

— *System* — управляет системой автоматов, содержит экземпляры классов *ChangeServer*, *AutoEventServer*, которые описаны ниже; он также хранит список всех автоматов системы; предполагается, что пользователь может создавать потомков класса *System*;

— *Change* — управляет системным переходом; при возникновении исключительной ситуации во время выполнения системного перехода отвечает за разворачивание (*unwind*) стека выполненных выходных воздействий; этот класс является абстрактным; библиотека STOOL предоставляет два потомка этого класса — *SingleTaskChange* и *MultiTaskChange* (для однопоточной и многопоточной работы); эти классы могут быть расширены пользователем;

— *ChangeServer* — разработан для использования в однопоточных и многопоточных системах; управляет созданием и уничтожением переходов.

Дальнейшее описание библиотеки приводится в последующих разделах работы. Так, описываются методы *Output::action()*, *Output::activity()* и *Output::jumpAction()*, реализующие различные типы выходных воздействий.

Применение библиотеки STOOL

Выделение состояния системы в целом. Пользователь библиотеки STOOL может получить список всех автоматов системы, а у каждого автомата — его состояние. Пользователь может также получить информацию о любом выполняющемся в данный момент системном переходе.

Информацию обо всех автоматах системы можно получить с помощью метода *System::enumerate()*, а обо всех выполняющихся переходах — с помощью экземпляра класса *ChangeServer*, возвращаемого методом *System::getChangeServer()*.

Приведем пример кода, распечатывающий список автоматов системы, в котором используется метод *System::enumerate()*:

```
...
struct AutoReceiver
  : public ItemsReceiver<const Auto&>
{
  virtual bool receiveCount( int _count ) {
    cout << "всего автоматов: " << _count <<
    endl;
    return true;
  }
  virtual bool receiveItem( int _index, const Auto&
  _item )
  {
    cout << "автомат #" << _index << ": " <<
    endl;
    cout << " имя класса автоматов: " <<
    _item.getInfo().getClassName() << endl;
    cout << " имя автомата: " <<
    _item.getInfo().getInstanceName() << endl;
    cout << "состояние:"<<
    _item.getInfo().getStateName(_item.getState())
    << endl;
    return true;
  }
};
...
system.enumerate( AutoReceiver() );
```

Действия и деятельности. Для упрощения шаблона, реализующего автоматы, в котором используются два оператора *switch*, в библиотеке STOOL имеются средства для различения действий и деятельности в вершинах.

Деятельности в вершинах реализуется методом *Output::activity()*, а действия — методом *Output::action()*. Объект выходного воздействия сам определяет, находится ли вызывающий автомат в первом шаге этапа. Для выполнения действия на переходе предназначен метод *Output::jumpAction()*.

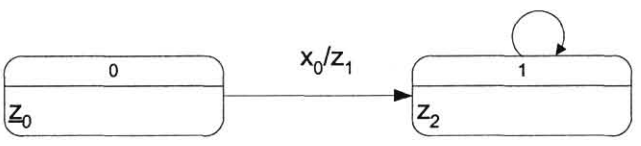
При описании поведения автомата в виде графа переходов действие в вершине и действие на переходе будем обозначать символом z_i, а деятельность — символом $\underline{z_i}$.

На рис. 1 приведен граф переходов. Ниже приведена функция, реализующая граф переходов с использованием библиотеки STOOL:

```
virtual void execution( State& state ) {
  switch ( state ) {
  case 0:
    io.z0().activity( *this );
    if ( io.x0().is( *this ) ) {
      io.z1().jumpAction( *this );
      state = 1;
    }
    break;
  case 1:
    io.z2().action( *this );
    break;
  }
}
```

В этом примере действия z_1 и z_2 выполняются не более одного раза.

Повторное использование автоматов. Автоматы предлагается реализовывать на основе патерна, в котором определяется набор необходимых входных (*I*) и выходных (*O*) воздействий.



■ ***Рис. 1.*** *Граф переходов*

```
class Ai : public Auto {
public:
  struct IO {
//Входные воздействия
    virtual Input& xk() = 0;
    virtual Input& xl() = 0;

//Выходные воздействия
    virtual Output& zm() = 0;
    virtual Output& zn() = 0;
  };
private:
  IO& io;

protected:
//Реализация графа перехода
  virtual void execution( State& state ) {
    switch ( state ) {
    case 0:
      if ( io.xk().is( *this ) )
        state = 1;
      ...
      break;
    ...
    }
  }

public:

//Конструктор
  Ai( IO& _io, const string& _instance_name,
  System& _system )
    : Auto( _instance_name, "Ai", _system )
    , io( _io ) {}
};
```

Такая реализация автомата не является системнозависимой, и поэтому появляется возможность повторно использовать классы автоматов.

Автоматическое протоколирование. Библиотека STOOL предоставляет средства для организации автоматического протоколирования изменений состояний системы. При разработке системы программист должен создать объект, реализующий детали протоколирования (куда записываются протокол, формат протокола и т. д.). Библиотека поддерживает автоматическое протоколирование:

— изменений состояния каждого автомата системы;
— создания и уничтожения автомата;
— начала и конца выполнения автомата;
— опроса входного воздействия;
— выполнения выходного воздействия;
— возникновения исключительной ситуации.

Гарантируется, что объект, реализующий протоколирование, будет уведомлен обо всех вышеуказанных изменениях.

Ниже приведен пример организации протоколирования:

```
class LogSystem : public AutoEventSync {
  int change_number;
public:
  LogSystem( Lockable& _locakble )
    : AutoEventSync( _locakble )
    , change_number( 0 ) {}
  void preamble() {
    cout << ++change_number << ".\t";
  }

  void onEvent( const Event _event, const Auto-
  EventSync::EventItem& _item ) {
    Lock lock( *this );
    switch ( _event ) {
    case AutoEventSync::E_AFTER_STATE_
    CHANGED:
      preamble();
      const AutoEventSync::StateEventData& data
      = _item.getStateEventData();
      const Auto& inst = data.state.getAuto();
      cout << "автомат " << inst.getInfo().getIn-
      stanceName()
           << "(" <<
           inst.getInfo().getClassName() << ")"
           << " перешел в состояние: " <<
           data.state
           << " (старое состояние: " <<
           data.old_state << ")"
           << "." << std::endl;
      break;
    }
  }
};
```

Механизм обработки ошибок. Для обеспечения устойчивости системы к ошибкам (исключительным ситуациям) вводится следующее ограничение: если не удалось осуществить системный переход, то система остается в исходном состоянии. Таким образом, если причина возникновения исключительной ситуации будет устранена, то при повторном запуске системный переход осуществится так, как если бы исключительной ситуации не возникало. Это аналогично транзакциям в системах баз данных, которые либо выполняются полностью, либо не выполняются вообще.

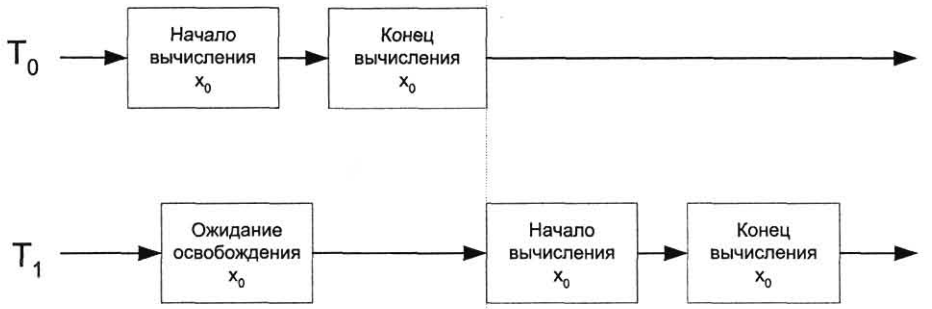
Для обеспечения устойчивости к ошибкам все выходные воздействия должны иметь возможность отката. Для этого выходные воздействия должны поддерживать следующие три операции:

1) «Выполнить» — воздействие выполняется, но при этом сохраняется вся информация, необходимая для отката; если реализовать откат сложно, то захватываются ресурсы, необходимые для выполнения воздействия, а оно само выполняется в операции «Подтвердить»;

2) «Откатить» — если воздействие было выполнено, производится откат выполненных изменений; если были захвачены ресурсы, то они освобождаются;

3) «Подтвердить» — если воздействие не было выполнено в операции «Выполнить», то оно выполняется, ресурсы освобождаются; вероятность возникновения исключительной ситуации во время выполнения этой операции должна быть сведена к минимуму.

Например, необходимо реализовать выходное воздействие, удаляющее некоторый файл. В операции «Выполнить» блокируется доступ к файлу. Если блокировать файл не удалось (например, он заблокирован другим приложением), то генерируется исключительная ситуация. В операции «От-



■ ***Рис. 2.*** *Разделение объекта между системными переходами*

катить» отменяется блокировка файла, а в операции «Подтвердить» удаляется файл.

Если программист не нуждается в обеспечении устойчивости к ошибкам, то для всех выходных воздействий он может реализовать только операцию «Выполнить».

Параллельные вычисления. Для решения многих задач бывает целесообразно организовать несколько потоков выполнения (многопоточные системы). В терминах систем с явным выделением состояний многопоточная система — это такая система, в которой в один и тот же момент может осуществляться несколько системных переходов.

Основной сложностью при разработке многопоточных систем является организация безопасного использования объекта несколькими потоками.

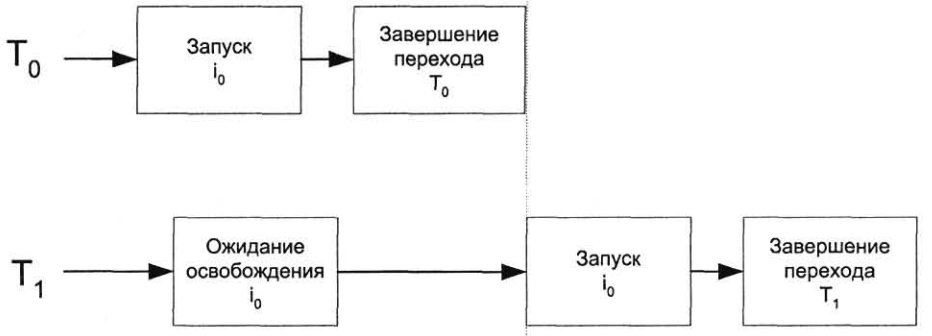
Библиотека STOOL может работать как в однопоточном, так и в многопоточном режиме. Переключение между этими режимами осуществляется передачей конструктору объекта *System* объекта-фабрики [10], создающего объекты *Change*, *ChangeServer* и *Lockable*. В библиотеке реализованы две фабрики:

— класс *Factory* является фабрикой «по умолчанию»; он настраивает библиотеку на однопоточный режим;

— класс *MultiTaskFactory* настраивает библиотеку на многопоточный режим; многопоточные объекты (*MultiTaskChange* и *MultiTaskLock*) реализованы с помощью библиотеки *boost::thread* (http://www.boost.org).

Любой объект системы, кроме автоматов, может одновременно использоваться только в одном переходе. Пусть выполняются два системных перехода — T_0 и T_1, которые обращаются к входному воздействию x_0. Процесс разделения объекта x_0 между переходами, осуществляемый средствами библиотеки, показан на рис. 2.

Автоматы являются единственными объектами библиотеки STOOL, которые остаются «захваченными» до завершения активизирующего их потока. Если системный переход T_0 запускает некоторый автомат i_0, то объект, представляющий этот автомат, остается «захваченным» до завершения этого перехода. Процесс разделения автомата i_0 между переходами будет происходить, как показано на рис. 3.



■ ***Рис. 3.*** *Разделение автомата между системными переходами*

Реализация входных и выходных воздействий

Для реализации входного воздействия создается потомок класса *Input*, и в нем переопределяется метод *bool execution()*:

```
class X0 : public Input {
  Data& data;
protected:
  virtual bool execution() {
    return data.isEnd();
  }
public:
  X0( Data& _data, System& _system )
    : Input( "X0", _system )
    , data( _data ) {}
};
```

Для реализации выходного воздействия необходимо произвести два действия:

1) определить потомок класса *Impact*;

2) определить потомок класса *Output*, создающий экземпляр соответствующего потомка класса *Impact*.

Приведем каркас реализации потомка класса *Impact* и соответствующую реализацию потомка класса *Output*:

```
class IZ0 : public Impact {
public:

  virtual void execute() {
    //Действие
  }

  virtual void rollback(){
    //Откат
  }

  virtual void commit() {
    //Подтверждение
  }
};


class Z0 : public Output {

public:
  virtual Impact* create() {
    return new IZ0();
  }

public:
  Z0( System& _system )
    : Output( "Z0", _system ) {}
};
```

Таким образом, для реализации выходных воздействий программист должен реализовать два класса.

Библиотека STOOL предоставляет средства, позволяющие сократить объем кода за счет введения вспомогательных классов.

Для реализации выходных воздействий, запускающих другой автомат, предназначен вспомогательный класс *AutoOutput*, «принимающий» запускаемый автомат в качестве параметра конструктора.

Для того чтобы не создавать два класса для реализации выходных воздействий, предназначены вспомогательные классы *GOutput* и *GOutputPx*. Например, для создания выходного воздействия с некоторым потомком класса *Impact* достаточно следующей строки кода:

```
GOutput<IZ0> z0( "Z0", system );
```

В случае, если входные и выходные воздействия представлены набором функций, для упрощенного создания соответствующих объектов можно воспользоваться вспомогательными функциями *makeFInput* и *makeFOutput*:

```
bool fx0() {
  //Реализация входного воздействия x0
}

void fz0() {
  //Реализация выходного воздействия z0
}
...
Input* x0 = makeFInput( fx0, "x0", system );
Output* z0 = makeFOutput( fz0, "z0", system );
```

Если входные и выходные воздействия представлены методами некоторого класса, то можно воспользоваться функциями *makeFInput* и *makeFOutput* в комбинации с библиотекой *boost::bind:*

```
class Ctx {
public:
  bool fx0() {
    //Реализация входного воздействия x0
    return false;
  }

  void fz0() {
    //Реализация выходного воздействия z0
  }
};
...
Ctx ctx;
Input* x0 = makeFInput( bind( &Ctx::fx0, ref( ctx ) ),
"x0", system );
Output* z0 = makeFOutput( bind( &Ctx::fz0, ref( ctx
) ), "z0", system );
```

В случае, если объекты автоматов, входных и выходных воздействий создаются как члены некоторого класса, то могут быть полезны разработанные макросы:

— DECLARE_FUNC_INPUT — объявляет метод, возвращающий экземпляр входного воздействия, которое вызывает некоторую функцию;

— DECLARE_FUNC_OUTPUT — объявляет метод, возвращающий экземпляр выходного воздействия, которое вызывает некоторую функцию;

— DECLARE_AUTO_OUTPUT — объявляет метод, возвращающий экземпляр выходного воздействия, которое запускает некоторый автомат;

— DECLARE_AUTO — объявляет метод, возвращающий экземпляр класса автоматов.

Использование приведенных макросов позволяет сократить реализацию объектов системы.

Пример использования библиотеки STOOL

Пусть на вход подается строка символов, завершающаяся нулем. *Словом* назовем непрерывную последовательность не пробельных символов. *Числом* назовем слово, состоящее только из цифр. Необходимо за один проход перевернуть все числа в строке и подсчитать количество слов. Например, при входной строке «test 123» результирующей строкой будет «test 321», а количество слов будет равно двум.

Для решения задачи требуется три класса автоматов:

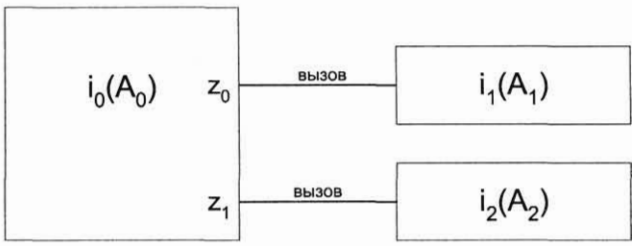
1) управляющих итерированием по строке — A_0;
2) переворачивающих числа — A_1;
3) подсчитывающих слова — A_2.

Создадим по одному экземпляру каждого класса автоматов (i_0, i_1 и i_2). На рис. 4 приведена схема взаимодействия автоматов, структурные схемы классов автоматов A_0, A_1 и A_2 приведены на рис. 5, а графы переходов автоматов A_0, A_1 и A_2 — на рис. 6.

Следует обратить внимание, что в первом графе переходов использован символ z_x, соответствующий завершению работы системы.

