ISSN 1684–8853

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЛАБОРАТОРИЯ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ ГЕОИНФОРМАЦИОННЫХ СИСТЕМ САНКТ-ПЕТЕРБУРГСКОГО ИНСТИТУТА ИНФОРМАТИКИ И АВТОМАТИЗАЦИИ РОССИЙСКОЙ АКАДЕМИИ НАУК

5(12)/2004

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

5(12)/2004

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Адрес редакции: 190000, Санкт-Петербург, Б. Морская ул., д. 67
Тел.: (812) 110-66-42, (812) 313-70-88
Факс: (812) 313-70-18
E-mail: ius@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 каталогам агентства «Роспечать»:
«Газеты и журналы» – № 15385,
«Издания органов НТИ» – № 69291

ОБРАБОТКА ИНФОРМАЦИИ И УПРАВЛЕНИЕ

Розов А. К., Бухарцев М. Н. *Классификация морских объектов* 2

ПРОГРАММНЫЕ И АППАРАТНЫЕ СРЕДСТВА

Дехканбаев Д. С. *Уменьшение временной сложности алгоритмов вычисления фрактальной размерности трехмерных точечных фракталов* 7

Шамгунов Н. Н., Корнеев Г. А., Шалыто А. А. *State Machine – новый паттерн объектно-ориентированного проектирования* 13

ЗАЩИТА ИНФОРМАЦИИ

Бубликов А. Б. *О свойствах двоичных матриц, используемых в алгоритмах кодирования информации булевыми преобразованиями* 26

МОДЕЛИРОВАНИЕ СИСТЕМ И ПРОЦЕССОВ

Алексеев К. В., Решетникова Н. Н. *3D интерактивная модель наблюдений астероидов. Позиционирование телескопа и ПЗС-наблюдения* 28

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ОБРАЗОВАНИЕ

Федоров И. Б., Коршунов С. В., Советов Б. Я. *Перспективы подготовки кадров по направлению «Информационные системы»* 38

Яковлев С. А. *Методические основы использования имитационного моделирования в учебном процессе при подготовке по направлению 654700 – информационные системы* 44

КРАТКИЕ СООБЩЕНИЯ

Бестугин А. Р. *Иерархия протоколов для технологии ATM* 50

ХРОНИКА И ИНФОРМАЦИЯ

Рекурсивные вычислительные системы 54

СВЕДЕНИЯ ОБ АВТОРАХ 58

АННОТАЦИИ 61

ЛР № 010292 от 18.08.98.
Сдано в набор 30.09.2004. Подписано в печать 25.10.2004. Формат 60×901/8.
Бумага офсетная. Гарнитура Pragmatica. Печать офсетная.
Усл. печ. л. 8,0. Уч.-изд. л. 9,0. Тираж 1000 экз. Заказ 420.

Оригинал-макет изготовлен
в отделе электронных публикаций и библиографии ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

Отпечатано с готовых диапозитивов
в отделе оперативной полиграфии ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

УДК 681.516.7.015.2

КЛАССИФИКАЦИЯ МОРСКИХ ОБЪЕКТОВ

А. К. Розов,
доктор техн. наук
Военно-морская академия им. Н. Г. Кузнецова
М. Н. Бухарцев,
доктор техн. наук, профессор
Федеральное Государственное унитарное предприятие «Первый центральный научно-исследовательский институт Министерства обороны России»

Классификация морских объектов может быть осуществлена с использованием аппарата стохастических дифференциальных уравнений. Полученные в результате решения уравнений апостериорные вероятности гипотез находят применение в алгоритме принятия решений. Приводятся примеры, иллюстрирующие процедуру классификации.

Classification of sea objects can be carried out by using stochastic difference equation. Inverse probabilities obtained as solutions of these equations are applied to the decision-making algorithms. Examples of classification are provided.

В настоящее время высказывается много предложений по использованию различий в параметрах эхо-сигналов для классификации объектов. Это различие в доплеровском сдвиге частот, различие в удлинении эхо-сигнала, ширине спектра, числе бликов в бликовом портрете целей и др. При этом каждый из признаков, как правило, работает только в «своих» условиях: доплеровский сдвиг частоты – при различии в относительных скоростях объектов, удлинение сигнала – при различии в протяженности объектов, уширение спектра – в случаях, когда есть то и другое различие и т. д. Для того чтобы они работали одновременно, алгоритм классификации должен учитывать связи между параметрами эхо-сигналов. А эти связи зависят, главным образом, от поведенческих характеристик объектов.

В свою очередь, поведенческие характеристики объектов определяются задачами, для решения которых они предназначены, и характером их поведения. По ряду причин поведенческие характеристики в определенной мере случайны, поскольку случайны параметры, их определяющие: координаты объекта – дистанция от объекта d и направление на него φ, а также элементы его движения – скорость v и курс q. Количественной мерой случайности является вероятностное распределение $F_t(d, \varphi, v, q)$, где t – время, прошедшее после начала движения локатора.

Поскольку локатор имеет дело не с данными, характеризующими движущийся объект, а с зависящими от них задержкой сигнала Θ и значениями его параметров $\lambda_1, \ldots, \lambda_n$, то распределение $F_t(d, \varphi, v, q)$ может быть с помощью детерминированных зависимостей $\lambda_i = f_i(d, \varphi, v, q)$, $i = 1, \ldots, n$, пересчитано в распределение $F_t(\Theta, \lambda_1, \ldots, \lambda_n)$.

Далее следует классическая схема разрешения гипотез о том, чем является классифицируемый объект. При таком подходе все признаки (параметры), их связь между собой, а также зависимость от поведенческих характеристик можно представить одним функционалом – апостериорной вероятностью той или иной гипотезы относительно классифицируемого объекта.

Важно, что изложенный подход учитывает множественность признаков, а предлагаемые способы использования отдельных признаков могут быть получены как частные варианты этого общего подхода. Однако эффективность таких частных вариантов будет тем меньше, чем меньше параметров эхо-сигнала участвует в классификации.

Важно и другое. Существующие методы обработки наблюдаемых воздействий базируются на использовании корреляционно-спектральных подходов. Последние достаточны, если предполагать возможность непосредственного измерения параметров сигнала (см. примеры 2 и 3), но оказываются непригодными при наличии помех.

В условиях воздействия помех алгоритм классификации должен строиться с использованием аппарата стохастических дифференциальных уравнений. Этот математический аппарат обработки наблюдаемых воздействий пока еще не нашел практического применения, но за ним боль-

шое будущее. Становлению и развитию этого аппарата посвящен ряд работ [1–5]. Здесь остановимся лишь на отличии стохастических дифференциальных уравнений от обыкновенных.

Изучение эволюции меняющихся во времени явлений часто начинается с написания дифференциального уравнения вида

$$\frac{dx_t}{dt} = f(x_t, t)$$

относительно функции x_t. Когда производная $f(x_t)$ детерминированно определена, нет необходимости применять теорию случайных процессов. Когда же $f(x_t, t)$ с течением времени подвергается случайным воздействиям, уравнение для x_t может быть представлено в виде

$$dx_t = a(x_t, t)dt + b(x_t, t)dw_t,$$

где dw_t – дифференциал винеровского процесса.

При этом важно следующее: для того чтобы x_t сохранял случайный характер, дифференциал dw_t должен иметь $\sqrt{\Delta}$ порядок малости. Именно этим свойством обладает винеровский процесс, для которого

$$M(w_{t_{K+1}} - w_{t_K})^2 = t_{K+1} - t_K = \Delta.$$

Выбор состава параметров. Могут быть приняты разные подходы или условия выбора состава параметров сигнала. Таким условием, например, может выступать требование о независимости параметров от дистанции до объекта. Но можно выбирать параметры в зависимости от способа решения задачи:

можно состав параметров выбирать таким, чтобы их распределение допускало его аппроксимацию нормальным или другим распределением в аналитической форме, что позволит придать правилу принятия решений также аналитический характер;

можно предположить, что помимо возможности нормальной аппроксимации параметры сигнала линейно входят в представляющее сигнал дифференциальное уравнение, что позволит представить сигнал условно-гауссовским процессом и, тем самым, использовать открывающиеся при этом возможности для оценивания параметров сигнала.

Как правило, исходное распределение элементов движения в силу неполноты информации может быть принято равномерным. В ходе трансформации оно обычно превращается в неравномерное распределение параметров сигнала. Эта неравномерность может быть учтена использованием байесовского подхода в формировании процедуры классификации и, тем самым, могут быть получены все связанные с данным подходом преимущества. А они немалые – байесовский подход приводит к выбору гипотез относительно принадлежности объекта к одному из классов на базе вычисления их апостериорных вероятностей – статистик, наиболее полно использующих информацию об остановке.

Байесовский подход должен учитывать завершающий характер процедуры классификации, предусматривающий остановку наблюдений. Составить алгоритм такой процедуры позволяет теория оптимальных правил остановки [4].

При байесовском подходе с каждым правилом $\delta(\nu, d)$, в котором ν – момент прекращения наблюдений, d – решение относительно принадлежности объекта к классу K, связаны потери, учитываемые средним риском

$$R(\delta) = cM\nu + M\bar{W}(d, K),$$

где $cM\nu$ – стоимость наблюдений; $MW(d, K)$ – слагаемое, учитывающее вероятность ошибочных решений.

Согласно байесовскому правилу, в каждый момент наблюдений решается вопрос о целесообразности продолжения наблюдений, которая определяется путем сопоставления ожидаемых потерь при продолжении наблюдений и потерь от остановки.

Момент остановки определяется из условия

$$\nu^* = \min\left\{t : R_t\left(\eta_0^t\right) = R_t^T\left(\eta_0^t\right)\right\},$$

где $R_t(\eta_0^t)$ – риск от продолжения наблюдений η_0^t на интервале $[0, t]$; $R_t^T\left(\eta_0^t\right)$ – риск от продолжения наблюдений на интервале $[0, T]$.

Решающее правило предполагает сравнение максимальной составляющей статистики $\pi_l(t)$, $l = 1, \ldots, m$, с границей Γ_l областей принятия решений:

$$d^* = \begin{cases} \nu^* = \inf\left[t > 0, \max \pi_l(\nu) \geq \Gamma_l(t)\right]; \\ d_1 \max \pi_l(t) = \pi_1\left(\nu^*\right) \geq \Gamma_1(t); \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots; \\ d_m, \max \pi_l(t) = \pi_m\left(\nu^*\right) \geq \Gamma_m(t). \end{cases}$$

Реализация байесовского подхода возможна при представлении сигнала и помехи с помощью потраекторного динамического метода.

Многомерный сигнал

$$\Theta_t = \left[\Theta_1(t), \ \ldots, \Theta_K(t)\right]$$

можно представить стохастическим дифференциальным уравнением

$$d\Theta_t = \left[a_0(\lambda, t) + a_1(\lambda, t)\Theta_t\right]dt + b(\lambda, t)dw_t^{(S)},$$

где a_0 и a_1 – коэффициенты, зависящие от параметров сигнала; $W_t^{(S)}$ – винеровский процесс. Наблюдаемое воздействие

$$\eta_t = \left[\eta_1(t), \ \ldots, \eta_n(t)\right]$$

представим уравнением

$$d\eta_t = \left[A_0(\lambda,t) + A_1(\lambda,t)\Theta_t\right]dt + B(\lambda,t)dw_t^{(n)}.$$

Дифференциальное представление сигнала и наблюдаемых воздействий предопределяет построение процедуры классификации на дифференциальной основе.

Можно показать, что уравнения для одномерных вероятностей имеют вид

$$d\pi_l(t) = \frac{1}{B^2}\pi_l(t)\left[A_1 m_\Theta(t \mid F_l) - \sum_{j=1}^{m} A_1 m_\Theta(t \mid F_j)\pi_j(t)\right] \times$$

$$\times\left[d\eta_t - \sum_{j=1}^{m} A_1 m_\Theta(t \mid F_j)\pi_j(t)dt\right];$$

$$\pi_l(0) = P_l,$$

где $m_\Theta(t \mid F_l) = \left[m_{\Theta 1}(t \mid F_l),\ \ldots, m_{\Theta K}(t \mid F_l)\right], l = 1,\ \ldots, m$ – вектор оценок текущих значений сигнала, вычисляемых с использованием уравнений Калмана – Бьюси.

В приведенные уравнения наблюдаемое воздействие η_t входит в виде дифференциала $d\eta_t$, а при их решении с помощью рекуррентных соотношений – в виде приращений

$$\Delta\eta_{t_K} = \eta_{t_{K+1}} - \eta_{t_K}.$$

Такой характер считывания обусловлен тем, что сам процесс

$$\eta_t = \int_0^t \Theta_s ds + Bw_t^{(n)}$$

нестационарен, в то время как приращения

$$\Delta\eta_{t_k} = \int_{t_K}^{t_{K+1}} \Theta_s ds + B\left(w_{t_{K+1}}^{(n)} - w_{t_K}^{(n)}\right)$$

стационарны, хотя и требуют интегрирования наблюдаемых воздействий за время шага временной дискретизации

$$\Delta = t_{K+1} - t_K.$$

Необходимые значения B, входящие в уравнения в виде коэффициентов $1/B^2$, находятся в результате вычисления $M(\Delta\eta_{t_K})^2$ и приравнивания его к $B^2\Delta$.

Нахождение оценок $m_\Theta(t \mid F_l)$ возможно при дискретизации распределения F_l, $l = 1, \ldots, m$, что делает целесообразным изначально получать его в виде гистограммы. Последняя получается в результате розыгрыша элементов движения объекта и использования детерминированной зависимости от них параметров сигнала.

Может иметь место частный случай, когда помехи малы и возможно непосредственное измерение параметров сигнала без затраты времени наблюдения. В этом случае выбор гипотезы относительно принадлежности сигнала к одному из классов производится в результате нахождения апостериорной вероятности $\pi_l(t)$, $l = 1, 2$, по формуле Баеса

$$\pi_l^{ij} = \frac{P(\lambda_{1i}, \lambda_{2j} \mid F_l)P(F_l)}{\sum_{K=1}^{2} P(\lambda_{1i}, \lambda_{2j} \mid F_K)P(F_K)},$$

где λ_{1i}, λ_{2j} – измеренные значения параметров сигнала, соответствующие клеткам двумерной гистограммы и сравнения $\max_l \pi_l$ с границей Γ.

Наряду с гистограммным вычислением π_l^{ij} по формуле Байеса целесообразен аналитический вариант, когда гистограмма аппроксимируется непрерывным распределением, например, нормальным. В этом случае отпадает необходимость удержания в памяти ЭВМ гистограмм, достаточно иметь в памяти только моменты (математическое ожидание, дисперсию и т. д.).

Проиллюстрируем сказанное примерами. Будем предполагать, что объекты имеют одинаковую длину L = 100 м, отстоят от классификатора на расстояние 2000 м, а элементы движения равномерно распределены в интервалах

$$\upsilon_1 \in [5\ \text{м/с},\ 15\ \text{м/с}],$$
$$q_1 \in [0^\circ,\ 90^\circ];$$
$$\upsilon_2 \in [9\ \text{м/с},\ 11\ \text{м/с}],$$
$$q_2 \in [0^\circ,\ 18^\circ].$$

Пример 1. Классификация объектов, для которых эхо-сигнал представляется высокочастотным процессом с дифференциалом

$$d\Theta_t = I_t dt;$$

$$d\nu_t = -(b_1 I_t + b_0\Theta_t)dt + b_1^*\sqrt{\left(b_0^* - \frac{b_1^{2^*}}{4}\right)}C_1 dw_t^{(S)},$$

в котором параметры b_0 и b_1 определяются формулами

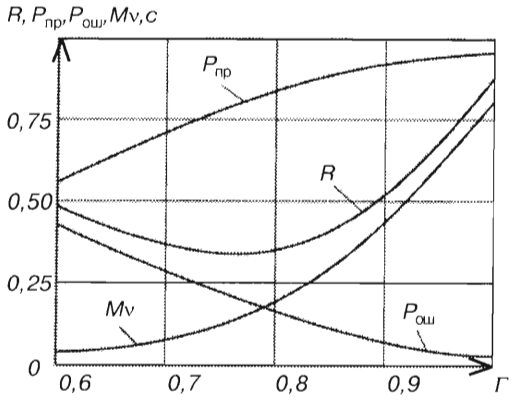
$$b_0 = \sqrt{\nu};$$
$$b_1 = \Delta\nu_1,$$

где ν – доплеровский сдвиг частоты; $\Delta\nu_1$ – ширина спектра сигнала (см. пример 2).

Наблюдаемое воздействие η_t представляется дифференциалом

$$d\eta_t = \Theta_t dt + \sqrt{C_2}dw_t^{(n)}.$$

Выбор гипотезы относительно принадлежности сигнала к одному из классов (и соответственно, объектов) производится в результате решения приведенных уравнений для $\pi_l(t)$ в двумерном варианте и сопоставления $\max_l \pi l(t)$ с границами областей принятия решений.



■ ***Рис. 1.*** *Вероятность правильной ($P_{пр}$) и ошибочной ($P_{ош}$) классификации, затрачиваемого на классификацию времени ($M\nu$) и среднего риска (R) в зависимости от границы (Γ) областей принятия решений*

Моделирование проводилось при $C_1 = 1$ с и параметрах b_0 и b_1, распределенных по нормальным законам:

$$N_1\left[0{,}5b_0^*;\ 0{,}5b_1^*,\ \left(\frac{1}{4}b_0^*\right)^2,\ \left(\frac{1}{4}b_1^*\right)^2\right];$$

$$N_2\left[b_0^*;\ b_1^*,\ \left(\frac{1}{4}b_0^*\right)^2,\ \left(\frac{1}{4}b_1^*\right)^2\right],$$

$$b_0^* = 2518\pi^2\ \text{с}^{-2},$$

$$b_1^* = 6\pi\ \text{с}^{-1},$$

$$P_1 = P_2 = \frac{1}{2}.$$

Помеха – белый шум с $C_2 = 0{,}4$ с.

Результаты моделирования – зависимость вероятности правильной и ошибочной классификации, затрачиваемого времени $M\nu$ и среднего риска $R = 0{,}1M\nu + P_{ош}$ от границы областей принятия решений Γ – представлены на рис. 1.

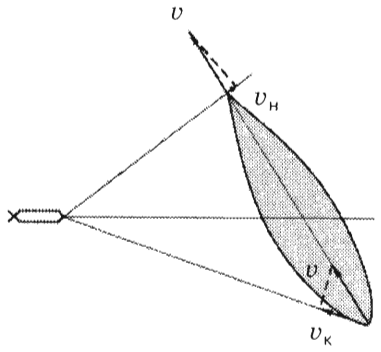
Пример 2. Классификация объектов, для которых возможно непосредственное измерение параметров сигнала – доплеровского сдвига частоты ν и протяженности энергетического спектра $\Delta\nu$, определяемых формулами

$$\nu = \frac{2\upsilon\cos q \cdot f}{C_{зв}};$$

$$\Delta\nu = \nu_н - \nu_к,$$

где f – частота локатора, Гц (для расчетов принята равной 30 кГц); $C_{зв} = 1500$ м/с – скорость звука в воде; $\nu_н$ и $\nu_к$ – доплеровские сдвиги частоты, обусловленные проекциями скорости υ на концевые участки объектов – $\upsilon_н$ и $\upsilon_к$ (рис. 2).

В отличие от примера 1, в этом случае предполагается, что ходовая помеха, связанная с движением локатора, отсутствует или настолько мала,



■ ***Рис. 2.*** *Схема вычисления скоростей концевых участков ($\upsilon_н$, $\upsilon_к$) объекта, движущегося со скоростью υ*

что ею можно пренебречь. Такая ситуация имеет место на сравнительно малых дистанциях до классифицируемых объектов.

Полученные в результате статистического эксперимента $F_l(\nu, \Delta\nu)$, $l = 1, 2$, двумерные гистограммы были аппроксимированы двумерным нормальным распределением с моментами

№ объекта	$M\nu$, Гц	$D\nu$, Гц^2	$M\Delta\nu$, Гц	$D\Delta\nu$, Гц^2	r
1	38,2	3,19×104	16,6	9,23	0,53
2	275	1,49×103	13,5	1,2	–0,49

Из приведенных данных видно, что параметры эхо-сигналов от объекта с широким диапазоном изменения элементов движения распределены в весьма широких областях, в то время как параметры эхо-сигналов от объекта с суженным диапазоном группируются в локальных областях. Это определяет, как уже отмечалось, целесообразность использования байесовского подхода для решения задач классификации.

Вероятности правильной классификации $P_{пр}$, вычисленные по двум гистограммам и аналитически с аппроксимацией их нормальными распределениями, оказались равными $P_{гист} = 0{,}95$ и $P_{аппр} = 0{,}93$. Проигрыш от введения аппроксимации оказался небольшим – 0,02. Вероятности $P_{пр}$, вычисленные с использованием одномерных гистограмм по ν и $\Delta\nu$, оказались меньшими соответственно на 0,13 и 0,15.

Пример 3. Классификация объектов, для которых возможно непосредственное измерение параметров – доплеровского сдвига и удлинения сигнала Δl, определяемых формулами

$$\nu = \frac{2\upsilon\cos q \cdot f}{C_{зв}};$$

$$\Delta l = L\cos q.$$

Отличительной особенностью данного примера является то, что для классификации объектов

используются такие параметры сигнала (ν, Δl), которые не зависят от дистанции до классифицируемых объектов и от их угловой протяженности. Отсутствие необходимости учитывать дистанцию до классифицируемых объектов намного упрощает процедуру классификации.

Полученные в результате статистического эксперимента $F_l(\nu, \Delta l)$, $l = 1, 2$, двумерные гистограммы были аппроксимированы двумерным нормальным распределением с моментами

№ объекта	$M\nu$, Гц	$D\nu$, Гц^2	$M\Delta l$, м	$D\Delta l$, м^2	r
1	276	$2{,}28 \cdot 10^4$	66,3	992	0,84
2	420	803	103	19,5	0,11

Вероятности правильной классификации ($P_{пр}$), вычисленные по двумерным гистограммам и аналитически с аппроксимацией их нормальными распределениями, оказались равными $P_{гист} = 0{,}95$ и $P_{аппр} = 0{,}89$. Проигрыш от введения аппроксимации оказался равным 0,06. Вероятности $P_{пр}$, вычисленные с использованием одномерных гистограмм по ν и Δl, оказались меньшими соответственно на 0,15 и 0,18.

Таким образом, теоретически обоснована и подтверждена моделированием перспективность байесовского подхода для составления алгоритмов, обеспечивающих высокую (0,8–0,9) вероятность правильной классификации объектов.

Литература

1. **Калман Р. Е, Бьюси Р. С.** Новые результаты в линейной фильтрации и теория предсказания / Пер. с англ. // Техническая механика. – № 83. Сер. Д. 1, 1961. – С. 95–107.
2. **Липцер Р. Ш., Ширяев А. Н.** Статистика случайных процессов. – М.: Наука, 1974. – 696 с.
3. **Скороход А. В.** Асимптотические методы теории стохастических дифференциальных уравнений. – Киев: Наукова думка, 1987.
4. **Роббинс Г., Сигмунд Д., Чао И.** Теория оптимальных правил остановки / Пер. с англ. – М.: Наука, 1975. – 168 с.
5. **Эллиот Р.** Стохастический анализ и его приложения. – М.: Мир, 1986.

УДК 681.3:519.2

УМЕНЬШЕНИЕ ВРЕМЕННО́Й СЛОЖНОСТИ АЛГОРИТМОВ ВЫЧИСЛЕНИЯ ФРАКТАЛЬНОЙ РАЗМЕРНОСТИ ТРЕХМЕРНЫХ ТОЧЕЧНЫХ ФРАКТАЛОВ

Д. С. Дехканбаев,
инженер
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Рассматриваются методы вычисления фрактальной размерности трехмерных точечных фракталов. Получены оценки временно́й сложности алгоритмов и времени выполнения программ. Предлагается модификация алгоритмов для уменьшения временно́й сложности и распараллеливание программ для уменьшения времени выполнения.

The methods for fractal dimension calculation of 3D point-like structures are considered. Estimate of time complexity of algorithms and run time of programs are derivated. Modifications of algorithms and scheme of parallelism are offered. Time complexity of algorithms and run time of programs are reduced.

Введение

Фрактальная размерность является наиболее употребительной характеристикой точечных фракталов. За исключением небольшого количества классических точечных фракталов, обладающих специальными свойствами, фрактальная размерность вычисляется только численно. Трехмерные точечные фракталы применяются, например, в астрофизике, при моделировании крупномасштабной структуры Вселенной [1].

Алгоритмы вычисления фрактальных размерностей основаны на учете взаимного расположения всех точек структуры, что приводит к большой временно́й сложности. Другое свойство этих алгоритмов – эффективное распараллеливание: фрактальную размерность можно вычислять одновременно в окрестности каждой точки либо одновременно на каждом масштабе.

Целью работы является исследование алгоритмов вычисления фрактальной размерности методом объемной условной плотности, оболочечной условной плотности и методом выделенных цилиндров.

Модели трехмерных точечных фракталов

Мандельброт определил фрактал так: «фрактал – это структура, состоящая из частей, которые в каком-то смысле подобны целому» [2]. Это определение не конструктивно, так как не содержит способа построения фрактала. В противоположность этому фрактальная размерность определена конструктивно. Рассмотрим одну из фрактальных размерностей – размерность подобия [3]. Размерность D объекта появляется как показатель степени r в соотношении между числом равных подобъектов N и коэффициентом подобия r:

$$Nr^D = 1.$$

Величину D называют фрактальной размерностью или размерностью подобия. Выражение для D через N и r находится логарифмированием обеих частей:

$$D = \frac{\log N}{\log(1/r)}. \qquad (1)$$

Другой фрактальной размерностью является массовая размерность [4]. Определение плотности непрерывной среды основано на предположении о том, что значение плотности не зависит от величины объема. В случае точечных фракталов понятия плотности точек не существует, так как в каждой части структуры содержится иерархия кластеров. В этом случае говорят об условной плотности, значение которой зависит от объема. Для ее описания необходимо ввести независи-

мую переменную – радиус области R, в которой производится подсчет точек. Например, для самоподобной структуры число точек растет по степенному закону:

$$N_p(R) \sim R^D,$$

где $N_p(R)$ – число точек в области радиуса R. Фрактальная размерность, называемая массовой размерностью, вычисляется по формуле

$$D = \frac{\log\left[N_p(R)\right]}{\log R}.$$

Для характеристики плотности точечных фракталов также используется функция $n_p(R)$ [5]:

$$n_p(R) = \frac{N_p(R)}{V} \approx R^{-(3-D)},$$

где $n_p(R)$ – концентрация точек в области радиуса R; $3-D$ – фрактальная коразмерность. Фрактальная размерность вычисляется по формуле

$$D = \frac{\log\left[n_p(R)\right]}{\log R} + 3. \qquad (2)$$

Численные методы нахождения фрактальной размерности трехмерных точечных структур

Существует большое количество определений различных фрактальных размерностей. Для самоподобных фракталов величина размерности не зависит от выбора определения. В случае самоафинных фракталов при любом определении фрактальной размерности существенен лишь способ нахождения размерности: интерполяция или экстраполяция; при этом получаются два разных значения размерности – локальное (справедливое для масштабов, меньших некоторого критического) и глобальное (справедливое для масштабов, больших критического). От того, какой способ нахождения размерности был использован, оба эти значения не зависят [6]. Чем более сложные фракталы рассматриваются, тем большее число фрактальных размерностей оказывается необходимо для их описания. Не все определения фрактальных размерностей реализованы в виде численных методов. Наиболее употребительными методами вычисления фрактальной размерности точечных структур в статистической физике и астрономии являются метод корреляционных функций, метод объемной условной плотности и метод оболочечной условной плотности [1, 7]. В нашей работе оценивается временна́я сложность и модифицируются методы объемной и оболочечной условной плотности и метод выделенных цилиндров [8].

В методе объемной условной плотности вычисление фрактальной размерности производится за два действия: 1) вычисление условной плотности как функции масштаба; 2) аппроксимация полученных данных прямой.

Действие 1. Вычисление условной плотности:

точку № 1 помещаем в центр куба со стороной R, находим среднюю плотность точек в этом кубе, деля количество точек в кубе на его объем;

повторяем эту операцию для каждой из N точек;

находим среднее арифметическое полученных N значений плотностей. Таким образом, мы получили одну точку на графике в координатах: логарифм средней плотности $\lg\left[n_p(R)\right]$ (по вертикали); логарифм характерного масштаба $\lg(R)$ (по горизонтали). Далее повторяем эту процедуру, изменяя длину стороны куба R, и получаем график зависимости $\lg\left[n_p(R)\right]$ от $\lg(R)$. В качестве пределов длины стороны куба используется среднее расстояние между точками и размер всей системы.

Действие 2. Аппроксимация полученных данных производится прямой $y = a + bx$ методом наименьших квадратов. Фрактальная размерность согласно формуле (2) определяется как $D = b + 3$.

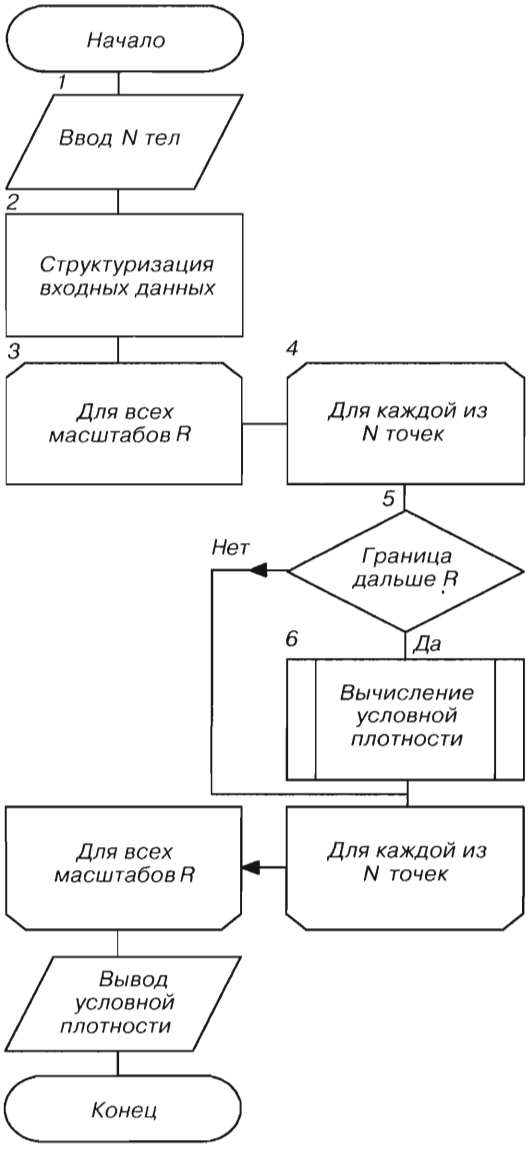
Метод оболочечной условной плотности отличается видом области, в которой вычисляется условная плотность: в методе объемной условной плотности – это шар с центром в выделенной точке; в методе оболочечной условной плотности – это сферический слой вокруг выделенной точки. Вместо сферы в этих методах также можно использовать куб. Другим отличием является величина шага масштаба: в методе объемной условной плотности он изменяется равномерно в логарифмическом масштабе, а в методе оболочечной условной плотности – равномерно в линейном масштабе. Ввиду такого сходства, временна́я сложность алгоритмов, реализующих эти методы, одинакова. Время выполнения обоих методов также сходно. Непосредственное сравнение их затруднено разным шагом – логарифмическим у объемного и линейным у оболочечного метода. По причине линейного шага метод оболочечной условной плотности может ближе подойти к максимально возможному масштабу – толщине объекта. Минимальный масштаб в методе оболочек не может быть меньше величины шага масштаба, т. е. толщины оболочки. Сравнение методов объемной и оболочечной условной плотности приведено в табл. 1.

Метод цилиндров существенно отличается от обоих методов условной плотности. Вычисление фрактальной размерности производится за два действия: 1) вычисление вероятности распределения точек $P(a)$ в цилиндрах, соединяющих пары точек; 2) аппроксимация полученных данных при помощи формулы

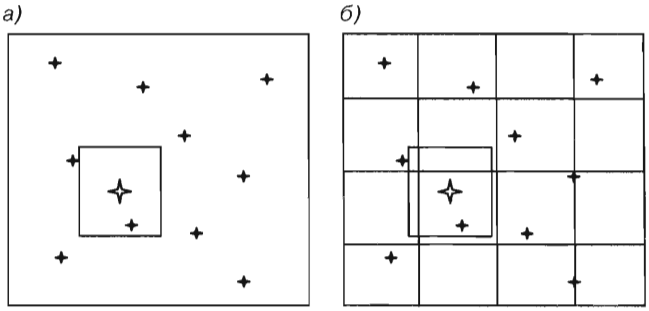
$$P(a) = A\left\{\left(\frac{r}{x}\right)^{-\alpha}\left[a^{-\alpha} + (1-a)^{-\alpha}\right] + B\right\},$$

■ ***Таблица 1.*** *Сравнение методов объемной и оболочечной условной плотности*

Критерии	Метод условной плотности	
	объемной	оболочечной
Наибольший максимальный масштаб	–	+
Наименьший минимальный масштаб	+	–
Малая дисперсия	+	–
Нет систематических ошибок	–	+



■ ***Рис. 1.*** *Укрупненная схема алгоритма реализации методов объемной и оболочечной условной плотности*



■ ***Рис. 2.*** *Разбиение пространства на ячейки для ускорения поиска соседних точек*

где a – масштаб, $0 \le a \le 1$; r и x – максимальный и минимальный масштабы, на которых проявляется фрактальность; α – фрактальная коразмерность; A и B – константы.

