Институт проблем управления
им. В.А. Трапезникова РАН

УПРАВЛЕНИЕ БОЛЬШИМИ СИСТЕМАМИ

Выпуск 66
Март 2017

СБОРНИК ТРУДОВ

ISSN 1819-2467

Регистрационный номер Эл №ФС77-44158 от 09 марта 2011 г.

Москва – 2017

РОССИЙСКАЯ АКАДЕМИЯ НАУК
Институт проблем управления
им. В.А. Трапезникова

УПРАВЛЕНИЕ БОЛЬШИМИ СИСТЕМАМИ

СБОРНИК ТРУДОВ

Выпуск 66

Москва – 2017

УДК 519 ISSN 1819-2467
ББК 32.81
У 67
Управление большими системами / Сборник трудов. Выпуск 66. М.: ИПУ РАН, 2017. – 197 с. **Дата опубликования:** 31.03.2017.

Адрес редакции: 117997, г. Москва, ул. Профсоюзная, д. 65.
Адрес в интернете: ubs.mtas.ru.

СОДЕРЖАНИЕ

УДК 004.421
ББК 32.97

АЛГОРИТМЫ ИНТЕРПРЕТАЦИИ ПРОСОДИЧЕСКИХ ПРИЗНАКОВ РЕЧИ ПРИ ЕЕ ОБРАБОТКЕ НИЗКОСКОРОСТНЫМИ КОДЕКАМИ

Бессонов М. А.[1]**,**
(*ФГАОУ ВО «Российский университет дружбы народов», Москва*)
Фархадов М. П.[2]
(*ФГБУН Институт проблем управления им. В.А. Трапезникова РАН, Москва*)

В рамках решения задачи определения языка аудиосообщения на основе просодического подхода предложены два алгоритма интерпретации просодических признаков речи и методика их использования – алгоритм на основе широких фонетических категорий и алгоритм на основе кросскорреляционной функции от мелодики речевого сигнала и последовательности кратковременных энергий. Проводится экспериментальная оценка алгоритмов. В качестве решающего правила используются нейронные сети.

Ключевые слова: идентификация языка, нейронные сети, просодические признаки речи, широкие фонетические категории.

1. Введение

Определение языка аудиосообщения является актуальной задачей в связи с развитием множества сетевых человеко-машинных интерфейсов, при этом в данные системы закладывается поддержка множества языков. Выделяют четыре подхода ее

[1] *Максим Александрович Бессонов*, *аспирант* (*bessonovma@gmail.com*).
[2] *Маис Паша Оглы Фархадов*, *д.т.н., с.н.с.* (*mais@ipu.ru*).

решения – акустический, фонотактический, лексический и просодический. Так или иначе, первые три строятся на одних параметрах речевого сигнала – акустических – мел-кепстральных коэффициентах, смещенных мел-кепстральных коэффициентах и т.д. Просодический подход [5–9] использует такие параметра как мелодика речи, ритм, тембр и т.д. Просодические параметры сложно поддаются описанию и математической интерпретации. И поэтому в данной статье предлагаются два алгоритма для комплексного описания просодических признаков речи с целью их использования в системах автоматического определения языка аудиосообщения. Первый алгоритм основан на широких фонетических категориях [4], второй на кросскорреляционной функции мелодии речи и последовательности кратковременных энергий.

Данные алгоритмы отличаются от известных тем, что они применимы для определения языка аудиосообщения по речи, прошедшей через низкоскоростные кодеки. Это обусловлено тем, что в низкоскоростных кодеках в канал связи передаются такие параметры, как частота основного тона, сигнал тон-шум и усиление на квазипериодических отрезках.

2. Алгоритмы интерпретации просодических признаков

2.1. АЛГОРИТМ НА ОСНОВЕ ШИРОКИХ ФОНЕТИЧЕСКИХ КАТЕГОРИЙ

Пусть множество $L = \{L_1, L_2, \ldots, L_N\}$ есть множество языков, на котором осуществляется процедура определения языка аудиосообщения, где N – общее число языков. Пусть каждый язык L_i представляется множеством аудиозаписей различных дикторов этого языка $L_i = \{l_1, l_2, \ldots, l_{Mi}\}$, где M_i – общее число аудиозаписей языка L_i.

Аудиозапись разбивается на квазистационарные сегменты $s_i(m)$ длительностью K отсчетов, где i – номер сегмента речевого сигнала, $i = 1, 2, \ldots, P$, P – общее число сегментов в аудиозаписи речевого сигнала, $m = 1, \ldots, K-1$. На каждом сегменте i вычисляется признак в соответствии с природой сегмента – вокализованный, невокализованный или пауза

(1) $A_i = T\left(s_i\left(m\right)\right), i = 1, 2, ..., P$,

где T – операция вычисления типа сегмента, а также кратковременная энергия сегмента

(2) $E_{k_i} = E\left(s_i\left(m\right)\right), i = 1, 2, ..., P$,

где E – операция вычисления кратковременной энергии сегмента. При работе алгоритма без восстановления исходной формы речевого сигнала параметры A_i и Ek_i берутся из кадров вокодерной передачи. Соответственно формируются последовательности $\vec{A} = \left(A_1, A_2, ..., A_P\right)$ и $\vec{E}k = \left(Ek_1, Ek_2, ..., Ek_P\right)$. Если сегмент классифицирован как пауза, то $A_i = 0$, если классифицирован как невокализованный, то $A_i = 1$. На каждом вокализованном сегменте вычисляется частота основного тона