Автоматы A_0, A_1 и A_2 реализуются на основе изложенного подхода следующим образом:

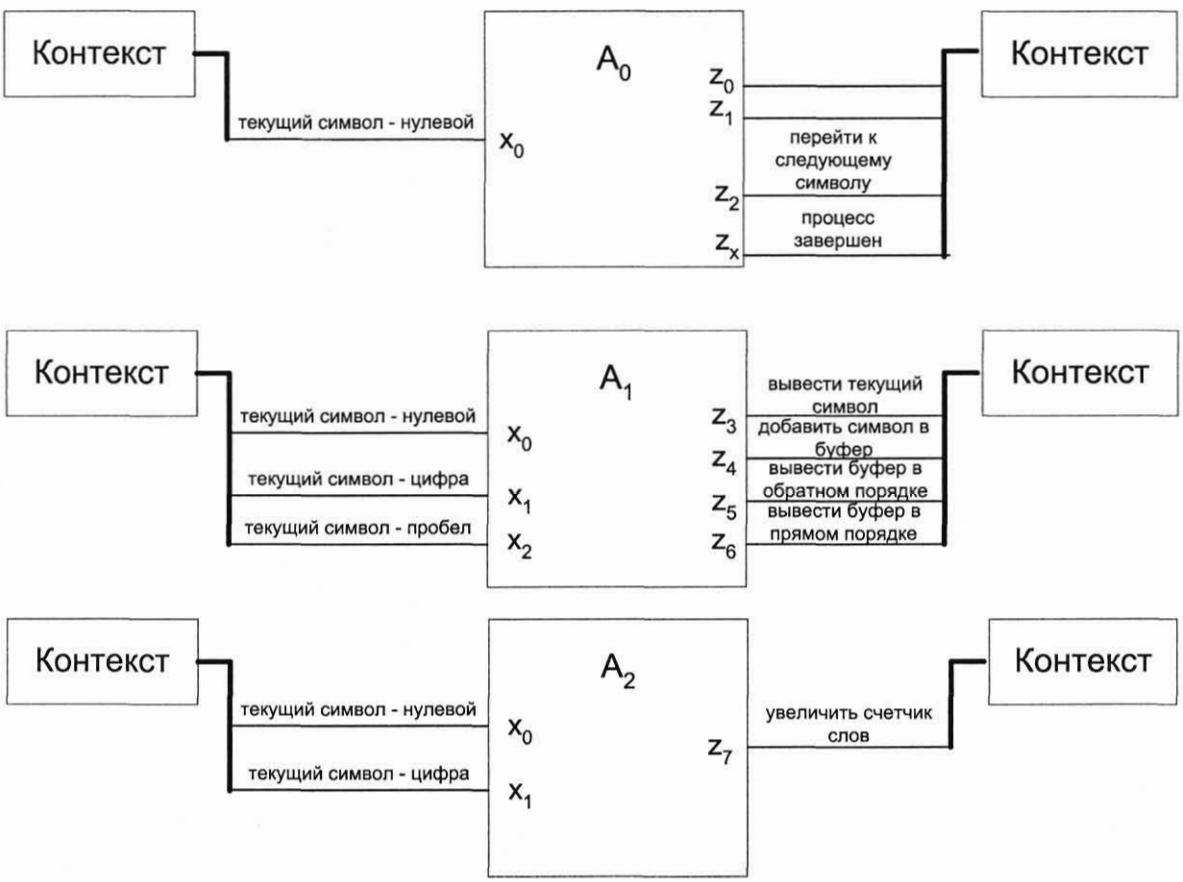
```
class A0 : public Auto {
public:
```



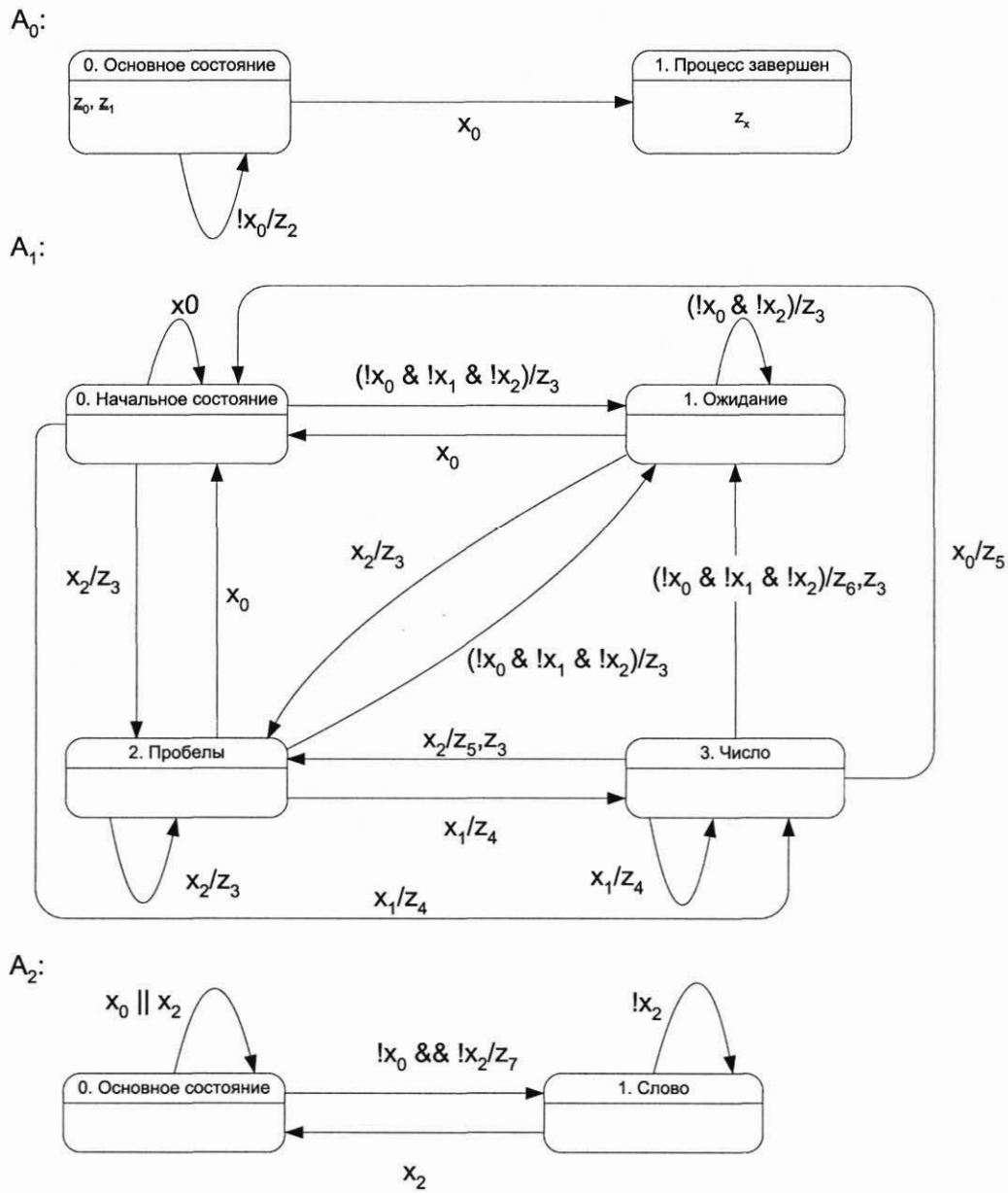
■ ***Рис. 4.*** *Схема взаимодействия автоматов*

```
  struct IO {
    virtual Input& x0() = 0;
    virtual Output& z0() = 0;
    virtual Output& z1() = 0;
    virtual Output& z2() = 0;
    virtual Output& zx() = 0;
  };
private:
  IO& io;

protected:
  virtual void execution( State& state ) {
    switch ( state ) {
    case 0:
      io.z0().activity( *this );
      io.z1().activity( *this );
      if ( io.x0().is( *this ) )
```



■ ***Рис. 5.*** *Структурные схемы классов автоматов A_0, A_1 и A_2*



■ ***Рис. 6.*** *Графы переходов классов автоматов A_0, A_1 и A_2*

```
        state = 1;
      else if ( !io.x0().is( *this ) )
        io.z2().jumpAction( *this );
      break;
    case 1:
      io.zx().action( *this );
      break;
    }
  }

public:
  A0( IO& _io, const string& _instance_name,
  System& _system )
    : Auto( _instance_name, "A0", _system )
    , io( _io ) {}
};
```

```
class A1 : public Auto {
public:
  struct IO {
    virtual Input& x0() = 0;
    virtual Input& x1() = 0;
    virtual Input& x2() = 0;
    virtual Output& z3() = 0;
    virtual Output& z4() = 0;
    virtual Output& z5() = 0;
    virtual Output& z6() = 0;
  };
private:
  IO& io;

protected:
  virtual void execution( State& state ) {
    switch ( state ) {
```

```
    case 0:
      if ( !io.x0().is( *this ) && !io.x1().is( *this )
      && !io.x2().is( *this ) ) {
        io.z3().jumpAction( *this );
        state = 1;
      } else if ( io.x1().is( *this ) ) {
        io.z4().jumpAction( *this );
        state = 3;
      } else if ( io.x2().is( *this ) ) {
        io.z3().jumpAction( *this );
        state = 2;
      } else if ( io.x0().is( *this ) )
      {}
      break;
    case 1:
      if ( !io.x0().is( *this ) && !io.x2().is( *this ) )
        io.z3().jumpAction( *this );
      else if ( io.x2().is( *this ) ) {
        io.z3().jumpAction( *this );
        state = 2;
      } else if ( io.x0().is( *this ) )
        state = 0;
      break;
    case 2:
      if ( io.x1().is( *this ) ) {
        io.z4().jumpAction( *this );
        state = 3;
      } else if ( io.x2().is( *this ) )
        io.z3().jumpAction( *this );
      else if ( !io.x0().is(*this) && !io.x1().is(*this)
      && !io.x2().is(*this)) {
        io.z3().jumpAction( *this );;
        state = 1;
      } else if ( io.x0().is( *this ) )
        state = 0;
      break;
    case 3:
      if ( io.x1().is( *this ) ) {
        io.z4().jumpAction( *this );
      } else if ( io.x2().is( *this ) ) {
        io.z5().jumpAction( *this );;
        io.z3().jumpAction( *this );;
        state = 2;
      } else if ( io.x0().is( *this ) ) {
        io.z5().jumpAction( *this );;
        state = 0;
      } else if(!io.x0().is(*this) && !io.x1().is(*this)
      && !io.x2().is(*this)) {
        io.z6().jumpAction( *this );;
        io.z3().jumpAction( *this );;
        state = 1;
      }
      break;
    }
  }

public:
  A1( IO& _io, const string& _instance_name,
  System& _system )
    : Auto( _instance_name, "A1", _system )
    , io( _io ) {}
};

class A2 : public Auto {
public:
  struct IO {
    virtual Input& x0() = 0;
    virtual Input& x2() = 0;
    virtual Output& z7() = 0;
  };
private:
  IO& io;

protected:
  virtual void execution( State& state ) {
    switch ( state ) {
    case 0:
      if ( !io.x0().is( *this ) && !io.x2().is( *this ) ) {
        io.z7().jumpAction( *this );
        state = 1;
      } else if ( io.x0().is( *this ) || io.x2().is( *this
      ) )
      {}
      break;
    case 1:
      if ( io.x2().is( *this ) ) {
        state = 0;
      }
    }
  }

public:
  A2( IO& _io, const string& _instance_name,
  System& _system )
    : Auto( _instance_name, "A2", _system )
    , io( _io ) {}
};
```

Приведенная реализация классов автоматов изоморфна графам переходов.

Завершив построение автоматной части программы, перейдем к реализации ее «контекста», реализующего такие функции, как, например, проверка на конец строки:

```
class Data {
  string in;
  string out;
  string buffer;
  string::iterator cursor;
  int nwords;

public:
  Data( const string& _in )
    : in( _in )
    , nwords( 0 )
    , cursor( in.begin() ) {}
  bool isEnd() { return in.end() == cursor; }
  bool isDigit() { return isdigit( *cursor ) != 0; }
  bool isSpace() { return isspace( *cursor ) != 0; }
  void forward() { cursor++; }
  void back() { cursor--; }
  void output() { out += *cursor; }
  void push() { buffer += *cursor; }
  void outBuffer() { out = out + buffer;
  buffer.clear(); }
  void outBufRev() { reverse( buffer.begin(),
  buffer.end() ); outBuffer(); }
  void incWords() { nwords++; }
  string getOutput() const { return out; }
  int getWordsCount() const { return nwords; }
};
```

Этот класс не содержит никакой «логики».

Теперь рассмотрим класс *MySystem* (потомок класса *stool::System*) — часть инфраструктуры системы. Этот класс создает и связывает между собой автоматы, входные и выходные воздействия, а также контекст:

```
class MySystem : public System, public A0::IO, public
A1::IO, public A2::IO {
  Data data;
  bool stopped;
  DECLARE_AUTO(A0, i0, *this, *this);
  DECLARE_AUTO(A1, i1, *this, *this);
  DECLARE_AUTO(A2, i2, *this, *this);

  DECLARE_FUNC_INPUT ( x0, bind( &Data::isEnd,
  ref( data) ), *this );
  DECLARE_FUNC_INPUT ( x1, bind( &Data::isDigit,
  ref( data) ), *this );
  DECLARE_FUNC_INPUT ( x2, bind( &Data::isSpace,
  ref( data) ), *this );
  DECLARE_AUTO_OUTPUT( z0, i1(), *this );
  DECLARE_AUTO_OUTPUT( z1, i2(), *this );
  DECLARE_FUNC_OUTPUT(      z2,      bind(
  &Data::forward, ref( data) ), *this );
  DECLARE_FUNC_OUTPUT(      z3,      bind(
  &Data::output, ref( data) ), *this );
  DECLARE_FUNC_OUTPUT( z4, bind( &Data::push,
  ref( data) ), *this );
  DECLARE_FUNC_OUTPUT(      z5,      bind(
  &Data::outBufRev, ref( data) ), *this );
  DECLARE_FUNC_OUTPUT(      z6,      bind(
  &Data::outBuffer, ref( data) ), *this );
  DECLARE_FUNC_OUTPUT(      z7,      bind(
  &Data::incWords, ref( data) ), *this );
  DECLARE_FUNC_OUTPUT(      zx,      bind(
  &MySystem::stop, ref( *this) ), *this );
protected:
  virtual void stop() { stopped = true; }
public:
  MySystem( const string& _in, Factory& _factory )
    : System( _factory )
    , data( _in )
    , stopped( false ) {}

  void run() {
    while ( !stopped )
      getChangeServer().start( i0() );
  }

  const Data& getData() const { return data; }
};
```

В инфраструктуру входит также объект, реализующий протоколирование:

```
class LogSystem : public AutoEventSync {
  int change_number;
  ostream& stream;
public:
  LogSystem( Lockable& _locakble, ostream&
  _stream )
    : AutoEventSync( _locakble )
    , change_number( 0 )
    , stream( _stream ) {}

  void preamble() {
    stream << ++change_number << ".\t";
  }

  void onEvent( const Event _event, const
  AutoEventSync::EventItem& _item ) {
    Lock lock( *this );
    switch ( _event ) {
    case AutoEventSync::E_AFTER_STATE_
    CHANGED:
      preamble();
      stream << "автомат "
        << _item.getStateEventData().state.
        getAuto().getInfo().getInstanceName()
        << "(" <<_item.getStateEventData().state.
        getAuto().getInfo().getClassName()
        << ")" << " перешел в состояние: " <<
        _item.getStateEventData().state
        << " (старое состояние: " << _item.get
        StateEventData().old_state << ")"
        << "." << std::endl;
      break;
    case AutoEventSync::E_AFTER_INPUT_
    CHECKED:
      preamble();
      stream << "опрошено входное воздей-
      ствие "
        << _item.getInputResultEventData().input.
        getName()
        << ", результат: " << _item.getInputResult
        EventData().result
        << "." << std::endl;
      break;
    case AutoEventSync::E_AFTER_OUTPUT_
    ACTIVATED:
      preamble();
      stream << "выполнено выходное воздей-
      ствие "
        << _item.getOutputEventData().output.get
        Name() << "." << std::endl;
      break;
    case AutoEventSync::E_AFTER_ EXCEPTION:
      preamble();
      std::exception* ex = _item.getException
      EventData().exception;
      if ( ex )
        stream << "возникло исключение: " <<
        ex->what();
      else
        stream << "возникло неизвестное ис-
        ключение";
      stream << std::endl;
      break;
    }
  }
};
```

При этом функция *main* реализуется следующим образом:

```
int main()
{
  const int INPUT_SIZE = 256;
  char input[ INPUT_SIZE + 1 ];
  cout << "input string: ";
  cin.getline( input, INPUT_SIZE );
  Factory factory;
  ofstream log_stream("log.txt");
  LogSystem log( factory.makeLockable(),
  log_stream );
  MySystem system( input, factory );
  system.getAutoEventService().inject( log );
  system.run();
  cout << "output: " << system.getData().get
  Output() << endl;
  cout << "words count: " << system.getData().get
  WordsCount() << endl;
  cout << "press any key...";
  _getch();
  return 0;
}
```

Как следует из рассмотренного примера, его реализация практически не содержит никакой «лишней информации» (кроме протоколирования). В реальных системах объем кода, обеспечивающего протоколирование, существенно меньше по сравнению с размером кода остальной части системы.

Таким образом, можно заключить, что предложенный подход позволил устранить недостатки SWITCH-технологии, перечисленные во введении.

Литература

1. **Сацкий С.** Дизайн шаблона конечного автомата на C++ //RSDN Magazine. — 2003. — № 1. — С. 20–24.
2. **Буч Г., Рамбо Д., Джекобсон А.** Язык UML. Руководство пользователя. — М.: ДМК, 2000. — 432 с.
3. **Любченко В. С.** О бильярде с Microsoft Visual C++ 5.0 //Мир ПК. — 1998. — № 1. — С. 202–206.
4. **Шалыто А. А., Туккель Н. И.** Танки и автоматы // BYTE/Россия. — 2003. — № 2. — С. 69–73. http://is.ifmo.ru (раздел «Статьи»).
5. **Туккель Н. И., Шалыто А. А.** Система управления танком для игры «Robocode». Объектно-ориентированное программирование с явным выделением состояний. 49 с. http://is.ifmo.ru (раздел «Проекты»).
6. **Шалыто А. А., Туккель Н. И.** Программирование с явным выделением состояний // Мир ПК. — 2001. — № 8. — С. 116–121; №9. — С. 132–138 http://is.ifmo.ru (раздел «Статьи»).
7. **Шалыто А. А., Туккель Н. И.** Реализация автоматов при программировании событийных систем // Программист. — 2002. — № 4. — С. 74–80. http://is.ifmo.ru (раздел «Статьи»).
8. **Шалыто А. А.** SWITCH-технология. Алгоритмизация и программирование задач логического управления. — СПб.: Наука, 1998. — 628 с.
9. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.** Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб.: Питер, 2001. — 368 с.

Соложенцев Е. Д.

Сценарное логико-вероятностное управление риском в бизнесе и технике. — СПб.: Изд. дом «Бизнес-пресса», 2004. — 420 с.

Рассмотрены методологические аспекты сценарного логико-вероятностного (ЛВ) управления риском неуспеха, следующие из анализа связей управления и риска, персонала и риска, а также управления риском на стадиях проектирования, испытаний и эксплуатации экономических и технических систем.

Изложены теоретические основы сценарного ЛВ-управления риском в бизнесе и технике, включающие в себя ЛВ-исчисление, ЛВ-методы, методологию и технологию автоматизированного структурно-логического моделирования, ЛВ-теорию риска с группами несовместных событий (ГНС). Описаны методики для сценарного ЛВ-управления риском в проблемах классификации, инвестирования и эффективности с произвольными распределениями случайных событий.