Поскольку в этом методе цилиндр строится на паре выделенных точек, он позволяет вычислять фрактальную размерность на максимальных масштабах, недостижимых для методов условной плотности в случае вытянутого фрактала. Максимальный масштаб методов условной плотности ограничен толщиной объекта.

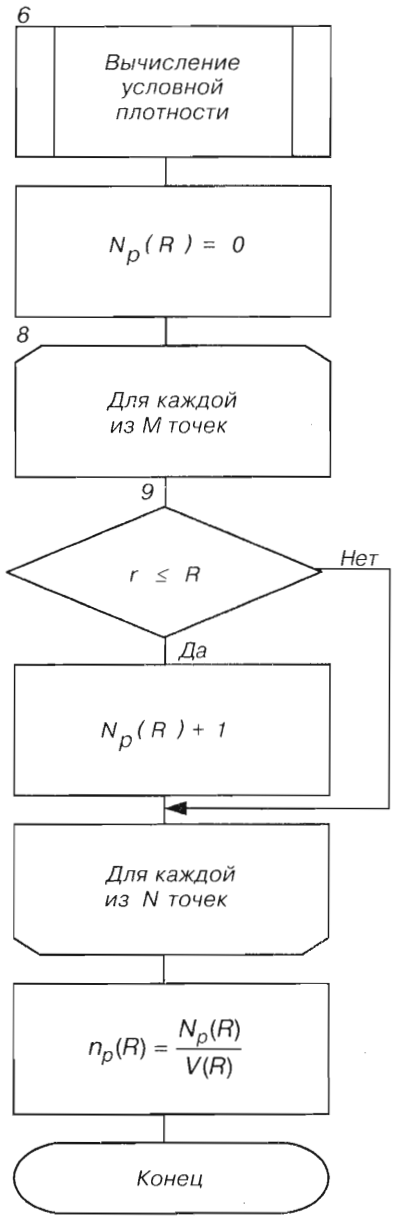
Методы объемной и оболочечной условной плотности отличаются только в выборе шага масштаба R и процедуре вычисления условной плотности, поэтому на рис. 1 для них приведена общая схема работы. В отличие от методов условной плотности, в методе цилиндров шаг 3 отсутствует, а на шаге 4 перебираются не точки, а пары точек, следовательно, цикл содержит не N итераций, а $1/2N^2$.

Для ускорения работы программ вычисления фрактальной размерности предлагается структурировать входные данные. После ввода исходных данных вычисляется объем параллелепипеда, описанного вокруг всех точек структуры, и строится трехмерный массив ячеек, заполняющих весь параллелепипед (на рис. 2 приведена проекция массива ячеек на координатную плоскость). Для каждой ячейки создается список точек, попадающих в нее. Тогда для нахождения условной плотности (шаг 6) не надо перебирать все N точек, достаточно проверить лишь M точек ($M \le N$), попадающих в смежные ячейки, которые перекрываются ячейкой, построенной вокруг выделенной точки (см. рис. 2).

Как видно на рис. 2, для нахождения плотности в выделенном кубе надо перебрать 11 точек (см. рис. 2, *а*) или 3 точки (см. рис. 2, *б*), где $N = 11$, $M = 3$. Процедура вычисления объемной условной плотности с использованием такой структуризации данных приведена на рис. 3.

Временна́я сложность

Для оценки временно́й сложности алгоритмов известны вероятностный и статистический подходы [9]. Вероятностный анализ строится на анализе алгоритма (без его реализации), основыва-



■ ***Рис. 3.*** *Схема алгоритма вычисления объемной условной плотности*

ясь на вероятностных характеристиках входных данных. Статистический анализ строится на основании результатов многократного выполнения реализованного на ЭВМ алгоритма статистическими методами. Рассмотрим сначала временну́ю сложность алгоритма при помощи вероятностного подхода.

Время работы алгоритма (см. рис. 1 и 3) зависит [10] от следующих величин: числа операций, выполняемых на каждом шаге алгоритма; числа выполнений циклов 3, 4 и 8 и вероятности ответа «да» на шагах 5 и 9.

Обозначим число операций на шагах 1, 2, ..., 9 через $A_1, A_2, \ldots, A_9$; число выполнений циклов 3, 4 и 8 через R, N и M; вероятность ответа «да» на шагах 5 и 9 через P_1 и P_2. Число операций на непронумерованных шагах (см. рис. 1 и 3) учтем в ближайших сверху пронумерованных шагах.

■ ***Таблица 2.*** *Параметры алгоритма условной плотности*

$A_1 = N$	$A_4 = 27$	$A_8 = 17$	$P_1 = P_1(R)$
$A_2 = N + k$	$A_5 = 30$	$A_9 = 1$	$P_2 = P_2(R)$
$A_3 = 12$	$A_7 = N$	R – параметр	$M \in \left[\frac{N}{k}, N\right]$

Входными данными являются N точек фрактала, число масштабов R, на которых вычисляется условная плотность, и число ячеек k. В табл. 2 приведены численные значения параметров алгоритма.

Время выполнения программы зависит от длины входных данных N и параметров R и k. Нас будет интересовать зависимость только от N и k, поэтому обозначим время выполнения программы через $t = f(N, k)$. Исследуем зависимость времени выполнения программы от длины входных данных N:

$$f(N) = A_1 + A_2 + RA_3N\left[A_4 + P_1A_5M\left(A_8 + P_2A_9\right)\right] + A_7$$

или

$$f(N) = 3N + k + cN + b\frac{N^2}{\sqrt{k}}, \qquad (3)$$

где $c = RA_3A_4$; $b = P_1A_5(A_8 + P_2A_9)c / A_4$. Поскольку постоянные множители не учитываются при оценке временно́й сложности, временна́я сложность – $O(N^2)$. Чтобы уменьшить временну́ю сложность, свяжем параметр k с длиной входных данных N – рассмотрим зависимость времени выполнения программы от k:

$$f(k) = 3N + k + cN + b\frac{N^2}{\sqrt{k}}.$$

Эта функция имеет минимум в точке $k_m = N^{\frac{4}{3}}\left(\frac{b}{2}\right)^{\frac{2}{3}}$. Подставив k_m в уравнение (3) получим:

$$f(N) = (3 + c)N + N^{\frac{4}{3}}\left(\frac{b}{2}\right)^{\frac{2}{3}} + b^{\frac{2}{3}}2^{-\frac{1}{3}}N^{\frac{4}{3}} =$$

$$= (3 + c)N + b^{\frac{2}{3}}\left(2^{-\frac{2}{3}} + 2^{-\frac{1}{3}}\right)N^{\frac{4}{3}}.$$

В этом случае временна́я сложность $O(N^{4/3})$.

Временна́я сложность алгоритмов, реализующих методы объемной и оболочечной условной плотности, одинакова.

Применив метод ячеек к вычислению фрактальной размерности точечных структур и установив число ячеек $k = k_m$, вместо временно́й сложности $O(N^2)$ получаем $O(N^{4/3})$. Теперь перейдем к

статистическому анализу программ, реализующих эти алгоритмы.

Время выполнения

Для уменьшения времени выполнения программ, реализующих методы вычисления фрактальной размерности, применяется MPI (Message passing interface) – стандарт на программный инструментарий для обеспечения связи между ветвями параллельного приложения. Основную часть времени вычислений выполняется внешний цикл (шаг 3 на рис. 1), в котором перебираются все точки (в методе условной плотности) или все пары точек (в методе цилиндров). Операции ввода-вывода до и после него выполняются сравнительно быстро. Для параллельного выполнения программы на нескольких машинах есть следующие схемы: 1) распараллелить цикл 3; 2) распараллелить цикл 4. Рассмотрим преимущества и недостатки обоих способов.

При вычислении на нескольких процессорах время выполнения t определяется формулой:

$$t = \frac{t_{LOOP}}{N_{CPU}} + t_{IO} = \left| \frac{N_{LOOP}}{N_{CPU}} + \frac{1}{2} \right| \cdot t_1 + t_{IO}, \qquad (4)$$

где t_{LOOP} – время выполнения распараллеленного цикла; N_{CPU} – число процессоров; t_{IO} – время выполнения операций ввода-вывода и установки начальных значений (не распараллеливается); N_{LOOP} – число проходов цикла; t_1 – время одного прохода цикла [11].

При распараллеливании цикла 3 время t_{LOOP} максимально, следовательно, время выполнения программы t минимально. Недостатком такой схемы распараллеливания является небольшое число ветвей (R, см. рис. 1). Это может приводить к неполной загрузке процессоров при $N_{CPU} \approx R$ и к простою процессоров, если $\frac{R}{N_{CPU}} \neq i$, где i – целое число.

При распараллеливании цикла 4 число ветвей максимально (равно N; в случае фрактальных структур обычно $N > 10^3$). Недостаток этой схемы в том, что значение t_{LOOP} не максимально, так как шаг 3 выполняется на одном (главном) компьютере. Следовательно, эффективность второй схемы меньше.

Предлагается пользоваться первой схемой, выбирая параметр $R = iN_{CPU}$. Тогда процессоры будут загружены равномерно (как во второй схеме) при сохранении всех преимуществ первой схемы.

При распараллеливании алгоритма выделенных цилиндров такого выбора не возникает, так как в нем нет цикла по R (шага 3). Поэтому шаг 4 разбивается на $1/2N^2$ ветвей, что позволяет равномерно загрузить все процессоры (так как $1/2N^2 >> N_{CPU}$) и достичь максимальной длины ветвей, т. е. максимального значение t_{LOOP} и, следовательно, минимального времени выполнения t.

■ ***Таблица 3.*** *Время выполнения программ на разном числе процессоров*

Методы, реализуемые программой вычисления фрактальной размерности	Число точек N	Число процессоров N_{CPU}	Время выполнения t, с
Метод условной плотности	3200	1	90
		3	30
		5	18
	6400	1	358
		3	120
		5	72
Метод цилиндров	1600	1	81
		3	27
		5	16
	3200	1	187
		3	63
		5	38

Для оценки эффективности параллельных вычислений используется формула

$$E = 100\ \% \frac{t_s}{N_{CPU} t},$$

где t_s – время выполнения программы одним процессором [11].

Оценим эффективность параллельных программ численно. В качестве тестового набора данных используем однородное распределение точек в кубе. Код написан на языке GNU C++ [12] с использованием MPI под Red Hat Linux 6.0 с ядром 2.2.14-5.0. Вычисления проводились на кластерной вычислительной системе кафедры вычислительных систем и сетей – 5 AMD Athlon (tm) XP 1600, L1 Cache 128K, L2 Cache 256K, RAM 256 Mb. В табл. 3 приведено время выполнения программ на разном числе процессоров.

Как видно из табл. 3, эффективность параллельных вычислений $E \approx 99$ %.

Перейдем к статистической оценке временной сложности программ вычисления фрактальной размерности – исследуем зависимость времени выполнения программ от длины входных данных (табл. 4). В табл. 4 методы объемной и оболочечной условной плотности объединены, так как время их выполнения почти одинаково.

■ ***Таблица 4.*** *Зависимость времени выполнения программ от длины входных данных*

Число точек N	Время выполнения t, с	
	Методы условной плотности	Метод цилиндров
100	0,0	0,0
200	0,1	0,1
400	0,5	0,6
800	2,0	3,8
1600	7,7	25,5
3200	30,1	186,7
6400	119,5	1395,6

Как видно из табл. 4, при увеличении числа точек N в два раза, время выполнения программ условной плотности возрастает в 3,9 раз, а программы цилиндров – в 7,5 раз. Это соответствует временно́й сложности $O(N^{1,97})$ и $O(N^{2,91})$ соответственно, что хорошо согласуется с оценкой временно́й сложности, полученной при анализе алгоритмов без их реализации – $O(N^2)$ и $O(N^3)$.

Выводы

Уменьшена временна́я сложность алгоритмов вычисления фрактальной размерности трехмерных точечных структур, реализующих методы объемной и оболочечной условной плотности – $O(N^{4/3})$ вместо $O(N^2)$. Предложены способы распараллеливания программ, реализующих алгоритмы объемной и оболочечной условной плотности и алгоритм выделенных цилиндров с эффективностью $E \approx 99$ %.

Литература

1. **Gabrielli A., Labini F. S., Joyce M., Pietronero L.** Statistical physics for cosmic structures. – USA, N. Y.: Springer-Verlag, 2004. – 422 p.
2. **Mandelbrot B.** Fractals // Encyclopedia of Physical Science and Technology. – 1987. – Vol. 5. – P. 579–593.
3. **Кроновер Р. М.** Фракталы и хаос в динамических системах. – М.: Постмаркет, 2000. – 352 с.
4. **Шредер М.** Фракталы, хаос, степенные законы. – Ижевск: РХД, 2001. – 528 с.
5. **Peitgen H. O., Jurgens H., Saupe D.** Chaos and fractals New frontiers of science. – USA, N. Y.: Springer-Verlag, 1993. – 984 p.
6. **Фракталы в физике** // Тр. VI международного симпозиума по фракталам в физике, Италия, Триест, 9–12 июля 1985 г. / МЦТФ; Под ред. Л. Петронеро и Э. Тозатти. – М.: Мир, 1988. – 670 с.
7. **Martinez V. J., Saar E.** Statistics of the Galaxy Distribution. – USA, CRC Press LLC, 2002. N. W. – 432 p.
8. **Барышев Ю. В., Бухмастова Ю. Л.** Метод условной лучевой концентрации для оценки фрактальной размерности распределения галактик // Письма в астрономический журнал. – 2004. – Т. 30. – № 6. – С.1–7.
9. **Ахо А. В., Хопкрофт Дж., Ульман Дж. Д.** Структуры данных и алгоритмы. – М.: Изд. дом «Вильямс», 2001. – 590 с.
10. **Кнут Д.** Искусство программирования для ЭВМ: В 3 т. – Т. 3: Сортировка и поиск. – М.: Мир, 1978. – 843 с.
11. **Немнюгин С., Стесик О.** Параллельное программирование для многопроцессорных вычислительных систем. – СПб.: БХВ, 2002. – 436 с.
12. **Страуструп Б.** Язык программирования C++. – М.: Бином; СПб.: Невский Диалект, 2000. – 990 с.

УДК 681.3.06

STATE MACHINE – НОВЫЙ ПАТТЕРН ОБЪЕКТНО–ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ

Н. Н. Шамгунов,
аспирант
Г. А. Корнеев,
аспирант
А. А. Шалыто,
доктор техн. наук, профессор
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

В статье предлагается новый паттерн объектно-ориентированного проектирования, названный State Machine. Этот паттерн расширяет возможности паттерна State, предназначенного для реализации объектов, поведение которых зависит от их состояния. Также предложено использовать события для уведомления об изменении состояния. Это позволяет проектировать объекты такого рода из независимых друг от друга классов. Предлагаемый паттерн по сравнению с паттерном State лучше приспособлен для повторного использования входящих в него классов.

This paper presents a new pattern for object-oriented design — State Machine. This pattern extends capabilities of State design pattern. These patterns allow an object to alter its behavior when its internal state changes. Introduced event-driven approach allows to decrease coupling. Thus automaton could be constructed from independent state classes. The classes designed with State Machine pattern are more reusable than State pattern.

Введение

Начало формальному изучению систем с конечным числом состояний было положено исследованием [1].

Различные модели детерминированных конечных автоматов, которые обычно называют «конечными автоматами» или просто «автоматами», были разработаны в середине 50-х годов прошлого века [2–5]. При этом если первые две из этих работ были посвящены синтезу схем (аппаратуры), то остальные носили более абстрактный характер. Считается, что итог этапу становления теории автоматов был подведен выпуском сборника статей [6], который, в частности, содержал работы [3, 5]. Интересно отметить, что перевод этого сборника на русский язык появился в СССР в том же году. Недетерминированные автоматы и их эквивалентность детерминированным автоматам были рассмотрены в работе [7].

В дальнейшем применительно к построению аппаратуры теория автоматов «распалась» на две взаимосвязанные теории: абстрактную и структурную [8], для первой из которых характерна последовательная обработка информации, а для второй – параллельная.

В программировании автоматы начали применяться после появления работы [9], в которой были введены регулярные выражения и приведено доказательство их эквивалентности конечным автоматам. При этом абстрактная теория автоматов используется в основном для обработки текстов [10–12], а структурная – для программного управления [13].

Еще одна область применения конечных автоматов – объектно-ориентированное программирование, где они используются для описания логики объектов, поведение которых зависит от состояния. При этом у объекта выделяются состояния, влияющие на его поведение (так называемые управляющие состояния). Заметим, что в этой области могут применяться конечные автоматы, существенно отличающиеся от абстрактных и структурных автоматов, например, тем, что в них исполь-

зуются понятия объектно-ориентированного программирования. В частности, объекты проектируются в терминах интерфейсов и методов (понятиях, отсутствующих в классических автоматах), а не в терминах входных и выходных воздействий. В данной статье рассматриваются вопросы реализации именно таких объектов.

В объектно-ориентированном программировании под поведением объекта обычно понимается функциональность его методов (действий). Однако во многих приложениях такого определения понятия «поведение объекта» недостаточно, так как необходимо учитывать и его внутреннее состояние.

Наиболее известной реализацией объекта, изменяющего поведение в зависимости от состояния, является паттерн *State* [14]. В указанной работе данный паттерн недостаточно полно специфицирован, поэтому в разных источниках, например в работах [15, 16], он реализуется поразному (в частности, громоздко и малопонятно). По этой причине многие программисты считают, что паттерн *State* не предназначен для реализации автоматов. Другой недостаток паттерна *State* состоит в том, что разделение реализации состояний по различным классам приводит и к распределению логики переходов по ним, что усложняет понимание программы. При этом не обеспечивается независимость классов, реализующих состояния, друг от друга. Таким образом, создание иерархии классов состояний и их повторное использование затруднено. Несмотря на эти недостатки, паттерн *State* достаточно широко применяется в программировании, например, при реализации синхронизации с источником данных в библиотеке *Java Data Objects (JDO)* [17].

Заметим, что проблемы с паттерном *State* возникают не только при его описании, но даже в определении, в том числе, из-за неправильного перевода. Так, в работе [18] приведено следующее определение: «шаблон *State* – состояние объекта контролирует его поведение», в то время как более правильным было бы следующее: «шаблон *State* – состояния объекта управляют его поведением». Эти определения имеют разный смысл, так как с английского языка слово *control* переводится как *управление*; в русском языке управление – это действие на объект, а контроль – мониторинг и проверка выполняемых действий на соответствие заданному поведению. Не случайно часто говорят «система управления и контроля».

Кроме работы [14], паттерн *State*, как отмечалось выше, описывается в работах [15, 16]. В них авторы рассматривают реализацию графов переходов автоматов с помощью этого паттерна. В работе [15] код реализации графа переходов с двумя вершинами занимает более десяти страниц текста. Такую же странную ситуацию можно видеть и в работе [16]. Автор этой книги легко справился с описанием и примерами к сорока шести паттернам, и только для паттерна *State* ему понадобилась помощь рецензента, который предоставил ему пример. Однако приведенный в этом примере граф переходов с четырьмя вершинами реализован таким образом, будто бы автор недостаточно разобрался в паттерне.

В настоящей работе предлагается новый паттерн, объединяющий достоинства реализации автоматов в *SWITCH-технологии* [13] (централизация логики переходов) и паттерна *State* (локализация кода, зависящего от состояния в отдельных классах).

Новый паттерн назван *State Machine*. Обратим внимание, что в работе [19] уже был предложен паттерн с аналогичным названием, предназначенный для программирования параллельных систем реального времени на языке Ada 95. Тем не менее, авторы выбрали именно это название, как наиболее точно отражающее суть предлагаемого паттерна.

В отличие от паттерна *State*, новый паттерн является истинно объектно-ориентированным, поскольку позволяет наследовать как автоматы целиком, так и отдельные их части. Для обеспечения повторного использования классов состояний, входящих в паттерн, предложено применять механизм *событий*, которые используются состояниями для уведомления автомата о необходимости смены состояния. Это позволяет централизовать логику переходов автомата и устранить «осведомленность» классов состояний друг о друге. При этом реализация логики переходов может осуществляться различными способами, например, с использованием таблицы переходов или оператора выбора (оператор `switch` в языке C++).

Более двадцати методов реализации объектов, изменяющих поведение в зависимости от состояния, рассмотрены в работе [20]. Паттерн *State Machine* мог бы продолжить этот список. Наиболее близким к новому паттерну является объединение паттернов *State* и *Observer*, рассмотренное в работе [21]. Однако предложенный в этой работе подход достаточно сложен, так как добавляет новый уровень абстракции — класс `ChangeManager`. В паттерне *State Machine* используется более простая модель событий, не привлекающая относительно тяжелую реализацию паттерна *Observer*. В работе [22] предложена реализация паттерна *State*, позволяющая создавать иерархии классов состояний. Зависимость между классами состояний снижается за счет того, что переход в новое состояние осуществляется по его имени. Такая реализация, тем не менее, не снимает семантической зависимости между классами состояний.

Описание паттерна

В названиях разделов будем придерживаться соглашений, введенных в работе [14].

Назначение. Паттерн *State Machine* предназначен для создания объектов, поведение которых варьируется в зависимости от состояния. При этом у клиента создается впечатление, что изменился класс объекта. Таким образом, назначение предлагаемого паттерна фактически совпадает с таковым для паттерна *State* [14], однако ниже будет показано, что область применимости последнего более узка.

Отметим, что в определении имеются в виду так называемые *управляющие*, а не *вычислительные* состояния [23]. Их различие может быть проиллюстрировано на следующем примере. При создании вычислительной системы для банка имеет смысл выделить режимы нормальной работы и банкротства в разные управляющие состояния, так как в этих режимах поведение банка может существенно отличаться. В то же время, конкретные суммы денег в банковском балансе будут представлять вычислительное состояние.

Мотивация. Предположим, что требуется спроектировать класс `Connection`, представляющий сетевое соединение. Простейшее сетевое соединение имеет два управляющих состояния: *соединено* и *разъединено*. Переход между этими состояниями происходит или при возникновении ошибки или посредством вызовов методов *установить соединение* (`connect`) и *разорвать соединение* (`disconnect`). В состоянии *соединено* может производиться получение (метод `receive`) и отправка (метод `send`) данных по соединению. В случае возникновения ошибки при передаче данных генерируется исключительная ситуация (`IOException`) и сетевое соединение разъединяется. В состоянии *разъединено* прием и отправка данных невозможны. При попытке осуществить передачу данных в этом состоянии объект также генерирует исключительную ситуацию.

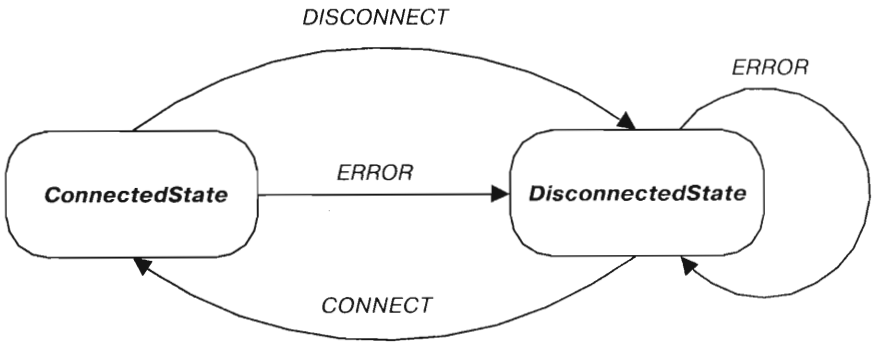
Таким образом, интерфейс, который необходимо реализовать в классе `Connection`, должен выглядеть следующим образом (здесь и далее примеры приводятся на языке Java):

```
package connection;
import java.io.IOException;
public interface IConnection {
  void connect() throws IOException;
  void disconnect() throws IOException;
  int receive() throws IOException;
  void send(int value) throws IOException;
}
```

Основная идея паттерна *State Machine* заключается в разделении классов, реализующих логику переходов (контекст), конкретных состояний и модели данных. Для осуществления взаимодействия конкретных состояний с контекстом используются события, представляющие собой объекты, передаваемые состояниями контексту. Отличие от паттерна *State* состоит в методе определения следующего состояния при осуществлении перехода. Если в паттерне *State* следующее состояние указывается текущим состоянием, то в предлагаемом паттерне это выполняется путем уведомления класса контекста о наступлении события. После этого, в зависимости от события и текущего состояния, контекст устанавливает следующее состояние в соответствии с графом переходов.

Преимуществом такого подхода является то, что классам, реализующим состояния, не требуется «знать» друг о друге, так как выбор состояния, в которое производится переход, осуществляется контекстом в зависимости от текущего состояния и события.

Отметим, что графы переходов, применяемые для описания логики переходов при проектировании с использованием паттерна *State Machine*, отличаются от графов переходов, рассмотренных в работах [11, 13, 24]. Применяемые графы переходов состоят только из состояний и переходов, помеченных событиями. Переход из текущего состояния S в следующее состояние S^* осуществляется по событию E, если на графе переходов существует дуга из S в S^*, помеченная событием E. При этом из одного состояния не могут выходить две дуги, помеченные одним и тем же событием. Отметим, что на графе переходов не отображаются ни методы, входящие в интерфейс реализуемого объекта, ни условия порождения событий.



■ ***Рис. 1.*** *Граф переходов для класса* `Connection`

Граф переходов для описания поведения класса `Connection` приведен на рис. 1. В нем классы, реализующие состояния *соединено* и *разъединено*, называются, соответственно, `ConnectedState` и `DisconnectedState`, тогда как событие `CONNECT` обозначает установление связи, `DISCONNECT` – обрыв связи, а `ERROR` – ошибку передачи данных.

Рассмотрим в качестве примера обработку разрыва соединения при ошибке передачи данных. При реализации с использованием паттерна *State* состояние `ConnectedState` укажет контексту, что следует перейти в состояние `DisconnectedState`. В случае же паттерна *State Machine* контекст будет уведомлен о наступлении события `ERROR` и осуществит переход в состояние `DisconnectedState`. Таким образом, классы `ConnectedState` и `DisconnectedState` не «знают» о существовании друг друга.

Применимость. Паттерн *State Machine* может быть использован в следующих случаях.

1. Поведение объекта существенно зависит от управляющего состояния. При этом реализация поведения объекта в каждом состоянии будет сконцентрирована в одном классе. Этот вариант использования паттерна иллюстрируется в данной работе на примере класса, реализующего сетевое соединение.

2. Рефакторинг кода [25], зависящего от состояния объекта. Примером может служить рефакторинг кода, проверяющего права доступа к тем или иным функциям программного обеспечения в зависимости от текущего пользователя или приобретенной лицензии.

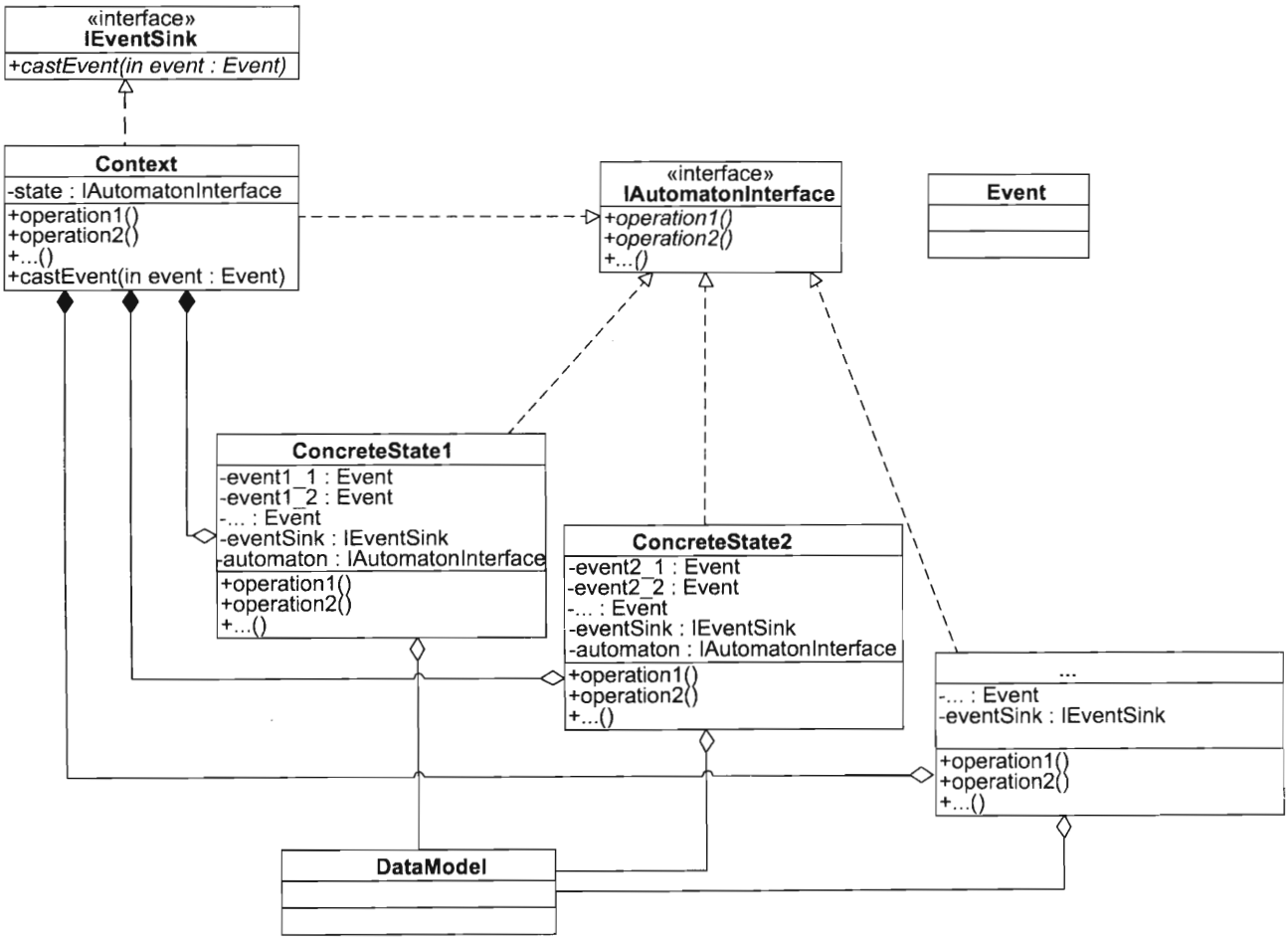
3. Повторное использование классов, входящих в паттерн, в том числе посредством создания иерархии классов состояний. Пример такого случая будет рассмотрен ниже.

4. Эмуляции абстрактных и структурных автоматов.

Таким образом, область применимости паттерна *State Machine* шире, чем у паттерна *State*.

Структура. На рис. 2 изображена структура паттерна *State Machine*.

Здесь `IAutomatonInterface` – интерфейс реализуемого объекта, а `operation1, operation2, ...` – его методы. Этот интерфейс реализуется основным классом `Context` и классами состояний `ConcreteState1, ConcreteState2, ...`. Для смены состояния объекта используются события `event1_1, event2_1, ..., event2_1, event2_2, ...`, являющиеся объектами класса `Event`. Класс `Context` содержит ссылки на все состояния объекта (`ConcreteState1` и `ConcreteState2`), а также на текущее состояние (`state`). В свою оче-



■ ***Рис. 2.*** *Структура паттерна Sate Machine*

редь, классы состояний содержат ссылку на модель данных (dataModel) и интерфейс уведомления о событиях (eventSink). Для того чтобы не загромождать рисунок, на нем не отражены связи классов состояний и класса Event.

Классы Context, ConcreteState1, ConcreteState2, ... реализуют интерфейс IAutomatonInterface. Класс Context содержит переменные типа IAutomatonInterface. Одна из них – текущее состояние автомата, а остальные хранят ссылки на классы состояний автомата. Отметим, что стрелки, соответствующие ссылкам на классы состояний, ведут к интерфейсу, а не к этим классам. Это следствие того, что все взаимодействие между контекстом и классами состояний производится через интерфейс автомата. Связи между контекстом и классами состояний отмечены стрелками с ромбом (используется агрегация).

Участники. Паттерн *State Machine* состоит из следующих частей.

1. *Интерфейс автомата* (IAutomatonInterface) – реализуется контекстом и является единственным способом взаимодействия клиента с автоматом. Этот же интерфейс реализуется классами состояний.

2. *Контекст* (Context) – класс, в котором инкапсулирована логика переходов. Он реализует интерфейс автомата, хранит экземпляры модели данных и текущего состояния.