Приведены ЛВ-модели риска и результаты компьютерных исследований для кредитных рисков, риска мошенничеств в бизнесе, риска портфеля ценных бумаг, риска потери качества и эффективности, и надежности систем со многими состояниями элементов. Рассматривается большое число новых задач оценки, анализа и управления риском. ЛВ-модели риска показали почти вдвое большую точность и в семь раз большую робастность, чем известные модели риска. Описаны программные средства для решения задач риска на основе ЛВ-методов, ЛВ-теории с ГНС и алгебры кортежей.

Книга предназначена для специалистов в области риска экономических, технических и организационных систем, для решения задач риска в проблемах классификации, инвестиций и эффективности, а также для студентов и аспирантов соответствующих университетов.

УДК 681.391.1

ПЕРЕДАЧА СО СКРЫТЫМ СМЫСЛОМ

И. Л. Ерош,
д-р техн. наук, профессор
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

В статье решается задача, в которой сообщение, принятое различными группами пользователей, может быть прочитано не одинаково, так как в сообщении заложен многократный скрытый смысл. При этом передача сообщения может выполняться как в открытом виде, так и в зашифрованном (во втором случае его необходимо предварительно расшифровать). Рассмотрены два варианта решения этой задачи. В первом при передаче сообщения в осмысленный текст вставляется некоторый фрагмент, который разными группами пользователей должен читаться по-разному. Во втором каждая группа пользователей читает сообщение, адресованное только ей, остальные пользователи получают бессмысленный набор символов.

The article is dedicated to resolution of a problem of interpretation of messages being received by different groups of users that may comprehend given message in different ways,because of multiple hidden implications associated with them. Message itself can be transmitted as well in clear as encrypted format. Prior decryption is required in the second case. Two possible resolutions of this problem are considered. The first resolution implies the insertion of a certain fragment into meaningful text of a message. Different users should interpret this fragment in different way. The second resolution assumes that each group of the users receives meaningful message addressed especially to it. The rest of the users are getting meaningless set of characters.

Введение

В ряде случаев при передаче сообщений требуется обеспечить дополнительную секретность, смысл которой состоит в том, чтобы некоторые пользователи, имеющие доступ к расшифрованному сообщению, не получали полной информации о точных датах, суммах и других данных, предназначенных для особого круга лиц из числа пользователей.

Реализация передачи со скрытым смыслом

Рассмотрим задачу передачи сообщения, в которое заложен скрытый смысл, при этом для каждой группы пользователей (получателей сообщения) этот смысл может быть различным.

Для выявления скрытого смысла сообщения будем использовать булевы преобразования двоичных последовательностей. Напомним основные теоремы из [1], с помощью которых будет обеспечиваться скрытый смысл полученного сообщения.

Пусть $A(a_1, a_2, \ldots, a_n)$ и $B(b_1, b_2, \ldots, b_n)$ — две двоичных последовательности длины n. Каждый элемент b_i является результатом преобразования булевой функцией F элемента a_i и некоторых элементов из «окружения» a_i. Так, если последовательность A имеет вид 0100100101001110 и задана функция $F = 0^* \urcorner 3 \vee -2$, это означает, что каждый элемент

$$b_i = a_i^* \urcorner (a_{i-3}) \vee a_{i-2},$$

где * — конъюнкция, $\urcorner$ — инверсия, $\vee$ — дизъюнкция.

Последовательность B в этом случае имеет вид 0101001001010111. В шестнадцатеричном коде такое преобразование будет записываться как 494E → → 5257, а в кодах ASCII этому преобразованию соответствует in → rw.

Теорема 1. Для того чтобы существовала булева функция F, такая, что $F(A) = B$, *необходимо и достаточно*, чтобы A была ненулевой последовательностью.

Теорема 2. Пусть $A_1, A_2, \ldots, A_s$ и $B_1, B_2, \ldots, B_s$ — два набора двоичных последовательностей длины n; s — целое, равное числу пар последовательностей. Для того чтобы существовала функция F, преобразующая любую последовательность A_i в последовательность B_i, *достаточно*, чтобы в A_i не было двух последовательностей, связанных сдвигом.

Для теоремы 2 можно сформулировать *необходимое и достаточное* условие: в таблице истинности функции F, построенной по методике работы [1], не должно быть двух одинаковых строк (набо-

ров), на которых функция принимала бы различные значения. К сожалению, это *необходимое и достаточное* условие проверяется труднее, чем просто *достаточное*.

Например, пусть требуется обеспечить преобразование A → A, 5 → D. Последовательности A и 5 связаны сдвигом, т. е. не удовлетворяют *достаточному* условию, но из таблицы истинности видно, что в ней нет одинаковых строк, на которых функция принимала бы разные значения (*необходимое и достаточное* условие). Поэтому находится общая функция преобразования: $F = \neg 1$. Если же взять A → 2, 5 → D, то такая функция не может быть построена, так как на одинаковых наборах в таблице истинности функция должна принимать различные значения.

Рассмотрим два варианта решения задачи, которые реализуются с помощью различных классов булевых функций.

Вариант 1. В передачу произвольной длины вставляется некоторый фрагмент, при обработке которого функцией F_i каждый пользователь (например, солдат, офицер и генерал) читает текст, предназначенный только ему, т. е. фрагмент содержит разный смысл для разных пользователей (рис. 1).

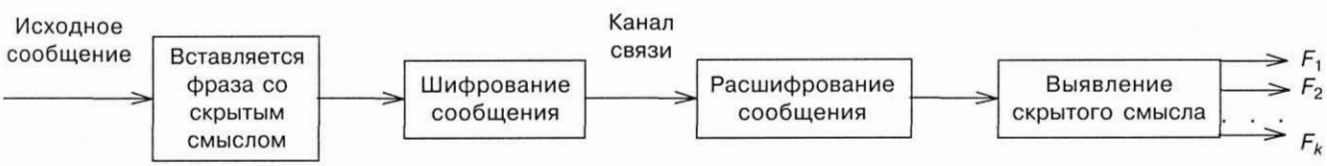
Пусть, например, выбраны следующие вставляемые в осмысленный текст фразы (таблица, столбец 1).

В первом столбце таблицы приведены фразы на русском языке, соответствующие им фразы на английском языке и английский вариант в кодах ASCII в шестнадцатеричном представлении. Во втором, третьем и четвертом столбцах приведены фразы, которые должны читаться соответственно солдатом (с помощью функции F_1), офицером (с помощью функции F_2) и генералом (с помощью функции F_3).

Каждую функцию F_i будем находить для преобразования 16-байтового блока сообщения. Если блоки сообщения оказываются короче 16 байт, то в конце фразы добавляются пробелы. Можно находить преобразования любой длины; блоки по 16 байт выбраны для упрощения примера.

Солдат, офицер и генерал этот текст читают совершенно по-разному с помощью собственных функций F_i. Для данного примера, используя процедуру, описанную в [1], вычисляем:

$$F_1 = \neg 5^* \neg 4^* 0^* \neg - 1^* - 2^* \neg - 3^* \neg - 4 \vee \neg 3^* \neg 1^* 0^* \neg - 2^* - 3^* \neg - 5 \vee$$
$$4^* \neg 3^* \neg 2^* \neg 1^* \neg 0^* - 1^* \neg - 3 \vee 4^* 0^* \neg - 1^* \neg - 2^* \neg - 4 \vee$$
$$8^* 2^* \neg 1^* \neg 0^* \neg - 1^* \neg - 3^* \neg - 6 \vee 7^* 2^* 0^* - 1^* - 7 \vee$$
$$2^* \neg 1^* 0^* - 1^* - 4 \vee 4^* 0^* \neg - 1^* - 2^* - 3 \vee$$
$$\neg 6^* 3^* \neg 0^* \neg - 1^* \neg - 5^* \neg - 6^* - 7 \vee 6^* 2^* 0^* \neg - 2^* \neg - 6^* \neg - 7 \vee$$
$$5^* \neg 3^* \neg 2^* 1^* - 1^* \neg - 2^* - 5 \vee 7^* \neg 3^* \neg 2^* \neg 1^* 0^* \neg - 1^* - 2 \vee$$
$$3^* \neg 1^* - 1^* - 3^* \neg - 4^* - 7 \vee \neg 5^* \neg 4^* 2^* \neg 1^* \neg 0^* - 8 \vee$$
$$6^* \neg 2^* \neg 1^* 0^* \neg - 2^* \neg - 3 \vee 7^* 2^* \neg - 1^* \neg - 2 \vee$$



■ ***Рис. 1.*** *Выявление скрытого смысла по варианту 1*

■ ***Таблица.*** *Выявление смысла исходных сообщений разными группами пользователей (с функциями F_i)*

Фраза, включенная в передаваемое сообщение	Солдат с F_1 читает	Офицер с F_2 читает	Генерал с F_3 читает
Сейчас хорошая погода Good weather now 474F4F44574 54154484552 4E4F572020	Оставаться на месте To remain on a place 544F52454D41 494E4F4E4150 4C414345	Готовь оружие Prepare the weapon 50524550415245 54484557454150 4F4E	Позвонить командующему Call to commander 43414C4C54484543 4F4D4D414E444552
Пора обедать Time tohave dinner 54494D4554 4F4841564544 494E4E4552	Взять противогазы To take gas masks 544F54414B4547 41534D41534B53 2020	Проверить охрану To check up guard 544F434845434B 55504755415244 2020	Завтра наступление Tomorrow approach 544F4D4F52524F57 415050524F414348
Полная луна The complete moon 544845434F4D504C 4554454D4F4F4E20	Отбой через час Tosleep in one hour 544F534C454550 494E4F4E45484F 5552	Танцы до утра Dance till morning 44414E43455449 4C4C4D4F524E 494E47	Полезная встреча Useful meeting 55534546554C4D45 4554494E47202020

4*3*⌉2*0⌉5*0*⌉–1*–2*–3∨8*2*0*–5*–7∨
8*⌉5*1*0*–2*–8∨4*⌉3*1*0*–1∨
⌉3*⌉2*⌉0*–1*⌉–3*⌉–4*–5∨2*⌉0*⌉–1*⌉–2*⌉–3*–8∨
⌉2*0*⌉–1*⌉–5*–9∨9*⌉3*2*⌉0*–1*⌉–3*–4∨
3*2*1*–4*⌉–6∨0*⌉–1*–2*⌉–3*⌉–4*–8∨⌉6*2*⌉1*0*–1∨
⌉4*3*2*⌉1*0∨7*1*⌉0*⌉–2*⌉–5*–6∨⌉6*2*⌉1*0*⌉–6*–7∨
⌉4*2*⌉1*⌉0*⌉–1*–2*⌉–4∨⌉2*1*⌉0*⌉–1*–2*⌉–3*⌉–5*⌉–8∨
5*⌉3*1*0*–5∨3*⌉1*0*–1*–7∨
⌉1*⌉0*–1*–2*–8∨⌉4*3*⌉–1*–2*⌉–4∨⌉6*3*⌉1*0*–1∨
5*⌉1*⌉0*–1*–2*⌉–5∨3*⌉2*–1*⌉–3*–4*–6∨
7*4*⌉3*⌉0*–1*⌉–2*–3∨5*⌉1*0*–9∨⌉9*3*⌉0*⌉–1*–4∨
6*⌉5*1*0*–1*–2∨7*2*⌉0*⌉–1∨
2*1*⌉0*–1*⌉–5∨5*⌉1*⌉0*⌉–1*–5∨⌉3*⌉2*⌉1*0*–2*⌉–8∨
7*⌉6*3*1*⌉–1*⌉–5*⌉–6∨
⌉4*1*⌉0*–1*⌉–3*–5*⌉–7*⌉–8∨8*2*0*–2*–4∨
0*–2*–3*–4∨3*⌉1*⌉–1*⌉–2*⌉–3∨
⌉5*1*⌉0*⌉–1*⌉–2*⌉–3*⌉–4∨⌉3*⌉0*⌉–1*–3*⌉–7∨
1*⌉0*⌉–1*⌉–2*⌉–3*⌉–4*⌉–9;

F_2 = 12*⌉3*⌉1*⌉0*⌉–1*⌉–2*–3∨4*0*⌉–1*⌉–2*⌉–4∨
7*3*⌉2*⌉0*⌉–1*–3*⌉–6∨⌉5*1*⌉0*–1*⌉–3*–5∨
2*1*0*–1*⌉–8∨⌉7*2*0*–1*–2∨⌉3*⌉2*⌉1*⌉0*–3*–7∨
1*⌉0*⌉–1*⌉–2*–4∨⌉3*⌉2*1*0*⌉–1*⌉–2∨⌉6*3*1*⌉0*
⌉–1*–3*⌉–6*–7∨⌉8*5*⌉2*1*–1*⌉–2*–3*–5∨
4*⌉2*⌉1*0*⌉–3*⌉–7∨⌉3*2*1*⌉0*–2*–6∨
2*1*–1*–6*⌉–7∨8*3*⌉–1*–8∨⌉8*2*⌉1*0*–1∨
5*1*0*⌉–1*–3∨⌉0*⌉–1*⌉–2*–4*–5∨
⌉2*⌉1*⌉0*⌉–1*⌉–2*⌉–3*–9∨12*⌉1*0*⌉–6∨
3*⌉1*0*⌉–1*–9∨⌉9*⌉3*2*⌉0*⌉–2*⌉–3*–4∨
⌉8*3*⌉2*1*⌉–1*–2∨⌉5*⌉4*0*–2*–3∨
6*⌉4*⌉1*⌉0*⌉–1*⌉–3∨⌉5*1*–1*–2*⌉–7∨5*⌉1*0*–5∨
3*⌉1*–2*–7∨⌉6*⌉3*⌉1*⌉0*⌉–1*⌉–2*–3∨0*–3*–4*⌉–5∨
6*⌉4*0*⌉–1*⌉–3*⌉–4*–6∨
⌉4*2*⌉1*⌉0*–2*⌉–3*⌉–4∨⌉4*3*⌉0*⌉–1*–2*⌉–5∨
3*⌉2*0*–1*⌉–3*⌉–6∨3*⌉2*⌉0*–1*⌉–3*–4*–6∨
7*4*⌉3*⌉0*–1*⌉–2*–3∨3*2*⌉1*⌉0*–2∨
2*1*–2*–4*–6*⌉–7∨9*8*⌉5*1*–2∨⌉7*1*0*–1*–6∨
1*⌉0*⌉–1*–4*–5∨5*2*⌉1*0*⌉–2∨⌉5*⌉1*0*–2*–4*⌉–6∨
3*⌉1*–3*⌉–5*–13∨⌉6*4*⌉1*–2*⌉–9∨6*3*⌉2*0*⌉–1∨
5*⌉1*⌉0*–1*–2*⌉–8∨4*0*–1*–2∨⌉5*1*⌉–1*–2*–3∨
⌉5*1*⌉–1*⌉–2*⌉–3*⌉–4∨⌉3*⌉2*⌉1*–2*⌉–6;

F_3 = 0*⌉–1*–2*⌉–4*⌉–6∨⌉8*⌉4*2*⌉1*⌉0*–2*⌉–3∨
⌉4*1*⌉–1*–2*⌉–3*⌉–4*⌉–12∨
⌉13*3*⌉2*⌉–1*–2*⌉–4∨⌉3*2*⌉0*⌉–1*⌉–2*–3∨
⌉7*⌉1*0*⌉–2*⌉–7∨⌉6*⌉3*2*0*–2*⌉–3∨5*0*⌉–1*⌉–3∨
4*3*1*⌉–1*⌉–6∨2*1*0*–1*⌉–8∨3*⌉2*1*0*⌉–1∨
⌉1*0*–1*⌉–2*–4∨⌉3*⌉2*⌉1*–1*–3*–7∨
⌉4*2*⌉–1*⌉–2*–4∨⌉1*⌉0*⌉–1*–2*–3*⌉–4∨
⌉3*2*0*⌉–1*–4*–6∨⌉7*4*⌉1*0*⌉–1*–2*⌉–3∨
6*⌉2*0*⌉–2*⌉–3∨⌉7*3*1*⌉0*⌉–1*⌉–2*⌉–6*–7∨
6*3*⌉0*–1*–2*⌉–3∨1*0*–1*–6*⌉–7∨
1*0*–1*–2*⌉–8∨8*⌉4*0*–1*–3∨10*⌉0*–1*–4*–6∨
⌉5*3*2*–3*–4∨3*⌉1*0*⌉–5*–9∨
⌉9*⌉3*2*⌉0*–1*⌉–3*–4∨3*1*⌉0*⌉–1*⌉–2*–5∨
1*–1*⌉–2*⌉–3*–5*–7∨6*⌉1*–1*–3*–7∨
3*1*–2*⌉–4*⌉–5∨⌉4*⌉2*0*–2*–3∨3*⌉1*⌉0*–2*–7∨
4*⌉2*⌉1*⌉–1*⌉–4∨2*⌉0*⌉–1*⌉–2*⌉–3*⌉–6∨
2*⌉1*⌉–1*⌉–4*–5*⌉–7∨⌉2*1*⌉0*⌉–1*–2*⌉–3*⌉–5*⌉–8∨
5*⌉3*0*–5∨6*⌉2*⌉1*0*–1∨2*⌉0*–1*–2*–8∨
⌉5*2*⌉0*–1*–2*⌉–5∨5*1*⌉0*–1*⌉–2*–8∨⌉5*⌉3*⌉1*
–1*⌉–2*–3*⌉–4∨2*–1*–2*⌉–3*⌉–4∨
⌉2*0*–2*–3*⌉–5∨6*⌉1*0*⌉–1*–11∨⌉8*3*1*0*⌉–1*–5∨
⌉7*2*1*0*–1∨⌉3*⌉2*0*–2*–3∨
⌉3*1*⌉0*⌉–1*⌉–2*–6∨5*⌉3*⌉2*1*–5*⌉–7∨
4*⌉2*⌉1*0*⌉–8∨⌉6*⌉1*⌉0*–2*–4*–8∨
⌉3*⌉1*0*⌉–2*⌉–5*–13∨4*⌉2*⌉1*–1*⌉–3*–14∨
7*1*⌉–1*–3*⌉–7∨⌉3*2*1*0*–3*⌉–7∨
7*⌉2*0*–1*⌉–2*⌉–3∨⌉5*1*⌉–1*–2*–3∨
⌉3*⌉2*–1*–4*–5∨4*⌉2*⌉1*⌉0*⌉–1*⌉–2.