3. *Классы состояний* (ConcreteState1, ConcreteState2, ...) – определяют поведение в конкретном состоянии. Реализуют интерфейс автомата.

4. *События* (event1_1, event1_2, ...) – инициируются состояниями и передаются контексту, который осуществляет переход в соответствии с текущим состоянием и событием.

5. *Интерфейс уведомления о событиях* (IEventSink) – реализуется контекстом и является единственным способом взаимодействия объектов состояний с контекстом.

6. *Модель данных* (DataModel) – класс предназначен для хранения и обмена данными между состояниями.

Отметим, что в предлагаемом паттерне интерфейс автомата реализуется как контекстом, так и классами состояний. Это позволяет добиться проверки целостности еще на этапе компиляции. В паттерне *State* такая проверка невозможна из-за различия интерфейсов контекста и классов состояний.

Отношения. При инициализации контекст создает экземпляр модели данных и использует его при конструировании экземпляров состояний. Кроме модели данных в конструктор класса состояния также передается интерфейс уведомления о событии (ссылка на контекст).

В процессе работы контекст делегирует вызовы методов интерфейса текущему экземпляру состояния. При исполнении делегированного метода объект, реализующий состояние, может сгенерировать событие – уведомить об этом контекст по интерфейсу уведомления о событиях.

Решение о смене состояния принимает контекст на основе события, пришедшего от конкретного объекта состояния.

Результаты. Сформулируем результаты, получаемые при использовании паттерна *State Machine*.

1. Также как и в паттерне *State*, поведение, зависящее от состояния, локализовано в отдельных классах состояний.

2. В отличие от паттерна *State* в предлагаемом паттерне логика переходов (сконцентрированная в классе контекста) отделена от реализации поведения в конкретных состояниях. В свою очередь, классы состояний обязаны только уведомить контекст о наступлении события (например, о разрыве соединения).

3. Реализация интерфейса автомата в классе контекста может быть сгенерирована автоматически, а реализация логики переходов – по графу переходов.

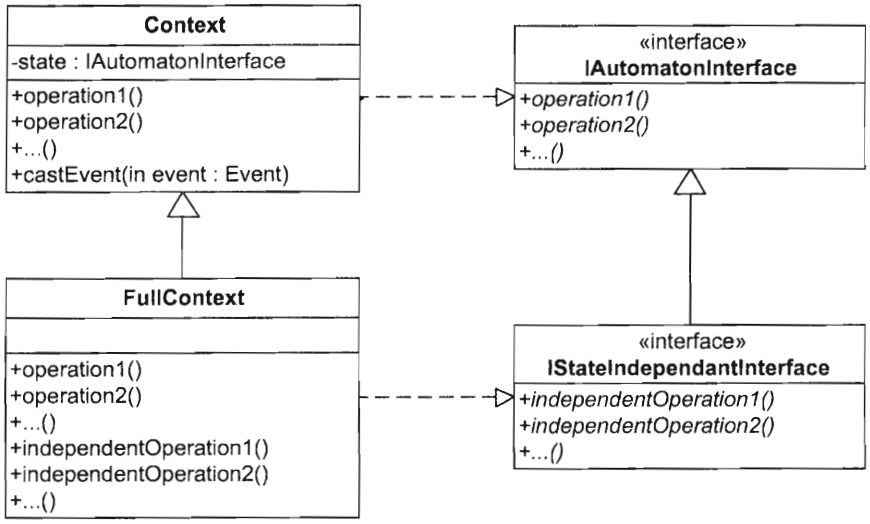
4. Логика переходов может быть реализована табличным способом, что повышает скорость смены состояний.

5. Паттерн *State Machine* предоставляет «чистый» (без лишних методов) интерфейс для пользователя. Для того чтобы клиенты не имели доступа к интерфейсу IEventSink, реализуемому классом контекста, следует использовать закрытое (private) наследование (например, в языке C++) или определить закрытый конструктор и статический метод, создающий экземпляр контекста, который возвращает интерфейс автомата. Соответствующий фрагмент кода имеет вид:

```
class Automaton {
  private Automaton() {}
  public static IAutomataInterface
            CreateAutomaton() {
      return new Automaton();
  }
  ...
}
```

6. В отличие от паттерна *State* паттерн *State Machine* не содержит дублирующих интерфейсов для контекста и классов состояний.

7. Возможно повторное использование классов состояний, в том числе посредством создания их иерархии. В работе [14] сказано, что «поскольку зависящий от состояния код целиком находится в одном из подклассов класса State, то добавлять новые состояния и переходы можно просто путем порождения новых подклассов». На самом же деле, добавление нового состояния зачастую влечет за собой модификацию остальных классов состояний, так как иначе переход в данное состояние не может быть осуществлен. Таким образом, расширение автомата, построенного на основе паттерна *State*, является проблематичным. Более того, при реализации наследования в паттерне *State* также затруднено и расширение интерфейса автомата. Скорее всего, именно по этим причинам в работе [14] не описано наследование автоматов.



■ ***Рис. 3.*** *Реализация методов, не зависящих от состояния*

8. В работе [26] рассматривается задача реализации объектов, часть методов которых не зависит от состояния. Для решения этой задачи предложен паттерн *Three Level FSM*. Данная задача может быть также решена и при помощи паттерна *State Machine*, для чего следует разделить интерфейс реализуемого объекта и интерфейс автомата. При этом последний реализуется контекстом в соответствии с описываемым паттерном. Для реализации полного интерфейса объекта создается наследник контекста, в котором определяются методы, не зависящие от состояния (рис. 3).

Реализация. Рассмотрим возможные модификации паттерна *State Machine*.

1. *Хранение модели данных*. Контекст можно реализовать таким образом, чтобы он включал в себя модель данных, как предлагается в паттерне *State*. Тогда в конструктор объекта состояния передается только один параметр. Однако при таком подходе зависимость реализации состояний от контекста увеличивается, что усложняет повторное использование классов состояний.

2. *Stateless и stateful классы состояний*. В паттерне *State Machine* контекст и классы состояний реализуют интерфейс автомата. Это достигается за счет того, что указанные классы содержат ссылки на модель данных и интерфейс уведомления о событиях. Таким образом, классы состояний являются *stateful* (зависят от предыстории). Такой подход не всегда приемлем, поскольку в некоторых ситуациях критичен расход памяти. Паттерн *State Machine* можно видоизменить так, чтобы состояния были *stateless* (не зависят от предыстории). При этом для классов состояний придется определить новый интерфейс, отличающийся от интерфейса автомата тем, что в каждый метод добавлены параметры, через которые передаются ссылки на модель данных и интерфейс уведомления о событиях. Это приводит к фактическому дублированию интерфейсов, что затрудняет модификацию кода и его повторное использование, но экономит память.

3. *Задание переходов*. Переходы между состояниями задаются в контексте. Это можно сделать, например, при помощи конструкции `switch`, как предлагается в SWITCH-технологии [27]. Переходы также можно задать таблицей, отображающей пару <*текущее состояние, событие*> в <*новое состояние*>. Инфраструктуру для реализации табличного подхода можно реализовать в базовом для всех контекстов классе.

4. *Протоколирование*. Вынесение логики переходов в контекст позволяет осуществлять протоколирование переходов автоматов в терминах состояний и событий.

5. *Создание модели данных*. Экземпляр модели данных может либо создаваться конструктором контекста (как описано выше), либо передаваться в параметрах конструктора контекста.

Пример кода. Коды всех примеров, описанных в статье, доступны по адресу [28].

В следующем примере приведен код на языке Java, реализующий класс `Connected`. Это упрощенная модель произвольного удаленного соединения, через которое можно передавать данные.

Интерфейсы и базовые классы, которые используются в данном примере, вынесены в пакет `ru.ifmo.is.sm` (`sm` – сокращение от *State Machine*). Диаграмма классов этого пакета приведена на рис. 4.

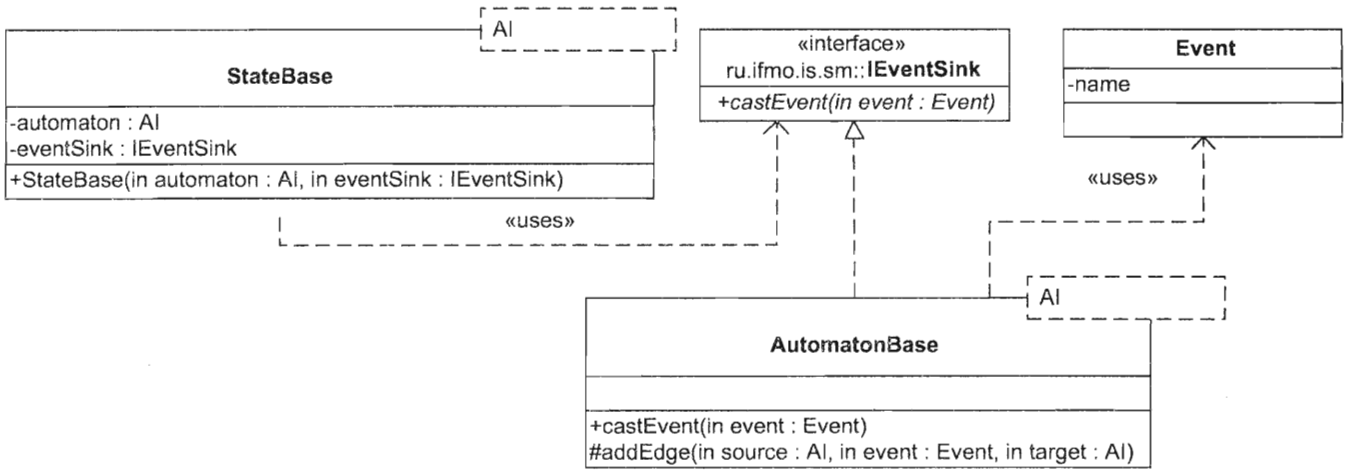
Опишем классы и интерфейсы, входящие в него.

1. `IEventSink` – интерфейс уведомления о событии:

```
package ru.ifmo.is.sm;

public interface IEventSink {
  public void castEvent(Event event);
}
```

2. `Event` – класс события. Используется для уведомления контекста из классов состояний:

```
package ru.ifmo.is.sm;
```



■ ***Рис. 4.*** *Диаграмма классов пакета* `ru.ifmo.is.sm`

```
public final class Event {
  private final String name;

  public Event(String name) {
    if (name == null) {
      throw new NullPointerException();
    }
    this.name = name;
  }
  public String getName() {
    return name;
  }
}
```

3. `StateBase` – базовый класс для состояний. Для проверки типов во время компиляции применяются *параметры типа* (*generic, template*), появившееся в Java 5.0. В конструкторе запоминается интерфейс для приема событий:

```
package ru.ifmo.is.sm;

public abstract class StateBase<AI> {
  protected final AI automaton;
  protected final IEventSink eventSink;

  public StateBase(AI automaton, IEventSink
              eventSink) {
    if (automaton == null ||
          eventSink == null) {
      throw new NullPointerException();
    }
    this.automaton = automaton;
    this.eventSink = eventSink;
  }

  protected void castEvent(Event event) {
    eventSink.castEvent(event);
  }
}
```

4. `AutomatonBase` – базовый класс для всех автоматов. Он предоставляет возможность наследнику регистрировать переходы, используя метод `addEdge`. Дополнительно класс `AutomatonBase` реализует интерфейс уведомления о событии:

```
package ru.ifmo.is.sm;

import java.util.*;

public abstract class AutomatonBase<AI>
            implements IEventSink {
  protected AI state;
  private final Map<AI, Map<Event, AI>>
    edges = new HashMap<AI, Map<Event,AI>>();

  protected void addEdge(AI source, Event
            event, AI target) {
    Map<Event, AI> row = edges.get(source);
    if (null == row) {
      row = new IdentityHashMap<Event, AI>();
      edges.put(source, row);
    }
    row.put(event, target);
  }

  public void castEvent(Event event) {
    try {
      state = edges.get(state).get(event);
    } catch (NullPointerException e) {
      throw new IllegalStateException(
          "Edge is not defined");
    }
  }
}
```

Классы, созданные в соответствии с паттерном *State Machine*, образуют пакет `connection`. Диаграмма классов этого пакета приведена на рис. 5.

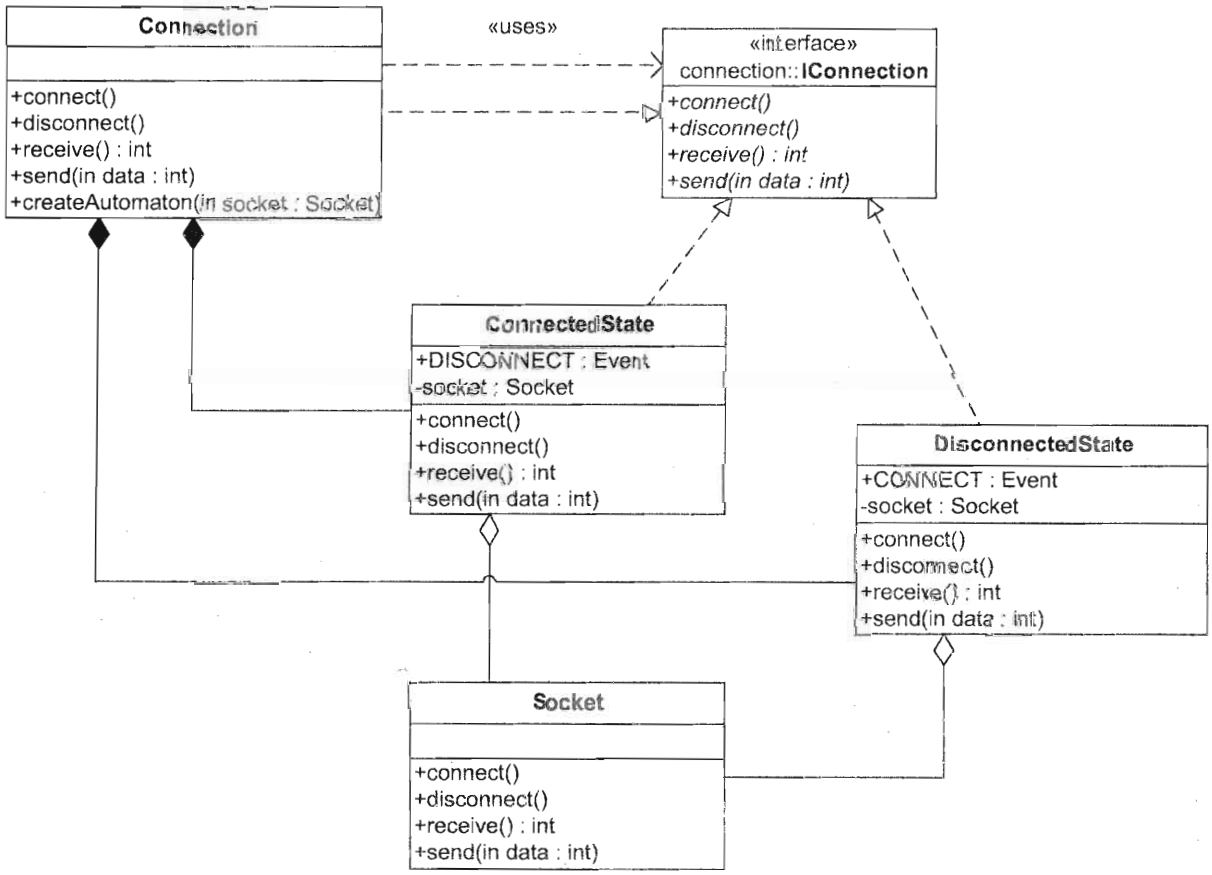
В качестве модели данных автомата используется класс `Socket`, в рассматриваемом случае реализующий интерфейс `IConnection`.

После определения интерфейса для клиента и модели данных необходимо перейти к определению управляющих состояний автомата. Для данного примера реализуем классы `ConnectedState` и `DisconnectedState`. В состоянии `ConnectedState` могут произойти события `ERROR` и `DISCONNECT`, а в состоянии `DisconnectedState` – события `CONNECT` и `ERROR` (см. рис. 1).

Обратим внимание, что на рис. 1 присутствуют дуги, помеченные одинаковыми событиями. В данном примере объекты событий создаются в классе состояния, из которого исходит переход. Например, событие `ERROR` на переходе из состояния `ConnectedState` в состояние `DisconnectedState` не совпадает с аналогичным событием на петле из состояния `DisconnectedState`. При другой реализации для события `ERROR` мог бы быть создан только один объект.

Приведем код классов состояний.

Класс `ConnectedState`:



■ ***Рис. 5.*** *Диаграмма классов пакета* `connection`

```
package connection;

import ru.ifmo.is.sm.*;
import java.io.IOException;

public class ConnectedState <AI extends
            IConnection>
    extends StateBase<AI> implements
            IConnection {
  public static final Event DISCONNECT
      = new Event("DISCONNECT");
  public static final Event ERROR
      = new Event("ERROR");

  protected final Socket socket;

  public ConnectedState(AI automaton,
            IEventSink eventSink, Socket
            socket) {
    super(automaton, eventSink);
    this.socket = socket;
  }

  public void connect() throws IOException {}

  public void disconnect()
      throws IOException {
    try {
      socket.disconnect();
    } finally {
      eventSink.castEvent(DISCONNECT);
    }
  }

  public int receive() throws IOException {
    try {
      return socket.receive();
    } catch (IOException e) {
      eventSink.castEvent(ERROR);
      throw e;
    }
  }

  public void send(int value) throws
            IOException {
    try {
      socket.send(value);
    } catch (IOException e) {
      eventSink.castEvent(ERROR);
      throw e;
    }
  }
}
```

Обратим внимание, что класс состояния только частично специализирует параметр типа класса `StateBase`. При расширении интерфейса автомата наследники класса состояния еще более специализируют этот тип. Окончательная специализация указывается при создании класса состояния в конструкторе автомата. Это обеспечивает возможность последующего расширения интерфейса автомата и классов состояний.

Класс `DisconnectedState`:

```
package connection;

import java.io.IOException;
import ru.ifmo.is.sm.*;

public class DisconnectedState <AI extends
            IConnection>
    extends StateBase<AI> implements
            IConnection {
  public static final Event CONNECT
      = new Event("CONNECT");
```

```
    public static final Event ERROR
        = new Event("ERROR");

    protected final Socket socket;

    public DisconnectedState(AI automaton,
              IEventSink eventSink, Socket
              socket) {
      super(automaton, eventSink);
      this.socket = socket;
    }

    public void connect() throws IOException {
      try {
        socket.connect();
      } catch (IOException e) {
        eventSink.castEvent(ERROR);
        throw e;
      }
      eventSink.castEvent(CONNECT);
    }

    public void disconnect() throws IOException
              {
    }

    public int receive() throws IOException {
      throw new IOException(
          "Connection is closed (receive)");
    }

    public void send(int value) throws
              IOException {
      throw new IOException(
          "Connection is closed (send)");
    }
  }
```

Остается описать класс `Connection` – контекст. В этом классе реализована логика переходов, в соответствии с графом переходов на рис. 1. Заметим, что последние четыре метода этого класса – делегирование интерфейса текущему состоянию:

```
package connection;

import java.io.IOException;
import ru.ifmo.is.sm.AutomatonBase;

public class Connection extends
          AutomatonBase<IConnection>
    implements IConnection {
  private Connection() {
    Socket socket = new Socket();

    // Создание объектов состояний
    IConnection connected = new
          ConnectedState<Connection>(this,
          this, socket);
    IConnection disconnected = new
          DisconnectedState<Connection>(this,
          this, socket);

    // Логика переходов
    addEdge(connected,
          ConnectedState.DISCONNECT,
          disconnected);
    addEdge(connected, ConnectedState.ERROR,
          disconnected);
    addEdge(disconnected,
          DisconnectedState.CONNECT,
          connected);
    addEdge(disconnected,
          DisconnectedState.ERROR,
          disconnected);

    // Начальное состояние
    state = disconnected;
  }

  // Создание экземпляра автомата
  public static IConnection createAutomaton()
          {
    return new Connection();
  }

  // Делегирование методов интерфейса
  public void connect() throws IOException
      { state.connect(); }
  public void disconnect() throws IOException
      { state.disconnect(); }
  public int receive() throws IOException
      { return state.receive(); }
  public void send(int value) throws
          IOException
      { state.send(value); }
}
```

Обратим внимание, что в классах состояний определена только логика генерации событий, а логика переходов сконцентрирована в классе контекста.

Повторное использование классов состояний

Рассмотрим два расширения класса `Connection`. Первое расширение будет демонстрировать возможность добавления методов в интерфейс класса, а второе – изменение поведения за счет введения новых состояний.

Расширение интерфейса автомата. Проиллюстрируем возможность расширения интерфейса автомата на примере добавления возможности возврата данных в объект соединения для их последующего считывания. Введем интерфейс `IPushBackConnection`, расширяющий интерфейс `IConnection` методом `pushBack`. Таким образом, расширенный интерфейс выглядит следующим образом:

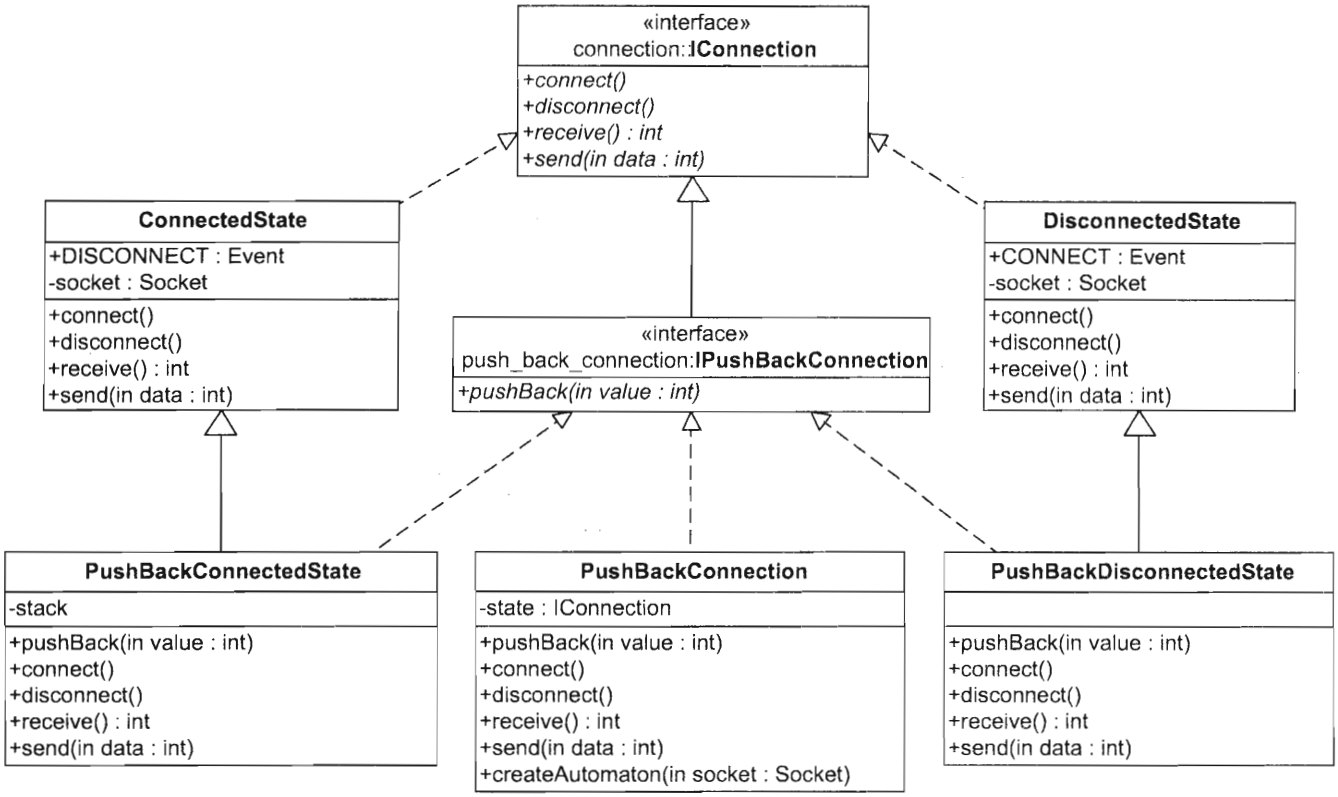
```
package push_back_connection;

import connection.IConnection;
import java.io.IOException;

public interface IPushBackConnection extends
          IConnection {
  void pushBack(int value) throws
          IOException;
}
```

Отметим, что код для этого примера помещен в пакет `push_back_connection`, диаграмма классов которого представлена на рис. 6.

При вызове метода `pushBack(int value)` значение, переданное в параметре этого метода, заносится в стек. При последующем вызове метода receive возвращается элемент из вершины стека, а если стек пуст, то значение извлекается из объекта `socket`.

Заметим, что в рассматриваемом случае количество управляющих состояний автомата не изменится, равно как и граф переходов автомата, но контекст и классы состояний должны реализовывать более широкий интерфейс `IPushBackConnection`. Назовем контекст нового автомата `BushBackConnection`, а новые состояния – `PushBackConnectedState` и `PushBackDisconnectedState`.

Приведем реализацию класса `PushBack ConnectedState` (класс `PushBackDisconnected State` реализуется аналогично). При этом класс



■ ***Рис. 6.*** *Диаграмма классов пакета* `push_back_connection`

`PushBackConnectedState` расширяет класс `ConnectedState`, наследуя его логику:

```
package push_back_connection;

import connection.*;
import ru.ifmo.is.sm.IEventSink;
import java.util.Stack;
import java.io.IOException;

public class PushBackConnectedState
    <AI extends IPushBackConnection>
    extends ConnectedState<AI> implements
            IPushBackConnection {
  Stack<Integer> stack
      = new Stack<Integer>();

  public PushBackConnectedState(AI automaton,
            IEventSink eventSink, Socket
            socket) {
    super(automaton, eventSink, socket);
  }

  public int receive() throws IOException {
    if (stack.empty()) {
      return super.receive();
    }

    return stack.pop().intValue();
  }

  public void pushBack(int value) {
    stack.push(new Integer(value));
  }
}
```

Отметим, что в классе `PushBackConnectedState` параметр типа специализируется еще более, чем в классе `ConnectedState`. Это требуется для того, чтобы поле `automaton`, определенное в классе `StateBase`, имело правильный тип – `IPushBackConnection`. Таким образом, интерфейс автомата «протягивается» сквозь всю иерархию классов состояний через параметр типа.

Созданные классы состояний используются для реализации класса контекста `PushBackConnection`:

```
package push_back_connection;

import connection.Socket;
import ru.ifmo.is.sm.AutomatonBase;
import java.io.IOException;

public class PushBackConnection extends
            AutomatonBase<IPushBackConnection>
    implements IPushBackConnection {
  private PushBackConnection() {
    Socket socket = new Socket();

    // Создание объектов состояний
    IPushBackConnection connected =
        new PushBackConnectedState
        <PushBackConnection>(this, this,
            socket);
    IPushBackConnection disconnected =
         new PushBackDisconnectedState
         <PushBackConnection>(this, this,
            socket);

    // Логика переходов
    addEdge(connected,
            PushBackConnectedState.
            DISCONNECT, disconnected);
    addEdge(connected,
            PushBackConnectedState.ERROR,
            disconnected);
    addEdge(disconnected,
            PushBackDisconnectedState.CONNECT,
            connected);

    // Начальное состояние
    state = disconnected;
  }
```

```
    // Создание экземпляра автомата
    public static IPushBackConnection
              createAutomaton() {
      return new PushBackConnection();
    }

    // Делегирование методов интерфейса
    public void connect() throws IOException
      { state.connect(); }
    public void disconnect() throws IOException
      { state.disconnect(); }
    public int receive() throws IOException
      { return state.receive(); }
    public void send(int value) throws
              IOException
      { state.send(value); }
    public void pushBack(int value) throws
              IOException
      { state.pushBack(value); }
  }
```

Приведенный пример иллюстрирует, что классы состояний могут использоваться повторно в случае расширения интерфейса автомата.

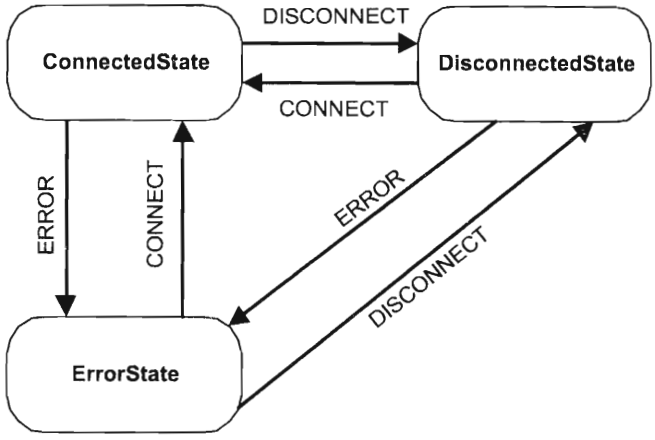
Расширение логики введением новых состояний. Расширение логики поведения будем рассматривать на примере сетевого соединения, которое в случае возникновения ошибки при передаче данных закрывает канал и бросает исключение. При очередной попытке передачи данных производится попытка восстановить соединение и передать данные.

Для описания такого поведения требуется ввести новое состояние – *ошибка*, переход в которое означает, что канал передачи данных закрыт из-за ошибки. Вызов любого метода передачи данных в этом состоянии будет приводить к установке нового соединения.

Таким образом, требуется реализовать класс `ResumableConnection`. Для этого необходимо дополнительно реализовать класс `ErrorState`, определяющий поведение в состоянии *ошибка*. Для состояний *соединено* и *разъединено* используются классы `ConnectedState` и `DisconnectedState`. Граф переходов для класса `ResumableConnection` изображен на рис. 7.

Класс `ErrorState` реализуется следующим образом:

```
package resumable_connection;

import connection.*;
import ru.ifmo.is.sm.*;
import java.io.IOException;

public class ErrorState
    <AI extends IConnection>
    extends StateBase<AI>
    implements IConnection {
  public static final Event CONNECT
      = new Event("CONNECT");
  public static final Event
     DISCONNECT = new Event("DISCONNECT");

  protected final Socket socket;

  public ErrorState(AI automata, IEventSink
            eventSink, Socket socket) {
    super(automata, eventSink);
    this.socket = socket;
  }

  public void connect() throws IOException {
    socket.connect();
```



■ ***Рис. 7.*** *Граф переходов класса* `ResumableConnection`

```
    castEvent(CONNECT);
  }

  public void disconnect() throws IOException
            {
    castEvent(DISCONNECT);
  }

  public int receive() throws IOException {
    connect();
    return automaton.receive();
  }

  public void send(int value) throws
            IOException {
    connect();
    automaton.send(value);
  }
}
```

Класс `ResumableConnection` реализуется так же:

```
package resumable_connection;

import connection.*;
import ru.ifmo.is.sm.AutomatonBase;
import java.io.IOException;

public class ResumableConnection extends
            AutomatonBase<IConnection>
    implements IConnection {
  private ResumableConnection() {
    Socket socket = new Socket();

    // Создание объектов состояний
    IConnection connected = new
            ConnectedState<ResumableConnection>(this,
            this, socket);
    IConnection disconnected = new
            DisconnectedState<ResumableConnection>(this,
            this, socket);
    IConnection error = new
            ErrorState<ResumableConnection>(this,
            this, socket);

    // Логика переходов
    addEdge(connected,
            ConnectedState.DISCONNECT,
            disconnected);
    addEdge(connected, ConnectedState.ERROR,
            error);
    addEdge(disconnected,
            DisconnectedState.CONNECT,
            connected);
    addEdge(disconnected,
            DisconnectedState.ERROR ,
            error);
    addEdge(error, ErrorState.CONNECT,
            connected);
```

```
        addEdge(error, ErrorState.DISCONNECT,
                disconnected);

        // Начальное состояние
        state = disconnected;
    }

    // Создание экземпляра автомата
    public static IConnection createAutomaton()
                {
        return new ResumableConnection();
    }

    // Делегирование методов интерфейса
    public void connect() throws IOException
        { state.connect(); }
    public void disconnect() throws IOException
        { state.disconnect(); }
    public int receive() throws IOException
        { return state.receive(); }
    public void send(int value) throws
                IOException
        { state.send(value); }
}
```

Из приведенного примера видно, что классы состояний могут быть использованы повторно при реализации других автоматов.

Выводы

Паттерн *State Machine* является усовершенствованием паттерна *State*. Он заимствует основную идею паттерна *State* – локализацию поведения, зависящего от состояния, в различных классах.

Новый паттерн устраняет ряд недостатков, присущих паттерну *State*.

1. Паттерн *State Machine* позволяет разрабатывать отдельные классы независимыми друг от друга. Поэтому один и тот же класс состояния можно использовать в нескольких автоматах, каждый со своим набором состояний. Таким образом, устраняется главный недостаток паттерна *State* – сложность повторного использования.

2. В паттерне *State* не описано, каким образом обеспечивается чистота интерфейса, предназначенного для клиента. Предлагаемый паттерн устраняет эту проблему.

3. В паттерне *State* логика переходов распределена по классам состояний, что порождает зависимости между классами и сложность восприятия логики переходов в целом. В паттерне *State Machine* логика переходов реализуется в контексте. Это позволяет разделить логику переходов и поведение в конкретном состоянии.

4. Использование паттерна *State Machine* не приводит к дублированию интерфейсов.

Тем не менее, паттерн *State Machine* не устраняет такой недостаток паттерна *State*, как необходимость производить делегирование интерфейса автомата текущему объекту состояния вручную. Это ограничение можно обойти, используя автоматическую генерацию кода контекста или языки программирования, поддерживающие динамическое делегирование. Автоматическая генерация кода обычно используется в CASE-средствах. Второй подход можно реализовать, например, на языке Self [29], в котором описанное выше делегирование можно выполнить при помощи динамического наследования. Это обеспечивается тем, что язык позволяет непосредственно изменять класс объекта во время выполнения, а объекты в этом языке могут делегировать операции другим объектам.

Литература

1. **McCulloch W., Pitts W.** A logical calculus of ideas immanent in nervous activity // Bull. Math. Biophysics. 1943, 5. – P. 115–133.
2. **Гаврилов М. А.** Теория релейно-контактных схем. – М.: Изд-во АН СССР, 1950. – 230 с.
3. **Huffman D. A.** The synthesis of sequential switching circuits // J. Franclin Inst. 1954. Vol. 257, N 3, 4. – P. 161–190.
4. **Mealy G.** A method for synthesizing sequential circuits // Bell system technical journal. 1955. Vol. 34. N 5. – P. 1045–1079.
5. **Moore E.** Gedanken experiments on sequential machines // Ed. C. E. Shannon, J. McCarthy. Princeton Univ. Press, 1956. – P. 129–153.
6. **Automata Studies** // Ed. C. E. Shannon, J. McCarthy. Princeton Univ. Press, 1956. – P. 400 (Автоматы // Под ред. К. Э. Шеннона, Дж. МакКарти. М.: Изд-во иностр. лит., 1956. – 451 с.).
7. **Rubin M., Scott D.** Finite automata and their decision problem // IBM J. Research and Development. 1959. Vol. 3. N 2. – P. 115–125 (Кибернетический сборник. Вып.4. М.: Изд-во иностр. лит., 1962).
8. **Глушков В. М.** Синтез цифровых автоматов. – М.: Изд-во физ.-мат. лит., 1962. – 476 с.
9. **Kleene S. C.** Representation of events in nerve nets and finite automata // Ed. C. E. Shannon, J. McCarthy. Princeton Univ. Press, 1956. – P. 3–41
10. **Thompson K.** Regular expression search algorithm // Communications of the ACM. 1966. – Vol. 11. N 6. – P. 419–422.
11. **Aho A., Sethi R., Ullman J.** Compilers: principles, techniques and tools. MA: Addison-Wesley, 1985. – 500 p. (Ахо А., Сети-Р., Ульман Дж. Компиляторы: принципы, технологии и инструменты. – М.: Вильямс, 2001. – 768 с.).
12. **Hopcroft J., Motwani R., Ullman J.** Introduction to automata theory, languages and computation. MA: Addison-Wesley, 2001. – 521 p. (Хопкрофт Д., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений. – М.: Вильямс, 2001. – 528 с.).
13. **Шалыто А. А.** SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. – 628 с.
14. **Gamma E., Helm R., Johnson R., Vlissides J.** Design Patterns. MA: Addison-Wesley Professional. 2001. – 395 (Гамма Э., Хелм Р., Джонсон Р., Влассидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2001. – 368 с.).
15. **Steling S., Maassen O.** Applied Java patterns. Pearson higher education. 2001, P. 608 (Стелтинг С., Массен О. Применение шаблонов Java. Библиотека профессионала. М.: Вильямс, 2002. – 564 с.).
16. **Grand M.** Patterns in Java: A catalog of reusable design patterns illustrated with UML. Wiley, 2002. – 544 p. (Гранд М. Шаблоны проектирования в Java. М.: Новое знание, 2004. – 560 с.).
17. Java Data Objects (JDO). (http://java.sun.com/products/jdo/index.jsp).
18. **Eliens A.** Principles of object-oriented software development. MA.: Addison-Wesley, 2000. – 502 p. (Элиенс А. Принципы

объектно-ориентированной разработки программ. М.: Вильямс, 2002. – 496 с.).
19. **Sandйn B.** The state-machine pattern // Proceedings of the conference on TRI-Ada '96 (http://java.sun.com/products/jdo/index.jsp).
20. **Adamczyk P.** The anthology of the finite state machine design patterns. (http://jerry.cs.uiuc.edu/~plop/plop2003/Papers/Adamczyk-State-Machine.pdf)
21. **Odrowski J., Sogaard P.** Pattern Integration – variations of state // Proceedings of PLoP96. (http://www.cs.wustl.edu/~schmidt/PLoP-96/odrowski.ps.gz).
22. **Sane A., Campbell R.**. Object-oriented state machines: subclassing, composition, delegation, and genericity // OOPSLA '95. (http://choices.cs.uiuc.edu/sane/home.html).
23. **Шалыто А. А., Туккель Н. И.** От тьюрингова программирования к автоматному. // Мир ПК. 2002. № 2. – С. 144–149. (http://is.ifmo.ru, раздел «Статьи»).
24. **Harel D.** Statecharts: A visual formalism for complex systems // Sci. Comput. Program. 1987. Vol.8. – P. 231–274
25. **Fawler M.** Refactoring. improving the design of existing code. MA: Addison-Wesley. – 1999. – 431 p. (Фаулер М. Рефакторинг. Улучшение существующего кода. – М.: Символ-плюс, 2003. — 432 с.).
26. **Martin R.** Three Level FSM // PLoPD, 1995. (http://cpptips.hyperformix.com/cpptips/fsm5).
27. **Шалыто А. А., Туккель Н. И.** SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5. С. 45–62. (http://is.ifmo.ru, раздел «Статьи»).
28. Раздел «Статьи» сайта кафедры «Технологии программирования» Санкт-Петербургского государственного университета информационных технологий, механики и оптики (http://is.ifmo.ru/articles).
29. **The Self Language** (http://research.sun.com/self/language.html).

УДК 681.325.5

О СВОЙСТВАХ ДВОИЧНЫХ МАТРИЦ, ИСПОЛЬЗУЕМЫХ В АЛГОРИТМАХ КОДИРОВАНИЯ ИНФОРМАЦИИ БУЛЕВЫМИ ПРЕОБРАЗОВАНИЯМИ

А. Б. Бубликов,
аспирант
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Формулируются требования к свойствам двоичных матриц, используемых для кодирования потоковой информации на основе булевых преобразований.

Requirements to properties of the binary matrixes used for coding of stream information with boolean transformations are formulated.

Введение

Основные положения использования двоичных матриц для защиты информации или ее передачи со скрытым смыслом рассмотрены в работе [1]. Дальнейшим развитием идеи кодирования информации на основе булевых преобразований является модификация, ориентированная на использование двоичных матриц с ненулевым определителем в поле Галуа GF(2).

В данной работе анализируются некоторые особенности двоичных матриц, полученных с помощью метода, описанного в работе [2]. Метод генерации ключевых двоичных квадратных матриц с заданными свойствами дает две матрицы – ключевую (обозначим ее через K) и матрицу K^{-1}, такую, что $KK^{-1} = E$, где E – единичная матрица. Одновременно с указанными матрицами вычисляется порядок некоммутативной группы, порождаемой матрицей K.

Особенности алгоритма кодирования

Предложенный в работе [1] алгоритм кодирования информации на основе булевых преобразований не является симметричным в общем смысле этого слова – повторное кодирование полученной информации с тем же ключом не приведет к ее декодированию. В рассматриваемой реализации кодирование осуществляется посредством матрицы K^{-1}, а декодирование – с использованием матрицы K. Однако следует отметить, что алгоритм симметричен относительно ключевой матрицы. Это значит, что абсолютно не важно, какая из матриц будет участвовать в процессе кодирования информации, поскольку процесс декодирования будет производиться с помощью второй матрицы. Эта особенность алгоритма приводит к тому, что выдвигаемые требования к качеству ключевой матрицы K должны быть справедливыми и для матрицы K^{-1}.

Рассмотрим некоторые особенности алгоритма. В общем случае в алгоритме, производящем операции над блоками информации, представляющими собой двумерный массив элементов кодируемой информации, слово может иметь длину m бит. Основная операция алгоритма – перемножение двух матриц: матрицы K^{-1} (K при декодировании) и очередного блока информации. Все операции производятся в поле Галуа GF(2). Размерность блока информации должна соответствовать размерности n ключевой матрицы. Для упрощения анализа ограничимся длиной слова, равной 8 битам, и, соответственно, размерностью матриц (8×8).

Проведем анализ матриц K и K^{-1}, полученных предложенным в работе [2] методом, и результатов их использования при кодировании.

$$K=\begin{bmatrix}0&0&0&1&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&1&0&0&1&1&0\\0&1&1&0&0&0&0&0\\0&1&0&0&1&0&0&0\\1&0&1&1&0&1&0&0\\0&0&1&1&0&0&0&0\\0&1&1&0&0&1&1&1\end{bmatrix};$$

$$K^{-1}=\begin{bmatrix}0&1&0&0&0&0&0&0\\1&0&0&1&0&0&1&0\\1&0&0&0&0&0&1&0\\1&0&0&0&0&0&0&0\\1&0&0&1&1&0&1&0\\0&1&0&0&0&1&1&0\\1&1&1&0&0&1&0&0\\1&0&1&1&0&0&1&1\end{bmatrix}.$$

Порядок некоммутативной группы, порождаемой матрицей K, равен 41, что является неплохим результатом для выбранной размерности матрицы и длины блока информации.

В обеих матрицах K и K^{-1} имеется по две строки, содержащие всего по одной единице.

Результат кодирования информации с использованием полученных матриц приведен ниже.

Исходная информации	Кодированная информация
Введите пароль:	пароль: №_Ccr~~iПи(
vikivikivikivikivikivikivikiviki	§
Номер матрицы: 1	§
1 1	Я
0 1	1 1 мамсяъм'бк<
Размер матрицы: 2×2	Чматрицы:M¤Ьс"уНЙнЦьЗ_Лкаца:
	1 1

Из приведенного примера видно, что именно наличие в матрице строк с одной единицей не позволяет кодировать все символы исходного текста.

Указанные особенности алгоритма и полученные результаты преобразования позволяют сформулировать некоторые очевидные требования к ключевой матрице и размеру блоков информации.

1. Ключевая матрица K и матрица K^{-1} должны содержать в своем составе равное количество единиц и нулей (достаточное условие) и не должны содержать строк и столбцов, в которых имеется только одна единица (необходимое условие). Преобладание в каждой строке ключевой матрицы единиц или наличие в строке только одной единицы приводят к тому, что при кодировании элементарное слово из состава очередного блока информации останется неизменным, поскольку операция перемножения матриц выполняется по модулю 2.

2. Размер блока информации L в байтах, вычисляемый как $L = m \times n^2 / 8$, зависит от длины (в битах) элементарного блока информации m, значение которого желательно иметь не кратным восьми, и размерности ключевой матрицы n, которую для улучшения качества кодирования рекомендуется выбирать как можно большей. Однако, поскольку алгоритм оперирует только полным блоком информации, это часто требует увеличения размера исходной информации до значения, кратного размеру блока L за счет дополнительной «незначащей информации».

3. Порядок некоммутативной группы, порождаемой ключевой матрицей K, должен быть по возможности большим, поскольку этот параметр отвечает за стойкость алгоритма к атакам типа «перебор». На этапе получения ключевой матрицы методом, предложенным в работе [2], это возможно заданием конкретного значения указанного параметра.

Выводы

Исходя из описанных выше требований и ограничений, налагаемых на ключевую матрицу и сам алгоритм, можно сделать выводы о целесообразности его применения для кодирования разных типов информации.

По типу информация, поступающая на кодирование, делится на потоковую и не потоковую. Применение алгоритма к потоковой информации позволит избежать недостатка, заключающегося в необходимости расширения последнего кодируемого блока до необходимого размера, за счет подбора необходимого сочетания параметров n и m для точного соответствия значению L.

Исходя из особенностей алгоритма, не рекомендуется применять алгоритм к информационному потоку, имеющему регулярную структуру и повторяющиеся фрагменты, при условии, что длина фрагмента меньше размерности ключевой матрицы. Несоответствие этому требованию приведет к тому, что все элементы одного или нескольких блоков информации будут одинаковы и, следовательно, не будут закодированы.

Таким образом, алгоритм наиболее подходит для кодирования двоичной аудио- и видеоинформации, особенно в потоковой реализации, и не применим для кодирования текстовой информации, имеющей много пробелов и регулярных структур.

Литература

1. **Ерош И. Л.** Разграничение доступа к ресурсам в системах коллективного пользования // Информационно-управляющие системы. – 2003. – № 2–3. – С. 63–66.
2. **Бубликов А. Б.** Методы получения двоичных матриц с заданными свойствами для использования в алгоритмах защиты информации на основе булевых преобразований / Седьмая научная сессия аспирантов ГУАП: Сб. докл.: В 2 ч. – Ч. I. Технические науки. – СПбГУАП, 2004. – С. 253–254.

УДК 681.3

3D ИНТЕРАКТИВНАЯ МОДЕЛЬ НАБЛЮДЕНИЙ АСТЕРОИДОВ. ПОЗИЦИОНИРОВАНИЕ ТЕЛЕСКОПА И ПЗС – НАБЛЮДЕНИЯ

К. В. Алексеев,
аспирант
Н. Н. Решетникова,
канд. техн. наук, доцент
Санкт-Петербургский государственный университет аэрокосмического приборостроения

В статье рассматриваются методы наблюдения малых планет из главного пояса Солнечной системы, астероидов АСЗ, объектов искусственного происхождения в околоземном космическом пространстве с помощью наземных ПЗС-телескопов. Эти методы используются в качестве основы для построения интерактивной трехмерной модели процесса наблюдения, визуализация которого позволит повысить точность реальных наблюдений для элементов орбит и параметров указанных объектов.

This article is devoted to the Solar system's minor planets, asteroids, near Earth space artifical satelites, observational methods by using land-based CCD telescopes. These methods are using as a base of interactive 3D model processing, and it's visualisation is to organize real observations to improve orbit of the elements and the parameters of pointed objects

Введение

В начале XIX века итальянский астроном Джузеппе Пиацци (1746–1826) открыл первую малую планету (астероид) диаметром около 933 км, которая была названа Церерой. Это открытие послужило началом эры исследования малых планет (МП). К настоящему времени обнаружено уже более 230 тыс. астероидов Главного пояса Солнечной системы, из которых более 90 тыс. получили постоянные номера, имеют определенные элементы орбит и диаметры от нескольких метров до сотен километров. Под воздействием гравитационных сил орбиты астероидов изменяются, в результате чего возможны столкновения их между собой или падения на поверхности больших планет, в том числе и на Землю. Как следствие, астероиды, сближающиеся с Землей (АСЗ), вызывают повышенный интерес. Современные приемники излучения, которыми оснащены наземные телескопы, позволяют регистрировать астероиды до минимальных размеров около метра в диаметре. Это стало возможным благодаря использованию в астрономии ПЗС-приемников, основой которых являются приборы с зарядовой связью, относящиеся к классу твердотельных полупроводниковых приборов.

В статье рассматриваются методы наблюдения посредством наземных ПЗС-телескопов малых планет из Главного пояса Солнечной системы, АСЗ, объектов искусственного происхождения в околоземном космическом пространстве, которые служат основой построения интерактивной трехмерной модели процесса наблюдения. Визуализация этого процесса позволит организовать реальные наблюдения с целью уточнения элементов орбит и параметров указанных объектов. Модель реализована с использованием технологий представления трехмерных виртуальных пространств в Интернет и демонстрирует работу наземного телескопа в режиме кадровых наблюдений, режиме суточного ведения и наблюдениях при неподвижном телескопе, а также обеспечивает возможность интерактивного взаимодействия с пользователем в реальном времени. По существу, речь идет о разработке и создании интерактивного планировщика астероидных наблюдений (ИПАН).

Актуальность моделирования обусловлена большим интересом к проекту создания всемирной виртуальной обсерватории, который откры-

вает широкие возможности доступа к последним данным наблюдений космических объектов как специалистам в области астрономии и космонавтики, так и любителям астрономии. В рамках проекта модель может быть использована в качестве интерфейса нового поколения.

Рассмотрим методы наблюдений астероидов на примере малой планеты (МП) Aurora-94. Первый этап реализации ИПАН включает подготовку установочных координат телескопа, реализацию модели позиционирования виртуального телескопа, моделирование наблюдений астероидов посредством регистрирующей ПЗС-камеры, просмотр ПЗС-кадров. Следующий этап развития ИПАН состоит из процедуры обработки ПЗС-изображений, отождествления полученных данных, обработки и получения координат астероида в системе опорных звезд, формирования базы первичных данных в формате виртуальной обсерватории (ВО) и является темой дальнейших разработок.

Режимы работы ПЗС-камеры

В основу работы ПЗС-камеры, расположенной на окулярном конце трубы телескопа, положен принцип использования ПЗС-приемника в режиме дрейфового сканирования при синхронном перемещении регистрируемых объектов в поле зрения телескопа и зарядовых пакетов, и в кадровом режиме, когда продолжительное накопление обеспечивается угловым перемещением трубы телескопа со скоростью суточного вращения Земли [2, 6]. Также существуют различные комбинированные режимы наблюдения небесных объектов. Рассмотрим моделирование двух базовых режимов.

1. Кадровый режим (stare or tracking mode). В этом режиме в течение установленного программой времени, т. е. времени экспозиции, выполняется накопление зарядовых пакетов при суточном ведении телескопа. Диапазон возможных экспозиций лежит в пределах от десятков миллисекунд до десятков минут, определяется яркостью регистрируемого объекта, точностью привода телескопа и ограничен величиной фона неба и темнового сигнала ПЗС-матрицы. Указанный режим широко распространен и используется для наблюдения любых объектов в видимой зоне небесной сферы.

2. Режим накопления с синхронным переносом зарядовых пакетов или режим сопровождения (drift scan mode). В этом режиме накопление выполняется одновременно с переносом зарядовых пакетов. Скорость переноса зарядовых пакетов вдоль столбцов матрицы должна равняться скорости перемещения изображения наблюдаемого объекта в плоскости матрицы. Режим используется при неподвижном телескопе и для объектов, перемещающихся в поле зрения телескопа. Время накопления определяется расстоянием, которое должно пройти изображение объекта по ПЗС-матрице. Угловой размер полученного кадра по склонению определяется линейным размером ПЗС-матрицы, по прямому восхождению устанавливается программно, ограничен продолжительностью ночного времени и стабильностью положения телескопа и ПЗС-камеры.

■ ***Рис. 1.*** *Алгоритм моделирования процесса наблюдения*

ПЗС-регистрирующее устройство предназначается для определения яркости в определенной фотометрической системе и координат регистрируемых объектов в прямоугольной системе координат, которая задается направлением столбцов и строк матрицы. В ходе дальнейшей обработки с использованием методов фотографической астрометрии определяются сферические координаты, на основании которых при наличии достаточного объема наблюдений определяются и уточняются элементы орбит.

Последовательность этапов моделирования для уточнения траектории МП

Процесс моделирования состоит из следующих операций (рис. 1).

1. По известным параметрам орбиты МП Aurora-94 с помощью программного пакета Epos Пулковской астрономической обсерватории [12] вычисляются ее экваториальные координаты для заданного периода наблюдений (01.03.05–31.03.05).

2. По траектории движения МП Aurora-94 на фоне опорных звезд небесной сферы определяется зона выборки опорных звезд из наиболее точного доступного каталога положений звезд, например, каталогов Tycho2, UCAC2, USNO-B1 и др. При этом ширина полосы опорных звезд определяется угловыми размерами кадра используемой ПЗС-матрицы, в зависимости от масштаба поля зрения телескопа.

3. Вычисление установочных координат телескопа.

4. Анимация виртуального телескопа (за основу выбран реально действующий телескоп Николаевской астрономической обсерватории – скоростной автоматический телескоп САК [11]).

5. Моделирование процесса установки (позиционирование) виртуального телескопа и ПЗС-наблюдений с ним.

6. Просмотр ПЗС-кадров.

Вычисление координат, определяющих траекторию движения МП Aurora-94

В табл. 1 приведены данные МП Aurora-94, вычисленные с помощью программного пакета Epos для заданного периода наблюдений [12].

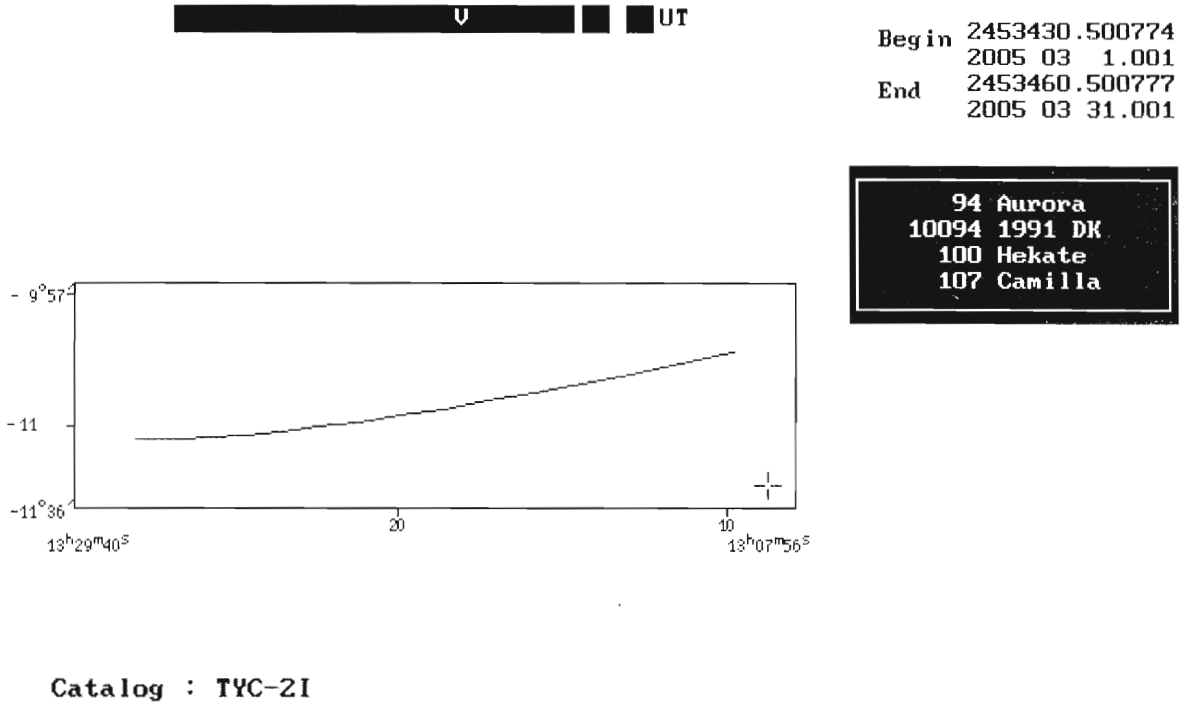
На рис. 2 показан участок траектории движения МП Aurora-94 (сплошная линия). Полоса регистрации ПЗС-камеры задается шириной ±15′ от центральной траектории МП. Значение ширины полосы 30′ выбрано исходя из реальных угловых размеров ПЗС-матрицы САК [10].

Используя встроенные или подключаемые к программной системе звездные каталоги Tycho2, UCAC2, USNO-B1 и другие [5, 7, 10], можно в любой выбранный на траектории МП момент времени T получить площадку с набором опорных звезд и вспомогательных данных – номера и количество звезд, их координаты и моменты времени, спектр, звездные величины, угловой диаметр (для астероида). На основе выявленных максимумов распреде-

■ ***Таблица 1.*** *Данные движения малой планеты Aurora-94 за период 01.03.05 – 31.03.05, полученные с помощью пакета Epos Пулковской обсерватории*

94 Aurora Compressed Ephemeris DE200/LE20 DT
Apparent coordinates Equinox: INST Accuracy ELLIPTIC
Nikolaev Long.= 31.97470 r*cos phi'= .683590 r*sin phi'= .727430

Дата SYYY MM DD.DD	Положение по α	Положение по δ	Расстояние от Земли	Расстояние от Солнца	Фаза вращения	Звездная величина	Величина вращения
2005 03 01.00	13 28 08.9055	-11 08 26.228	2.54067579	3.325	12.0	12.9	136.0W
2005 03 03.00	13 27 23.8204	-11 08 38.078	2.52158776	3.326	11.5	12.9	138.1W
2005 03 05.00	13 26 34.0417	-11 08 23.629	2.50331561	3.328	11.0	12.9	140.3W
2005 03 07.00	13 25 39.6659	-11 07 42.959	2.48589659	3.329	10.4	12.8	142.5W
2005 03 09.00	13 24 40.8159	-11 06 36.223	2.46936812	3.330	9.9	12.8	144.8W
2005 03 11.00	13 23 37.6591	-11 05 03.795	2.45376654	3.331	9.3	12.8	147.0W
2005 03 13.00	13 22 30.4076	-11 03 06.331	2.43912575	3.332	8.8	12.7	149.3W
2005 03 15.00	13 21 19.2938	-11 00 44.645	2.42547647	3.334	8.2	12.7	151.5W
2005 03 17.00	13 20 04.5556	-10 57 59.590	2.41284639	3.335	7.6	12.6	153.8W
2005 03 19.00	13 18 46.4397	-10 54 52.040	2.40126066	3.336	6.9	12.6	156.1W
2005 03 21.00	13 17 25.2062	-10 51 22.927	2.39074221	3.337	6.3	12.6	158.4W
2005 03 23.00	13 16 01.1314	-10 47 33.262	2.38131185	3.338	5.7	12.5	160.7W
2005 03 25.00	13 14 34.5084	-10 43 24.157	2.37298837	3.339	5.0	12.5	163.0W
2005 03 27.00	13 13 05.6469	-10 38 56.839	2.36578876	3.341	4.3	12.4	165.3W
2005 03 29.00	13 11 34.8694	-10 34 12.647	2.35972854	3.342	3.7	12.4	167.6W
2005 03 31.00	13 10 02.5036	-10 29 13.001	2.35482212	3.343	3.0	12.4	169.8W



■ ***Рис. 2.*** *Траектория движения МП Aurora-94 с 01.03.05 по 31.03.05*

ления опорных звезд в равномерно распределенных площадках, принимая минимально допустимое число звезд, необходимое для достоверного получения наиболее высокой точности положений определяемых объектов (астероиды, АСЗ и др.), равным, по крайней мере, трем, можно определить оптимальные моменты регистрации ПЗС-кадров (площадок), которые будут содержать наибольшее количество опорных звезд с точными координатами. Также можно найти оптимальную конфигурацию опорных звезд для наблюдений выбранного объекта или изучить участки неба, где пересекаются видимые пути двух или более объектов, с тем, чтобы выбрать общие опорные звезды.

При выборе времени регистрации ПЗС-кадров, а следовательно, и их положения на небесной сфере (по орбите астероида) следует принимать во внимание наличие в ПЗС-кадре таких вызывающих интерес дополнительных объектов, как внегалактические радиоисточники, другие нумерованные или неизвестные астероиды, объекты из стандартных, калибровочных площадок, явления видимых покрытий и сближений астероидов между собой и со звездами из высокоточных каталогов Hipparcos, Tycho2.

Для программно-управляемого телескопа выборка оптимальных моментов наблюдений может выполняться автоматически, т. е. реализуется адаптивный режим виртуального телескопа.

Расчет установочных координат

В основном цикле управления используется следующий порядок расчета установочных координат.

1. Расчет текущего звездного времени (осуществляется после считывания значений по координатам *A, Z*). Пересчет текущих координат α, δ в *A, Z*.
2. Расчет поправки на рефракцию в *Z* телескопа.
3. Расчет поправок в установочные координаты телескопа.
4. Учет поправок и рефракции в координаты *A* и *Z*, *P* телескопа и пересчет их в координаты α, δ телескопа.
5. Коррекция текущих координат α, δ с учетом скорости движущегося объекта

Здесь *A* – положение объекта по азимуту; *Z* – положение объекта по зенитному расстоянию; *P* – параллактический угол; α – прямое восхождение объекта; δ – склонение объекта.

Получение информации о текущем звездном времени

Для точного наведения телескопа на объект и дальнейшего сопровождения этого объекта необходим предварительный ввод текущего времени с заданной точностью. В представленной методике используется московское время и рассчитывается величина текущего звездного времени, которая в дальнейшем используется для пересчета координат α,δ в *A, Z*.

Методика расчета звездного времени опирается на известные из курса сферической астрономии формулы [3, 4]. Необходимо учитывать время, принятое в избранном часовом поясе, где долгота места наблюдения λ, а московское время M_0. Тогда перевод M_0 в стартовое звездное время S_{L0} производится по формуле

$$S_{L0} = S_{G0} + (M_0 - 3^h)K' + \lambda, \quad (1)$$

где S_{G0} – звездное истинное время в гринвичскую полночь;

M_0 – московское время наблюдения;

λ – долгота телескопа (для САК $\lambda = 2^h07^m54^s$);

K' – коэффициент перевода промежутков среднего солнечного времени в промежутки звездного времени, $K' = 1{,}0027379093$;

3^h – поправка на декретное время (или 4 часа для летнего времени).

Расчет азимутальных координат. Производится пересчет текущих экваториальных координат α и δ, на текущий момент звездного времени S, в азимутальные координаты A, Z [3, 4].

Вычисляется часовой угол объекта t (в градусах) по формуле

$$t = 15(S - \alpha). \quad (2)$$

Используя формулы сферической тригонометрии, можно получить:

$$\cos Z = \cos\varphi \cos\delta \cos t + \sin\varphi \sin\delta; \quad (3)$$

$$\operatorname{tg} A = \frac{\cos\delta \sin t}{\cos\delta \sin\varphi \cos t - \cos\varphi \sin t}. \quad (4)$$

Для вычисления параметра P используются следующие формулы:

$$\sin P = \frac{\sin t \cos\varphi}{\sin Z}; \quad (5)$$

$$\cos P = \frac{\sin\varphi \cos\delta - \sin\delta \cos\varphi \cos t}{\sin Z}; \quad (6)$$

$$P = \operatorname{arctg}\frac{\sin P}{\cos P}. \quad (7)$$

Для решения проблемы выбора из двух возможных положений по A в северной зоне (при $A > 120°$ или $A < -120°$) в алгоритм расчета A входит значение азимута сравнения. Выбирается значение ближайшее, к азимуту сравнения.

Расчет поправки на рефракцию. Явление преломления световых лучей при прохождении ими земной атмосферы называется астрономической рефракцией, или углом рефракции, или рефракцией r. Если видимое зенитное расстояние светила Z', а истинное зенитное расстояние Z, то Z можно определить как $Z = Z' + r$, т. е. истинное зенитное расстояние небесного объекта больше видимого на величину рефракции r.

Рефракция зависит не только от высоты объекта над горизонтом, но и от состояния атмосферы, главным образом от ее плотности, которая сама является функцией, в основном температуры и давления.

Поправки на рефракцию рассчитываются при давлении B и температуре t по формуле [3, 4]:

$$\Delta Z_{\text{реф}} = -60{,}25\frac{273B}{760(273 + t)} \times$$

$$\times \operatorname{tg} Z\left(1 - \frac{273 + t}{273}0{,}00726\sec^2 Z\right), \quad (8)$$

где B – давление (мм. рт. ст); t – температура атмосферы (°С).

Поправки на влияние рефракции ($\delta_{\text{реф}}$), нульпункты разделенных кругов по α и δ (α_0, δ_0) вводятся в вычисленные на моменты наблюдений положения небесных объектов ($\alpha_{\text{вид}}$, $\delta_{\text{вид}}$) и являются установочными параметрами телескопа:

$$\alpha_{\text{уст}} = \alpha_{\text{вид}} + \alpha_0; \quad (9)$$

$$\delta_{\text{уст}} = \delta_{\text{вид}} + \delta_0 + \delta_{\text{реф}}.$$

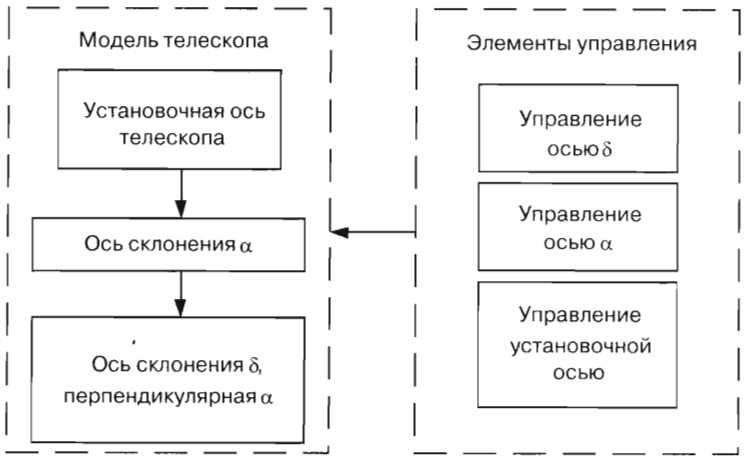
Влияние кривизны суточной параллели не учитывается, ввиду ее незначительности (менее 1″) при наблюдениях МП вблизи плоскости меридиана при отклонении не более 15°.