Булевы преобразования выполняются очень быстро и практически не влияют на скорость передачи сообщений.

Многократно вложенный смысл проиллюстрируем примером. Возьмем несколько сообщений, различных по смыслу:

A. Вперед в атаку!
B. Держим оборону.
C. Какие будут распоряжения?
D. Срочно назад!

В английском варианте и кодах ASCII это выглядит так:

A. Forwards in attack!
ASCII: 464F525741524453494E41545441434B.
B. We keep a defense.
ASCII: 57454B4545504144454645E53452020.
C. What orders will be?
ASCII: 574841544F524445525257494C4C4245.
D. Urgently we back!
ASCII: 555247454E544C5957454241434B2020.

Функция F, преобразующая $F(A) = B$; $F(B) = C$; $F(C) = D$, имеет вид

F = 1*0*⌉–1*–2*⌉–7∨⌉8*0*–1*–2*⌉–3∨0*–3*–4∨
⌉4*⌉2*⌉0*⌉–1*–2*⌉–3*–4∨3*⌉1*⌉0*–2*–5∨
⌉3*⌉1*0*⌉–3*⌉–4*–5∨4*⌉1*⌉0*⌉–1*⌉–2∨
5*⌉–1*⌉–2*⌉–3*–5*⌉–8∨3*2*1*0*⌉–6∨1*0*–1*–2*⌉–∨
⌉8*3*0*–4∨⌉6*⌉0*⌉–1*–2*⌉–3*–4*–7∨
⌉7*⌉3*⌉2*1*⌉–1*–3*–4*⌉–5∨⌉6*4*⌉2*⌉1*⌉–1*⌉–2∨
4*⌉2*⌉1*⌉–1*⌉–2*⌉–3*⌉–7∨4*2*⌉1*⌉–1*⌉–4*⌉–5∨
7*5*3*1*⌉0*–3∨5*⌉–1*–2*–4∨
6*⌉3*1*–2*⌉–5*⌉–7∨5*⌉2*0*⌉–1*⌉–6*–7∨
4*⌉3*⌉1*⌉0*–1*⌉–3*⌉–7∨⌉3*2*⌉1*0*⌉–1*–3∨
⌉2*1*⌉–1*–2*⌉–3*–4*⌉–5*⌉–6∨6*–1*–3*–8∨
4*3*⌉2*0*⌉–6∨⌉1*⌉0*⌉–1*–2*–3∨5*3*⌉1*⌉–3*⌉–5∨
7*1*0*–1∨⌉5*0*–1*–2*⌉–3∨1*–5*–6*–7∨
6*4*2*⌉0*⌉–6∨⌉7*⌉1*0*⌉–1*⌉–2*–4*–6∨
6*⌉3*⌉2*0*–2*–4∨5*1*–1*⌉–2*–5*–7∨
⌉6*2*⌉1*0*!–1*–2*–4∨⌉5*⌉2*⌉1*⌉0*⌉–1*–3∨
⌉2*⌉1*⌉–1*⌉–2*⌉–3*–6∨4*⌉3*0*⌉–1*–2*⌉–4∨
2*⌉1*⌉0*⌉–1*–2*⌉–4*⌉–5*–8∨
⌉5*3*1*–1*⌉–3*⌉–7∨3*2*⌉1*⌉0*–2∨
3*⌉1*0*–1*⌉–2∨4*2*⌉1*⌉–1*⌉–4*–6∨
4*2*⌉1*0*⌉–6*–7∨6*5*2∨⌉3*1*0*⌉–2*–7∨
4*⌉1*⌉0*–1*–2∨5*3*2*⌉1*–6∨4*⌉3*2*1*⌉0*–2∨
⌉5*3*⌉0*⌉–1*–3*–5∨
⌉5*1*⌉–1*⌉–2*⌉–3*⌉–4*⌉–5∨⌉3*⌉2*⌉1*⌉0*⌉–1*⌉–2*
⌉–3*⌉–4∨⌉3*1*0*⌉–2*⌉–3*⌉–6∨0*–2*–3*–9∨
7*⌉1*–1*–3*–5*–6∨⌉7*2*1*0*–2∨
⌉3*2*⌉1*0*⌉–4*⌉–6∨⌉5*0*⌉–1*–2*–3∨
⌉3*2*⌉1*⌉0*–1*⌉–3*–4*⌉–8∨⌉6*1*0*–1*⌉–2*⌉–3∨
7*5*1*–1*⌉–2*–3*–5∨5*3*⌉2*⌉1*⌉0*–1*⌉–4*–7∨
⌉1*0*⌉–1*–4*–6*⌉–7∨⌉8*⌉7*⌉3*2*0*–2*⌉–4∨
⌉6*⌉5*⌉4*⌉2*⌉1*⌉0*–3*–5∨⌉4*⌉1*⌉0*–2*–4*–6∨
⌉6*2*0*⌉–1*⌉–2*⌉–3*⌉–4∨4*⌉3*⌉2*⌉1*⌉0*–3∨
4*⌉1*–3*–4.

Следует заметить, что никто из пользователей, имея в руках преобразователь с функцией F, не получает информации о том, когда и какая фраза, переданная центром, в какую фразу будет преобразована с помощью этой функции. Этой информацией владеет только центр.

Вариант 2. В рассмотренном примере (вариант 1) можно использовать произвольный, но ограниченный набор фраз, которые каждый абонент воспринимает по-своему. В варианте 2 система построена таким образом, чтобы передаваемый текст был совершенно произвольным без каких-либо ограничений. Это можно сделать с использованием так называемых обратимых булевых функций, которые лежат в классе линейных [1], т. е. таких, что уравнение $F(X) = B$ всегда разрешимо при любых B относительно X. Тогда, снабдив каждую группу абонентов своей функцией F, можно передавать по каналу некоторый текст (X), а каждый абонент, используя свою функцию, будет читать только то, что предназначено центром непосредственно ему. Остальные абоненты будут извлекать из передаваемого текста бессмысленный набор букв и символов (рис. 2)

Если центр хочет передать конкретному пользователю некоторое сообщение B, он берет функцию F этого пользователя и вычисляет двоичную последовательность $F^{-1}(B) = A$, затем передает A в зашифрованном виде по каналу связи. После расшифрования только тот получатель, кто владеет преобразователем с этой конкретной функцией, может восстановить сообщение: $F(A) = B$. Остальные пользователи, пытаясь восстановить сообщение с помощью своих функций, получают бессмысленную последовательность символов. Не представляет трудности нахождение таких функций для побуквенного преобразования сообщения, однако в этом случае все сводится к обычному шифру замены, который легко вскрывается. Лучше выполнять преобразования сразу нескольких символов (например, 10–20 символов текста), а еще лучше — блоками битов, не кратных длине символов (т. е. не кратных четырем битам). Однако в приводимых примерах для большей наглядности длины блоков выбираются кратными четырем битам. Операции прямого и обратного преобразования выполняются по mod 2.

Пусть требуется передать конкретному получателю фразу: We act in a campaign tomorrow. В кодах ASCII эта фраза с учетом пробелов между словами будет выглядеть так: 57452041435420494E 204120434F4D504149474E20544F4D4F52524F5720.

■ ***Рис. 2.*** *Выявление скрытого смысла по варианту 2*

Разобьем сообщение на блоки по 20 бит, что будет соответствовать 2,5 буквам в каждом блоке:

57452 | 04143 | 54204 | 94E20 | 41204 | 34F4D |

50414 | 9474E | 20544 | F4D4F | 52524 | F5720.

Возьмем четырехразрядное обратимое преобразование с добавленным пятым столбцом и пятой строкой вида

$$F_1 = \begin{pmatrix} 01100 \\ 11111 \\ 01111 \\ 00111 \\ 00001 \end{pmatrix}; \quad F_1^{-1} = \begin{pmatrix} 01100 \\ 00110 \\ 10110 \\ 10101 \\ 00001 \end{pmatrix}.$$

Произведение этих матриц дает единичную матрицу: $F_1 \times F_1^{-1} = I$.

Правый столбец и нижняя строка этих матриц позволяют учитывать инверсию элементов при умножении по mod 2.

Умножим матрицу F_1^{-1} на фрагменты текста по четыре блока каждый. Последний блок состоит из всех единиц, т. е. может быть записан как FFFFF:

$$\begin{pmatrix} 01100 \\ 00110 \\ 10110 \\ 10101 \\ 00001 \end{pmatrix} \times \begin{pmatrix} 57452 \\ 04143 \\ 54204 \\ 94E20 \\ FFFFF \end{pmatrix} = \begin{pmatrix} 50347 \\ C0C24 \\ 97876 \\ FC9A9 \\ FFFFF \end{pmatrix};$$

$$\begin{pmatrix} 01100 \\ 00110 \\ 10110 \\ 10101 \\ 00001 \end{pmatrix} \times \begin{pmatrix} 41204 \\ 34F4D \\ 50414 \\ 9474E \\ FFFFF \end{pmatrix} = \begin{pmatrix} 64B59 \\ C435A \\ 8515E \\ EE9EF \\ FFFFF \end{pmatrix};$$

$$\begin{pmatrix} 01100 \\ 00110 \\ 10110 \\ 10101 \\ 00001 \end{pmatrix} \times \begin{pmatrix} 20544 \\ F4D4F \\ 52524 \\ F5720 \\ FFFFF \end{pmatrix} = \begin{pmatrix} A686B \\ A7204 \\ 87740 \\ 8DF9F \\ FFFFF \end{pmatrix}.$$

В результате получим закодированный текст: 50347C0C2497876FC9A964B59C435A8515EEE9EFA 686BA7204877408DF9E.

Только некоторые сочетания символов соответствуют в этом тексте каким-то использованным буквам кода ASCII, т. е. текст полностью бессмысленный. Однако если полученное сообщение преобразовать функцией F_1, то будет восстановлен передаваемый текст:

$$\begin{pmatrix} 01100 \\ 11111 \\ 01111 \\ 00111 \\ 00001 \end{pmatrix} \times \begin{pmatrix} 50347 \\ C0C24 \\ 97876 \\ FC9A9 \\ FFFFF \end{pmatrix} = \begin{pmatrix} 57452 \\ 04143 \\ 54204 \\ 94E20 \\ FFFFF \end{pmatrix};$$

$$\begin{pmatrix} 01100 \\ 11111 \\ 01111 \\ 00111 \\ 00001 \end{pmatrix} \times \begin{pmatrix} 64B59 \\ C435A \\ 8515E \\ EE9EF \\ FFFFF \end{pmatrix} = \begin{pmatrix} 41204 \\ 34F4D \\ 50414 \\ 9474E \\ FFFFF \end{pmatrix};$$

$$\begin{pmatrix} 01100 \\ 11111 \\ 01111 \\ 00111 \\ 00001 \end{pmatrix} \times \begin{pmatrix} A686B \\ A7204 \\ 87740 \\ 8DF9F \\ FFFFF \end{pmatrix} = \begin{pmatrix} 20544 \\ F4D4F \\ 52524 \\ F5720 \\ FFFFF \end{pmatrix}.$$

Действительно, результат преобразования имеет вид: 574520414354204 94E204120434F4D5041 49474E20544F4D4F52524F5720, что полностью совпадает с исходным сообщением: We act in a campaign tomorrow.

Возьмем теперь в качестве длины блока 12 бит сообщения, т. е. каждый блок будет соответствовать 1,5 букве. Выполним то же обратное преобразование с первыми четырьмя блоками:

$$\begin{pmatrix} 01100 \\ 00110 \\ 10110 \\ 10101 \\ 00001 \end{pmatrix} \times \begin{pmatrix} 574 \\ 520 \\ 414 \\ 352 \\ FFF \end{pmatrix} = \begin{pmatrix} 134 \\ 746 \\ 232 \\ E9F \\ FFF \end{pmatrix}.$$

Полученный фрагмент является отличным от предыдущего случая и также полностью бессмысленным. Однако после обработки его прямым преобразованием F получаем

$$\begin{pmatrix} 01100 \\ 11111 \\ 01111 \\ 00111 \\ 00001 \end{pmatrix} \times \begin{pmatrix} 134 \\ 746 \\ 232 \\ E9F \\ FFF \end{pmatrix} = \begin{pmatrix} 574 \\ 520 \\ 414 \\ 352 \\ FFF \end{pmatrix},$$

т. е. 574520414352. Это и есть начало передаваемой фразы: We act ... .

Заметим, что для реализации варианта 2 подходит любое обратимое преобразование

$$F: \{0,1\}^n \to \{0,1\}^n.$$

Рассмотрим множество матриц размера (n, n) с элементами и операциями из поля Галуа GF(2), т. е. будем выполнять операции сложения и умножения по mod 2. Множество таких матриц конечно и равно 2^{n^2}. Исключим из этого множества все вырожденные матрицы с определителем, равным 0 [в поле GF(2)]. Оставшееся множество матриц с определителем, равным 1, образует некоммутативную группу в поле GF(2). Для некоторой матрицы F найдем обратную матрицу F^{-1} такую, что $F \times F^{-1} = I$ — единичная матрица. Пусть сообщение, которое нужно передать, есть B. Найдем произведение $F^{-1} \times B = A$. Эту строку зашифруем и передадим по каналу связи. После расшифрования толь-

ко пользователь, который владеет матрицей преобразования F, может получить $FA = B$. Остальные будут читать бессмысленный текст. Если разных пользователей снабдить различными функциями F_i, то можно обеспечить адресную передачу сообщений.

Если имеется некоторое множество преобразований одинаковой размерности n: $F_1, F_2, \ldots, F_k$ и соответствующие им обратные преобразования F_1^{-1}, $F_2^{-1}, \ldots, F_k^{-1}$, то можно получить прямое преобразование $F = F_1, F_2, \ldots, F_k$, а обратное $F^{-1} = F_k^{-1}, F_{k-1}^{-1}, \ldots, F_1^{-1}$, так как множество матриц F образует некоммутативную группу по умножению с определителем, равным 1, над полем Галуа GF(2). Это позволяет увеличить количество различных преобразований и снабжать пользователей сети разными функциями F_i для адресной передачи сообщений.

Можно использовать матрицы с определителем, равным 1, произвольной различной размерности, меняя при этом и размеры преобразуемых блоков. Например, можно взять такие матрицы:

$$F_1 = \begin{pmatrix} 101 \\ 010 \\ 011 \end{pmatrix}, \quad F_2 = \begin{pmatrix} 1101 \\ 0101 \\ 1110 \\ 1100 \end{pmatrix}, \quad F_3 = \begin{pmatrix} 11011 \\ 01100 \\ 10001 \\ 01110 \\ 01011 \end{pmatrix}.$$

Соответственно, обратные матрицы будут иметь вид

$$F_1^{-1} = \begin{pmatrix} 111 \\ 010 \\ 011 \end{pmatrix}, \quad F_2^{-1} = \begin{pmatrix} 1100 \\ 1101 \\ 0011 \\ 1001 \end{pmatrix}, \quad F_3^{-1} = \begin{pmatrix} 10001 \\ 11110 \\ 10110 \\ 01010 \\ 10101 \end{pmatrix}.$$

Любая фраза, состоящая, например, из 60 символов, может быть последовательно преобразована матрицей F_1 (длина блока, например, 20 символов), затем F_2 (длина блока, например, 15 символов) и в заключение F_3 (длина блоков, например, 12 символов). Восстановление исходного текста выполняется в обратном порядке, т. е. сначала текст преобразуется матрицей F_3^{-1}, затем F_2^{-1} и в заключение матрицей F_1^{-1}.

Поиск потерянного смысла

Известно, что совокупность степеней любого элемента F_i некоммутативной группы конечного порядка порождает циклическую коммутативную группу некоторого порядка s_i, т. е. $F^{s_i} = I$. В связи с этим, можно изменить передачу по варианту 2. На передающем конце отправитель будет обрабатывать сообщение $F^d \times B = A$, где d — целое, $0 < d < s_i$, а получатель сообщения, имея ту же функцию F, будет выполнять следующее преобразование: $F^{s_i - d} \times A = F^{s_i - d} \times F^d \times B = F^{s_i} \times B = I \times B = B$. При этом сообщение будет восстановлено. Если получатель сообщения не знает величины d, он может последовательно умножать полученную последовательность A слева на F до тех пор, пока не отыщет потерянный смысл — последовательность B.

Рассмотрим, например, матрицу F вида

$$F = \begin{pmatrix} 11001 \\ 01101 \\ 10110 \\ 10011 \\ 00001 \end{pmatrix}.$$

Определитель этой матрицы в поле GF(2) равен 1. Совокупность степеней этой матрицы порождает коммутативную группу порядка 15, т. е. $\{F^0 = I, F^1, F^2, \ldots, F^{14}\}$ — коммутативная группа. Возьмем 14 произвольных сообщений $B_1, B_2, \ldots, B_{14}$ и зашифруем их путем умножения $F^i \times B_i = A_i$ для всех $i = 1, 2, 3, \ldots, 14$. Обозначим $F^3 = P$, тогда последовательность степеней $\{P^0, P^1, P^2, P^3, P^4\}$ тоже образует коммутативную группу, которая является подгруппой исходной коммутативной группы, составленной из степеней матрицы F. Аналогично построим коммутативную подгруппу $F^5 = Q$ и последовательность степеней $\{Q^0, Q^1, Q^2\}$. Снабдим каждую группу пользователей соответствующей функцией: F — самого ответственного получателя, а затем остальных получателей матрицами P и Q. Если теперь зашифровать все 14 сообщений и передать их по каналу связи, то после обработки полученного сообщения соответствующими функциями самый ответственный получатель прочитает все сообщения, следующий — только кратные трем и последний прочитает только сообщения, кратные пяти. Для второго и третьего часть сообщений окажутся потерянными.