Описание трехмерной модели телескопа

Моделирование технических процессов, а также создание моделей реальных объектов является актуальной задачей для решения широкого круга технических проблем. В рассматриваемом случае моделируется работа телескопа для изучения особенностей методов наблюдения МП с ПЗС-наземных телескопов. Конструирование модели, реализующей работу конкретного реального привода, требует длительного и трудоемкого его исследования. Поэтому в данной разработке реализована только общая модель, использующая основные принципы, алгоритмы и особенности конструкции САК-М1.0. В качестве модели приведена базовая конструкция монтировки телескопа САК-М1.0. Для моделирования процесса наблюдения выбрано 16 временных значений, в которых были проведены измерения координат МП Aurora-94 с 01.03.05 по 31.03.05 г. Для моделирования интерактивной части, а именно движения телескопа, движения малой планеты, создания спецэффектов, использовался язык моделирования виртуальной реальности VRML. Особенностью языка VRML является то, что сцены, созданные с его помощью, интерактивны и интерпретируются с использованием Internet Explorer с установленным VRML броузером.

Модель телескопа создана при помощи пакета трехмерного моделирования 3D MAX 5.0 с дальнейшим переводом в стандарт X3D/VRML [13]. Для демонстрации возможностей телескопа промоделированы режимы автоматической работы телескопа по наблюдению тел Солнечной системы на примере МП Aurora-94. Для обеспечения интерактивного взаимодействия пользователя с телескопом созданы элементы управления с использованием языка Java, VRML.

Нижеприведенный скрипт, а точнее, набор маршрутов языка VRML в сочетании с примитивами и скриптами языка VRML, позволяет управлять монтировкой телескопа по оси α.



■ ***Рис. 3.*** *Структура взаимодействия элементов телескопа*

```
ROUTE_1.trackPoint_changed TO _ScriptAlfa.a
                \ использование цилиндрического датчика, при работе которого начинает выполняться Java – скрипт (ScriptAlpha.a), отвечающий за вращение по оси склонения α \
ROUTE_ScriptAlfa.dr TO _Sigma.rotation
                \ угол вращения группы объектов Sigma \
ROUTE_PlanS1.translation_changed TO _begunok1. translation
                \ работа элемента управления для передачи угла поворота элементов телескопа по заданной оси \
ROUTE_14.orientation_changed TO _0.set_rotation
ROUTE_14.position_changed TO _0.translation
ROUTE_14.position_changed TO _14.center
ROUTE_PlanS1.translation_changed TO _ScriptAlfa.a
                \ использование элемента управления для передачи угла поворота оси склонения телескопа \
```

Для демонстрации режимов работы по сопровождению МП были интерполированы значения, приведенные в табл. 1, и рассчитан маршрут следования монтировки телескопа при наведении на МП в режимах сопровождения и сканирования.

К возможностям модели следует отнести:

– вращение телескопа по двум осям склонения;

– наблюдение двух режимов функционирования виртуальной модели МП Aurora-94 (режим суточного ведения, режим сопровождения);

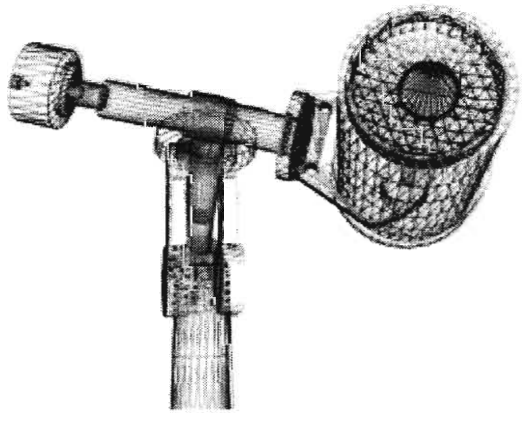
– получение представления о методологии создания ПЗС-кадров при помощи САК-1.0М;

– выполнение в кадровом режиме ПЗС-наблюдений МП Aurora-94 на виртуальной модели (суточное ведение).

Алгоритмы ведения по *A*, *Z* выполнены на основе данных программного пакета Epos (см. табл. 1). На основе этих данных производится интерполяция значений положения осей модели телескопа в соответствии с реально заложенными алгоритмами реального телескопа. Модель состоит из набора узлов, которые взаимодействуют по определенным принципам. Структура взаимодействия элементов приведена на рис. 3.

Трехмерная модель телескопа (рис. 4) представляет собой виртуальный аналог автоматического комплекса САК-М1.0 [9]. Виртуальный телескоп как отдельный сложный объект включен в трехмерную модель фрагмента карты звездного неба с движущейся по расчетной траектории планетой Aurora-94. Пульт управления, представленный в модели, позволяет корректировать положение телескопа в интерактивном режиме. Модель позволяет получить представление о работе комплекса САК-М1.0 и задачах автоматизации процесса наблюдения за движением МП и АСЗ в режимах: покадрового наблюдения; суточного ведения; наблюдения при неподвижном телескопе.

Используется экваториальная (параллактическая) монтировка с приводами на полярную ось и ось склонений. В качестве главной трубы, укрепленной на массивной колонне (установка немецкого типа), применен менисковый телескоп системы Д. Д. Максутова [11]. На окулярном конце трубы установлена ПЗС-камера; труба уравновешена противовесами и отцентрирована.



■ ***Рис. 4.*** *VRML-модель САК-М1.0 Николаевской астрономической обсерватории*

Расчет поправок на деформацию конструкции телескопа

Конструкция телескопа имеет значительный вес и подвержена деформациям и другим явлениям, которые, в свою очередь, вносят погрешность в наведении на наблюдаемый объект. Существует несколько методик определения конструкционных параметров телескопа, влияющих на точность наведения. На практике чаще всего применяют метод коррекции с использованием экспериментальных измерений и установки положений телескопа с учетом полученных поправок [8].

Формулы расчета поправок выглядят следующим образом:

$$\Delta A = K_0 + K_1\frac{1}{\operatorname{tg} Z} + K_2\frac{1}{\sin Z} - K_3\frac{\sin A}{\operatorname{tg} Z} + K_4\frac{\cos\delta\cos P}{\sin Z}; \quad (10)$$

$$\Delta Z = K_5 + K_6\sin Z + K_7\cos Z + K_3\cos A + K_4\cos\varphi\sin A, \quad (11)$$

где K_0 (A_0) – поправка нуль-пункта по азимуту A; K_1 (L) – поправка за наклон горизонтальной оси; K_2 (k) – коллимационная ошибка; K_3 (F) – широтная ошибка вертикальной оси; $K_4(\Delta S)$ – поправка часового угла; K_5 (Z_0) – поправка нуль-пункта по зенитному расстоянию Z; K_6 (d) – коэффициент горизонтального гнутия трубы; K_7 (d_1) – коэффициент вертикального гнутия трубы.

Общие принципы наведения

1. Управляемый разгон выполняется с ограниченным ускорением a, т. е. имеет место программное увеличение скорости задания v на каждом шаге управления на $a/10$.

Скорость возрастает либо до максимально установленного значения (v_{maxA}, v_{maxZ}), либо до меньшего (при переезде на небольшие расстояния).

2. Дальнейшее движение выполняется на постоянной скорости v_{max}.

3. Торможение выполняется как равнозамедленное движение с постоянным ускорением торможения $a_{торм}$.

4. Если по одной координате телескоп достиг заданного значения, а по другой еще нет – включается режим непрерывного отслеживания положения, который необходим либо из-за движения объекта, либо из-за разбаланса усилителя привода координаты.

5. Если рассогласование по координате меньше установленного значения $\Delta_{уст}$, то считается, что телескоп достиг нужного положения. Когда это условие выполняется для обеих координат, то включается счетчик времени $t_{уст}$, который должен принять определенное значение, чтобы режим наведения считался законченным.

Режим наведения

Наведение на объект. В режиме наведения производится дополнительный расчет $A_{упр}$ и $Z_{упр}$ объекта на момент звездного времени:

$$S_{упр} = S + \Delta t_{упр}, \quad (12)$$

где $\Delta t_{упр}$ – время упреждения, равное предполагаемому времени смены режима наведение – ведение. Оно складывается из предполагаемого времени остановки приводов и времени разгона привода ведения до базовой скорости объекта. Базовая скорость объекта получается из этого же расчета. Например, по азимуту:

$$v_{A\ баз} = \frac{A_{упр} - A_{тек}}{\Delta t_{упр}}. \quad (13)$$

Скорость по Z рассчитывается аналогично. В дальнейшем именно $A_{упр}$ и $Z_{упр}$ используются для расчета рассогласований. Относительная скорость торможения и отработки рассогласования, для слежения за объектом, рассчитывается по формуле:

$$v_{отн} = \sqrt{2|\Delta|a_{торм} + a^2} - a_{торм}, \quad (14)$$

где $a_{торм}$ – ускорение торможения; Δ – рассогласование.

Скорость задания на привод устанавливается как

$$v_{прив} = v_{отн} + v_{баз} + \Delta v, \quad (15)$$

где Δv – поправка скорости привода. В большинстве случаев она равна нулю. После определенного времени нахождения в зоне начинается приращение Δv на 0,01"/с.

Наведение на фиксированное положение по *A* и *Z*. При моделировании данного режима необходима установка телескопа в фиксированное положение с достаточной точностью приводами наведения. Для этого определяется минимальная скорость v_{min} = 1'/с, которая больше зоны нечувствительности привода, и зона движения на минимальной скорости Δ_{min}. Также рассчитывается зона Δ_{max} – начало равнозамедленного торможения от скорости v_{max} так, чтобы скорость движения снизилась до v_{min} к моменту подъезда к зоне Δ_{min}:

$$\Delta_{max} = \frac{v_{max}^2 - v_{min}^2}{2a_{торм}} + \tau v_{max} + \Delta_{min}, \quad (16)$$

где τ – время задержки отработки привода.

Скорость в зоне торможения рассчитывается по формуле:

$$v = \sqrt{2(|\Delta| - \Delta_{min})a_{торм} + v_{min}^2} - \tau a_{торм}. \quad (17)$$

В интервале значений от Δ_{min} до $\Delta_{уст}$ устанавливается постоянная скорость, а начиная с $\Delta_{уст}$ скорость равномерно уменьшается до нуля. Если телескоп по одной координате уже подъехал, а по

второй еще едет – включается расчет поправки скорости привода Δv.

Режим сопровождения

Алгоритм ведения. В отличие от алгоритма наведения алгоритм сопровождения имеет несколько этапов, которые последовательно сменяют друг друга. На каждом из этих этапов решаются разные задачи и, соответственно, работают разные алгоритмы управления приводом ведения. На каждом такте управления производится расчет базовых скоростей объекта путем расчета координат *A, Z, P* с упреждением на одну секунду. Разница с текущими координатами и дает значения этих скоростей. Далее рассчитывается рассогласование как разница между текущими (расчетными) и измеренными (реальными) координатами.

Слежение за объектом. В этом режиме обеспечивается точная установка телескопа на объект и максимально качественное его сопровождение. При этом возможны два режима работы: 1) учет рассогласования; 2) определение поправки скорости привода.

Скорость управления приводом рассчитывается по формуле

$$v_{прив} = v_{баз} + v_{расс} + v_{попр}, \qquad (18)$$

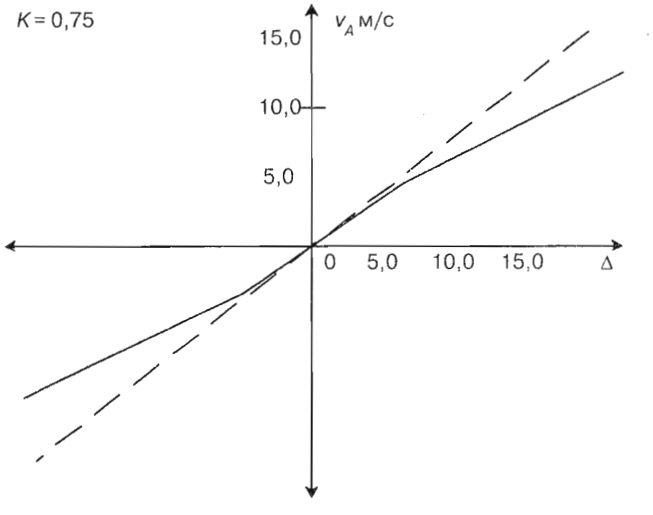
где $v_{баз}$ – базовая скорость объекта; $v_{расс}$ – добавка к скорости для слежения по рассогласованию; $v_{попр}$ – поправка скорости привода.

1. Учет рассогласования. Скорость рассогласования рассчитывается по формуле

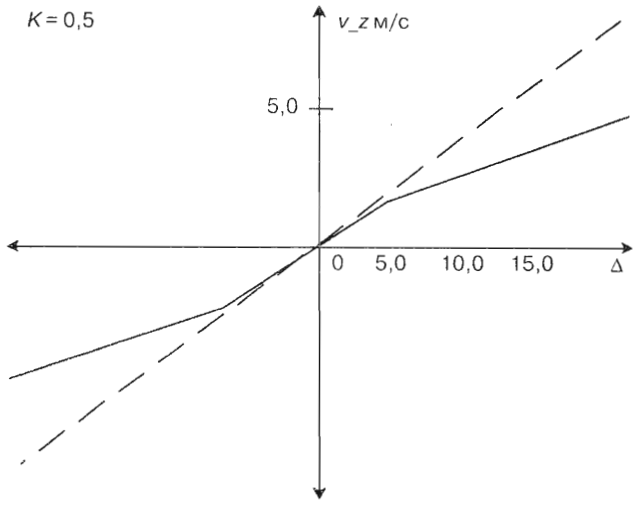
$$v_{расс} = \sqrt{2|\Delta|a + \left(\frac{a}{k}\right)^2} - \frac{a}{k}, \qquad (19)$$

где k – коэффициент отработки.

Для координаты A коэффициент $k_A = 0{,}75$, для Z коэффициент $k_Z = 0{,}5$.



■ ***Рис. 5.*** *Зависимость скорости v от рассогласования для A, K = 0,75*



■ ***Рис. 6.*** *Зависимость скорости v от рассогласования для Z, K = 0,5*

На рис. 5, 6 представлена зависимость скорости v от рассогласования для координат A и Z.

2. Расчет поправки скорости привода. Поправка скорости привода рассчитывается по формуле

$$v_{попр} = v_{попр(n-1)} + \Delta k_{попр}, \qquad (20)$$

где $\Delta k_{попр}$ – поправки по A, Z.

Условия внесения поправки:

рассогласования не слишком большие, а именно установленные в пределах (5% скорости привода для A, Z);

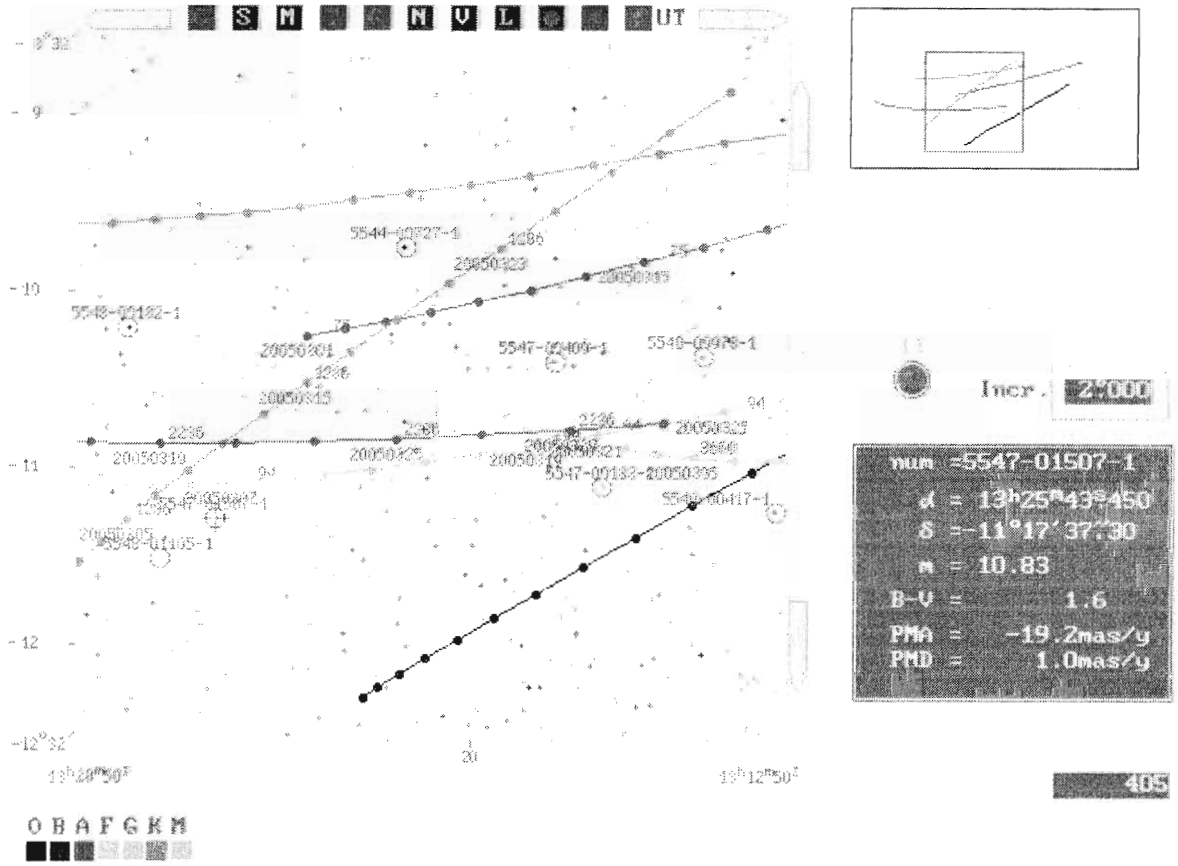
изменения за последние две секунды не превышают 0,35''.

Для проверки выполнения условий, а также для избежания колебаний телескопа на главной резонансной частоте со стороны управления имеет смысл использование цифрового фильтра, отслеживающего величину поправки. В случае, если рассогласование увеличилось до предельных значений, то фаза слежения возвратится в фазу переезда на объект.

Получение ПЗС-кадров

Рассмотрим наиболее широко применяемый кадровый режим получения ПЗС-изображений отдельных площадок неба, включающих искомый объект (объекты), опорные звезды и другие не программные объекты, в том числе и ложные изображения.

Используемая методика наблюдений состоит в определении положения астероида из серии последовательных наблюдений (кадров) в течение одной ночи. Продолжительность экспозиции одного кадра составляет для не слишком слабых объектов в среднем 3–5 мин. Это позволяет получить точечные изображения объекта и опорных звезд на одном кадре. Сравнение соседних в серии кадров (в количестве от двух до десяти) дает



■ ***Рис. 7.*** *Наблюдение движения МП Aurora-94 на фоне опорных звезд в поле 4,0' на 4,0'*

возможность выделить движущиеся объекты на фоне неподвижных звезд, среди которых могут находиться как дополнительные (нумерованные) объекты, так и новые объекты, требующие отождествления. Такая серия позволяет получить среднее положение объекта и оценить точность его координат за ночь, в зависимости от качества ПЗС-изображений, влияние атмосферы и электронной части ПЗС-камеры, привода телескопа и других факторов.

Практика ПЗС-наблюдений в астрономических обсерваториях показывает, что для надежного выявления астероидов Главного пояса достаточно перекрытий наблюдаемой площадки в течение часа. Для выявления астероидов с меньшей угловой скоростью необходимо увеличить время между повторными наблюдениями одной и той же площадки до суток и более.

На рис. 7 представлены кадры, совмещенные для серии наблюдений, где последовательными точками отмечены положения МП Aurora-94 за одну ночь. Опорные звезды выделены кружком и номером. Также представлено движение МП Aurora-94 за весь период наблюдений, указанный в табл.1. В этом случае наблюдения ограничены шестнадцатью средними положениями МП для моментов времени, значения которых приведены в табл. 1. Согласно этим измерениям, было промоделировано движение МП на фоне опорных звезд, а также положения телескопа в моменты измерения.

Заключение

Рассмотренные методы моделирования процесса наблюдения малых планет и АСЗ позволяют использовать реальные данные в виртуальной модели и выявлять неточности измерений. В совокупности с моделью, реальные телескопы могут стать мощным инструментом при наблюдении малых планет и АСЗ, позволят ускорить процесс наведения на объект, а также адаптировать процесс наблюдений под реальные условия. Моделирование процесса наблюдений способствует созданию методической базы, необходимой для изучения процесса взаимодействия астронома с телескопом.

Авторы искренне признательны директору Астрономической Обсерватории г. Николаева (Украина) профессору Геннадию Ивановичу Пинигину за оказанную помощь при подготовке статьи, а также выражают надежду на то, что виртуальный телескоп и дальнейшие трехмерные построения будут практически полезны исследователям космического пространства.

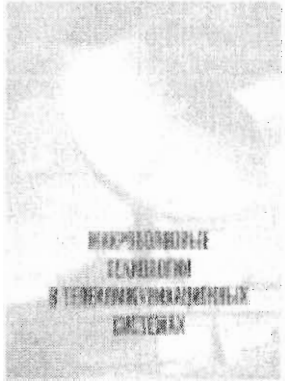
Литература

1. **Yearbook of Astronomy 2004** // Pan Macmillan, 2003 – P. 318
2. **Абросимов В. М., Ковальчук А. Н., Малевинский С. В.** и др. Изучение объектов в ближнем космосе с помощью телескопа АЗТ-8, оснащенного ПЗС-камерой // Космічна наука і технологія. – 2004. – Т. 10. – № 1. – С. 79–84.
3. **Жаров В.Е.** Сферическая астрономия (http://astronet.ru/db/msg/1190894/index.html)
4. **Кононович Э. В., Мороз В. И.** Общий курс астрономии. – М.: Едиториал УРСС, 2001. – 544 с.
5. **Львов В. Н., Смехачева Р. И., Цекмейстер С. Д.** ЭПОС. Пакет программ для решения эфемеридных задач, связанных с обьектами Солнечной системы. Руководство пользователя. Версия 3. 1999–2003. – СПб.: ГАО РАН, Пулково. – С. 32.
6. **Пинигин Г. И.** Телескопы наземной оптической астрометрии. Учебн. пособ. – Николаев: Атолл, 2000. – 104 с.
7. **Цекмейстер С. Д.** Новые возможности программной системы EPOS//Известия ГАО. – № 214. Астрометрия и небесная механика. СПб., 2000. – С. 469–478.
8. **Шергин В., Максимова В.** Алгоритм работы управляющего вычислительного комплекса БТА. Специальная астрофизическая обсерватория Российская академия наук (http://serv.sao.ru/hq/sekbta/Modern/bta_control/node1.html)
9. **Alekseev K., Pinigin G., Reshetnikova N.** 3D Interactive Model of Observational Process of the Minor Planet Iris (7) / / Research of Artificial and Natural NEOs and Other Solar System Bodies with CCD Ground-based Telescopes. Proceedings of the Conference, May 17–20, 2004, Nikolaev. – Atol, – 2004. – P. 13–14.
10. **Manual for Asteroid Catcher B-612** // Japan Spaceguard Association. – Tokyo, Japan, 2000. – P. 1–50. – CD-ROM.
11. **Besarab V., Kovalchuk A., Mazhev A.** et al., Current state of the Fast robotic Telescope // Research of Artificial and Natural NEOs and Other Solar System Bodies with CCD Ground-based Telescopes. Proceedings of the Conference, May 17–20, 2004, Nikolaev. – Atol, 2004. – P. 20–21.
12. **Львов В. Н., Смехачева Р. И., Цекмейстер С. Д.** EPOS software package, 1999–2004 by the Central Astronomical Observatory at Pulkovo. Saint Petersburg, Russia.
13. **Carey R., Bell G., Marrin C.,** ISO/IEC 14772-1:1997, Virtual Reality Modeling Language (VRML97), 1997. The VRML Consortium Incorporated. (http://www.vrml.org/Specifications/VRML97/copyright.html)

В. Ф. Михайлов, Т. Н. Нарытник, И. В. Брагин, В. Н. Мошкин.
Микроволновые технологии в телекоммуникационных системах: Учеб. пособие / СПбГУАП. СПб., 2003. – 337 с.: ил. ISBN 5-8088-0092-7

Приводятся и обсуждаются сведения о распространении радиоволн микроволнового диапазона, а также типы и характеристики линий передач этого диапазона. Рассматриваются методы формирования сигналов телевидения и звукового вещания, микроволновые телерадиоинформационные сети. Обсуждаются системы Интернет-доступа, включая доступ по телефонной сети к кабельному телевидению, спутниковым каналам и телерадиоинформационным сетям. Анализируется оборудование микроволновых телерадиоинформационных распределительных сетей и интегрированные информационные сети на их основе.

Пособие предназначено для студентов радиотехнических специальностей.



УДК 681.327.8

ПЕРСПЕКТИВЫ ПОДГОТОВКИ КАДРОВ ПО НАПРАВЛЕНИЮ «ИНФОРМАЦИОННЫЕ СИСТЕМЫ»

И. Б. Федоров,
доктор техн. наук, профессор
С. В. Коршунов,
канд. техн. наук, профессор
Московский государственный технический университет им. Н. Э. Баумана
Б. Я. Советов,
доктор техн. наук
Санкт-Петербургский государственный электротехнический университет

В статье рассматривается проблема подготовки разработчиков информационных технологий в условиях перехода к информационному обществу. Приводится содержание новых специальностей в области информационных систем.

The article deals with the issue of training information technology experts for condition towards an information society. Contents of new speciality in the sphere of information systems is given.

Российское образование своими истоками восходит к русскому просвещению. Просвещение – это не только высшая, но и самая выгодная политика для великой нации. Задачей просвещения является формирование новой высокоразвитой личности, которая заложит основы цивилизации будущего. Если будущее рассматривать в перспективе наступившего века, то следует учесть внешние объективные условия перехода человечества в информационное общество. Для информационного общества характерна лидирующая роль образования, и будущее образования следует рассматривать в тесной взаимосвязи с особенностями жизни людей в информационном обществе. Представляется, что России не удастся миновать таких прогнозируемых этапов перехода к информационному обществу, как компьютеризация, формирование информационных ресурсов, их социализация. Современный уровень информатизации в России, как и во всем мире, указывает на завершение первого этапа перехода к информационному обществу. Это подтверждается числом используемых компьютеров, непрерывно растущим количеством пользователей Интернетом, развитием рынка коммуникационных и информационных услуг. Интенсивное вхождение в информационное общество реально при опережающем развитии научно-технических направлений, непосредственно обеспечивающих создание и эффективное применение новых информационных технологий, модернизацию конструкторской, технологической и промышленных баз производства информационных средств и их элементов, экстенсивное распространение вычислительной и микропроцессорной техники на различные области человеческой деятельности. В основе этих процессов лежит новый подход к образованию как ведущему фактору развития будущего общества. Информационные технологии стали основой развития мировой системы образования. В настоящее время в образовательной системе эффективно применяются передовые информационные и коммуникационные технологии. Однако остается актуальным создание системы формирования всеобщей компьютерной грамотности как основы воспитания информационной культуры населения. Завершающий этап формирования информационного общества характеризуется обеспечением свободного доступа ко всей информации, накопленной человечеством, возможностью не только пользоваться, но и непосредственно пополнять городской, общегосударственный и мировой информационный фонды. Информационный фонд становится достоянием практически

каждого жителя страны. Таким образом, переход к информационному обществу должен базироваться на приоритетном развитии образования.

В конце XX века национальные и мировая системы образования пережили глубокий кризис. В качестве основных факторов, объективно дестабилизирующих работу системы образования, обычно называют следующие: социальная и экономическая нестабильность, острый дефицит средств, неполнота и противоречивость нормативно-правовой базы. Однако главным фактором остается неудовлетворительное финансирование, что является источником кризисных ситуаций в системе образования. Потребность в финансовых средствах для образовательных учреждений из бюджетных источников обеспечивается менее чем наполовину. По этой причине последние 15 лет российская система образования находилась в состоянии непрерывного реформирования. Можно выделить периоды «новаторства» (под руководством Г. А. Ягодина), «деполитизации и деидеологизации» (под руководством Э. Д. Днепрова), «демократизации и вхождения в рынок» (под руководством В. Г. Кинелева и А. Н. Тихонова), «модернизации» (под руководством В. М. Филиппова). Этап модернизации образования отличается от предшествующего реформирования наличием системных решений, частичного предварительного эксперимента, участием Президента и Государственного совета в этом процессе. Однако явно недостаточная финансовая поддержка образования требует поиска новых путей его развития. Многие считают, что одним из актуальных направлений может стать информатизация. Важно отметить, что, говоря о совокупности средств информатизации, необходимо иметь в виду не только средства вычислительной техники и некоторую «сумму информационных технологий», но также и сумму общественных знаний и умений по использованию указанных средств, которая может быть определена как уровень общественного (или организационного) обучения. Очевидно, что ни одна предметная область не может перешагнуть через некоторые объективные стадии такого общественного обучения. Этот факт впервые был облечен в форму модели стадий роста Р. Ноланом в 1979 г. Модель Нолана показывает, как изменяются расходы на информатизацию, которые определяют уровень организационного обучения, в зависимости от степени проникновения информационной технологии в деятельность организации. Важно обеспечить профессиональную подготовленность пользователей. Таким образом, информатизация должна опираться на обучение пользователей средств информатики. В то же время информатизация открывает возможность становления новой системы непрерывного открытого гибкого дистанционного образования на базе перспективных информационных технологий. Можно считать, что целью информатизации общества является создание гибридного интегрального интеллекта всей цивилизации, способного предвидеть и управлять развитием человечества.

Образовательная система в информационном обществе должна стать системой опережающей. Переход от консервативной образовательной системы к опережающей мог бы базироваться на опережающем формировании информационного пространства российского образования. Такое информационное пространство формируется уже в настоящее время, хотя в ряде случаев и стихийно. Появление информационных технологий способствовало ускорению смены технологий в производстве, проектировании, научных исследованиях, административном управлении. Прогнозируют, что цикл обновления производственных технологий уменьшится до шести-восьми лет, что означает неоднократную смену вида деятельности человека в течение его жизни. Уже в настоящее время перед многими встает задача смены профессии. Ускоренное получение дополнительного образования возможно за счет эффективного внедрения в процесс обучения новых информационных технологий. Поэтому проблема информатизации имеет глубокое социальное значение. В этом смысле проблема модернизации образования на всех его уровнях требует конструктивных решений и ее целесообразно решать в тесной связи с информатизацией общества. Одним из существенных направлений модернизации образования является подготовка разработчиков информационных технологий. Как показал опыт, ее можно реализовать в рамках направления подготовки бакалавров, магистров и дипломированных специалистов – «Информационные системы».

Общие статистические параметры современного Перечня направлений подготовки дипломированных специалистов и специальностей высшего профессионального образования на начало 2002 г. представляли собой следующее: 95 направлений подготовки бакалавров (магистров), 10 групп специальностей, 87 направлений подготовки дипломированных специалистов. Общее количество специальностей высшего профессионального образования – 480. В области техники и технологии было сформировано 81 направление подготовки, включающие 289 специальностей. Причем им соответствует только 41 направление бакалавриата (магистратуры), что явно недостаточно для сохранения единой идеологии подготовки кадров. Последнее потребовало подготовки Государственных образовательных стандартов (ГОС) по ряду направлений бакалавриата для создания завершенной системы Классификатора.

Интенсивное развитие информационной индустрии не может не отражаться на потребностях отраслей экономики России в подготовке квалифицированных специалистов в области информационных систем и технологий. Исторически пер-

вой по инициативе УМО вузов по образованию в области машиностроения и приборостроения при МГТУ им. Н. Э. Баумана была организована подготовка инженеров по межотраслевой специальности 071900 – Информационные системы (по областям применения). Приказом Государственного Комитета РФ по высшему образованию ГОС высшего профессионального образования и государственные требования к минимуму содержания и уровню подготовки выпускников по специальности 071900 – Информационные системы (по областям применения) были введены в действие с 1 декабря 1994 г. Начиная с 1995 года, специальность 071900 была активно востребована вузами, и к началу 1999 года подготовка по данной специальности была открыта более чем в 50 государственных университетах, в том числе в Архангельске, Владимире, Волгограде, Вологде, Воронеже, Вятке, Екатеринбурге, Казани, Костроме, Москве, Нижнем Новгороде, Обнинске, Орле, Пензе, Ростове-на-Дону, Санкт-Петербурге, Ставрополе, Сыктывкаре, Твери, Томске, Тюмени, Ульяновске, Ухте и ряде других городов.

В соответствии с приказом министра образования РФ от 02.03.2000 г. № 686 был утвержден перечень направлений подготовки и специальностей высшего профессионального образования, включающий направление 654700 – Информационные системы (ИС) с единственной специальностью 071900 – Информационные системы и технологии (ИСТ). Одновременно были разработаны и 27 марта 2000 г. утверждены ГОС высшего профессионального образования и примерный учебный план (ПУП) по направлению подготовки дипломированного специалиста 654700 – ИС (регистрационный номер 276 тех/дс).

Существование одной специальности 071900 – ИСТ в рамках направления подготовки дипломированных специалистов 654700 – ИС привело к возникновению в рамках данной специальности более чем 20 специализаций, обусловленных потребностями регионов России. Объектами профессиональной деятельности инженера по этой специальности были определены информационные системы и сети, их математическое, информационное и программное обеспечение, способы и методы проектирования, отладки, производства и эксплуатации программных средств информационных систем в следующих областях: машиностроение, приборостроение, наука и образование, металлургия, энергетика, техническая физика, административное управление, бизнес, менеджмент, ядерная энергетика, геология и нефтегазодобыча, химико-лесной комплекс, телекоммуникации, связь, горное дело, управление технологическими процессами, медицинские технологии, средства массовой информации, химико-лесной комплекс, текстильная и легкая промышленность, строительство, экология, дизайн, а также и в других областях человеческой деятельности.

Переход к информационному обществу требует кадровой поддержки, которая не может быть реализована в рамках существующих направлений подготовки дипломированных специалистов, а потому объективно возникла необходимость формирования самостоятельного единого направления подготовки бакалавров (магистров) и дипломированных специалистов по ряду новых специальностей в области информационных систем. Необходимость открытия ряда новых специальностей направления подготовки дипломированных специалистов 654700 – Информационные системы подтверждается тем, что возникшие к 2002 г. специализации специальности 071900 – Информационные системы и технологии носили отраслевой характер, что не могло служить основой для их дальнейшего эффективного развития в условиях, когда происходит интеграция различных отраслей. Необходима подготовка разработчиков информационных технологий в конкретных областях применения. Например, для создания учебных материалов на различных носителях, ведения учебного процесса необходимы и профессионалы-разработчики сетевых курсов с мультимедийными приложениями, и преподаватели-тьюторы, а также администраторы и системные программисты, разрабатывающие новые инструментальные средства для подготовки учебно-методических материалов. Поэтому с учетом мнения научно-педагогической общественности российской высшей школы было предложено сформировать новые специальности по областям деятельности, в которых наметились явные достижения в области использования информационных технологий. Экспертная оценка показала, что такими областями являются образование, научные исследования, средства массовой информации, менеджмент, дизайн, защита информации и т. д.

В целом актуальность открытия новых образовательных программ по этим областям деятельности в рамках направления подготовки дипломированных специалистов 654700 – Информационные системы обосновывается следующими причинами:

при наличии специальностей, обеспечивающих подготовку пользователей информационных технологий, ощущается явный недостаток в специалистах – разработчиках перспективных информационных технологий и систем;

проявились отдельные области человеческой деятельности, где явно просматривается специфика в разработке и применении информационных технологий и систем;

в ряде предметных областей уже создано значительное число информационных систем, которые требуют совершенствования, обновления и модернизации;

сокращение сроков разработки и внедрения информационных систем в отрасли народного хозяйства возможно лишь при участии специали-

стов, владеющих знаниями в области автоматизации проектирования таких систем.

На заседаниях Учебно-методического совета по направлению 654700 – Информационные системы проблема расширения перечня специальностей по данному направлению многократно обсуждалась с участием представителей УМО вузов по университетскому политехническому образованию при МГТУ им. Н. Э. Баумана и была поддержана научно-педагогической общественностью Москвы, Санкт-Петербурга, Нижнего Новгорода, Воронежа, Астрахани, Петрозаводска, Владимира, Екатеринбурга, Пензы, Ставрополя и ряда других городов. Представители вузов Российской Федерации подтвердили, что в настоящее время созданы необходимые объективные условия поддержки этих специальностей, определяемые потребностями развивающихся отраслей экономики России и наличием высококвалифицированных педагогических кадров, позволяющих организовать обучение бакалавров и инженеров по направлению «Информационные системы» без существенного привлечения дополнительных ресурсов.

Сегодня для руководства практически любой организации или предприятия необходим инженер по информационным системам, профессионально подготовленный для решения задач управления на основе эффективного использования информационных технологий, что можно осуществить в рамках существующего направления «Информационные системы» с расширенной номенклатурой специальностей. Дефицит таких специалистов на отечественном рынке труда не восполняется выпускниками направления 654600 – Информатика и вычислительная техника. Эти причины побудили руководство Учебно-методического совета выйти в Министерство образования РФ с предложением об открытии в рамках направления подготовки дипломированных специалистов 654700 – ИС следующих специальностей (коды специальностей указаны как предварительные и подлежат утверждению):

654701 – Информационные системы и технологии (071900),

654702 – Информационные технологии в образовании (073700),

654703 – Информационные технологии в дизайне,

654704 – Информационные технологии в медиаиндустрии,

654705 – Информационный менеджмент,

654706 – Информационные системы в научных исследованиях,

654707 – Безопасность информационных систем.

Единый подход к подготовке инженеров по этим специальностям должен предусматривать наличие бакалавриата (магистратуры) по направлению «Информационные системы». Поэтому было предложено наряду с подготовкой дипломированных специалистов по указанным специальностям организовать в России подготовку бакалавров по направлению 554400 – Информационные системы, содержание ГОСа которого стало базой для последующего обучения магистров данного направления и дипломированных специалистов по вновь открываемым специальностям. Исходя из этой идеологии, были разработаны ГОС и ПУП подготовки бакалавров по направлению 554400 – Информационные системы, ГОС подготовки магистров (эти документы утверждены Министерством образования РФ), а также ГОС и ПУП подготовки дипломированных специалистов по направлению 654700 – Информационные системы со специальностями 654701 – 654707.

В соответствии с разработанным ГОСом бакалавра информационные системы определяются как область науки и техники, которая включает совокупность средств, способов и методов человеческой деятельности, направленных на создание и применение систем сбора, представления, хранения, передачи и обработки информации. Базовую подготовку по данному направлению обеспечивает блок общепрофессиональных дисциплин общей трудоемкостью 2040 ч (в том числе, федеральный компонент – 1632 ч): электротехника и электроника; метрология, стандартизация и сертификация; безопасность жизнедеятельности; информационные технологии; теория информационных процессов и систем; интеллектуальные информационные системы; информационные сети; информационная безопасность и защита информации; моделирование систем; архитектура ЭВМ и систем; операционные системы; технология программирования; компьютерная геометрия и графика; банки и базы данных. Специальные дисциплины объемом 1156 ч устанавливаются вузом, включая дисциплины по выбору студента.

Разработанный ГОС стал основой линейной схемы подготовки инженера или магистра при последующем (свыше 4 лет) обучении. Специфика направления 554400 – Информационные системы проявилась в содержании основной образовательной программы и в перечне аннотированных магистерских программ, вошедших в структуру ГОСа магистра. В обязательный минимум специализированной подготовки магистра объемом 1134 ч (в том числе, федеральный компонент – 350 ч) включены дисциплины: архитектура современных информационных систем, теоретические основы информационных процессов, методологические основы информационных технологий. Предложен следующий перечень аннотированных магистерских программ: модели информационных процессов, архитектура информационных систем, информационные системы управления образованием, компьютерные обучающие системы, системы представления знаний, коммуникационные технологии, геоинформационные системы, web-технология, информационные системы в научных исследованиях, базы знаний, мультиме-

диа технология, технология управления знаниями, протоколы и интерфейсы информационных систем, информационный дизайн, информационные технологии в средствах массовой инфоромации, менеджмент информационных систем, безопасность информационных систем, анализ и синтез информационных систем, управление данными, администрирование в информационных системах, распределенные информационные системы, электронные библиотеки, корпоративные информационные системы, технология разработки информационных систем, человеко-машинное взаимодействие в информационных системах.

Магистр, освоивший основную образовательную программу высшего профессионального образования в рамках направления «Информационные системы», подготовлен для продолжения образования в аспирантуре по научным специальностям: 051305 – Элементы и устройства вычислительной техники и систем управления; 051306 – Автоматизация и управление технологическими процессами и производствами (по отраслям); 051311 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей; 051312 – Системы автоматизации проектирования (по отраслям); 051313 –Телекоммуникационные системы и компьютерные сети; 051315 – Вычислительные машины и системы; 051317 – Теоретические основы информатики; 051318 – Математическое моделирование, численные методы и комплексы; 051319 – Методы и системы защиты информации, информационная безопасность.

ГОС подготовки инженера по направлению 654700 – Информационные системы в настоящее время утвержден Министерством образования РФ с включением специальностей 071900 – Информационные системы и технологии и 073700 – Информационные технологии в образовании. Профессиональную подготовку по специальностям, входящим в это направление, обеспечивает блок специальных дисциплин объемом 2074 ч трудоемкости, в которые входят 1224 ч, отведенных на специализации.

Специальность 071900 – Информационные системы и технологии в предыдущем Классификаторе находилась в группе междисциплинарных специальностей (по областям применения). По этой специальности тремя учебно-методическими объединениями были разработаны ГОС ВПО в технической области (первым был утвержден Государственным Комитетом РФ по высшему образованию 1 декабря 1994 года ГОС ВПО, разработанный Учебно-методическим объединением вузов по образованию в области машиностроения и приборостроения на базе МГТУ им. Н. Э. Баумана и СПбГТУ), в области экономики и гуманитарной сфере, причем последние стандарты были ориентированы на подготовку, главным образом, пользователя в профессиональной сфере. В новом ГОСе предлагается следующий перечень специальных дисциплин по этой специальности: корпоративные информационные системы; представление знаний в информационных системах; администрирование в информационных системах; геоинформационные системы; мультимедиа технология; надежность информационных систем; проектирование информационных систем.

Специальность 073700-Информационные технологии в образовании открыта Министерством образования РФ в 2002 году и в соответствии с ГОСом подготовки дипломированных специалистов по направлению 654700 характеризуется следующим перечнем специальных дисциплин: информационные системы в управлении учебным процессом; мультимедиа технологии в образовании; дистанционные технологии в образовании; мировые информационные образовательные ресурсы; психолого-педагогические основы проектирования информационных систем в образовании; проектирование информационных систем в образовании.

Специальности с условными номерами 654703 – 654704 предложены Учебно-методическим советом в структуре проекта ГОСа подготовки дипломированных специалистов по направлению 654700 и переданы на рассмотрение Министерству образования РФ. ГОС подготовки по специальностям 654705 – 654707 в настоящее время дорабатывается. Перспектива открытия этих специальностей учтена в перечне общепрофессиональных дисциплин, в соответствии с чем подготовлены предложения по блокам специальных дисциплин этих специальностей.

Специальность 654703 – Информационные технологии в дизайне характеризуется следующим перечнем специальных дисциплин: основы информационных технологий в дизайне; компьютерная обработка изображений; технические средства дизайна; введение в теорию дизайна; технология сетевого дизайна и ее программное обеспечение; инструментальные средства визуальной коммуникации и прикладной дизайн.

Специальность 654704 – Информационные технологии в медиаиндустрии включает следующий перечень специальных дисциплин: техника аудиовизуальных средств информации; языковые средства создания гипертекстовых и гиперграфических документов; интегрированные системы и технологии в медиаиндустрии; проектирование и эксплуатация информационных систем в медиаиндустрии; презентация и анимационная графика; дизайн и оформление средств массовой информации; цифровая обработка информации.

Специальность 654705 – Информационный менеджмент опирается на следующие специальные дисциплины: теоретические основы информационного менеджмента; принятие решений в информационном менеджменте; формирование и менеджмент информационных ресурсов; инфор-

мационный бизнес; экономика информатизации; проектирование систем информационного менеджмента.

Специальность 654706 – Информационные системы в научных исследованиях включает следующие специальные дисциплины: модели и методы планирования экспериментов; идентификация объектов научных исследований; методы и средства обработки экспериментальной информации; информационно-управляющие системы реального времени; интерфейсы информационных систем; проектирование и эксплуатация научно-испытательных комплексов

Специальность 654707 – Безопасность информационных систем характеризуется следующим перечнем специальных дисциплин: теоретические основы безопасности информационных систем; криптографические методы защиты информационных систем; безопасность сетевых технологий; программно-аппаратные средства обеспечения безопасности информационных систем; проектирование средств безопасности информационных систем.

Перечень указанных специальностей с учетом рассмотрения данных предложений в Министерстве образования требует совершенствования в связи с интенсивным процессом создания и внедрения информационных технологий в новые сферы человеческой деятельности.

В настоящее время содержание ГОСов по направлениям 554400 и 654700 отличается единым подходом и преследует цель создания эффективной многоуровневой системы образования с подготовкой бакалавров, магистров и дипломированных специалистов с квалификацией «инженер». Данный подход позволяет при дальнейшем развитии перейти к подготовке бакалавров по специальности и магистров по специальности. Организация предложенной подготовки по направлению «Информационные системы» может стать конструктивной основой модернизации образования на базе современных информационных технологий. Специалисты данного профиля займут нишу разработчиков перспективных информационных технологий. Часть из них, работая в сфере образования, несомненно, будет способствовать повышению качества образования. С их участием перспективными направлениями дальнейшего развития образования станут следующие: включенное обучение на основе телекоммуникационной среды, самостоятельная работа в коммуникационных средах, проведение лекционных, практических, лабораторных, самостоятельных занятий в условиях использования современных информационных технологий по направлениям подготовки, компьютерная поддержка профессиональной ориентации и отбора абитуриентов, компьютерная поддержка последипломного образования, компьютерный контроль знаний и тестирования квалификации, компьютерная поддержка аккредитации специальностей и аттестации вузов. Наличие специалистов позволит осуществить разработку научных основ методологии использования информационных и коммуникационных технологий. Сюда можно отнести системный анализ развития и внедрения информационных технологий, разработку новых принципов организации вычислительного процесса, создание описания предметных областей и построение математических моделей формализуемых и трудно формализуемых функциональных задач, разработку новых информационных процессов, проектирование и внедрение новых информационных технологий в виде интерактивных, аудио-, видео-, компьютерных, телекоммуникационных средств.

Становление и развитие направления подготовки кадров в области информационных систем и технологий должно быть согласовано с другими смежными направлениями, что на стадии проектирования ГОСов, естественно, детально учитывалось. Не менее важен содержательный аспект. Разработанные примерные программы дисциплин, на наш взгляд, учитывают достижения мировой образовательной системы и современного научно-технического прогресса. Можно надеяться, что подготовка разработчиков информационных систем и технологий будет способствовать конкурентоспособности российского образования и ускорит продвижение России к информационному обществу.

Литература

1. **Федоров И. Б., Коршунов С. В., Советов Б. Я.** Новые специальности в направлении подготовки «Информационные системы» // Информационные технологии. – 2002. – № 8.
2. **Советов Б. Я.** Проблемы модернизации непрерывного образования в условиях перехода к информационному обществу // Тенденции в модернизации современного образования: Вестник Северо-Западного отделения Российской академии образования. – Вып. 7. – СПб., 2002.

УДК 681.387.8

МЕТОДИЧЕСКИЕ ОСНОВЫ ИСПОЛЬЗОВАНИЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ В УЧЕБНОМ ПРОЦЕССЕ ПРИ ПОДГОТОВКЕ ПО НАПРАВЛЕНИЮ 654700 – ИНФОРМАЦИОННЫЕ СИСТЕМЫ

С. А. Яковлев,
доктор техн. наук, профессор
Санкт-Петербургский государственный электротехнический университет

Рассматривается машинное моделирование как эффективный инструмент исследования характеристик процесса функционирования информационно-управляющих систем реального времени как на этапе их проектирования, так и эксплуатации. Особое внимание уделяется задаче построения прикладной теории эволюционного и десиженсного моделирования таких систем.

Machine modelling is considered as the effective tool of research of characteristics of process of functioning is information - managing systems of real time as at a stage of their designing, and operation. The special attention is given a problem of construction of the applied theory evolutionary and designation modelling of such systems.

Нельзя говорить конкретно о моделировании систем вообще. Поэтому сразу ограничим круг рассматриваемых объектов моделирования теми объектами, которые лежат в рамках направлений 654600 – Информатика и вычислительная техника и 654700 – Информационные системы. Обобщено их можно назвать информационно-управляющими системами (ИУС) реального времени. Таким образом, речь в статье с точки зрения методологии моделирования пойдет об ИУС, а не о социологических, биологических и других системах (хотя подчеркиваю, возможности имитационного моделирования, несомненно, позволяют расширить круг объектов моделирования).

Машинное моделирование является эффективным инструментом исследования характеристик процесса функционирования ИУС как на этапе их проектирования, так и эксплуатации. Но этим возможности данного метода не ограничиваются: в современных системах управления машинное моделирование используется непосредственно в контуре управления, на его основе решаются задачи прогнозирования для принятия решений по управлению объектом, т. е. реализуются адаптивные системы управления. Построение таких адаптивных систем стало возможным, с одной стороны, после решения ряда вопросов информационного подхода к проблеме управления, а с другой стороны, после проработки задач моделирования в реальном масштабе времени на современных ЭВМ с учетом ограниченности ресурсов в системе управления объектом.

Одной из самых актуальных задач в информатике является создание методологии исследования объектов внешнего мира средствами вычислительной техники, т. е. в конечном итоге, создание информационных моделей, на основе которых можно объяснить теоретическую значимость и практическую ценность любой системы S (в том числе и программной системы) в условиях воздействий внешней среды E.

Создание системы управления различными объектами требует наличия большого объема информации как о самом объекте, так и о его входных и выходных переменных. Эта информация необходима для построения адекватной модели ИУС, на основе которой может быть эффективно осуществлен процесс управления. При этом следует различать два вида информации, необходимой для построения и совершенствования модели и ИУС: априорную и текущую. Априорная информация об объекте управления (ОУ), его

входных и выходных переменных, внутренних состояниях необходима для построения модели, по которой будет создаваться система управления этим объектом: выбираться структура, алгоритмы и параметры ИУС, критерий функционирования. Обычно для сложных вновь проектируемых ОУ отсутствует необходимая для создания ИУС модель, и задача управления должна решаться в условиях недостаточной или вовсе отсутствующей априорной информации об объекте. Речь идет об отсутствии информационной («управленческой») модели ОУ, устанавливающей взаимосвязь между выходными и входными переменными.

Особенности ИУС

Проблема создания ИУС неизбежно возникает при разработке ОУ и при их модернизации. На первый взгляд может показаться, что в тех случаях, когда новая ИУС разрабатывается для уже давно функционирующей системы *S*, длительное время находящейся в эксплуатации, положение с априорной информацией лучше и построение модели проще. Опыт показывает, что это не так и получение информационной модели в этом случае также весьма трудоемко. Таким образом, как в случае вновь проектируемой системы *S*, так и при наличии уже функционирующей возникает проблема получения дополнительной информации для создания ИУС. Единственным эффективным путем получения такой информации в настоящее время является машинное моделирование.

В том случае, когда ИУС создана и функционирует, существует необходимость в получении текущей информации, вызванная в основном двумя причинами. Во-первых, это потребность в совершенствовании ИУС, а во-вторых, необходимость уточнения поведения системы и возникающих в ней ситуаций с целью компенсации изменений характеристик системы *S* как ОУ. Процессы, с которыми связана текущая информация первого вида, являются достаточно медленными и для управления ими необходима подсистема эволюционного управления. Процессы второго типа являются более быстрыми и для управления ими необходима подсистема оперативного управления в реальном масштабе времени (РМВ).

Следует подчеркнуть, что по темпу принятия решений и месту решения задач подсистемы эволюционного и оперативного управления существенно отличаются друг от друга. Так, процессы оперативного управления могут протекать на несколько порядков быстрее, чем процессы эволюционного управления.

Важнейшей задачей современной теории и практики управления является построение модели ОУ, т. е. формализация закономерностей функционирования объекта. На основе этой модели определяются структура, алгоритмы и параметры ИУС, выбираются аппаратно-программные средства реализации системы. Одним из эффективных методов построения модели сложного объекта является идентификация.

Появление в настоящее время большого количества работ по формализации процессов и построению их моделей во многих областях исследований (технике, экономике, социологии и т. д.) преследует две основные цели. Первая из них связана со значительным расширением возможностей изучения на базе ЭВМ сложных процессов функционирования различных объектов при помощи метода моделирования, для чего необходимо математическое описание исследуемого процесса. Не меньшее значение в технических системах имеют модели, используемые для достижения второй цели, т. е. применяемые непосредственно в контуре управления объектами.

Эволюционные и десиженсные модели

Невозможность ограничиться только одной универсальной моделью связана с тем, что, с одной стороны, перед этими моделями ставятся различные цели, а с другой стороны, они описывают процессы, протекающие в различных масштабах времени, причем степень полноты модели и ее соответствия реальному объекту зависят от целей, для которых эта модель используется. Модели первого типа имеют в основном гносеологический характер, от них требуется тесная связь с методами той конкретной области знаний, для которой они строятся. Модели такого типа являются достаточно «инерционными» в своем развитии, так как отражают эволюцию в конкретной области знаний. Такие модели будем называть эволюционными. Модели второго типа имеют информационный характер и должны соответствовать конкретным целям по принятию решений в управлении объектом, который они описывают. Такие модели будем называть десиженсными. Деление на гносеологические (эволюционные) и информационные (десиженсные) модели достаточно условно, но оно удобно для отражения целей моделирования.

В информационных моделях, используемых непосредственно для принятия решений в ИУС, требование оперативности является одним из основных. Оно вызвано тем, что при каждом воздействии на ОУ необходимо учесть в модели действительные изменения, происшедшие в объекте, и внешние возмущения, на основе которых рассчитывается управление. Это требование оперативности, т. е. необходимость работы такой модели в РМВ, часто приводит к отказу от сложных и точных моделей и к разработке специальных, так называемых робастных, алгоритмов построения моделей, использование которых в ИУС обычно ведет к поставленной цели.

Появление идентификации в начале 1960-х годов было связано с острой необходимостью разработки методов построения именно информаци-

онных моделей ОУ. Отсутствие таких моделей сдерживало процесс автоматизации этих объектов, использования ЭВМ в контуре управления. Объекты оказались неподготовленными к внедрению вычислительной техники из-за отсутствия их математического описания, их информационных моделей. Построение информационной модели методами идентификации должно быть направлено на ликвидацию этого разрыва и разработку методов оперативного получения модели ОУ. При этом методы идентификации должны предусматривать использование ЭВМ для решения задач построения информационной модели.

Элементы теории моделирования

Отсутствие формальных методов перехода от гносеологических моделей к информационным в современной теории управления не дает возможности получить по имеющейся информации адекватное описание, необходимое для создания ИУС. Но учет сведений, содержащихся в гносеологических моделях, может значительно увеличить объем априорной информации о рассматриваемом ОУ. Поставив цель построения гносеологической модели процесса функционирования системы *S* для получения необходимой априорной информации с целью создания эффективной ИУС и сузив класс объектов моделирования до конкретного, т. е. до поведения конкретной системы *S*, решим задачу построения *прикладной теории эволюционного и десиженсного моделирования*, позволяющей эффективно (в реализационном аспекте) перейти от гносеологических («исследовательских») моделей к информационным («управленческим») моделям. Наиболее просто такой переход можно совершить, если оба эти класса моделей будут базироваться на единой концептуальной модели, использовать единую систему информации (базу знаний) и иметь единую критериальную систему. Рассмотрим сначала особенности гносеологических и информационных моделей.

Вопрос применимости некоторой математической модели к изучению рассматриваемого объекта не является чисто математическим вопросом и не может быть решен математическими методами. Только критерий практики позволяет сравнивать различные гипотетические модели и выбрать из них такую, которая является наиболее простой и в то же время правильно передает свойства изучаемого объекта, т. е. системы *S*.

Ориентируясь на общие вопросы методологии моделирования сложных технических систем, сформулируем требования к прикладной теории моделирования, а точнее – к элементам этой теории в ее приложении для решения конкретно поставленной задачи. Как уже отмечалось выше, эта задача ставится следующим образом. Необходимо сначала построить и реализовать на ЭВМ эволюционную модель процесса функционирования системы *S*, полученную в ходе стратегической идентификации ОУ, а затем на ее базе построить десиженсную модель, используемую для решения практических задач оперативного управления в адаптивной ИУС сетью. Или, используя терминологию теории идентификации, необходимо построить конкретную дискретную адаптивную систему управления с идентификатором и предсказателем (комбинированную) в цепи обратной связи (ДАСК), т. е. реализовать сначала стратегический идентификатор, а затем на его базе тактический оперативный идентификатор и предсказатель, рассматривая в качестве ОУ не реальную систему *S* (ввиду ее отсутствия), а машинную модель процесса ее функционирования.

Таким образом, поставленную задачу можно трактовать и как задачу автоматизации исследования объекта (машинной модели *Мм*) в целях синтеза тактической и оперативной модели, используемой непосредственно в контуре управления системой *S*, а затем для проверки эффективности управления в целом.

Прежде чем переходить к изложению элементов теории моделирования процессов в системе *S*, дадим ряд определений. Напомним, что под моделированием будем понимать исследование объекта посредством изучения его модели, т. е. другого объекта, более удобного для этой цели. Под сложностью моделируемого объекта фактически будем понимать сложность сведений о нем (его описания), зависящую от целей моделирования и уровня, на котором выполняется описание. Таким образом, сложность возрастает не только при введении в рассмотрение новых качеств, но и при переходе к более детальному описанию процесса функционирования объекта моделирования, т. е. системы *S*.

Задачу прикладной теории моделирования сформулируем, исходя из тех требований, которые будет предъявлять к ней пользователь (исследователь, разработчик системы *S*), проводящий эксперименты с процессами функционирования системы *S* и ее элементов для решения конкретной прикладной задачи. В таком контексте основной задачей при решении проблем управления является выбор моделей на уровне оперативного управления, сохраняющих при этом существенные для ИУС черты *S* с учетом ограничений реализации в РМВ (особенно при оперативном управлении). В дальнейшем модель, практически реализуемую с учетом ограниченности ресурсов, будем называть *трактабельной*. Таким образом, помимо теоретических вопросов построения модели вообще будем рассматривать вопросы трактабельности модели, связанные с формальным представлением ее описания, его упрощением, проверкой адекватности упрощенной модели и т. д.

Тот факт, что моделируемая система *S* существует лишь как замысел разработчика, вносит в про-

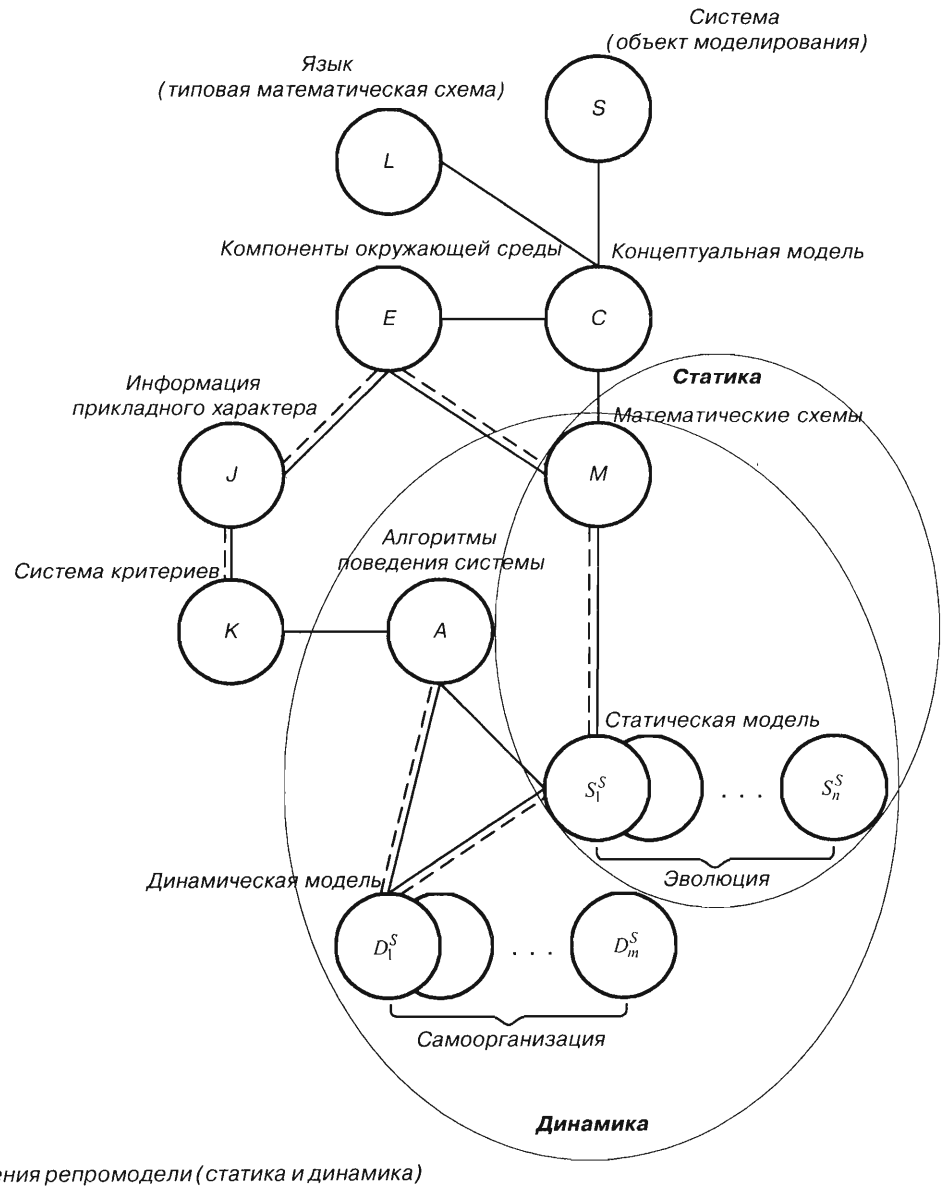
блему разработки такой теории значительные трудности. В частности, не удается непосредственно проверить адекватность модели процесса функционирования системы S с помощью реального объекта. Частично эта трудность устраняется путем проведения натурных экспериментов с элементами S. Ряд существенных трудностей возникает из-за неполноты исходной информации об объекте моделирования.

Большой объем знаний о системах и их элементах, накопленный к настоящему времени, подлежащий объединению в рамках теории моделирования и несоизмеримый с познавательными возможностями одного исследователя, выдвигает необходимость организации и детализации таких знаний (теории) в систему, затрагивающую лишь существенно ограниченное число объектов при сохранении общности подхода. При этом развитие отдельных методов статистического моделирования, языков моделирования, теории планирования машинных экспериментов оказывается недостаточным.

Создание прикладной теории, обеспечивающей конкретные потребности разработчика модели и охватывающей весь процесс моделирования в широком смысле этого слова, требует системного подхода и прежде всего установления основ теории: понятий об объекте, предмете, содержании, структуре и логике теории.

Объект прикладной теории моделирования

Объектом разрабатываемой прикладной теории является непосредственно процесс моделирования поведения системы S, т. е. процесс перехода от моделируемого объекта (системы S) сначала к статической модели S^S, используемой при стратегической идентификации, а затем и к динамической модели D^S, непосредственно исполь-



■ *Схема построения репромодели (статика и динамика)*

зуемой при оперативном управлении с использованием методов и алгоритмов ИУС. При этом ориентируются на критериальную систему K. Такой переход осуществляется через описание (концептуальную модель), фиксирующее сведения об объекте S в понятиях языка L (терминах типовых математических схем). При выборе математической схемы моделирования M вводится также понятие среды E, позволяющее использовать информацию прикладного характера J о целях моделирования, законах функционирования системы S, имеющемся математическом аппарате и другую информацию для исследования методов и алгоритмов управления системой A.

Таким образом, поскольку объектом данной прикладной теории моделирования является процесс моделирования, то возникает необходимость в построении и изучении «модели моделей», или *репромодели RM* (от англ. reproduce – воспроизводить, делать копию, порождать). Репромодель представляет собой упрощенный и наглядный прототип создаваемых моделей, используемых в ИУС, и дает возможность эффективного приближения к таким моделям с максимальным использованием априорной и оперативной информации о поведении системы S, поступающей в процессе ее функционирования. Схема построения репромодели приведена на рисунке. После того как сформулирована концептуальная модель C и введены понятия компонент сред S, основное содержание элементов прикладной теории моделирования для управления системой составляют компоненты M, A, S^S и D^S (критерий K считается заданным), причем переход от M к S^S составляет *статику* моделирования, а переход от M к множеству D^S с привлечением информации из компонент S^S и A – *динамику* моделирования. Такое разделение на статику и динамику условно показано на рисунке пунктирной и сплошной линиями, соответственно.

Движение в пространстве статических моделей процесса функционирования системы S^S назовем *эволюцией* (или эволюционным моделированием), а движение в пространстве динамических (активных) моделей D^S, используемых в контуре управления, – *самоорганизацией* (или моделированием с самоорганизацией). Важно отметить, что компоненты объекта теории L, C, E, M имеют искусственное происхождение, базирующееся на эвристических представлениях, и могут при необходимости изменяться (развиваться) в интересах самой прикладной теории. Это существенно отличает прикладную теорию моделирования от естественнонаучных теорий.