Выбирая произвольные матрицы F с определителем, равным 1, в поле GF(2) и находя коммутативные подгруппы, порождаемые некоторыми элементами, можно адресовать сообщения разным пользователям.

Заключение

Для шифрования и расшифрования сообщений можно использовать любые методы, как с открытыми, так и с секретными ключами.

Основываясь на методах обеспечения скрытого смысла по варианту 1, можно организовать передачу с многократно вложенным смыслом. В этом случае каждый пользователь «прочитывая» несколько раз один и тот же фрагмент сообщения с одной функцией или разными функциями F, каждый раз извлекает из него осмысленный текст нового содержания. Возможна комбинация методов по вариантам 1 и 2.

Чтение сообщений со скрытым смыслом можно сравнить с индивидуальными очками, которые выдаются разным пользователям сети. Из расшиф-

рованного открытого текста каждый читает только то, что предназначено лично ему. В случае многократно вложенного смысла при каждом новом прочтении одного и того же фрагмента текста пользователь видит новый текст. Число вложений зависит от выбранной функции и может быть достаточно большим.

Литература

1. **Ерош И. Л.** Булевы функции. Комбинационные схемы. Преобразование двоичных последовательностей: Учебн. пособ. — СПб.: Изд-во СПбГУАП, 2001. — 30 с.
2. **Ерош И. Л.** Разграничение доступа к ресурсам в системах коллективного пользования // Информационно-управляющие системы. — 2003. — № 2–3. — С. 63–66.

ИЗДАТЕЛЬСТВО «ПОЛИТЕХНИКА» ПРЕДСТАВЛЯЕТ

Куприянов М. С., Матюшкин Б. Д.

Цифровая обработка сигналов: процессоры, алгоритмы, средства проектирования. — 2-е изд., перераб. и доп. — СПб.: Политехника, 2002. — 592 с.: ил.

Книга содержит три части. Первая часть «Процессоры цифровой обработки сигналов» посвящена архитектуре и особенностям организации DSP. Во второй части «Алгоритмы цифровой обработки сигналов» рассматриваются основы теории дискретных систем, методы анализа эффектов квантования сигналов при реализации алгоритмов обработки на DSP, базовые алгоритмы ЦОС и их реализация на DSP. Третья часть «Инструментальные средства проектирования систем ЦОС» содержит описание программных и аппаратных средств, используемых для решения задач проектирования и входящих в стартовый комплекс разработчика систем ЦОС. В приложении приведена система команд семейств DSP5600x и DSP5630x.

Книга рассчитана на инженерно-технических работников, занимающихся проектированием систем ЦОС, а также студентов соответствующих специальностей технических университетов.

Информационно-управляющие системы для подвижных объектов. Семинары ASK Lab 2001 / Под общ. ред. М. Б. Сергеева. — СПб.: Политехника, 2002. — 234 с.: ил.

В книге представлены статьи, посвященные актуальным проблемам в области разработки информационно-управляющих систем для подвижных объктов, вопросам их надежности, алгоритмического и аппаратного обеспечения, защиты информационных каналов.

Книга ориентирована на научных и инженерно-технических работников, специалистов в области встраиваемых систем управления не только авиационных комплексов, но и наземных подвижных дистанционно управляемых объектов различного назначения.

УДК 621-52:004.52

ТЕХНОЛОГИИ РЕЧЕВОГО УПРАВЛЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ

Я. Ю. Изилов,
канд. техн. наук
Санкт-Петербургский государственный политехнический университет

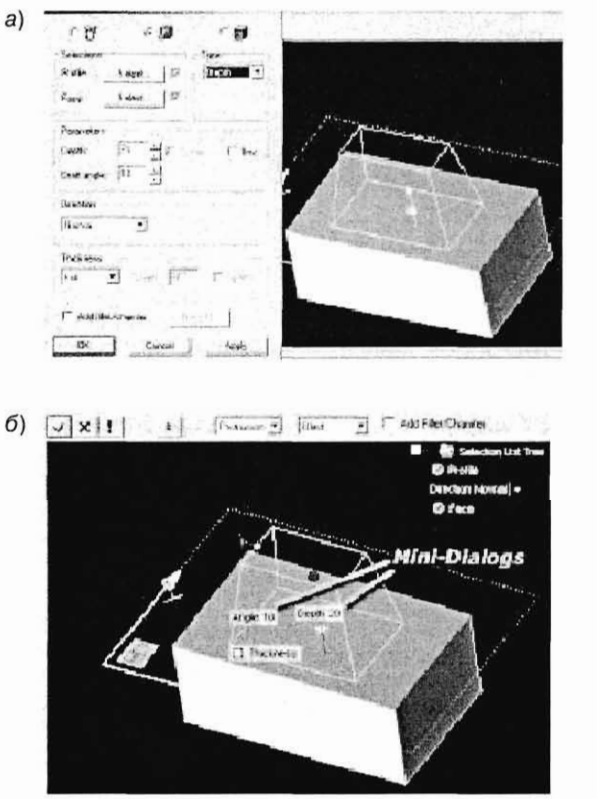
Рассматриваются новые технологии для автоматизации различных производственных процессов, которые позволяют осуществить ввод данных, контроль и обеспечить оперативное управление промышленными системами с помощью речевых команд.

This paper considers new technologies for various industrial processes automation which allows to input data, to carry out the checkup and to realize the operative control of industrial systems with the help of speech commands.

Современное промышленное производство не может обойтись без внедрения высоких технологий. В частности, революционной тенденцией в промышленности стало появление автоматизированных систем, позволяющих осуществить речевое управление, сочетающее множество функциональных возможностей, легкость обучения и эксплуатации.

Одной из них является машиностроительная система автоматического проектирования (САПР) *thinkdesign* производства итальянской компании *think3 S. p. A.* На выставке Euromold-2000 впервые была представлена версия этой системы с возможностями речевого управления [12]. Данная САПР функционирует на IBM PC-совместимом компьютере под управлением операционной системы Microsoft Windows. В последней версии САПР полностью интегрирован речевой интерфейс, который предоставляет пользователю простой, быстрый и более естественный способ создания конструкции. Такой подход позволяет ему сосредоточить внимание именно на конструкции, а не на работе с многочисленными меню, командными строками и т. д. Это нововведение отразилось на стиле графического интерфейса пользователя (рис. 1).

Старый стиль (рис.1, *а*) показывает традиционное диалоговое окно, которое содержит все опции управления, собранные в одном большом элементе графического интерфейса. Опции управления в диалоговом окне занимают значительную часть площади экрана и уменьшают обзорность проектируемой конструкции.



■ ***Рис. 1.*** *Графический интерфейс пользователя: а — старый стиль; б — новый стиль, содержащий мини-диалоги*

В отличие от старого, новый стиль (рис. 1, *б*) показывает элементы управления в графическом окне. При этом используются новые элементы графического интерфейса, так называемые мини-диалоги, которые не скрывают графическое поле конструкции. Команды управления, входные параметры и геометрические атрибуты конструкции, представленные в мини-диалогах, распределены в стратегических точках графического окна. Это позволяет увеличить обзорность проектируемой конструкции.

Все мини-диалоги имеют соответствующие голосовые команды или ключевые слова, которые отображены в графическом окне, например, «угол 10», «глубина 20», «длина 345,76», «толщина 0,52» и т. д. Кроме мини-диалогов, данная САПР имеет много других новых элементов графического интерфейса с соответствующими голосовыми командами.

Более подробно с возможностями САПР *thinkdesign* можно ознакомиться на официальном интернет-сайте компании *think3* [12].

В дополнение к сказанному отметим, что речевой интерфейс (РИ) предоставляет огромные возможности не только для оперативного управления САПРом, но и для ввода данных, например, прецизионных чисел. В качестве примера рассмотрим два сценария рисования окружности с радиусом 24 мм и координатами центра $X = 134{,}26$, $Y = 120{,}15$, которые могут быть реализованы в САПР *thinkdesign*. Один сценарий традиционно использует клавиатуру и «мышку», а другой — речевой интерфейс.

Сценарий 1.

1. Подвести «мышкой» курсор к меню «Insert».
2. Нажать левую кнопку «мышки».
3. В меню «Insert» с помощью «мышки» выбрать подменю «Drafting».
4. Нажать левую кнопку «мышки».
5. В подменю «Drafting» с помощью «мышки» выбрать вложенное меню «Circle and Arc».
6. Нажать левую кнопку «мышки».
7. Во вложенном меню «Circle and Arc» с помощью «мышки» выбрать команду «Center».
8. Нажать левую кнопку «мышки».
9. Подвести «мышкой» курсор к меню «Tools».
10. Нажать левую кнопку «мышки».
11. В меню «Tools» с помощью «мышки» выбрать подменю «Snap».
12. Нажать левую кнопку «мышки».
13. В подменю «Snap» с помощью «мышки» выбрать команду «Enable Point coordinates».
14. Нажать левую кнопку «мышки».
15. В появившемся окне «Point coordinates» подвести «мышкой» курсор к полю *X*.
16. Нажать левую кнопку «мышки».
17. С помощью клавиатуры ввести число 134,26.
18. В окне «Point coordinates» подвести «мышкой» курсор к полю *Y*.
19. Нажать левую кнопку «мышки».
20. С помощью клавиатуры ввести число 120,15.
21. В окне «Point coordinates» подвести «мышкой» курсор к кнопке «OK».
22. Нажать левую кнопку «мышки».
23. С помощью «мышки», перемещая курсор в поле чертежа, выбрать значение радиуса окружности «Radius 24».

Сценарий 2.

1. Произнести «Center circle».
2. Произнести «Coordinate input».
3. Произнести «Ex one hundred thirty four point twenty six».
4. Произнести «Wai one hundred twenty point fifteen».
5. Произнести «Radius twenty four».

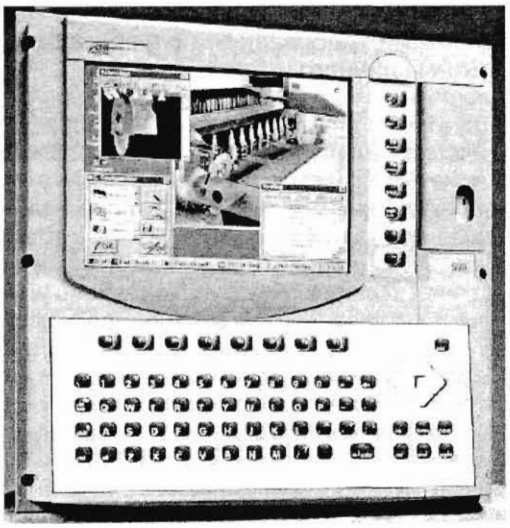
Продемонстрированные сценарии показывают, что РИ значительно упрощает использование данной САПР. При этом действительно обеспечивается более быстрый и более естественный способ создания конструкции. Для выполнения сценария 1 автору потребовалось около 40 с, для сценария 2 — около 15 с. В целом, по опыту автора, спроектировать новую конструкцию с использованием РИ можно как минимум на 20–30 % быстрее по сравнению с традиционными САПР. Однако использование РИ в данной САПР возможно только на английском языке. Вместе с этим, в работах [6–9] проводились исследования по эффективности использования РИ в сравнении с клавиатурой и «мышкой» при работе с графическими приложениями. Отмечается, что использование РИ позволяет ускорить процесс создания изображений в зависимости от их сложности до 50 %.

В нашей стране ежегодно доля изделий машино- и приборостроения в общем объеме продукции неуклонно увеличивается. Эти изделия становятся все более сложными и точными. Следовательно, усложняется их разработка и изготовление, увеличивается цикл и сложность подготовки их производства.

Общеизвестно, что за последние два десятилетия период производства изделия значительно сократился, а средняя продолжительность цикла технической подготовки увеличилась. В условиях многономенклатурного мелкосерийного и единичного производства продолжительность цикла технической подготовки стала соизмеримой с продолжительностью производства изделия, а во многих случаях превышает ее. Это обусловлено ростом трудоемкости и сложности процессов конструкторской и технологической подготовки, а также технологической наладки автоматизированного оборудования.

Упростить и ускорить процесс технологической наладки станка позволяет использование новой системы управления *iLENIA* (рис. 2), разработанной итальянской компанией *CNi Informatica* в 2001 году [10]. Данная система относится к классу Industrial PC и может использоваться для управления различным промышленным оборудованием, таким, как станки или промышленные роботы, а также осуществлять их групповое управление.

Среди множества функциональных возможностей системы *iLENIA* особого внимания заслуживает новый способ взаимодействия с пользователем, а именно при помощи речевых команд. Это достигается посредством использования встроенного программно-аппаратного речевого интерфейса «*Voice Master*». Он позволяет дублировать нажатие клавиш на клавиатуре системы, запрограммировать



■ ***Рис. 2.*** *Общий вид системы управления iLenia*

необходимые команды по желанию пользователя и осуществлять оперативное управление технологическим оборудованием непосредственно голосом. Однако возможность ввода речевых команд управления на русском языке в данной системе отсутствует.

Другим решением для ускорения процесса технологической наладки станка является использование новой системы *iPRO*. Она разработана в 2001 году английской компанией *Newall Electronics* [11]. В основу системы заложен фундаментальный принцип: если что-нибудь отвлекает квалифицированного наладчика или оператора станка от работы, то это, вероятно, снижает производительность его труда.

Новая разработка представляет собой беспроводную персональную систему, которая крепится на голове и на поясе пользователя. На пояс натягивается ремень, к которому присоединен микрокомпьютер, на голову надевается устройство со встроенными микрофоном, наушником и подвесным проекционным мини-экраном (рис. 3). Радиус действия системы составляет несколько сотен метров. Такое нововведение позволяет пользователю свободно передвигаться вокруг станка, осуществлять контроль и обеспечивать оперативное управление станком с помощью речевых команд.

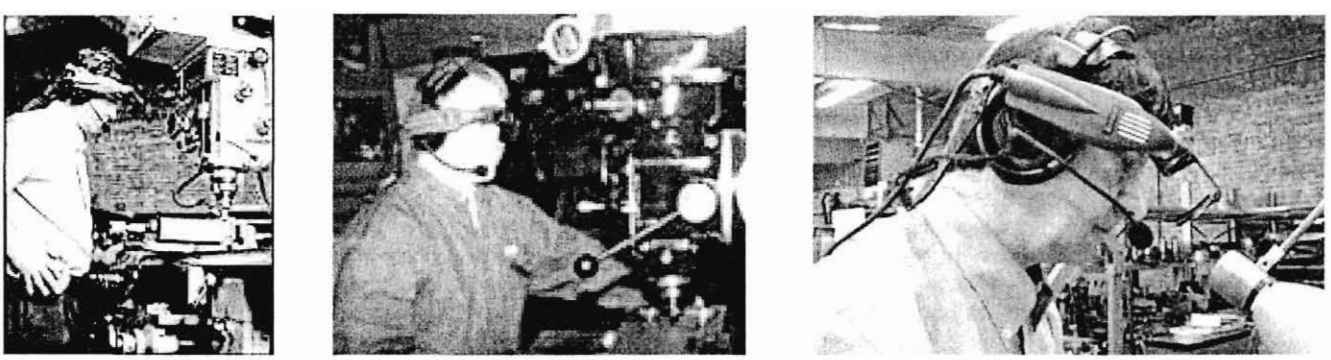
По заявлению президента компании *Newall Electronics* Д. Донэлдсона, система *iPRO* предлагает беспрецедентную безопасность, удобство и производительность. Она была разработана специально для того, чтобы помочь наладчикам и операторам станков лучше выполнять их задания, намного быстрее, с увеличенной точностью и комфортом [11].

Система *iPRO* позволяет наладчику или оператору станка всегда иметь необходимую информацию прямо перед глазами, не отвлекаясь от работы. Это стало возможным благодаря использованию расположенного на уровне глаз подвесного проекционного мини-экрана. Он сделан из специальной оптики и вмонтированного в нее 1,1″ полупрозрачного микродисплея. Габариты подвесного экрана в два раза меньше пластиковой кредитной карточки. За счет применения специальной оптики проектируемое изображение кажется достаточно больших размеров и таким, как на полноцветном 15-дюймовом мониторе.

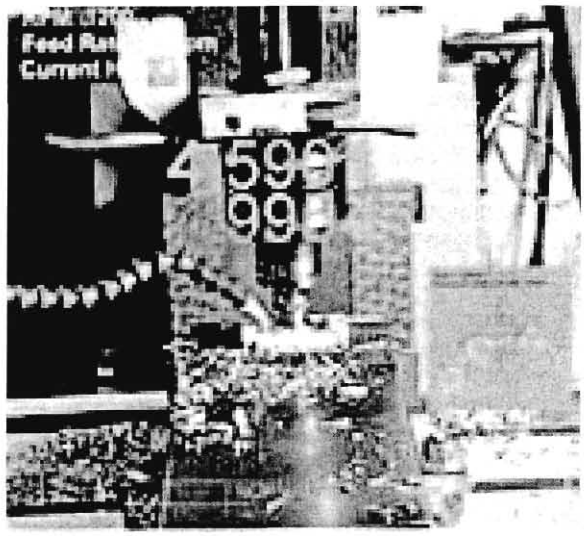
На подвесной мини-экран с помощью речевых команд может быть выведена информация, например, о значениях координат положения исполнительных механизмов станка, числа оборотов вращения шпинделя, скорости подачи, глубины резания без нарушения обзорности зоны обработки. На рис. 4 представлен фрагмент подобной информации.

Несмотря на очевидные достоинства системы *iPRO*, осуществлять в ней речевое управление представляется возможным только на английском языке.

С учетом вышеизложенного, можно заключить, что способ речевого управления и ввода данных позволяет избавить технологов и наладчиков оборудования с УЧПУ от рутинной работы при создании математических моделей новых деталей, технологической наладке станков с УЧПУ, робототехнических систем и другого автоматизированного оборудования. Речевой диалог является легким в использовании интерфейсом. Он предусматривает увеличение эффективности взаимодействия с автоматизированными системами посредством



■ ***Рис. 3.*** *Использование системы iPRO при технологической наладке металлообрабатывающих станков*



■ ***Рис. 4.*** *Фрагмент видеоинформации, выводимой на экран системы iPRO*

ускорения ввода данных при выполнении задач, когда глаза и руки оператора заняты. В последнем случае речевая форма управления является наиболее оптимальной по сравнению, например, с клавиатурой, джойстиком, световым пером, сенсорным экраном или манипулятором курсора типа «мышь». Подсистема речевого управления не будет полностью заменять указанные устройства. Однако она может быть общим компонентом следующего поколения автоматизированных систем различного назначения.