Предмет прикладной теории моделирования

Высказывания, составляющие любую теорию, формируются относительно предмета теории, а именно системы понятий, отображающих с той или иной степенью обобщения объект теории (репромодель RM). Таким образом, задание предмета прикладной теории моделирования процессов в системе S равносильно заданию репромоделей. Очевидно, что различным репромоделям должны соответствовать различные аспекты теории. Применительно к ИУС сужаем круг этих аспектов за счет конкретизации целей моделирования путем введения в репромодель компонент A, ограниченных методами и алгоритмами оперативного управления. Построение репромодели по схеме, приведенной на рисунке, позволяет использовать как информацию общего характера о процессах моделирования и управления J, так и конкретную информацию о методах и алгоритмах управления системой S с учетом выбранных критериев оценки эффективности K.

Содержание, структура и логика прикладной теории

Содержание прикладной теории моделирования охватывает две части: 1) базис теории, включающий систему эвристических принципов, полученных при обобщении имеющегося опыта моделирования сложных объектов вообще; 2) тело теории, содержащее эвристические правила машинной реализации конкретных моделей процесса функционирования S (S^S и D^S).

Предложения теории, относящиеся к компонентам M, A, S^S и D^S или возможным переходам между ними, содержат множество условий, позволяющих точно их сформулировать лишь для простейших случаев. В пределе предложения сводятся к описанию фактов, относящихся к отдельным реализациям процесса моделирования, которые назовем *прецедентами Pr*. Отметим, что Pr составляют эмпирическую основу прикладной теории моделирования, а множество $\{Pr\}$, классифицированное по условиям, может рассматриваться как обобщенное предложение теории, содержащее весь зафиксированный в $\{Pr\}$ опыт моделирования сложных систем вообще.

Более определенные предложения теории могут быть получены на основе системного подхода с детализацией репромодели по этапам построения и реализации S^S и D^S, когда ставятся различные цели при моделировании процессов в системе S. В общем случае репромодель, т. е. ее базис, задается множеством *принципов* $\{\pi r\}$, определяющих желаемые свойства моделей (S^S и D^S) и другие ограничения. Использование $\{\pi r\}$ регламентируется предложениями теории, относящимися к ограниченному множеству обобщенных ситуаций. Поиск этих ситуаций в множестве известных прецедентов $\{Pr\}$ позволяет накопить необходимые факты в количестве, достаточном для формулировки обобщенных предложений.

Говоря о прикладной теории моделирования с системных позиций, невозможно обойти ее реализационный аспект. В теории это отражено введением понятия *трактабельности* модели, т. е. ее реализуемости в рамках принятых ресурсных ограничений (например, на оперативную память и быстродействие ЭВМ). Особенно важна трактабельность десиженсных моделей, непосредственно используемых в ИУС, так как часто от нее зависит эффективность конкретного метода и алгоритма управления (а иногда и возможность его использования вообще). Вопросы трактабельности модели ставятся во главу угла при проведении стратегического и тактического планирования машинных экспериментов. Поэтому не будем останавливаться на этих вопросах детально, отметим только, что трактабельность модели достигается выполнением набора практических *правил* реализации модели $\{\rho r\}$, которые и составляют тело прикладной теории моделирования.

Таким образом, в конечном итоге множество прецедентов выражается через меньшее число эвристических принципов $\{\pi r\}$ и практических правил реализации $\{\rho r\}$ (базис и тело теории). Это позволяет считать репромодель и систему $[\{\pi r\}, \{\rho r\}]$ основой «системного» аспекта прикладной теории моделирования. При практическом применении неизбежно объединение «прецедентного» и «системного» аспектов теории моделирования на основе логического понятия дополнительности. В данном случае это способствует сужению общей проблемы моделирования за счет введения в прикладную теорию компоненты *A*. Для обеспечения возможности развития репромодель должна строиться как открытая система, т. е. с соблюдением принципов архитектуры открытых систем, что нашло свое отражение при машинной реализации моделей (например, в системе моделирования GPSS, хотя авторы при ее разработке вряд ли подозревали об этом и получили эффект открытости интуитивно, на опыте аналогичных разработок).

Относительно логики прикладной теории моделирования отметим, что она опирается на индуктивный подход, т. е. обобщение и классификацию множества прецедентов $\{Pr\}$, оставляя место для дедуктивного подхода в рамках конкретных математических схем *M*.

Вопросы практического воплощения прикладной теории моделирования непосредственно связаны с реализацией соответствующих инструментальных средств моделирования и возможностью ее использования для решения задач моделирования конкретных ИУС, работающих в РМВ.

Литература

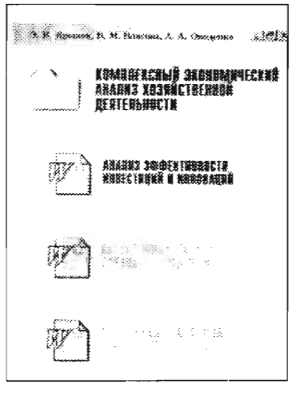
1. **Арсеньев Б. П., Яковлев С. А.** Интеграция распределенных баз данных. – СПб.: Лань, 2001. – 464 с.
2. **Колбанев М. О., Яковлев С. А.** Модели и методы оценки характеристик обработки информации в интеллектуальных сетях связи. – СПб.: Изд-во СПбГУ, 2002. – 230с.
3. **Советов Б. Я., Яковлев С. А.** Моделирование систем / 3-е изд. – М.: Высшая школа, 2001. – 343 с.
4. **Советов Б. Я., Яковлев С. А.** Моделирование систем. Практикум / 2-е изд. – М.: Высшая школа, 2003. – 295 с.
5. **Швецов А. Н., Яковлев С. А.** Распределенные интеллектуальные информационные системы. – СПб.: Изд-во СПбГЭТУ, 2003. – 318 с.

Крылов Э. И., Власова В. М., Оводенко А. А.

Анализ эффективности инвестиций и инноваций: Учеб. пособие / СПбГУАП. СПб., 2003. 506 с.: ил. ISBN 5-8088-0093-5

В учебном пособии рассматриваются задачи анализа эффективности инвестиций и инноваций. Исследованы взаимосвязи между инвестиционной, инновационной, финансовой и производственной деятельностью предприятия в рамках учебных курсов "Комплексный экономический анализ хозяйственной деятельности", "Инвестиции".

Пособие предназначено для студентов, обучающихся по экономическим специальностям 060400 "Финансы и кредит", 060500 "Бухгалтерский учет и аудит", 060600 "Мировая экономика", и может быть использовано студентами междисциплинарных специальностей 3514 "Прикладная информатика (в экономике)", 3512 "Налоги и налогообложение".



УДК 621391.28:518.5

ИЕРАРХИЯ ПРОТОКОЛОВ ДЛЯ ТЕХНОЛОГИИ АТМ

А. Р. Бестугин,
канд. техн. наук, доцент
Санкт-Петербургский государственный университет аэрокосмического приборостроения

В статье рассматривается текущее состояние эталонной модели иерархии протоколов для технологии АТМ.

The current reference model for ATM is considered.

Создание мультисервисных сетей связи, которые строятся на основе АТМ-технологии, вызывает в настоящее время определенный интерес. АТМ-технология является, возможно, самой сложной из известных технологий, поэтому число публикаций, касающихся различных ее сторон, постоянно растет. В основном в публикациях рассматриваются вопросы управления трафиком и перегрузкой, а также качества обслуживания. Однако все эти вопросы решаются только на основе модели протоколов Ш-ЦСИО (широкополосной цифровой сети интегрального обслуживания), которая является расширением Эталонной модели взаимодействия открытых систем (ЭМВОС) и полностью соответствует принципам, положенным при разработке в ее основу. Вместе с тем модель имеет существенные отличия. Прежде всего, она включает в себя три плоскости: пользователя, управления и администрирования (рис. 1). Иногда рассматривается четвертая плоскость (контрольная – для обработки контрольной информации). Все плоскости разбиты на уровни. Функции уровней эталонной модели протоколов в настоящее время относительно детально разработаны только для первых трех уровней модели. Это физический уровень; уровень АТМ (где происходит структурирование ячеек); уровень адаптации АТМ (который поддерживает услуги более высокого уровня, например, эмуляцию каналов, высокоскоростную передачу данных без установления соединения, ретрансляцию кадров и т. п.). При этом модель продолжает развиваться и дополняется новыми функциями.

Физический уровень (Physical Layer) предназначен для передачи по физическим каналам и состоит из двух подуровней:

1) адаптации к физической среде передачи (PMD – Phsical Media Dependent Sublayer);

2) преобразования передачи (TC – Transmission Convergence Sublayer).

На физическом уровне реализуется два типа функций, зависящих от типа физической линии и функции преобразования пакетов АТМ в последовательность единичных элементов.

В первую группу входят функции, обеспечивающие передачу двоичных сигналов, тактовую синхронизацию и электронно-оптические преобразования.

Во вторую группу входят:

генерация и восстановление кадров, состоящих из множества пакетов АТМ;

структурирование кадров в соответствии с выбранным принципом цифровой передачи, например, SONET/SDN, DS3 (45 Мбит/с), E3 (34 Мбит/с), DS1 (1,5 Мбит/с), E1 (2 Мбит/с) и др.;

структурирование потока пакетов АТМ, обеспечивающее определение границ пакетов АТМ на приемной стороне, для чего на передающей стороне поток пакетов скремблируется;

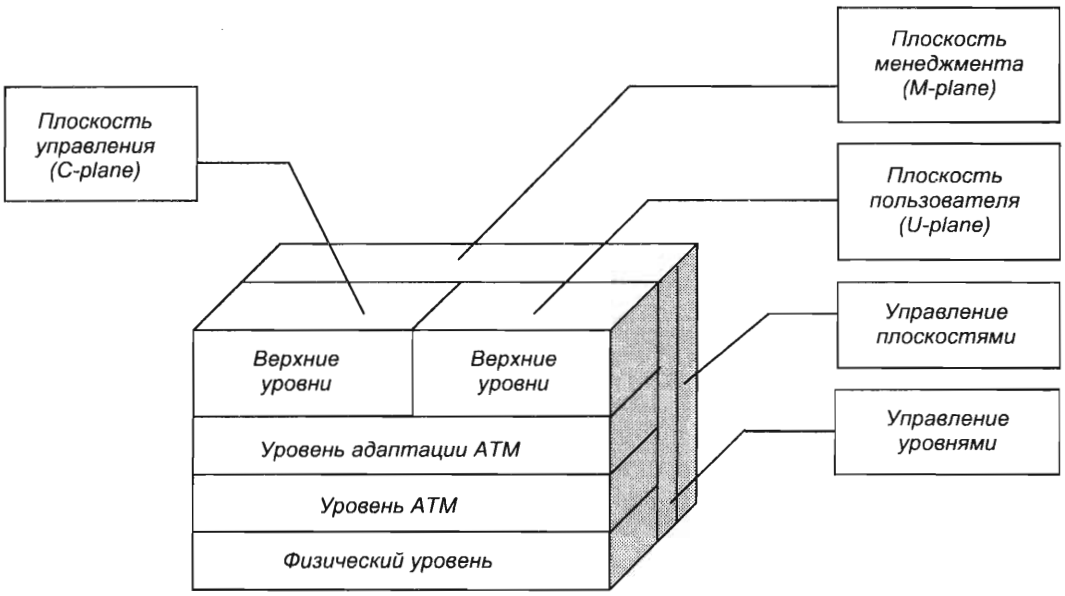
определение циклической последовательности для защиты заголовка пакета АТМ от ошибок (на передающей стороне);

проверка и, если возможно, исправление ошибок (на приемной стороне); пакеты АТМ с пораженным заголовком удаляются;

согласование скоростей пакетов АТМ, состоящее во вставке и подавлении «пустых» пакетов, используемых для согласования свойств потока пакетов и передающей системы.

Уровень асинхронного режима доставки (ATM Layer – Asynchronous Transfer Mode Layer) отвечает за создание АТМ-пакетов (или ячеек). Он принимает 48-байтовые пакеты с уровня адаптации и добавляет 5-байтный заголовок. На уровне АТМ реализуются следующие функции:

мультиплексирование и демультиплексирование пакетов АТМ; в направлении передачи пакеты АТМ на различных виртуальных путях (ВП) и



■ ***Рис. 1.*** *Эталонная модель протоколов Ш-ЦСИО*

виртуальных каналах (ВК) объединяются в общий поток на основе асинхронного временно́го разделения; при приеме пакеты АТМ распределяются на соответствующие ВП и ВК; преобразование идентификатора виртуального пути (ИВП) и идентификатора виртуального канала (ИВК) производится в коммутаторах АТМ путем отображений значений входящих ИВП/ИВК в исходящие;

функция генерации/ выделения заголовка пакета АТМ предполагает генерацию в направлении передачи полного заголовка, за исключением поля защиты от ошибок; поле заголовка генерируется на основе информации, поступающей с верхнего уровня, при приеме заголовок пакета АТМ удаляется;

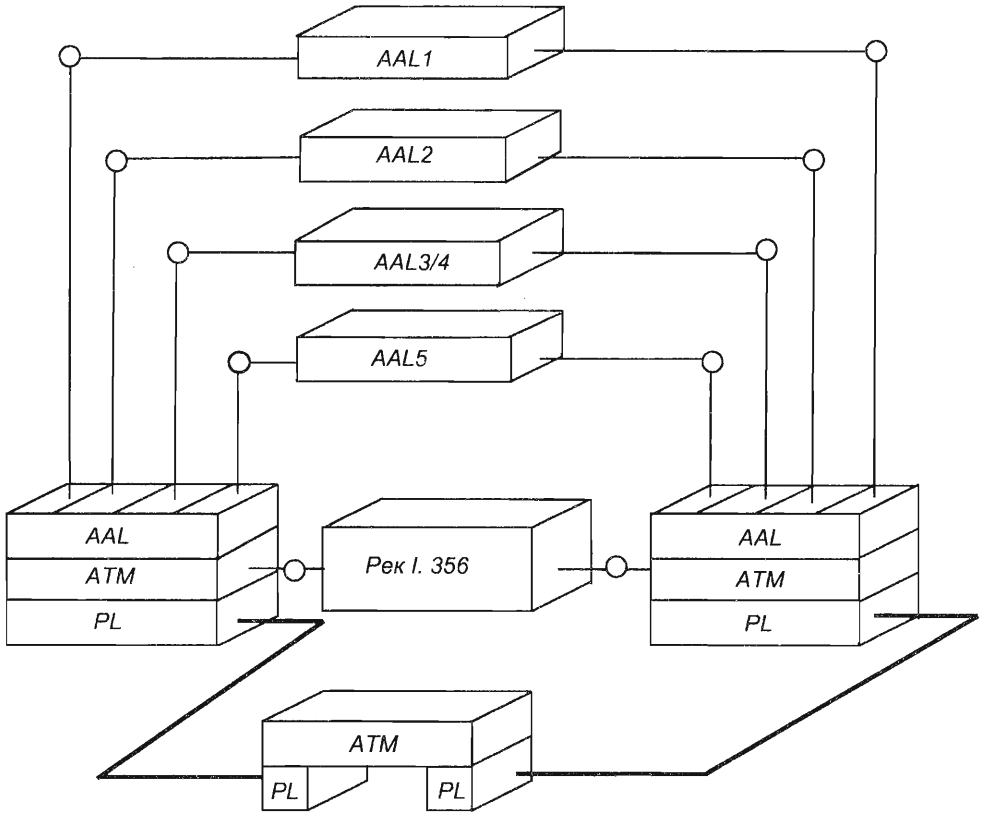
общее управление потоком, процедуры, которые специально генерируются на уровне АТМ-пакетов.

Функции передачи сигналов, управление трафиком уровня АТМ подобны функциям канального уровня Эталонной модели, а функции установления соединения ближе всего к функциям маршрутизации, которые определены стандартами Эталонной модели для сетевого уровня. Вопросы управления рассмотрены в работах [1,3, 5].

Ориентация на соединение – уникальная характеристика сетей АТМ, которая определяет сложность протоколов АТМ. Сам факт, что АТМ ориентирован на соединение, подразумевает потребность в специальных для АТМ протоколах сигнализации и структурах адресации наряду с протоколами маршрутизации запросов на соединение через сеть АТМ. Эти протоколы АТМ, в свою очередь, влияют на способ, посредством которого протоколы высших уровней могут функционировать через сети АТМ. Последние могут быть реализованы по-разному, и каждой реализации присущи свои достоинства и недостатки.

АТМ использует принцип виртуальных соединений между конечными точками сети. Различают два вида соединений: 1) PVC (Permanent Virtual Cir-cuit) – постоянный виртуальный канал; 2) SVC (Switched Virtual Circuit) – коммутируемый виртуальный канал. PVC представляет собой соединение между конечными точками, которое существует постоянно и может устанавливаться или разрываться вручную. SVC – соединение между конечными точками, устанавливаемое или закрываемое динамически специальными процедурами в АТМ-устройствах, участвующих в соединении. Все протоколы высших уровней, действующие через АТМ, первоначально используют SVC. Коммутируемые виртуальные соединения автоматически устанавливаются (посредством протокола сигнализации) и разрываются по требованию программного обеспечения АТМ-устройств или по другим причинам, без вмешательства оператора АТМ-сети. Процессы формирования ячеек и их передача не различаются для обоих вариантов соединений. Единственное их отличие состоит в способах установления соединения.

АТМ использует принципы виртуальных путей (VP – Virtual Path) и виртуальных каналов (VC – Virtual Chanel) между конечными точками сети. Они необходимы для связи одного АТМ-устройства с несколькими одновременно. Виртуальные пути (ВП) и виртуальные каналы (ВК) используют для идентификации отдельных виртуальных соединений в АТМ-сети. ВП – это логическая конструкция, которая объединяет виртуальные каналы по определенному признаку. ВП и ВК не существуют параллельно. Понятие ВК описывает в общем случае логическое соединение, обеспечивающее однонаправленную транспортировку пакетов АТМ между узлами АТМ. Каждое соединение в канале имеет уникальные идентификаторы ВП (ИВП) и ВК (ИВК). ИВП/ИВК включаются в заголовок АТМ-пакета и предназначаются для его маршрутизации через сеть. Несколько последовательных ВК об-



■ ***Рис. 2.*** *Протокольные типы уровня AAL*

разуют соединения (СВК). Концевые точки СВК представляют собой элементы сети, в которых информационное поле пакета ATM рассматривается как протокольная единица при обмене между уровнем ATM и уровнем адаптации ATM. Последовательная совокупность ВП образует соединение СВП, расположенное в транспортной сети между двумя окончаниями СВП.

Система сигнализации является выделенной и функционирует независимо от системы передачи пользователей информации. Система сигнализации должна обеспечивать установление и управление СВП и СВК. При этом соединения могут устанавливаться по требованию или быть полупостоянными и постоянными. Среди других функций сигнализации необходимо отметить следующие:

поддержка соединений в различных сетевых конфигурациях (двухточечные, многоточечные, вещательные);

согласование характеристик трафика и ресурсов сети при установлении соединения;

возможность реконфигурации установленных соединений, в том числе путем добавления и удаления отдельных элементов ВК и ВП;

поддержка соединений со службами, не входящими в перечень служб Ш–ЦСИО (например, со службами аналоговой сети).

Для передачи управляющих сигналов при необходимости двухточечного соединения организуется канал сигнализации, использующий соединения ВК между двумя концевыми точками сети сигнализации. Образованный канал, называемый двухточечным каналом сигнализации, идентифицируется парой ИВП/ИВК. Аналогичным образом могут быть организованы каналы сигнализации при формировании многоточечных и вещательных конфигураций.

Процедуры для установления, проверки и поддержания ВК сигнализации определены в Рекомендации I.311 как мета-сигнализация (МС). Процедура МС реализуется в постоянном соединении ВК, называемом ВК-мета-сигнализацией, которое устанавливается для каждого направления и идентифицируется парой ИВП/ИВК. К функциям системы мета-сигнализации относятся:

управление ресурсами системы сигнализации (в первую очередь, пропускная способность каналов сигнализации);

установление и управление каналами сигнализации;

управление доступом требований пользователей к системе сигнализации.

Уровень адаптации ATM (AAL – ATM Adaptation Layer) в соответствии с эталонной моделью протоколов AAL расположен между уровнем ATM и сетевым уровнем. AAL обеспечивает доступ пользовательских приложений к коммутирующим устройствам ATM, так как многие приложения не имеют прямого доступа к сервису ATM. Уровень AAL содержит два подуровня:

1) общий подуровень CP(Common Part);

2) подуровень объединения конкретных услуг SSCS (Service Specific Convergence Sublayer); подуровень SSCS, в свою очередь, делится еще на два подуровня:

1) SSCF (Service Specific Coordination Function) — функция координации конкретных услуг);

2) SSCOP (Service Specific Connection Oriented Protocol) — протокол, ориентированный на соединение конкретной услуги).

Этот уровень, в свою очередь, разбит еще на два подуровня:

1) SAR (Segmentation and Reassembly) — разборка и сборка;

2) CS (Convergence Sublayer) — подуровень преобразования.

Как ясно из названия, SAR-подуровень разбирает и собирает большие пакеты, а CS-подуровень обеспечивает синхронизацию для различных классов сервиса. На подуровне разборки/ сборки пакеты, полученные после преобразования, которые называются CS-PDU (Convergence Sublayer Data Unit) разбиваются на 48-байтовые отрезки. Это вызвано тем, что уровень ATM не может оперировать пакетами данных длиной более 53 байт. Подуровень разборки/ сборки гарантирует, что ни один пакет длиной, отличной от этого размера, не пересечет «границу» между уровнями AAL и ATM с той или с другой стороны.

Уровень AAL имеет контрольные функции, называемые LME (Layer Managment Entry) или ME (Managment Entry). В задачу этих функций входят инициализация, контроль и координация представления пользовательских данных и контрольной информации при обращении к уровню ATM с высших уровней.

Подуровень CS отвечает за подготовку пакетов информации, поступивших с пользовательской платформы (например, IP-пакетов), для сегментации. Такая операция необходима для возможности правильного восстановления исходного пакета на приемном конце. Для корректной сегментации пакетов в них добавляется некоторая контрольная информация, использование которой зависит от типа адаптационного сервиса.

Таким образом, основными функциями AAL являются сегментация информации, поступающей с верхних уровней на блоки, размер которых должен соответствовать стандарту уровня ATM, и сборка информационных блоков из пакетов, поступающих с уровня ATM. Функции AAL зависят от вида службы, поддерживаемой в Ш-ЦСИО. С учетом этой особенности в Рекомендации I.362 предложена классификация служб с тем, чтобы минимизировать количество протоколов AAL. Всего были введены четыре класса служб (A,B,C,D). Недавно был введен класс X. В качестве классификационных признаков были выбраны следующие основные характеристики [1]:

синхронизация устройств между конечными точками передачи (требуется или нет);

скорость передачи битов информации (постоянная или изменяющаяся);

режим соединения (с установкой или без).

Классификационные признаки и классы служб подробно описаны в работе [5]. Можно выделить несколько общих для классов A и B функций:

сегментация и сборка информации пользователя;

контроль и регулировка задержки пакетов ATM;

контроль прохождения пакетов через сеть;

контроль тактовой синхронизации на отдельных участках сети.

Основными функциями для классов C и D являются сегментация и сборка информации пользователя и обнаружение ошибок в пользовательских данных. Для класса D в состав функций включаются также процедуры, требуемые для поддержания режима без установления соединения. Эти функции определены в самом общем виде, однако существует их связь с функциями адресации и маршрутизации сетевого уровня. В стандартах Ш-ЦСИО для этой цели определяются следующие типы протоколов: AAL1, AAL2, AAL3/4, AAL5 (рис.2). Форум ATM разработал только три из них (AAL1, AAL3/4, AAL5). Каждый протокол AAL упаковывает данные в ячейки своим способом. Все протоколы, за исключением AAL5, добавляют некоторую служебную информацию к 48 байтам данных в ячейки ATM. Информация включает в себя команды обработки для каждой ячейки, которые используются для обеспечения различных категорий сервиса.

Недавно был разработан новый протокол AAL-CU (ATM Adaptation Layer – Composit User, т. е. уровень адаптации к ATM для составного пользователя), который затем был переименован в AAL2. Существующие протоколы адаптации приспособлены для переноса в сеть небольших пакетов, чувствительных к задержкам. Таким образом, чтобы приемник мог легко выделить нужные пакеты, требуется частичное заполнение ячейками ATM, что ведет к неэффективной передаче. Новый протокол позволяет устранить этот недостаток и решает две задачи: защиту от ошибок и выделение пакетов. В результате этого сокращаются ошибки, повышаются скорость выделения пакетов и общая эффективность.

Литература

1. **Бестугин А. Р., Богданова А. Ф., Стогов Г. В.** Контроль и диагностирование телекоммуникационных сетей. – СПб.: «Политехника», 2003. – 174 с.
2. **Бондаренко Д.** ATM – сетевая технология будущего // Компьютер Пресс. – 1995. – № 2. – С. 87–89; № 5. – С. 48–50; № 6. – С. 89–91; № 7. – С. 99–100.
3. **Ефимушкин В. А., Ледовских Т. В.** Механизмы управления трафиком в сетях ATM // Электросвязь. – 2003. – № 1. – С. 39–45; № 2. – С. 33–36.
4. **Кучерявый А. Е., Пяттаев В. О., Моисеев С. М.** Технология ATM на российских сетях связи. – М.: Радио и связь, 2002. – 308 с.
5. **Eckberg A. E. B**-ISDN/ ATM Traffic and Congestion Control // IEEE Network, September 1992. – P. 28–42.

РЕКУРСИВНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Из прошлого через настоящее в будущее

Прошло 25 лет с того времени, когда в 1979 году на кафедре вычислительных систем и сетей Ленинградского института авиационного приборостроения был запущен первый в мире экспериментальный образец рекурсивной многопроцессорной вычислительной машины высокой производительности и надежности. Рассказывает основатель этой кафедры и ее заведующий в течение 30 лет доктор технических наук, профессор, лауреат Государственной премии СССР М. Б. Игнатьев

Я окончил электромеханический факультет Ленинградского политехнического института (ЛПИ) по специальности «Автоматика и телемеханика» в 1955 году и был распределен на работу на предприятие Министерства среднего машиностроения, в атомную промышленность, где я впервые встретился с роботами-манипуляторами для работы в активных зонах. Тогда они так не назывались, ведь в начале пятидесятых годов кибернетика была в СССР объявлена лженаукой. За годы учебы в Политехническом институте мы ни разу не слышали слово «кибернетика», но заведующий кафедрой автоматики и телемеханики ЛПИ профессор Б. И. Доманский так организовал учебный процесс, что мы были полностью готовы к восприятию идей кибернетики и на предприятиях сразу включились в решение проблем автоматизации очень сложных объектов. Проблема создания все более совершенных роботов захватила меня полностью, и с тех пор я занимаюсь этой проблемой, несмотря на различные перипетии моей жизни. Несколько раз я менял место работы, но как только в Ленинградском институте авиационного приборостроения (ЛИАП) открылась возможность вплотную заняться проблемой роботов, я начал там работать. К 1972 году нами был создан спектр различных робототехнических систем – от адаптивных манипуляторов до шестиногой шагающей машины.

В 1972 году я был назначен заместителем главного конструктора по робототехнике, но об этой стороне моей деятельности я расскажу в другой публикации. Робот – это синтез трех систем: во-первых, сенсорных систем, которые поставляют информацию об окружающей среде, во-вторых, системы движителей, которые могут манипулировать с различными предметами в окружающей среде и перемещать робот, в-третьих, системы принятия решений, управляющей системы, которая является как бы аналогом «мозга» робота. Развитие робототехники тесно связано с развитием вычислительной техники, и возможности роботов определяются возможностями вычислительных структур разного уровня. Поэтому после организации кафедры вычислительных систем и сетей в ЛИАП в 1972 году важным направлением ее деятельности кроме робототехники стало создание развивающихся вычислительных систем нетрадиционной архитектуры. Чтобы понять логику такого решения, необходимо рассказать о состоянии мировой вычислительной техники в начале семидесятых годов.

В это время господствующее положение занимала фирма IBM, грубо нарушающая законы о монополиях и ведущая судебные процессы во многих штатах внутри США и других странах. Этот монополизм проявился и в компьютерной литературе – там описывались машины IBM и почти ничего не говорилось о машинах других фирм, таких как «Контрол Дейта Корпорейшен», «Бэрроуз» и др., которые выступали конкурентами IBM. В машинах фирмы IBM реализовывалась классическая фон-Неймановская архитектура, которая уже не могла удовлетворить потребителей. В Советском Союзе шла борьба двух тенденций: развитие своих собственных разработок, таких как БЭСМ, Урал и другие, и копирование зарубежного опыта, прежде всего копирование машин IBM. В этой ситуации наша молодая кафедра, выделившаяся из кафедры технической кибернетики ЛИАП в феврале 1972 года, решила развивать нетрадиционные многопроцессорные вычислительные системы, которые в перспективе обеспечивали высокую производительность и надежность. Для меня это решение было продолжением моих работ в области цифровых дифференциальных анализаторов, которые являлись многопроцессорными специализированными рекурсивными структурами с обратными связями, высокопроизводительными и надежными за счет введения избыточности методом избыточных переменных, разработанным мною ранее[1]. Важный шаг был сделан нашим доцентом В. А. Торгашевым, который предложил распрост-

---

[1] Воронов А. А., Гарбузов А. Р., Ермилов Б. Л., Игнатьев М. Б. и др. Цифровые аналоги для систем автоматического управления. – М.: Изд-во АН СССР, 1960.

Игнатьев М. Б. Голономные автоматические системы. – Л.; М.: Изд-во АН СССР, 1963.

Мальцев А. И. Алгоритмы и рекурсивные функции. – М., 1965.

Игнатьев М. Б. О совместном использовании принципов введения избыточности и обратной связи для построения ультраустойчивых систем // Тр. III Всесоюзного совещ. по автоматическому управлению. Т. 1. – Изд-во АН СССР, 1968.

Игнатьев М. Б. Метод избыточных переменных для функционального кодирования цифровых автоматов // Теория автоматов. № 4. – Киев: Изд-во ИК АН УССР, 1969.

Игнатьев М. Б. О лингвистическом подходе к анализу и синтезу сложных систем // Тез. Межвузовской научно-технич. конф. «Техническая кибернетика». – М.: Изд-во МВТУ, 1969.

Игнатьев М. Б. Избыточность в многоцелевых системах // Тр. IV симпозиума по проблеме избыточности. – Л., 1970.

Игнатьев М. Б., Кулаков Ф. М., Покровский А. М. Алгоритмы управления роботами-манипуляторами. – М.: Машиностроение, 1972 (США, Вирджиния пресс, 1973, 3-е издание 1977).

Бритов Г. С., Игнатьев М. Б., Мироновский Л. А., Смирнов Ю. М. Управление вычислительными процессами. – Л.: Изд-во ЛГУ, 1973.

INFORMATION PROCESSING 74 – NORTH-HOLLAND PUBLISHING COMPANY (1974)

RECURSIVE MACHINES AND COMPUTING TECHNOLOGY

V. M. GLUSHKOV*, M. B. IGNATYEV**, V. A. MYASNIKOV*** and V. A. TORGASHEV**

*USSR Academy of Sciences
Institute of Cybernetics, Ukrainian Academy of Sciences
Kiev, USSR
**Leningrad Institute of Aviation Instrument Making
Leningrad, USSR
***Main Department of Computing Engineering and Control Systems
State Committee of the USSR Council of Ministers for Science
and Technology
Moscow, USSR

The paper analyzes disadvantages of traditional computers. It shows that partial revision of von Neumann principles fails to provide a leap in the development of computing technology. New principles of program and structural organization of digital computers are offered. Their name, "recursive computers" (RC), stems from the recursive method of defining their internal language and structure. The main features of their internal language and their tentative architecture are considered, and their capabilities are evaluated.

■ ***Рис. 1.*** *Титул к историческому докладу на конгрессе ИФИП в Швеции*

ранить и развить эти принципы на универсальные вычислительные машины. В итоге родилась концепция рекурсивных машин, которая получила поддержку Государственного комитета по науке и технике в Москве и Института кибернетики во главе с академиком В. М. Глушковым в Киеве. Сложился коллектив из москвичей, которых представлял В. А. Мясников, из киевлян, которых представлял В. М. Глушков, и ленинградцев с общим центром в ЛИАПе. В наиболее ярком виде эта концепция была представлена в нашем докладе на международном конгрессе ИФИП в Стокгольме в 1974 году[1] (рис. 1). Доклад в Стокгольме делал я, советская делегация отнеслась ко мне очень холодно, зато иностранцы приветствовали доклад, который ниспровергал компьютерные авторитеты и традиционную архитектуру и провозглашал нетрадиционную рекурсивную, которая потом завоевала весь мир в виде систем клиент-сервис. Впервые советская компьютерная разработка была анонсирована на международной арене, что привлекло внимание с разных сторон. Итогом этой акции было, во-первых, включение работы в программу ГКНТ и выделение финансов на создание экспериментального образца рекурсивной машины, во-вторых, соглашение с фирмой «Контрол Дейта Корпорейшен» по созданию рекурсивной машины на основе наших архитектурных решений, в-третьих, предоставление самой лучшей для того времени элементной базы и средств отладки. Я стал руководителем рабочей группы по сотрудничеству с фирмой «Контрол Дейта Корпорейшен» и в этом качестве развивал как проект по рекурсивной машине, так и другие проекты, в числе которых была покупка машины Сайбер для Ленинградского научного центра АН СССР.

[1] Glushkov V., Ignatyev M., Miasnikov V., Torgashev V. Recursive machines and computing technology. Proceedings IFIP-74, computer hardware and architecture, p. 65–70, Stockholm, August 5–10, 1974.

На базе этой машины организовался сначала Ленинградский научно-исследовательский вычислительный центр, а потом Ленинградский институт информатики и автоматизации АН СССР. Следует отметить, это было время некоторого потепления в советско-американских отношениях, именно в это время реализовывался проект Союз–Апполон. Таким образом, в результате стечения благоприятных обстоятельств нам удалось развернуть работу по реальному созданию рекурсивной машины. Закипела работа, в которой принимали участие многие сотрудники нашей кафедры: В. А. Торгашев, В. И. Шкиртиль, С. В. Горбачев, В. Б. Смирнов, В. М. Кисельников, А. М. Лупал, Ю. Е. Шейнин и многие другие. В результате были изготовлены многие блоки машины, и осенью 1979 года экспериментальный образец рекурсивной машины был предъявлен государственной комиссии во главе в академиком А. А. Дородницыным. В специальном Постановлении ГКНТ СССР и Комиссии Президиума Совета Министров СССР от 14.09.1979 г. за № 472/276 отмечалось, что запуск первого в мире экспериментального образца многопроцессорной рекурсивной машины высокой производительности и надежности является достижением мирового уровня. Были разработаны планы дальнейшего развития этой работы, но в декабре 1979 года советские войска вошли в Афганистан и правительство США разорвало все научно-технические связи с СССР, в том числе и по линии фирмы «Контрол Дейта Корпорейшен», что нанесло нам большой ущерб. Но работа продолжалась, хотя наш коллектив разделился – часть сотрудников в январе 1980 года во главе с В. А. Торгашевым перешла в Ленинградский научно-исследовательский вычислительный центр АН СССР, другая часть продолжала работать на нашей кафедре над созданием различных модификаций многопроцессорных систем. В Институте кибернетики в Киеве был создан отдел рекурсивных машин. Таковы внешние контуры этой пионерской работы.

Принципы организации рекурсивных машин и систем

В математике существует большой раздел – рекурсивные функции[1]. Долгое время термин «рекурсия» употреблялся математиками, не будучи четко определенным. Его приблизительный интуитивный смысл можно описать следующим образом. Значение искомой функции Ф в произвольной точке *X* (под точкой подразумевается набор значений аргументов) определяется, вообще говоря, через значения этой же функции в других точках *H*, которые в какомто смысле предшествуют *X*. Само слово «рекурсия» означает возвращение[2]. Рекурсивные функции – это вычислимые функции. По сути дела, все вычислимые на компьютерах функции – это рекурсивные функции, но разные компьютерные архитектуры поразному ведут вычислительные процессы. Чем лучше соответствует структура компьютера структуре задач, тем меньше затраты памяти и времени. Так что когда мы говорим о рекурсивных машинах, мы говорим о соответствии структур машины и задач, а так как задачи бывают разные, то структура машин должна гибко подстраиваться к структурам задач. Математика в настоящее время погружена в программирование, и в программировании рекурсивные операции распространены.

ЭВМ выступает как средство материализации логико-математических преобразований. ЭВМ являет собой иллюстрацию концепции потенциальной осуществимости, поскольку при отсутствии ограничений на время работы и емкость памяти любая ЭВМ в состоянии провести любые вычисления. Конкретное же протекание процессов вычисления проявляется лишь на уровне организации преобразований информации (задействуются конкретные регистры, коммутаторы, процессоры, линии передачи данных в определенном порядке и сочетании и т. д.). С этой точки зрения «архитектура ЭВМ» – это ее структура в состоянии (процессе) реализации алгоритма, т. е. как бы ожившая структура. Философской основой такого представления является теория отражения, раскрывающая отображение категорий и явлений одной природы (числа, алгоритмы) на объекты другой природы (физические элементы, сигналы). Причем это отображение взаимно неоднозначно – алгоритму *aj* может соответствовать множество архитектур {*A*} и обратно – архитектуре *Aj* непосредственно не соответствует какой-либо алгоритм *aj*. Специфика взаимодействия {*a*} и {*A*} раскрывает глубинные свойства диалектического процесса развития математики и вычислительной техники как частного случая взаимодействия абстрактного и конкретного. Как отмечает С. А. Яновская, «лицо машинной математики все более зависит от развития философских и логических оснований математики»[3]. Не представляется возможным непротиворечивая формализация отображения {*a*} а {*A*} из-за его неоднозначности. Поэтому построить соответствующую аксиоматическую теорию проектирования ЭВМ не представляется возможным[4].

Когда мы формулировали принципы организации рекурсивных машин, мы исходили из потребностей развития вычислительных машин и систем, получили множество авторских свидетельств[5]. Это был интересный творческий процесс, и с точки зрения достоверности сделанного тогда, в 1974–1979 годах, стоило бы полностью воспроизвести наш доклад на конгрессе ИФИП в Стокгольме. Этот доклад содержал анализ недостатков машин традиционной архитектуры, ревизию принципов фон Неймана, принципы архитектуры рекурсивных машин, основные особенности языка рекурсивных машин, фрагментарное описание рекурсивной машины. В качестве иллюстрации рекурсивной структуры можно привести систему 3М – модульную микропроцессорную систему[6]. Система 3М строится из модулей трех типов – операционных, коммуникационных и интерфейсных (рис. 2, рис. 3). Операционные модули выполняют основную работу по обработке данных, реализации объектов математической памяти, процессов определения готовности и выполнения операторов программы на внутреннем языке. Коммуникационный модуль предназначен для реализации коммуникационной системы – установления логического соединения между модулями, обмена информацией между модулями, поиска в системе ресурсов запрошенного типа. Интерфейсные модули подключаются к внешним устройствам своими блоками ввода-вывода. Вопросы организации обмена информацией с внешним миром имеют большое значение для существенно многопроцессорных систем, оказывают значительное влияние на их фактические характеристики. Различные классы задач требуют различной интенсивности обмена с внешними устройствами. Вычислительная система должна обеспечивать построение таких ее конфигураций для каждого конкретного применения, которые бы обладали оптимальными для этого применения характеристиками по вводу-выводу. Система 3М обеспечивает инкрементное наращивание вычислительной мощности до любого необходимого

---

[1] Мальцев А. И. Алгоритмы и рекурсивные функции. – М., 1965.

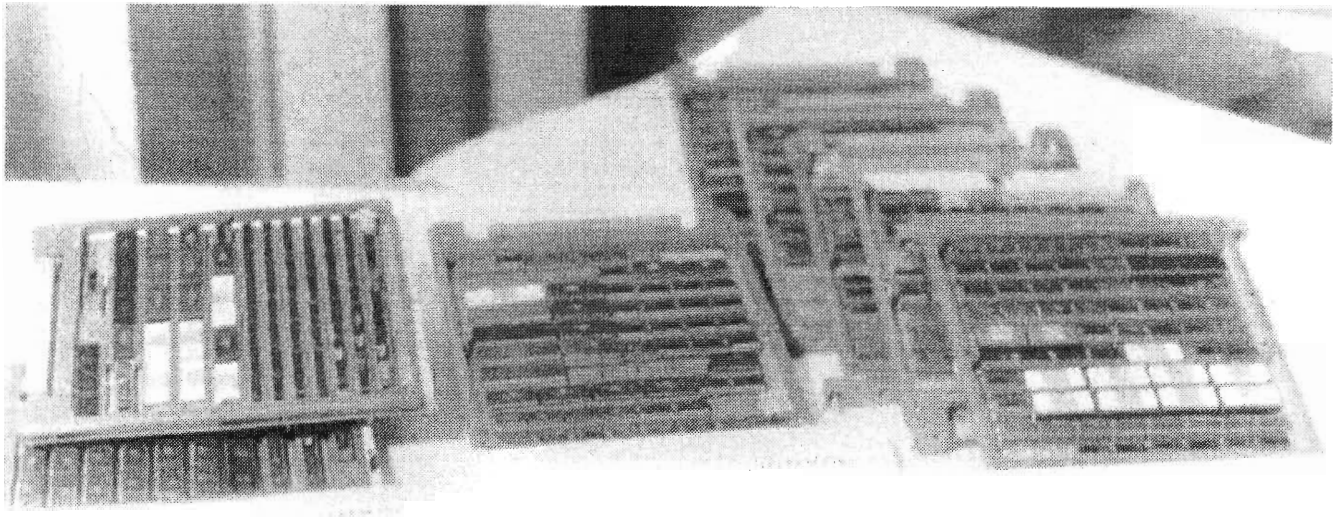
[2] Математическая энциклопедия. Т. 4. (статья «Рекурсия») – С. 962–966. – М., 1984.

[3] Яновская С. А. Методологические проблемы науки. – М., 1972.

[4] Суворова П. Г. Диалектика абстрактного и конкретного в понятии «архитектура ЭВМ // Новые идеи в философии науки и научном познании / Под ред. Ю. И. Мирошникова. – Екатеринбург, 2002.

[5] Игнатьев М. Б., Кисельников В. М., Торгашев В. А., Смирнов В. Б. Ассоциативное запоминающее устройство. А. с. № 4844562. Бюлл. изобретений № 34, 1975.
Бекасова А. А., Горбачев С. В., Игнатьев М. Б., Мясников В. А., Торгашев В. А. Процессор с микропрограммным управлением. А. с. № 814118 с приоритетом от 18.10.1979.
Горбачев С.В., Игнатьев М.Б., Кисельников В.М., Мясников В. А., Торгашев В.А. Многопроцессорная вычислительная система. А. с. № 962965 с приоритетом от 27 августа 1974 г. Бюлл. изобретений № 36, 1982.

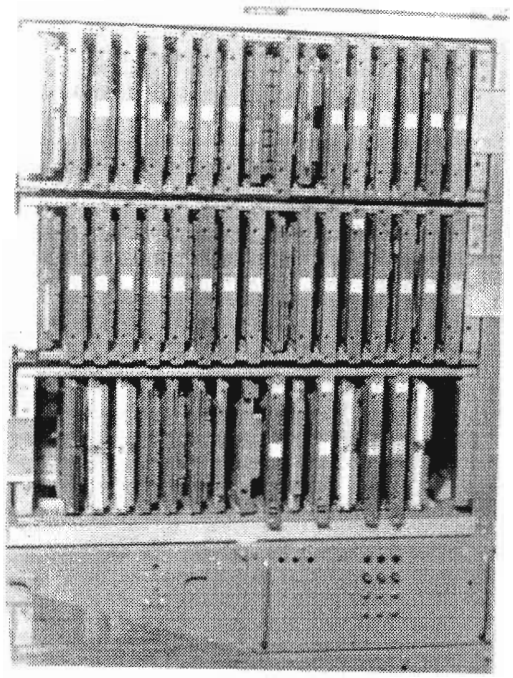
[6] Игнатьев М. Б., Кочкин А. И., Шейнин Ю. Е. Микропроцессорные ситемы – архитектура, программирование и отладка. – М.: Изд-во Атомэнергоиздат, 1983.



а) б) в)

■ ***Рис. 2.*** *Система 3М: а) комуникационный модуль; б) интерфейсный модуль; в) вычислительный модуль*

значения путем подключения дополнительных блоков без внесения изменений в имеющуюся систему и ее программное обеспечение как на этапе разработки системы, так и в ходе ее эксплуатации. Методология проектирования и реализации системы 3М базируется на рассмотрении вычислительной системы как иерархии виртуальных машин. Система 3М имеет рекурсивно-организованную многоуровневую структуру. Рекурсивность структуры состоит в том, что структура всякой модификации системы задается рекурсивным определением. Динамически меняющиеся в ходе вычислений виртуальные процессы требуют постоянной динамической реконфигурации связей между модулями. Сейчас реализуются системы, содержащие тысячи и миллионы процессоров.



■ ***Рис. 3.*** *Модуль системы 3М в собранном виде*

Перспективы развития вычислительных систем

В связи с изложенным хотелось рассмотреть проблемы развития вычислительной техники. Вычислительные машины предназначены для решения задач. Общая схема решения задач имеет вид

$$Я_{чел} \rightarrow Я_{ос} \rightarrow Я_{пр} \rightarrow Я_{маш} \rightarrow Я_{рез},$$

где $Я_{чел}$ – формулировка задачи на естественном языке; $Я_{ос}$ – формулировка задачи на языке основных соотношений; $Я_{пр}$ – формулировка задачи на языке программирования; $Я_{маш}$ – формулировка задачи на машинном языке; $Я_{рез}$ – формулировка задачи на языке результата в виде графиков, таблиц, изображений, текстов, звуков и т. п. К сожалению, для большинства задач имеется только формулировка на естественном языке, большинство задач плохо формализованы. Поэтому актуальным является переход от описания на естественном языке на язык основных соотношений, лингво-комбинаторное моделирование является одним из способов такой формализации[1]. В результате такой формализации порождаются рекурсивные структуры со структурированной неопределенностью. Таким образом, рекурсивная структура машин должна включать три составляющих – явления, смыслы и структурированную неопределенность, которые наличествуют в любой задаче.

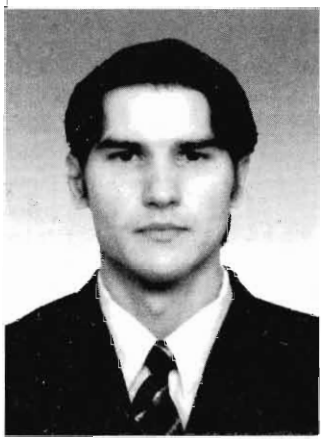
В заключение я призываю высказаться на страницах этого журнала других участников пионерского проекта по рекурсивным машинам, чтобы многогранно осветить проблему.

М. Б. Игнатьев

[1] Игнатьев М. Б. Лингво-комбинаторное моделирование плохо формализованных систем // Информационно-управляющие системы, 2003. – № 6. – С. 34–37.

СВЕДЕНИЯ ОБ АВТОРАХ

АЛЕКСЕЕВ Константин Владимирович



Аспирант Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 2002 году окончил Санкт-Петербургский государственный университет аэрокосмического приборостроения.
Является автором четырех научных публикаций.
Область научных интересов – компьютерная графика, технологии мультимедиа и виртуальных миров.

БЕСТУГИН Александр Роальдович

Доцент кафедры прикладной математики Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 1982 году окончил математико-механический факультет Ленинградского государственного университета им. А. А. Жданова.
В 2000 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором более 50 научных публикаций и соавтором одной монографии.
Область научных интересов – системы управления и связи.

БУБЛИКОВ Андрей Борисович

Ассистент кафедры вычислительных систем и сетей Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 2001 году окончил Санкт-Петербургский государственный университет аэрокосмического приборостроения по специальности "Вычислительные машины, комплексы, системы и сети".
Является автором пяти научных публикаций.
Область научных интересов – защита потоковой информации в сетях общего доступа.

БУХАРЦЕВ Михаил Николаевич

Профессор, главный научный сотрудник Первого Центрального научно-исследовательского института Министерства обороны Российской Федерации.
В 1962 году окончил Высшее военно-морское училище им. М. В. Фрунзе, в 1972 году – Военно-морскую академию.
В 1987 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 80 научных публикаций.
Область научных интересов – исследовательское проектирование.

ДЕХКАНБАЕВ Дмитрий Саттаркулович

Инженер кафедры вычислительных систем и сетей Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 2001 году окончил Санкт-Петербургский государственный университет аэрокосмического приборостроения.
Является автором шести научных публикаций.
Область научных интересов – численные методы моделирования и исследования точечных фракталов, параллельные вычисления.

КОРНЕЕВ Георгий Александрович

Аспирант Санкт-Петербургского государственного университета информационных технологий, механики и оптики.
В 2004 году окончил Санкт-Петербургский государственный университет информационных технологий, механики и оптики.
Является автором шести научных публикаций.
Область научных интересов – технологии программирования, теория программирования, конечные автоматы, объектно-ориентированное проектирование.

КОРШУНОВ Сергей Валерьевич

Доцент, проректор по учебно-методической работе Московского государственного технического университета им. Н. Э. Баумана, действительный член Международной академии открытого образования и Академии проблем качества, член-корреспондент Российской Академии естественных наук, лауреат премии правительства в области образования за 2001 г.
В 1976 году окончил Московское высшее техническое училище им. Н.Э. Баумана.
В 1981 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором более100 научных публикаций.
Область научных интересов – проектирование и испытания автономных приборных устройств.

РОЗОВ Алексей Константинович

Старший научный сотрудник Военно-морской академии им. Н. Г. Кузнецова.
В 1947 году окончил Высшее инженерно-техническое училище ВМФ по специальности инженер-электромеханик.
В 1968 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором 26 научных публикаций.
Область научных интересов – применение статистических методов в задачах обнаружения, классификации и оценивания сигналов.

ФЕДОРОВ Игорь Борисович

Профессор, ректор Московского государственного технического университета им. Н. Э. Баумана, заведующий кафедрой радиоэлектронных систем и устройств, заслуженный деятель науки и техники РФ, член-корреспондент Российской Академии наук, президент Ассоциации технических университетов, действительный член Российской Академии естественных наук и ряда других академий.
В 1963 году окончил Московское высшее техническое училище им. Н. Э. Баумана.
В 1983 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 200 научных публикаций.
Область научных интересов – радиолокационная техника.

РЕШЕТНИКОВА Нина Николаевна

Доцент кафедры вычислительных систем и сетей Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 1979 году окончила Ленинградский институт авиационного приборостроения.
В 1991 году защитила диссертацию на соискание ученой степени кандидата технических наук.
Является автором более 100 научных публикаций.
Область научных интересов – компьютерная графика, технологии мультимедиа и виртуальных миров.

СОВЕТОВ Борис Яковлевич

Профессор, заведующий кафедрой автоматизированных систем обработки информации и управления Санкт-Петербургского государственного электротехнического университета.
Заслуженный деятель науки и техники РФ, действительный член Российской академии образования.
В 1960 году окончил Ленинградский электротехнический институт им. В. И. Ульянова (Ленина).
В 1971 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 400 научных публикаций.
Область научных интересов – информатика и автоматизированное управление.

ШАЛЫТО Анатолий Абрамович

Заведующий кафедрой технологий программирования Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Ученый секретарь НПО "Аврора".
В 1971 году окончил Ленинградский электротехнический институт по специальности "Автоматика и телемеханика".
В 1999 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 250 научных публикаций, трех монографий и 70 изобретений.
Член редакционной коллегии журнала "Информационно-управляющие системы".
Области научных интересов – системы логического управления, автоматное программирование.

ШАМГУНОВ Никита Назимович

Аспирант Санкт-Петербургского государственного университета информационных технологий, механики и оптики.
В 2001 году окончил Уральский государственный университет.
Является автором пяти научных публикаций.
Область научных интересов – технологии программирования, теория программирования, конечные автоматы, объектно-ориентированное проектирование.

ЯКОВЛЕВ Сергей Алексеевич

Профессор кафедры автоматизированных систем обработки информации и управления Санкт-Петербургского государственного электротехнического университета. Действительный член Международной академии информатизации.
В 1970 году окончил Санкт-Петербургский государственный электротехнический университет.
В 1989 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 220 научных и учебно-методических публикаций, в том числе четырех монографий и 25 учебников и учебных пособий.
Область научных интересов – имитационное моделирование, сети интегрального обслуживания, интеллектуализация информационных систем.

ПАМЯТКА ДЛЯ АВТОРОВ

Поступающие в редакцию статьи проходят обязательное рецензирование.

При наличии положительной рецензии статья редактируется и рассматривается редакционной коллегией. Принятая в печать статья направляется автору для согласования редакторских правок. После согласования автор представляет в редакцию окончательный вариант текста статьи, а также фотографию и краткое изложение сведений о себе.

Процедуры согласования текста статьи, предоставления фото (размером 4×5,5 см) и сведений об авторе могут осуществляться как непосредственно в редакции, так и по e-mail (электронный вариант фото в виде файла *.tif, *.jpg с разрешением 300 dpi).

При отклонении статьи редакция представляет автору мотивированное заключение и рецензию. При необходимости доработать статью — рецензию.

Редакция журнала напоминает, что ответственность за подбор, достоверность и точность фактов, экономико-статистических и технических показателей, собственных имен и прочих сведений, а также за то, что в материалах не содержится сведений, не подлежащих открытой публикации, несут авторы публикуемых в журнале материалов и рекламодатели.

УДК 681.516.7.015.2

Классификация морских объектов

Розов А. К., Бухарцев М. Н. – Информационно-управляющие системы, 2004. – № 5. – С. 2–6.

Классификация морских объектов может быть осуществлена с использованием аппарата стохастических дифференциальных уравнений. Полученные в результате решения уравнений апостериорные вероятности гипотез находят применение в алгоритме принятия решений. Приводятся примеры, иллюстрирующие процедуру классификации.

Список лит.: 5 назв.

UDK 681.516.7.015.2

Classification of sea objects

Rozov A. K., Buharcev M. N. – IUS, 2004. – N 5. – P. 2–6.

Classification of sea objects can be carried out by using stochastic difference equation. Inverse probabilities obtained as solutions of these equations are applied to the decision-making algorithms. Examples of classification are provided.

Refs: 5 titles.

УДК 681.3:519.2

Уменьшение временно́й сложности алгоритмов вычисления фрактальной размерности трехмерных точечных фракталов

Дехканбаев Д. С. – Информационно-управляющие системы, 2004. – № 5. – С. 7–12.

Рассматриваются методы вычисления фрактальной размерности трехмерных точечных фракталов. Получены оценки временно́й сложности алгоритмов и времени выполнения программ. Предлагается модификация алгоритмов для уменьшения временно́й сложности и распараллеливание программ для уменьшения времени выполнения.

Список лит.: 12 назв.

UDK 681.3:519.2

Reducing of time complexity of algorithms for fractal dimension calculation of 3D point-like structures

Dehkanbaev D. S. – IUS, 2004. – N 5. – P. 7–12.

The methods for fractal dimension calculation of 3D point-like structures are considered. Estimate of time complexity of algorithms and run time of programs are derivated. Modifications of algorithms and scheme of parallelism are offered. Time complexity of algorithms and run time of programs are reduced.

Refs: 12 titles.

УДК 681.3.06

State Machine – новый паттерн объектно-ориентированного проектирования

Шамгунов Н. Н., Корнеев Г. А., Шалыто А. А. – Информационно-управляющие системы, 2004. – № 5. – С. 13–25.

В статье предлагается новый паттерн объектно-ориентированного проектирования, названный *State Machine*. Этот паттерн расширяет возможности паттерна State, предназначенного для реализации объектов, поведение которых зависит от их состояния. Также предложено использовать события для уведомления об изменении состояния. Это позволяет проектировать объекты такого рода из независимых друг от друга классов. Предлагаемый паттерн по сравнению с паттерном *State* лучше приспособлен для повторного использования входящих в него классов.

Список лит.: 29 назв.

UDK 681.3.06

State Machine – a new design pattern

Shamgunov N. N., Korneev G. A., Shalyto A. A. – IUS, 2004. – N 5. – P. 13–25.

This paper presents a new pattern for object-oriented design – *State Machine*. This pattern extends capabilities of State design pattern. These patterns allow an object to alter its behavior when its internal state changes. Introduced event-driven approach allows to decrease coupling. Thus automaton could be constructed from independent state classes. The classes designed with *State Machine* pattern are more reusable than *State* pattern.

Refs: 29 titles.

УДК 681.325.5

О свойствах двоичных матриц, используемых в алгоритмах кодирования информации булевыми преобразованиями

Бубликов А. Б. – Информационно-управляющие системы, 2004. – № 5. – С. 26–27.

Формулируются требования к свойствам двоичных матриц, используемых для кодирования потоковой информации на основе булевых преобразований.

Список лит.: 2 назв.

UDK 681.325.5

About properties of the binary matrixes used in algorithms of information coding with boolean transformations

Bublikov A. B. – IUS, 2004. – N 5. – P. 26–27.

Requirements to properties of the binary matrixes used for coding of stream information with boolean transformations are formulated.

Refs: 2 titles.

УДК 681.3

3D интерактивная модель наблюдений астероидов. Позиционирование телескопа и ПЗС-наблюдений

Алексеев К. В., Решетникова Н. Н. – Информационно-управляющие системы, 2004. – № 5. – С. 28–37.

В статье рассматриваются методы наблюдения малых планет из главного пояса Солнечной системы, астероидов АСЗ, объектов искусственного происхождения в околоземном космическом пространстве с помощью наземных ПЗС-телескопов. Эти методы используются в качестве математической основы для построения интерактивной трехмерной модели процесса наблюдения, визуализация которого позволит повысить точность реальных наблюдений с целью уточнения элементов орбит и параметров указанных объектов.

Список лит.: 13 назв.

UDK 681.3

3D interactive model of asteroids observation. Positioning of telescope and CCD observations

Alekseev K. V., Reshetnikova N. N. – IUS, 2004. – N 5. – P. 28–37.

This article is devoted to the Solar system's minor planets, asteroids, near Earth space artifical satelites, observational methods by using land-based CCD telescopes. These methods are using as a mathematical base of interactive 3D model processing, and it's visualisation is to organize real observations to improve orbit of the elements and the of pointed objects.

Refs: 13 titles.

УДК 681.327.8

Перспективы подготовки кадров по направлению «Информационные системы»

Федоров И. Б., Коршунов С. В., Советов Б. Я. – Информационно-управляющие системы, 2004. – № 5. – С. 38–43.

В статье рассматривается проблема подготовки разработчиков информационных технологий в условиях перехода к информационному обществу. Приводится содержание новых специальностей в области информационных систем.

Список лит.: 2 назв.

UDK 681.327.8

The perspectives of specialists training in the sphere of information systems

Fedorov I. B., Korshunov S. V., Sovetov B. Y. – IUS, 2004. – N 5. – P. 38–43.

The article deals with the issue of training information technology experts for condition towards an information society. Contents of new speciality in the sphere of information systems is given.

Refs: 2 titles.

УДК 681.387.8

Методические основы использования имитационного моделирования в учебном процессе при подготовке по направлению 654700 – информационные системы

Яковлев С. А. – Информационно-управляющие системы, 2004. – № 5. – С. 44–49.

Рассматривается машинное моделирование как эффективный инструмент исследования характеристик процесса функционирования информационно-управляющих систем реального времени как на этапе их проектирования, так и эксплуатации. Особое внимание уделяется задаче построения прикладной теории эволюционного и десиженсного моделирования таких систем.

Список лит.: 5 назв.

UDK 681.387.8

Methodical bases of use of simulation in educational process by preparation on a direction 654700 – Information systems

Ykovlev S. A. – IUS, 2004. – N 5. – P. 44–49.

Machine modelling is considered as the effective tool of research of characteristics of process of functioning is information – managing systems of real time as at a stage of their designing, and operation. The special attention is given a problem of construction of the applied theory evolutionary and designation modelling of such systems.

Refs: 5 titles.

УДК 621391.28:518.5

Иерархия протоколов для технологии ATM

Бестугин А. Р. – Информационно-управляющие системы, 2004. – № 5. – С. 50–53.

В статье рассматривается текущее состояние эталонной модели иерархии протоколов для технологии ATM.

Список лит.: 5 назв.

UDK 621391.28:518.5

Hierarchy of protocols for ATM- technology.

Bestugin A. R. – IUS, 2004. – N 5. – P. 50–53.

The current reference model for ATM is considered.

Refs: 5 titles.

Ю. П. Иванов, В. Г. Никитин, В. Ю. Чернов

Контроль и диагностика измерительно-вычислительных комплексов: Учеб. пособие / СПб.: СПбГУАП, 2004. – 98 с.: ил. ISBN 5-8088-0114-1

Изложены основные понятия, задачи, методы и способы контроля и диагностики технического состояния измерительно-вычислительных комплексов летательных аппаратов. Описаны обобщенная структура и основные характеристики систем контроля, основные показатели достоверности контроля и ее составляющие, а также выбор допусков на параметры контроля и методы принятия решений в процессе контроля. Рассмотрены методы, аппаратные и программные средства цифровых измерительно-вычислительных комплексов с помощью систем встроенного контроля и диагностики.

Предназначено для студентов, обучающихся по специальностям 190300 «Авиационные приборы и измерительно-вычислительные комплексы», 131000 «Техническая эксплуатация авиационных электросистем и пилотажно-навигационных комплексов», а также магистров по направлению 5515 «Приборостроение» и бакалавров по направлению 5520 «Эксплуатация авиационной и космической техники».

Уважаемые авторы журнала

«ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ»!

При подготовке рукописей статей редакция просит вас руководствоваться следующими рекомендациями:

Объем статьи (текст, таблицы, иллюстрации и библиография) не должен превышать эквивалента в 20 страниц, напечатанных на бумаге формата А4 на одной стороне через 1,5 интервала Word (не более 70 знаков в строке) шрифтом Times New Roman размером 13.

Обязательными элементами оформления статьи являются: индекс УДК, заглавие, инициалы и фамилия автора (авторов), ученая степень, полное название организации, аннотация (5–10 строк) на русском и английском языках.

В редакцию предоставляются:

- отпечатанный (формат А4) текст статьи, подписанный всеми авторами с указанием даты представления, и иллюстрации, пронумерованные с подрисуночными подписями (в двух экземплярах);
- статья в виде файла Microsoft Word или Adobe PageMaker (шрифты Times New Roman, Arial, тексты программ – Courier New) на дискетах 1,44Mb или CD;
- иллюстрации в текст не заверстываются и предоставляются отдельными исходными файлами, поддающимися редактированию:
  растровые – в формате *.tif, *.bmp, *.jpg, *.gif, *.png с максимальным разрешением,
  векторные – в формате *.doc, *.vsd, *.cdr, *.ai, *.p65, *.pmd с теми же шрифтами;
- формулы в формульном редакторе, желательно латиницей;
- аннотация (5–10 строк) на русском и английском языках;
- название статьи, фамилия, имя и отчество автора на английском языке;
- сведения об авторе (фамилия, имя, отчество, место работы, должность, год окончания и наименование вуза, год защиты диссертации, количество научных публикаций, область научных интересов, домашний и служебный адреса и телефоны, e-mail, факс), фото автора (можно в электронном виде с разрешением на 300 точек при размере 40x55 мм);
- экспертное заключение (при необходимости).

Список литературы составляется по порядку ссылок в тексте и оформляется следующим образом:

- для книг и сборников — фамилия и инициалы авторов, полное название книги (сборника), город, издательство, год, общее число страниц;
- для журнальных статей — фамилия и инициалы авторов, полное название статьи, название журнала, год издания, номер журнала, номера страниц;
- ссылки на иностранную литературу следует давать на языке оригинала без сокращений.

Адрес редакции:
190000, Санкт-Петербург, Б. Морская ул., 67,
Факс: (812) 313-70-18
Тел.: (812) 313-70-88, 110-66-42
E-mail: ius@aanet.ru

Научно-исследовательская лаборатория Объектно-Ориентированных Геоинформационных Систем (НИЛ ООГИС) образована при Санкт-Петербургском институте информатики и автоматизации Российской Академии Наук (СПИИРАН) в 2000 году.

Это самое молодое и динамично развивающееся подразделение института. На сегодняшний день в лаборатории работает 10 сотрудников, из которых 1 доктор и 4 кандидата технических наук. Лаборатория оснащена новейшим современным оборудованием для проведения фундаментальных исследований и комплексных наукоемких разработок по созданию, развитию и сопровождению различных программных систем на базе геоинформационных технологий.

За время существования НИЛ ООГИС сотрудники лаборатории выполнили, помимо плановых институтских исследований, комплексные проекты по заказу Военно-морского флота РФ, Главного управления Навигации и Океанографии Министерства Обороны РФ, Международного Научно-Технического Центра РАН, Европейской штаб-квартиры научной лаборатории ВВС США. На ряд инновационных технологий, разработанных в группе и успешно внедренных в практику, получены патенты РФ.

Научно-исследовательская лаборатория Объектно-Ориентированных Геоинформационных Систем Санкт-Петербургского института информатики и автоматизации Российской Академии Наук открыта для широкого взаимовыгодного сотрудничества и готова прийти на помощь Вам в разрешении Ваших проблем.

199178, Санкт-Петербург,
14-я линия Васильевского острова, дом 39,
СПИИРАН, НИЛ ООГИС
Контактный адрес:
(E-mail: popovich@mail.iias.spb.su)
http://NIGGIS.iias.spb.su