Практическая ценность в организации новой технологии человеко-машинного интерфейса с использованием речевого ввода заключается в том, что она обеспечивает прямой доступ к управлению современными высокопроизводительными системами специалистам с невысокой квалификацией. Использование систем программного управления с вводом данных голосом повышает производительность операций, так как уменьшается время работы с клавишной панелью [2, 3].

Таким образом, проведенный автором в данной работе и в работе [3] обзор устройств и систем с речевым вводом информации показывает, что их номенклатура и сфера применения постоянно расширяется. В то же время, серийных устройств и систем отечественного производства, способных воспринимать русский язык, автором не обнаружено. В этой связи, актуальной остается проблема улучшения характеристик существующих [1, 5] и создания новых речевых систем, которые могли бы функционировать без УЧПУ. Это особенно важно для автоматизированных тяжелых крупногабаритных станков, функционирующих без УЧПУ (например, с цифровой индикацией). Необходимость и своевременность проведения научно-исследовательских работ в данном направлении обусловлены объективными требованиями совершенствования человеко-машинного интерфейса.

Условием для широкого использования речевых технологий является повышение вероятности автоматической идентификации произносимых слов и увеличение быстродействия систем речевого взаимодействия. Для решения данной задачи необходимо проведение научных исследований, направленных на создание новых методов и систем автоматического анализа русской речи с улучшенными характеристиками и интеграции их в современные системы управления. Вместе с этим вопросы расширения функциональных возможностей существующих систем управления на основе технологии речевого управления остаются открытыми.

Литература

1. **Изилов Я. Ю.** Новые возможности оперативного управления металлообрабатывающими станками с ЧПУ // Металлообработка. — 2003 — № 1 (13). — С. 41–43.
2. **Изилов Я. Ю.** Совершенствование процесса технологической подготовки оборудования с ЧПУ // Проблемы машиноведения и машиностроения: Межвуз. сб. — Вып. 30. — СПб.: Изд-во СЗТУ, 2003.
3. **Изилов Я. Ю.** Некоторые аспекты моделирования речевых сигналов. — СПб.: Изд-во СПбГТУ, 2001.
4. **Изилов Я. Ю., Федотов А. И.** Расширение функциональных возможностей робототехнических систем // Научно-технические ведомости СПбГТУ 1899–1999. — № 3. — СПб.: Изд-во СПбГТУ, 1999.
5. **Изилов Я. Ю.** и др. Система речевого ввода информации в ЭВМ на естественном языке // Тезисы докл. и сообщ. научн. военно-техн. конф. «Автоматизация процессов управления соединениями и частями ПВО, информационные технологии. Состояние и перспективы создания единой автоматизированной радиолокационной системы» 15–16 мая 1996 г. — СПб.: СПВУРЭ ПВО, 1996.
6. **Bolt R.** Put-That-There: Voice & Gesture at the Graphics Interface // Computer Graphics. — 1980. — 14. — 3. — P. 262–270.
7. **Hauptmann A.** Speech and Gestures for Graphic Image Manipulation // Human Factors in Computer Systems (SIGCHI). — 1989. — P. 241–245.
8. **Martin G. F.** The Utility of Speech Input in User-Computer Interfaces // International Journal of Man-Machine Studies. — 1989. — Vol. 30. — P. 355–375.
9. **Pausch R., Leatherby J. H.** A Study Comparing Mouse-Only Input vs. Mouse Plus-Voice Input for a Graphical Editor // Proceedings of the AVIOS '90 Voice I/O Systems Applications Conference, September 1990. — P. 227–231.
10. http://www.cniinformatica.it
11. http://www.newall.co.uk
12. http://www.think3.com

УДК 681.325.5

МОДЕЛЬ ПРЕДСТАВЛЕНИЯ УЧЕБНОГО МАТЕРИАЛА И СПОСОБ ДИАГНОСТИРОВАНИЯ ОШИБОК ОПЕРАТОРА В АВТОМАТИЗИРОВАННОЙ ОБУЧАЮЩЕЙ СИСТЕМЕ

Д. А. Горбунов,
аспирант
В. Я. Мамаев,
доцент
К. К. Петров,
аспирант
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Построена структурная модель учебного материала, обладающая свойством полной выводимости целевого знания. Предложен способ диагностирования ошибок оператора, основанный на анализе характеристик целевого знания эталонной и фактической структурных моделей предметной области. Рассмотрен пример диагностирования возможных ошибок оператора-навигатора при решении одной из задач воздушной навигации.

The structural model of a teaching material is constructed, which one has property of full deducibility of target knowledge. The way of diagnosing of errors of the operator-navigator is offered, which one is based on the analysis of characteristics of target knowledge of reference and actual structural models of data domain. The example of diagnosing of possible errors of the operator-navigator is considered at the solution of one of problems of air navigating.

Структурная модель учебного материала

Построим структурную модель учебного материала (УМ). Для этого введем в рассмотрение конечное множество $M = (\mu_1, \mu_2, \ldots, \mu_i, \ldots, \mu_n)$, где μ_i — обучающий блок (ОБ), который соответствует порции УМ и порции отношений на M, являющихся отображениями вида:

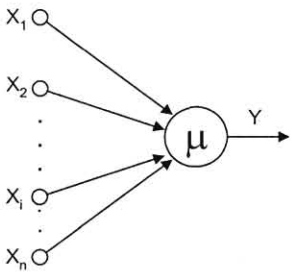
1) $\mu = S(\mu_1, \mu_2, \ldots, \mu_i, \ldots, \mu_n)$ — отношение непосредственной связности по информации или выводимости блока μ из блоков $(\mu_1, \mu_2, \ldots, \mu_i, \ldots, \mu_n)$;

2) $\mu = \delta(\mu_1, \mu_2, \ldots, \mu_i, \ldots, \mu_n)$ — отношение детализации знания μ, которое состоит из значений $M = (\mu_1, \mu_2, \ldots, \mu_i, \ldots, \mu_n)$.

Тогда структурной моделью УМ назовем тройку M, S, δ, где M — множество учебных блоков; S — отношение информационной связности; δ — отношение детализации. Отношения информационной связности ОБ таковы, что знания имеют свои источники (μ_0), промежуточные или выводимые знания (μ_S) и конечные, т. е. целевые (μ_K) знания.

Обучающие μ-блоки можно связать в сеть знаний следующим образом. Каждой вершине сети сопоставляется единственный μ-блок, а каждой ее дуге соотносится маркер, который является кодом описания соответствующего значения (формулы), заключенного в учебный блок. Далее маркеры (метки) будем обозначать большими буквами латинского алфавита.

Исходя из того, что μ-блок определяет отображение (преобразование одного множества X в другое множество Y), введем понятие формулы вывода.



■ ***Рис. 1.*** *Логический блок обучения с формулой вывода (1)*

Формулой вывода назовем выражение

$$\mu(X_1, X_2, ..., X_i, ..., X_n) \to Y, \qquad (1)$$

где (X_1, X_2, ..., X_i, ..., X_n) — поставляемые в μ-блок исходные значения; Y — целевые (выходные) знания, полученные в результате процедуры обучения (вывода); «→» обозначает некоммутативную операцию «следует».

Каждый μ-блок должен иметь единственный выход, т. е. все исходящие из μ-блока дуги должны иметь одинаковый маркер. Формула (1) читается так: знание Y является следствием процесса научения из знаний X_1 & X_2 & ... & X_i & ... & X_n. Графически это представлено на рис. 1.

Граф на рис. 1 показывает, из каких составляющих и как складывается целевое знание. Для его построения необходимо выполнить следующие процедуры:

1) выделить множества $M = \{\mu_1, \mu_2, ..., \mu_k\}$ возможных порций обучения, имеющих законченный смысловой характер;

2) сгруппировать знания для каждого μ-блока в виде логических формул вывода $\mu(X_1, X_2, ..., X_i, ..., X_n) \to Y$, где Y — результирующее значение, выведенное из составляющих знаний $X_1, X_2, ..., X_i, ..., X_n$;

3) связать μ-блоки в обучающие кластеры при помощи подстановок знаний в формулы вывода.

Обучающим кластером назовем направленный граф $G <X, M, Y>$, вершины которого размечены μ-блоками, дуги Y — маркером знаний, каждой вершине соотнесена формула вывода, и каждая вершина (μ-блок) кластера выводима из начальных знаний либо является начальным знанием μ_0 [3].

Отсюда следует свойство полной выводимости целевого знания, что является необходимым условием. При отсутствии полной выводимости невозможно построить процессы контроля знаний и управления обучением.

Таким образом, в качестве модели предметной области будем рассматривать граф $G = <X, M, Y>$, объектные вершины которого $M = \|\mu_{ij}\|_{n \times n}$ есть фрагменты обучения. В конкретных случаях ими могут быть параграфы теоретического материала, диалоговые структуры, задачи или их фрагменты и т. д.

Способ диагностирования ошибок оператора

Предлагаемая процедура диагностики ошибок обучаемого оператора заключается в анализе характеристик целевого знания Y эталонной и фактической структурных моделей умения (ЭМ, ФМ), проводимых электронным инструктором (ЭИ) с целью выявления причин несоответствия вершин и дуг графа, в которых эти несоответствия наблюдаются.

Решения задачи диагностики можно достичь методом дедукции (составлением плана в обратном направлении, т. е. продвижением от конца к началу, что соответствует случаю, когда обучаемый, заработав оценку за решение задачи или полет в целом, хочет получить объяснения своим ошибкам) или индукции (составление плана в прямом направлении или продвижением от начала к концу, когда обучаемый хочет знать свои ошибки на каждом шаге) [6].

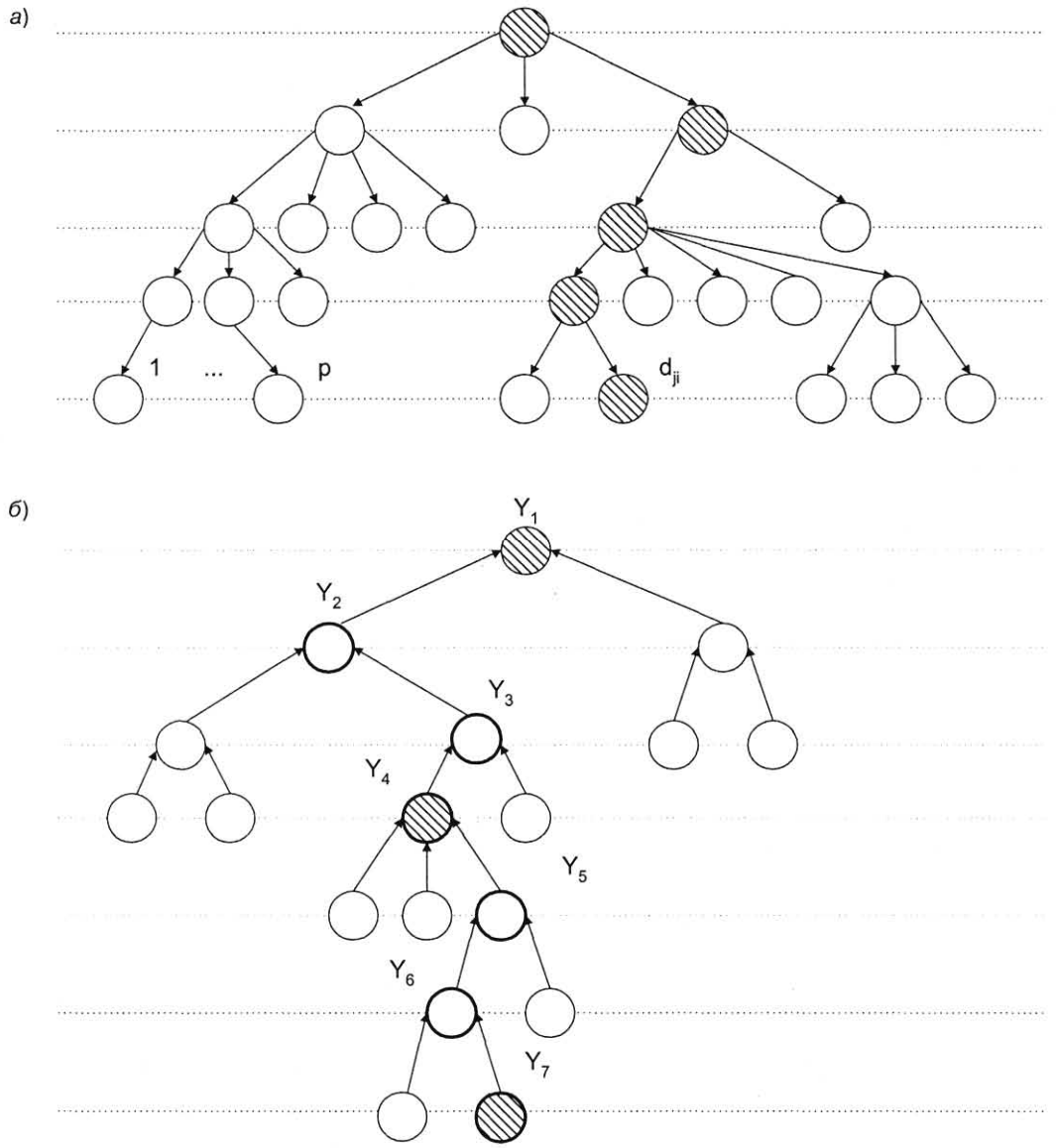
Модель диагностики деятельности обучаемого по решению последовательности задач при теоретическом обучении или в маршрутном полете методом дедукции в виде орграфа представлена на рис. 2, *а*. Здесь учитывается возможность различных вариантов развития событий вследствие изменения условий (например, исходных на полет) и потому продвижения обучаемого от начала к концу по той или иной ветви графа. Как видно из рис. 2 дедуктивный метод обеспечивает исключение из процесса диагностики всех не содержащих ошибки ветвей (содержащие ошибки вершины заштрихованы). Его недостатком является возможность достоверного определения ошибки лишь на нижнем (начальном) уровне. Для надежной идентификации всех ошибок на ветвях вышестоящих уровней иерархии необходимо неоднократное предъявление обучаемому фактической модели ОБ после применения процедуры исправления ошибки на нижестоящем уровне. В обучающих системах такая процедура недопустима, так как сопровождается отрицательным переносом навыка.

В предлагаемом способе диагностики деятельности обучаемого во избежание названных выше недостатков будем применять индуктивный метод, для чего в случае обнаружения ошибки на нижестоящем уровне в качестве эталона для сравнения на следующем уровне используем фактическую модель (рис. 2, *б*).

Модель коррекции эталонной модели по фактической на каждом шаге процедуры диагностики d_j можно представить в следующем виде:

$$\forall d_j \in D \begin{cases} \text{если } d_j = 1, \text{ ТО } X_j^э := Y_{j+1}^э \\ \text{если } d_j = 0, \text{ ТО } X_j^э := Y_{j+1}^ф \end{cases},$$

$$\text{где } d_j = \begin{cases} 1, \text{ при } Y_{j+1}^э = Y_{j+1}^ф \\ 0, \text{ при } Y_{j+1}^э \neq Y_{j+1}^ф, \quad j = (\overline{1, n}). \end{cases}$$



■ ***Рис. 2.*** *Диагностика деятельности обучаемого: а — дедуктивный метод; б — индуктивный метод*

В случае несовпадения результатов одиночной операции эталонной и фактической моделей на предыдущем уровне формирование операнда текущего уровня эталонной модели ($X_j^э$) предлагается реализовать путем искусственного присвоения результата операции предыдущего уровня фактической модели ($Y_{j+1}^ф$), а не за счет естественной трансформации результата операции предыдущего уровня эталонной модели ($Y_{j+1}^э$) [4].

Рассмотрим в качестве примера задачу «Расчет направления и скорости ветра по углу сноса и путевой скорости, измеренным на контрольном этапе», решение которой обучаемым должно быть проконтролировано. В этом случае объектными вершинами графа будут являться отдельные ее действия, а дугами — направления последовательности выполняемых действий.

Пример

Одним из важнейших дидактических средств подготовки оператора-навигатора является система задач, при решении которых он не только закрепляет полученные теоретические знания, но приобретает умения и навыки, необходимые в его профессиональной деятельности. Рассмотрим пример, когда по запросу оператора осуществляется диагностирование возможных ошибок и предъявление на экране результатов расчетов эталонных (выполненных ПЭВМ АОС) и фактических (выполненных обучаемым с помощью встроенного специализированного калькулятора, так как только в этом случае обеспечивается условие контролируемости) при решении задачи «Расчет направления и скорости

ветра по углу сноса и путевой скорости, измеренным на контрольном этапе».

Исходными данными задачи являются [5]:

— истинная воздушная скорость ($V_{ист}$ [$\frac{км}{ч}$]);

— угол курса (УК[°]); длина контрольного этапа ($S_{к.э}$[км]);

— фактический условный путевой угол (ФУПУ[°]);

— время пролета контрольного этапа ($t_{к.э}$[м · с]).

Алгоритм решения задачи следующий:

1) угол сноса: УС = ФУПУ – УК;

2) путевая скорость: $W = \frac{S_{к.э}}{t_{к.э}}$;

3) скорость ветра: $U = \sqrt{V^2 + W^2 - 2VW\cos(YC)}$;

4) направление ветра:

$$\sigma = \arcsin\left(\frac{W\sin(YC)}{U}\right) + УК;$$

$$\sigma = \begin{cases} \sigma - 360^\circ & \text{при} \quad \sigma > 0 \\ \sigma + 360^\circ & \text{при} \quad \sigma < 0 \end{cases}.$$

Введем обозначения:

$Y'_0 = УС = ФУПУ - УК;$

$Y'_1 = W = S_{к.э} / t_{к.э};$

$Y'_2 = \cos УС;$

$Y'_3 = VW;$

$Y'_4 = 2VW;$

$Y'_5 = 2VW\cos УС;$

$Y'_6 = V^2;$

$Y'_7 = W^2;$

$Y'_8 = V^2 + W^2;$

$Y'_9 = U^2 = V^2 + W^2 - 2VW\cos УС;$

$Y'_{10} = U = \sqrt{V^2 + W^2 - 2VW\cos УС};$

$Y'_{11} = \sin УС;$

$Y'_{12} = W\sin УС;$

$Y'_{13} = (W\sin УС)/U;$

$Y'_{14} = \arcsin((W\sin УС)/U);$

$Y'_{15} = \sigma = \arcsin((w \sin УС)/v) + УК;$

$Y'_{16} = \sigma = \sigma - 360^\circ \quad \text{при} \quad \sigma > 0;$

$Y'_{17} = \sigma = \sigma + 360^\circ \quad \text{при} \quad \sigma < 0.$

Граф $G(X, M, Y)$ данной задачи (рис. 3) описывается с помощью системы следующих формул:

$(\mu_{01}, \mu_{02})\mu_{11} \approx (\mu_{01} \overset{Y_{01}}{\rightarrow} \mu_{11}), (\mu_{02} \overset{Y_{02}}{\rightarrow} \mu_{11});$

$(\mu_{01}, \mu_{81})\mu_{91} \approx (\mu_{01} \overset{Y_{01}}{\rightarrow} \mu_{91}), (\mu_{81} \overset{Y_{81}}{\rightarrow} \mu_{91});$

$(\mu_{03}, \mu_{04})\mu_{12} \approx (\mu_{03} \overset{Y_{03}}{\rightarrow} \mu_{12}), (\mu_{04} \overset{Y_{04}}{\rightarrow} \mu_{12});$

$(\mu_{05})\mu_{24} \approx \mu_{05} \overset{Y_{05}}{\rightarrow} \mu_{41};$

$(\mu_{05}, \mu_{12})\mu_{25} \approx (\mu_{05} \overset{Y_{05}}{\rightarrow} \mu_{25}), (\mu_{12} \overset{Y_{12}}{\rightarrow} \mu_{25});$

$(\mu_{06}, \mu_{25})\mu_{33} \approx (\mu_{06} \overset{Y_{06}}{\rightarrow} \mu_{33}), (\mu_{25} \overset{Y_{25}}{\rightarrow} \mu_{33});$

$(\mu_{07}, \mu_{91})\mu_{101} \approx (\mu_{07} \overset{Y_{07}}{\rightarrow} \mu_{101}), (\mu_{91} \overset{Y_{91}}{\rightarrow} \mu_{101});$

$(\mu_{11})\mu_{21} \approx \mu_{11} \overset{Y_{11}}{\rightarrow} \mu_{21};$

$(\mu_{11})\mu_{22} \approx \mu_{11} \overset{Y_{11}}{\rightarrow} \mu_{22};$

$(\mu_{12}, \mu_{21})\mu_{31} \approx (\mu_{12} \overset{Y_{12}}{\rightarrow} \mu_{31}), (\mu_{21} \overset{Y_{21}}{\rightarrow} \mu_{31});$

$(\mu_{12})\mu_{23} \approx \mu_{12} \overset{Y_{12}}{\rightarrow} \mu_{23};$

$(\mu_{22}, \mu_{33})\mu_{41} \approx (\mu_{22} \overset{Y_{22}}{\rightarrow} \mu_{41}), (\mu_{33} \overset{Y_{33}}{\rightarrow} \mu_{41});$

$(\mu_{23}, \mu_{24})\mu_{32} \approx (\mu_{23} \overset{Y_{23}}{\rightarrow} \mu_{32}), (\mu_{24} \overset{Y_{24}}{\rightarrow} \mu_{32});$

$(\mu_{31}, \mu_{61})\mu_{71} \approx (\mu_{31} \overset{Y_{31}}{\rightarrow} \mu_{71}), (\mu_{61} \overset{Y_{61}}{\rightarrow} \mu_{71});$

$(\mu_{32}, \mu_{41})\mu_{51} \approx (\mu_{32} \overset{Y_{32}}{\rightarrow} \mu_{51}), (\mu_{41} \overset{Y_{41}}{\rightarrow} \mu_{51});$

$(\mu_{51})\mu_{61} \approx \mu_{51} \overset{Y_{51}}{\rightarrow} \mu_{61};$

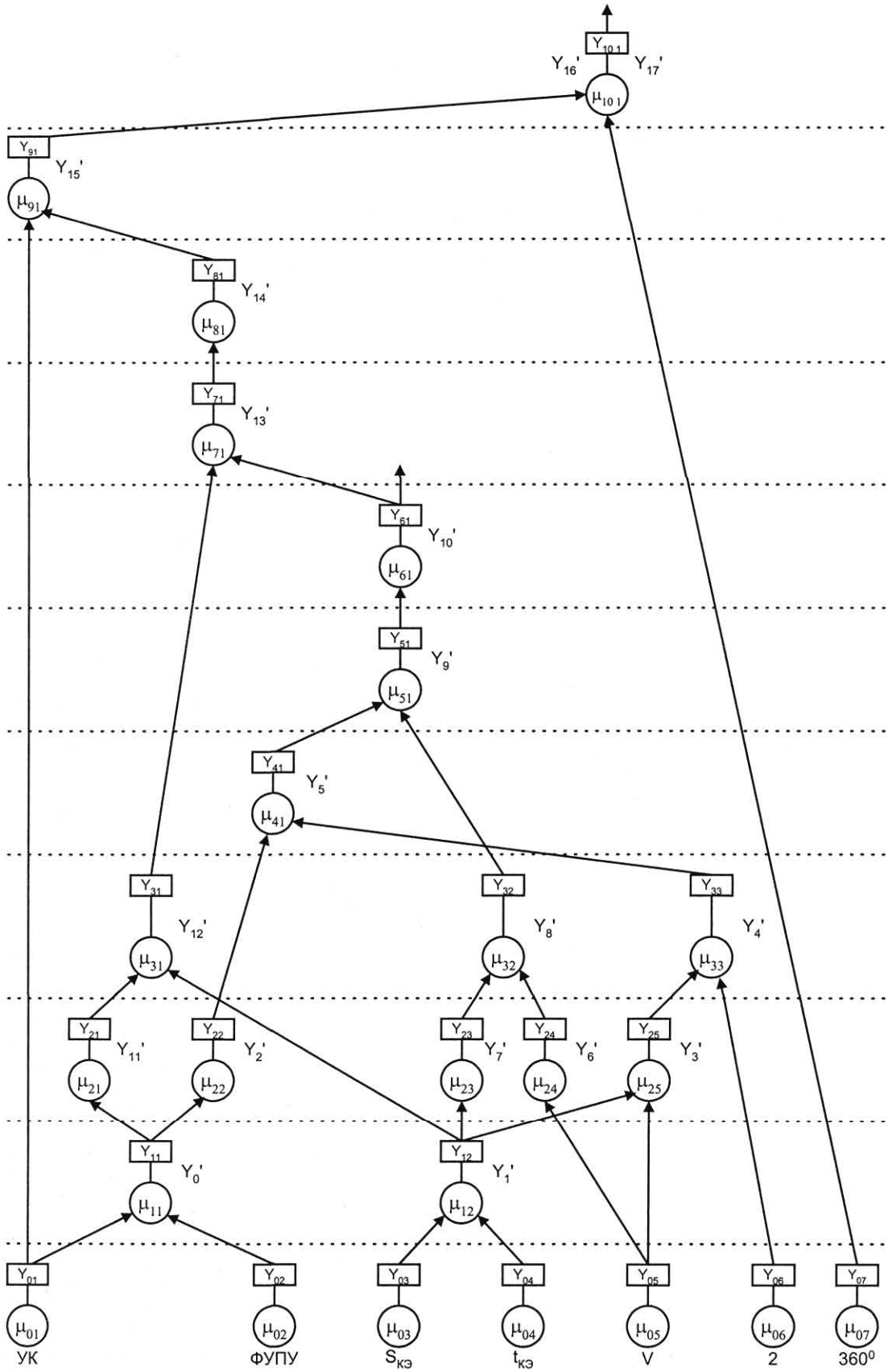
$(\mu_{71})\mu_{81} \approx \mu_{71} \overset{Y_{71}}{\rightarrow} \mu_{81}.$

Система формул определяет взаимосвязь между действиями в рассматриваемой задаче. Входными вершинами графа $\mu_{01}, \ldots, \mu_{07}$ являются угол курса, фактический условный путевой угол, длина контрольного этапа, время пролета контрольного этапа, истинная скорость полета, константы 2 и 180°. Выходных вершин в задаче две, что является условием получения целевых знаний, т. е. нахождения искомых значений. Выходными вершинами являются: μ_{61} — скорость ветра и μ_{101} — направление ветра.

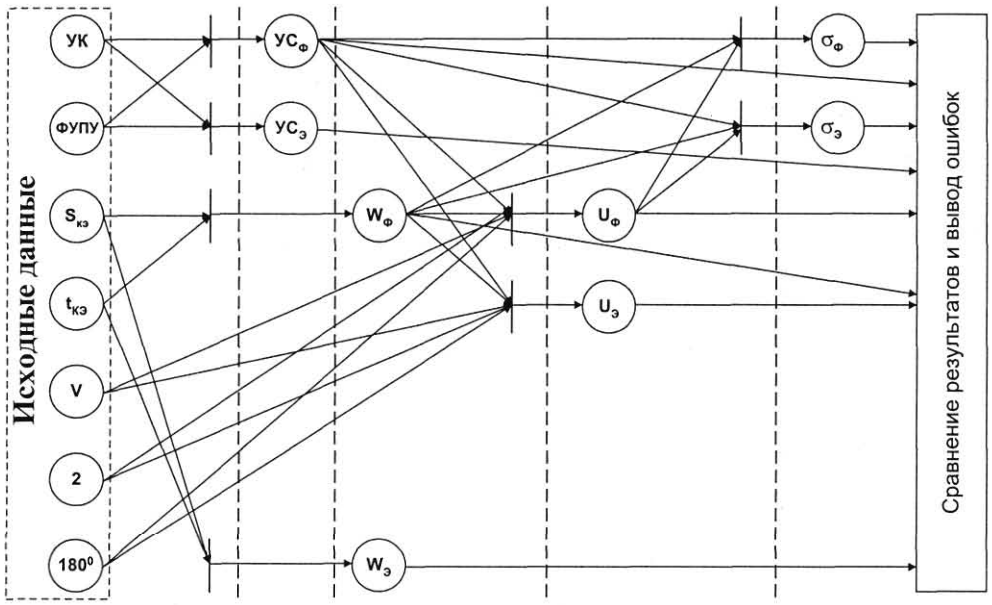
Данный граф описывает процесс решения рассматриваемой задачи и позволяет построить эталонную и фактическую структурные модели умения и проанализировать выходные вершины этих моделей, чтобы выявить причины несоответствия вершин и дуг графа. Методика диагностики ошибок обучаемого базируется на индуктивном способе, т. е. в случае обнаружения ошибки в предыдущей операции, например при расчете угла сноса, в качестве эталона для сравнения в следующей операции нахождения скорости ветра используется фактический угол сноса, а не эталонный (рис. 4).

Предложенная методика реализована программным способом с использованием языка Delphi, в основе идеологии которого лежит технология визуального проектирования и методология объектно-ориентированного программирования.

При решении задачи в АОС в открывшемся окне программы выбирается в меню «Задачи» одна из задач. Исходные данные задачи формируются случайным образом (см. рис. 7). Ответы вводятся обучаемым в поля ввода в левой панели экрана. В меню «Помощь» пункт «Калькулятор» позволяет вызвать

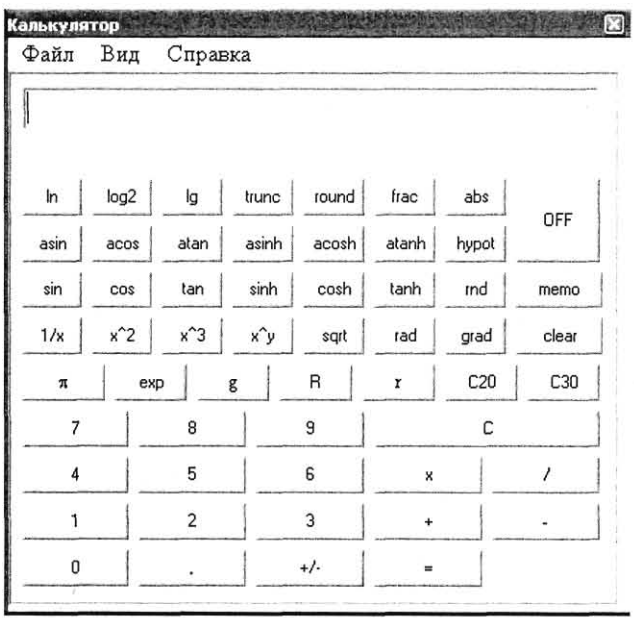


■ ***Рис. 3.*** *Граф G(X, M, Y) задачи «Расчет направления и скорости ветра по углу сноса и путевой скорости, измеренным на контрольном этапе»*



■ ***Рис. 4.*** *Применение методики диагностики ошибок обучаемого в задаче «Расчет направления и скорости ветра по углу сноса и путевой скорости, измеренным на контрольном этапе» при расчете скорости ветра*

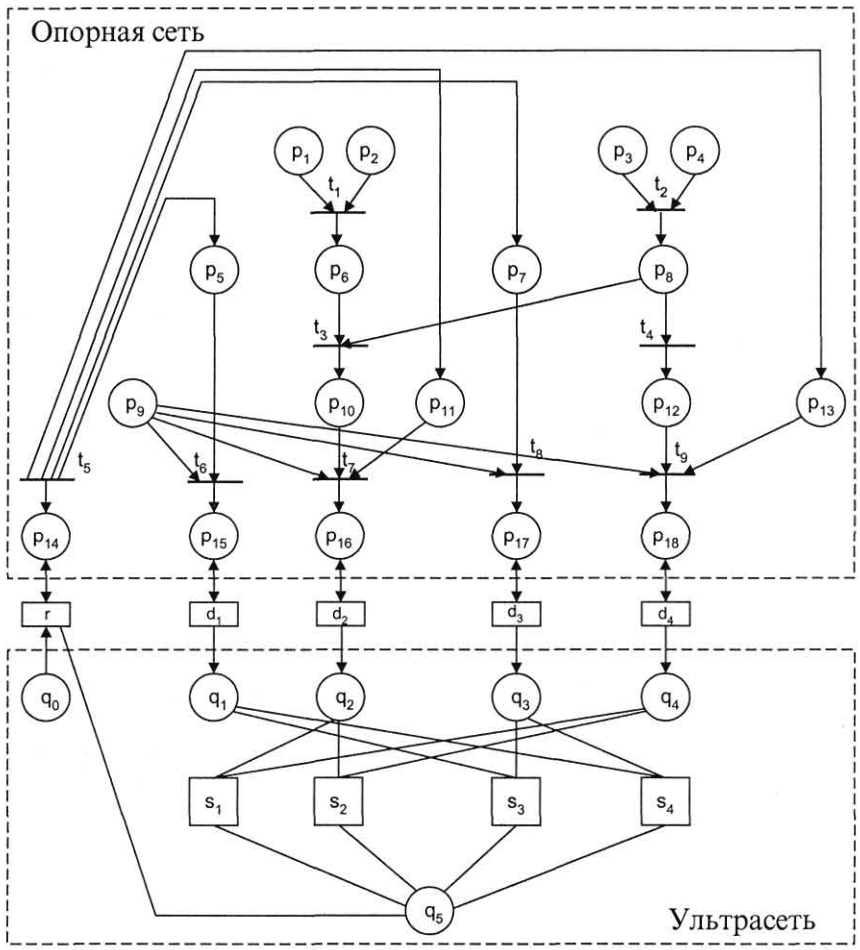
программу калькулятор (рис. 5). При нажатии кнопки «Результат» на экран выводятся эталонные значения, рассчитанные программой. На основе сравнения этих значений с результатами, введенными обучаемым, формируется оценка по каждому из действий и итоговый вывод, характеризующий общий итог решения задач. При нажатии на кнопку «Помощь» обучаемому предлагается возможность увидеть те формульные зависимости, которыми он должен был руководствоваться при реализации последовательности действий решения каждой конкретной задачи.



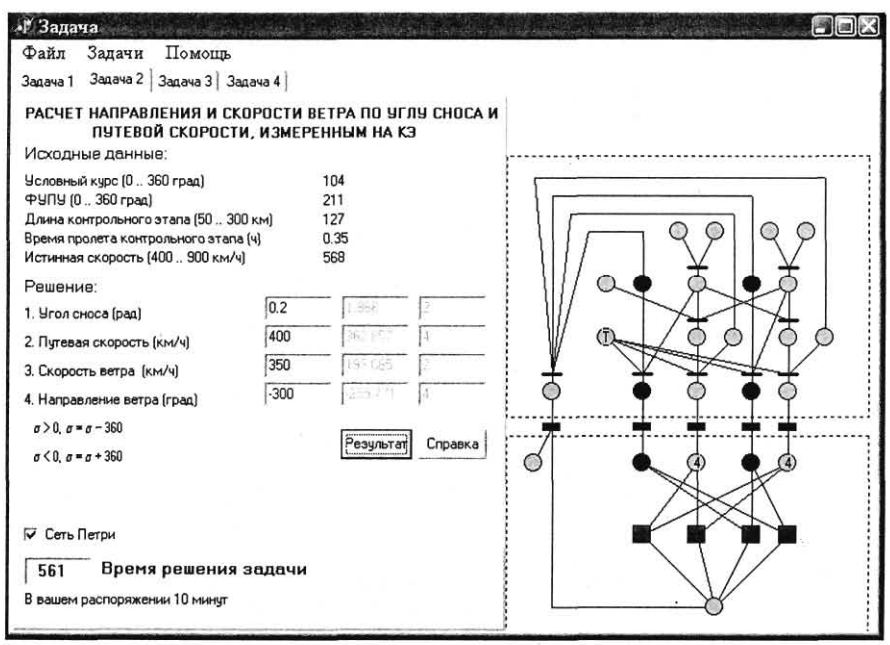
■ ***Рис. 5.*** *Встроенный калькулятор для контроля действий обучаемого*

На правой панели экрана представлена сетевая модель (сеть Петри) процесса решения задачи обучаемым оператором, которая может служить подсказкой обучаемому о месте сделанной ошибки (в режиме работы «Самоподготовки») или преподавателю, наблюдающему за ходом решения задачи (в режиме «Контроль знаний»).

Модель функционирует следующим образом (рис. 6). Исполнительный элемент (ИЭ) r через позицию p_{14} и переход t_5 осуществляет выполнение всех этапов вычислений, т. е. происходит расчет эталонных значений на каждом из этапов (P_5, P_7, P_{11}, P_{13}). Переходы t_6–t_9 осуществляют формирование ошибки на каждом этапе. Затем эти ошибки поступают в накопители q_1–q_4 через датчики d_1–d_4. Ультрасистемы S_1 и S_2 осуществляют снятие ошибки с последнего этапа и ее анализ. Если ошибка не превышает допустимого значения, то формируется ответный сигнал на ИЭ r о том, что задача решена успешно, иначе происходит обращение через ультрасистемы S_3 и S_4 об ошибках на предыдущих этапах. В результате на накопитель q_5 возвращается информация о наиболее существенной ошибке (величина ошибки и номер этапа, на котором она допущена). Затем все сведения об ошибке поступают через ИЭ r на накопитель q_0 [1]. Окраска позиций зеленым цветом говорит о правильном решении задачи (оценки «отлично», «хорошо», «удовлетворительно»), а красным — об ошибочных действиях обучаемого (оценка «неудовлетворительно») (рис. 7). Позиция P_9 осуществляет контроль времени решения задачи обучаемым.



■ ***Рис. 6.*** *Полная сетевая модель задачи «Расчет направления и скорости ветра по углу сноса и путевой скорости, измеренным на контрольном этапе»*



■ ***Рис. 7.*** *Результаты расчетов, выполненных обучаемым, и оценка его действий программой*

В том случае, если задача не решена в отведенный интервал времени, обучаемый получает оценку «неудовлетворительно» по временному критерию решения контрольного задания.

Обращение обучаемого к программе по результатам вывода о неправильности его действий осуществляется с помощью позиции q_0 (см. рис. 6).

Литература

1. **Горбунов Д. А.** Интеллектуальный интерфейс в диагностической обучаемой системе // Сб. докл. Пятой научной сессии аспирантов и соискателей ГУАП. — СПб., 2002.
2. **Змитрович А. Н.** Интеллектуальные информационные системы. — Минск: НТООО «Тетрасистемс», 1997. — 368 с.
3. **Курганская Г. С.** Модели, методы и технология дифференцированного обучения на базе Интернет / Автореф. ... канд. тех. наук. — М., 2001. — 34 с.
4. **Мамаев В. Я., Петров К. К., Синяков А. Н.** Диагностика ошибок обучаемого в навигационном тренажере штурмана // Матер. научно-технич. конф. «Тренажерные технологии и симуляторы-2002». — СПб.: СПбГПУ, 2002. — С. 159.
5. **Мамаев В. Я., Синяков А. Н., Петров К. К., Горбунов Д. А.** Воздушная навигация и элементы самолетовождения: Учебное пособие. — СПбГУАП, 2002. — 280 с.
6. **Петров К. К.** Диагностика ошибок штурмана в интеллектуальной обучающей системе // Сб. докл. Пятой научной сессии аспирантов и соискателей ГУАП. — СПб., 2002.

ИЗДАТЕЛЬСТВО «ПОЛИТЕХНИКА» ВЫПУСТИЛО В СВЕТ

Кербер Л. Л.

Туполев.— СПб.: Политехника, 1999. — 339 с.: ил.

Эта интереснейшая книга — воспоминание о выдающемся авиаконструкторе Андрее Николаевиче Туполеве. О человеке, помогавшем Н. Е. Жуковскому исследовать проблемы аэро- и гидродинамики, прошедшем столько испытаний, но не сломившемся и создавшем вместе со своим КБ более 100 типов военных и гражданских самолететов, среди которых АНТ-25, Ту-104 (первый реактивный самолет), Ту-114, Ту-144 (сверхзвуковой пассажирский). На его самолетах установлено 78 мировых рекордов, 28 уникальных перелетов с экипажами В. Чкалова и М. Громова, в том числе через Северный полюс в США.

А. Н. Туполев потряс Европу советскими самолетами и их достижениями.

Этой книгой издательство открывает публикацию серии книг о выдающихся отечественных авиаконструкторах. Книга рассчитана на широкий круг читателей.

Крылов А. Н.

Мои воспоминания. — 9-е изд., перераб. и доп. СПб.: Политехника, 2003. — 510 с.: ил.

Академик Алексей Николаевич Крылов — основоположник современной теории корабля — был ученым энциклопедического склада ума. Ему принадлежат оригинальные труды по различным вопросам математики, физики и астрономии; он автор многих изобретений и ряда прекрасно написанных учебных курсов по теории корабля, теоретической механике, дифференциальному и интегральному исчислениям и т. д.

Книга «Мои воспоминания» — это написанные прекрасным литературным языком рассказы большого ученого об основных периодах его научной и практической деятельности.

Книга рассчитана на широкий круг читателей, интересующихся историей отечественной науки, флота и судостроения.

СВЕДЕНИЯ ОБ АВТОРАХ

ДУБАРЕНКО Владимир Васильевич

Ученый секретарь Института проблем машиноведения РАН.
В 1963 г. окончил Ленинградский Военно-механический институт по специальности механика, в 1965 г. — по специальности системы управления. В 2002 г. защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором 70 научных публикаций.
Область научных интересов: интеллектуальные системы и системы управления.

КОНОВАЛОВ Александр Сергеевич

Заведующий кафедрой управления и информатики в технических системах СПбГУАП.
Заведующий лабораторией автоматизации института проблем машиноведения РАН.
Окончил Ленинградский институт авиационного приборостроения в 1968 г. по специальности «Электрооборудование ракет и других летательных аппаратов». Кандидатскую диссертацию защитил в 1981 г., докторскую — в 1997 г.
Автор более 120 научных публикаций.
Области научных интересов: синтез нелинейных систем автоматического управления сложными объектами; системы искусственного интеллекта; системы автоматизированного проектирования.

ШУМИЛОВ Павел Евгеньевич

Аспирант кафедры управления и информатики в технических системах. Окончил Санкт-Петербургскую государственную академию аэрокосмического приборостроения в 1997 г.
Автор 7 научных работ.
Область научных интересов: нечеткая логика и интеллектуальное управление системами торможения транспортных средств.

ГОРОДЕЦКИЙ Андрей Емельянович

Доктор технических наук, профессор, Институт проблем машиноведения РАН, заведующий лабораторией.
Область научных интересов: информационно-управляющие системы, нейронные сети, оптическое распознавание образов.

ТАРАСОВА Ирина Леонидовна

Кандидат технических наук, старший научный сотрудник, Институт проблем машиноведения РАН, старший научный сотрудник.
Область научных интересов: информационно-управляющие системы, нейронные сети.

ЛУКЬЯНОВА Людмила Михайловна

Доцент кафедры систем управления и вычислительной техники Калининградского государственного технического университета, канд. техн. наук, действительный член Международной Академии информатизации.
Область научных интересов: системные исследования, инженерная психология.

ШАЛЫТО Анатолий Абрамович

Заведующий кафедрой «Информационные системы» Санкт-Петербургского государственного университета информационных технологий, механики и оптики.
Окончил ЛЭТИ в 1971 г. по специальности «Автоматика и телемеханика».
Ученый секретарь НПО «Аврора».
Диссертацию на соискание ученой степени доктора технических наук защитил в 1999 г.
Автор большого числа научных трудов, в том числе 3 книг. Автор более 70 изобретений.
Член редакционной коллегии журнала «Информационно-управляющие системы».
Области научных интересов: системы логического управления; автоматное программирование

ЕРОШ Игорь Львович

Профессор кафедры вычислительных систем и сетей СПбГУАП.
Окончил в 1960 г. Ленинградский электротехнический институт (ЛЭТИ).
Диссертацию на соискание ученой степени доктора технических наук защитил в 1979 г.
Автор более 150 научных трудов, в том числе соавтор двух учебников и трех монографий.
Член-корреспондент Международной академии информатизации.
Области научных интересов: дискретная математика; распознавание образов; защита информации.

ГОРБУНОВ Дмитрий Анатольевич

Аспирант кафедры аэрокосмических приборов и измерительно-вычислительных комплексов.
Окончил Санкт-Петербургский государственный университет аэрокосмического приборостроения (СПбГУАП) в 2003 г. по направлению «Приборостроение».
Автор 4 научных публикаций.
Область научных интересов: интеллектуальные тренажерно-обучающие системы.

ШОПЫРИН Данил Геннадиевич

Инженер-программист фирмы «Транзас технологии».
В 2002 г. окончил Оренбургский государственный университет.
Область научных интересов: объектно-ориентированное и автоматное программирование.

МАМАЕВ Виктор Яковлевич

Доцент кафедры аэрокосмических приборов и измерительно-вычислительных комплексов.
Окончил Ленинградский институт авиационного приборостроения (ЛИАП) в 1958 г. по специальности «Электрооборудование летательных аппаратов». Защитил кандидатскую диссертацию в 1971 г. С 1974 г. на преподавательской работе.
Автор более 90 научных публикаций.
Область научных интересов: интеллектуальные тренажерно-обучающие системы.

ПЕТРОВ Кирилл Константинович

Аспирант кафедры аэрокосмических приборов и измерительно-вычислительных комплексов.
Окончил Санкт-Петербургский государственный университет аэрокосмического приборостроения (СПбГУАП) в 2003 г. по направлению «Приборостроение».
Автор 6 научных публикаций.
Область научных интересов: интеллектуальные тренажерно-обучающие системы.

УДК 519.71

Принципы логического управления динамическими объектами
Дубаренко В. В. Информационно-управляющие системы, 2003. — № 5. — С. 2–11.

Исследуются методы построения систем логического управления, обеспечивающих повышение качества нелинейных динамических объектов, и предлагаются алгоритмы их реализации. Список лит.: 30 назв.

UDK 519.71

Principles of logic automatic control by dynamic objects
Dubarenko V. V. IUS, 2003. — № 5. — P. 2–11.

Synthesizing methods of logic control systems are investigated. Methods provide improvement of quality of control by nonlinear dynamic objects. Realizing algorithms of these methods are offered. Refs: 30 titles.

УДК 629.7.062.5(075)

Применение нечеткой логики в авиационных системах антиюзовой автоматики
Коновалов А. С., Шумилов П. Е. Информационно-управляющие системы, 2003. — № 5. — С. 12–17.

Нечеткая логика уже получила широкое признание в задачах управления сложными системами. Авиационные системы антиюзовой автоматики характерны наличием существенных нелинейностей и высоким порядком математической модели объекта управления и представляют существенные сложности для классических методов синтеза регулятора, обеспечивающего эффективное торможение в различных условиях. В настоящей статье рассматриваются возможности использования нечеткой логики для синтеза регулятора авиационных систем антиюзовой автоматики и предлагается использование адаптивной базы правил управления регулятора для сохранения высокой эффективности торможения при различных состояниях взлетно-посадочной полосы. Список лит.: 16 назв.

UDK 629.7.062.5(075)

Application of fuzzy logic in aircraft antilock braking systems
Konovalov A. S., Shumilov P. E. IUS, 2003. — № 5. — P. 12–17.

Fuzzy logic has been already widely recognized in complex systems control. Aircraft antilock braking system (ABS) is a typical task for fuzzy logic, combining a number of significant non-linearities with high order of mathematical model of control object. For classical synthesis methods it is difficult to design an ABS regulator, which would have provided effective control in various braking conditions. This article discussed potentials of fuzzy logic in ABS regulator design and proposes the use of adaptive rule base to maintain high braking efficiency in different runway surface conditions. Refs: 16 titles.

УДК 621(075.8)

Логически прозрачные сети
Городецкий А. Е., Тарасова И. Л. Информационно-управляющие системы, 2003. — № 5. — С. 18–20.

Рассматриваются методы формирования логически прозрачных сетей. Выделяются следующие классы нейронных сетей: логико-лингвистические и логико-вероятностные. Анализируются условия и проблемы создания таких сетей. Список лит.: 2 назв.

UDK 621(075.8)

Logical-transparent nets
Gorodetsky A. E., Tarasova I. L. IUS, 2003. — № 5. — P. 18–20.

Methods of formation of logically transparent networks are considered The following classes of neural networks are entered: logical-linguistic and logical-probabilistic. It is analyzed conditions and problems of creation of such networks. Refs: 2 titles.

УДК 681.51:303.732+519.76

Структурно-целевой анализ в управлении системами производственной сферы
Лукьянова Л. М. Информационно-управляющие системы, 2003. — № 5. — С. 21–28.

Исследуется проблема повышения научно-технического уровня системного анализа при управлении рыбопромышленными объектами. Обсуждается возможный путь решения проблемы, основывающийся на структурно-целевой парадигме системного анализа. Рассматриваются постулаты, принципы, концепция поддержки структурно-целевого анализа и синтеза систем и ее реализация в полимодельной системе, основывающейся на логико-лингвистических формализациях. Список лит.: 20 назв.

UDK 681.51:303.732+519.76

Structure-and-purpose analysis in control of industrial systems
Lukianova L. M. IUS, 2003. — № 5. — P. 21–28.

The problem of brining scientific level of systems analysis in control for fishing industry objects up to standard is studied. Possible way of problem decision based on a structure-and-purpose paradigm of systems analysis is discussed. Postulates, principles, conception of structure-and-purpose analysis support, and its realization based on logical-and-linguistic formalizations are outlined. Refs: 20 titles.

УДК 681.3.06

Объектно-ориентированный подход к автоматному программированию
Шалыто А. А., Шопырин Д. Г. Информационно-управляющие системы, 2003. — № 5. — С. 29–39.

В данной работе предлагается подход к реализации объектно-ориентированных систем с явным выделением состояний. Рассмотрены вопросы повторного использования программных компонентов, параллельных вычислений, автоматического протоколирования работы системы и повышения отказоустойчивости системы. Список лит.: 9 назв.

UDK 681.3.06

Object-oriented approach to a automata programming
Shalyto A. A., Shopyrin D.G. IUS, 2003. — № 5. — P. 29–39.

In this work an approach to an implementation of object-oriented systems with obvious state dedication is given. Considered questions of reusing of software components, multithreading, automatic system work logging and increasing of fault tolerance of a system. Refs: 9 titles.

УДК 681.391.1

Передача со скрытым смыслом
Ерош И. Л. Информационно-управляющие системы, 2003. — № 5. — С. 40–46.

В статье решается задача, в которой сообщение, принятое различными группами пользователей, может быть прочитано не одинаково, так как в сообщении заложен многократный скрытый смысл. При этом передача сообщения может выполняться как в открытом виде, так и в зашифрованном (во втором случае его необходимо предварительно расшифровать). Рассмотрены два варианта решения этой задачи. В первом при передаче сообщения в осмысленный текст вставляется некоторый фрагмент, который разными группами пользователей должен читаться по-разному. Во втором каждая группа пользователей читает сообщение, адресованное только ей, остальные пользователи получают бессмысленный набор символов. Список лит.: 2 назв.

UDK 681.391.1

Transferring messages with hidden sense
Erosh I. L. IUS, 2003. — № 5. — P. 40–46.

The article is dedicated to resolution of a problem of interpretation of messages being received by different groups of users that may comprehend given message in different ways, because of multiple hidden implications associated with them. Message itself can be transmitted as well in clear as encrypted format. Prior decryption is required in the second case. Two possible resolutions of this problem are considered. The first resolution implies the insertion of a certain fragment into meaningful text of a message. Different users should interpret this fragment in different way. The second resolution assumes that each group of the users receives meaningful message addressed especially to it. The rest of the users are getting meaningless set of characters. Refs: 2 titles.

УДК 621-52:004.52

Технологии речевого управления для автоматизации производственных процессов
Изилов Я. Ю. Информационно-управляющие системы, 2003. — № 5. — С. 47–50.

Рассматриваются новые технологии для автоматизации различных производственных процессов, которые позволяют осуществить ввод данных, контроль и обеспечить оперативное управление промышленными системами с помощью речевых команд. Список лит.: 12 назв.

UDK 621-52:004.52

Speech control technologies for industrial processes automation
Izilov Y. U. IUS, 2003. — № 5. — P. 47–50.

This paper considers new technologies for various industrial processes automation which allows to input data, to carry out the checkup and to realize the operative control of industrial systems with the help of speech commands. Refs: 12 titles.

УДК 681.325.5

Модель представления учебного материала и способ диагностирования ошибок оператора в автоматизированной обучающей системе
Мамаев В. Я., Горбунов Д. А., Петров К. К. Информационно-управляющие системы, 2003. — № 5. — С. 51–58.

Построена структурная модель учебного материала, обладающая свойством полной выводимости целевого знания. Предложен способ диагностирования ошибок оператора, основанный на анализе характеристик целевого знания эталонной и фактической структурных моделей предметной области. Рассмотрен пример диагностирования возможных ошибок оператора-навигатора при решении одной из задач воздушной навигации. Список лит.: 6 назв.

UDK 681.325.5

Model of performance of a teaching material and way of diagnosing of errors of the operator-navigator in the automated learning system
Mamaev V. Y., Gorbunov D. A., Petrov K. K. IUS, 2003. — № 5. — P. 51–58.

The structural model of a teaching material is constructed, which one has property of full deducibility of target knowledge. The way of diagnosing of errors of the operator-navigator is offered, which one is based on the analysis of characteristics of target knowledge of reference and actual structural models of data domain. The example of diagnosing of possible errors of the operator-navigator is considered at the solution of one of problems of air navigating. Refs: 6 titles.

Редакция приносит извинения за опечатку, допущенную в номере 2–3/2003.
На стр. 44 следует читать: *Я. Ю. Изилов.*