(3) $F0_i = F\left(s_i\left(m\right)\right), i = 1, 2, ..., P$,

где F – операция вычисления частоты основного тона, и формируется последовательность $\overrightarrow{F0} = \left(F0_1, F0_2, ..., F0_P\right)$. При работе алгоритма без восстановления исходной формы речевого сигнала параметр $F0i$ берется их кадров вокодерной передачи. Диапазон изменения частоты основного тона аудиозаписей разбивается на 5 интервалов. Для вокализованных сегментов каждый сегмент обозначается цифрой в соответствии с тем, в какой интервал ЧОТ попадает значение частоты на данном сегменте

(4) $F0_{u_i} = UF\left(\overrightarrow{F0}\right), i = 1, 2, ..., P$,

где $F0u_i$ – уровень ЧОТ, UF – операция вычисления диапазона изменения ЧОТ и кодирования каждого сегмента цифровым обозначением, формируется последовательность $\overrightarrow{F0u} = \left(F0u_1, F0u_2, ..., F0u_P\right)$ – последовательность из значений ЧОТ на сегментах аудиозаписи. Далее вычисляются сегменты возрастания/убывания кратковременной энергии речевого сигнала

(5) $E_{u_i} = UE\left(\overrightarrow{E_k}\right), i = 1, 2, ..., P$,

Кодирующиеся Eu_i = (+/–)1 в зависимости от того, возрастает или убывает энергия соответственно, где UE – операция вычис-

ления возрастания/убывания кратковременной энергии речевого сигнала. Формируется последовательность $\overrightarrow{Eu} = (Eu_1, Eu_2, ..., Eu_P)$. Если данный сегмент относится к участку убыванию кратковременной энергии, цифровое значение ЧОТ умножается на (–1).

Для определения побочных и главных ударений определяется главный и побочный максимумы ЧОТ на отрезке между двумя паузами. Если положение максимума ЧОТ и кратковременной энергии совпадают во времени и максимальны на отрезке, то этот сегмент принимается за главный максимум, если максимумы во времени не совпадают, то сегмент принимается за побочный максимум $MAX_i = \Theta(\overrightarrow{F0u}, \overrightarrow{Eu})$, где Θ – операция определения главного и побочного максимумов ЧОТ и кратковременной энергии. Формируется последовательность

(6) $\overrightarrow{MAX} = (MAX_1, MAX_2, ..., MAX_P)$.

Таким образом, окончательная последовательность широких фонетических категорий (ШФК) аудиозаписи $\overrightarrow{X} = (X_1, X_2, ..., X_P)$ состоит из элементов X_i, где

(7) $$X_i = \begin{cases} 0, \text{ если } A_i - \text{пауза}, \\ 1, \text{ если } A_i - \text{невокализованный}, \\ 2, \text{ если } F0_{u_i} - \text{уровень 1}, \\ -2, \text{ если } F0_{u_i} - \text{уровень 1}, E_{u_i} = -1, \\ 3, \text{ если } F0_{u_i} - \text{уровень 2}, \\ -3, \text{ если } F0_{u_i} - \text{уровень 2}, E_{u_i} = -1, \\ 4, \text{ если } F0_{u_i} - \text{уровень 3}, \\ -4, \text{ если } F0_{u_i} - \text{уровень 3}, E_{u_i} = -1, \\ 5, \text{ если } F0_{u_i} - \text{уровень 4}, \\ -5, \text{ если } F0_{u_i} - \text{уровень 4}, E_{u_i} = -1, \\ 6, \text{ если } F0_{u_i} - \text{уровень 5}, \\ -6, \text{ если } F0_{u_i} - \text{уровень 5}, E_{u_i} = -1, \\ 7, \text{ если } MAX_i - \text{побочный максимум}, \\ 8, \text{ если } MAX_i - \text{главный максимум}. \end{cases}$$

На рис. 1 и рис. 2 приведены блок-схемы алгоритма кодирования сегментов речевого сигнала.

По последовательности широких фонетических категорий $\overrightarrow{X}$ вычисляется автокорреляционная функция $\vec{R} = \Psi(\overrightarrow{X})$, где Ψ – операция вычисления автокорреляционной функции.

В случае работы алгоритмов без восстановления исходной формы речевого сигнала значения ЧОТ берутся из кадров вокодерной передачи. В случае работы алгоритма с восстановлением исходной формы речевого сигнала требуется выбор алгоритма оценки частоты основного тона.

Для определения ЧОТ существуют различные алгоритмы [2]. В данной работе были проведены испытания готовых алгоритмов, реализуюших определение ЧОТ по автокорреляционной функции (АКФ) – алгоритм SIFT, по кратковременной функции средней разности (КФСР) – алгоритм AMDF, а также алгоритм оценки ЧОТ из алгоритма кодирования речи MELP. Проценты отрезков речевого сигнала с показателями *P*(ОТ) – правильно определенным основным тоном, *P*(НВ/В) – принятия вокализованного отрезка за невокализованный, *P*(В/НВ) – принятия невокализованного за вокализованный приведены в таблице 1.

Таблица 1.Показатели правильности оценки основного тона

Алгоритм	SIFT	AMDF	MELP
P(ОТ), %	87±1	89±1	95±1,5
P(НВ/В), %	7±1	6±1	3±0,5
P(В/НВ), %	0,5	0,5	0,5

Как следует из экспериментального сравнения представленных алгоритмов, наилучшим оказался MELP. Данный алгоритм был выбран для оценки основного тона.

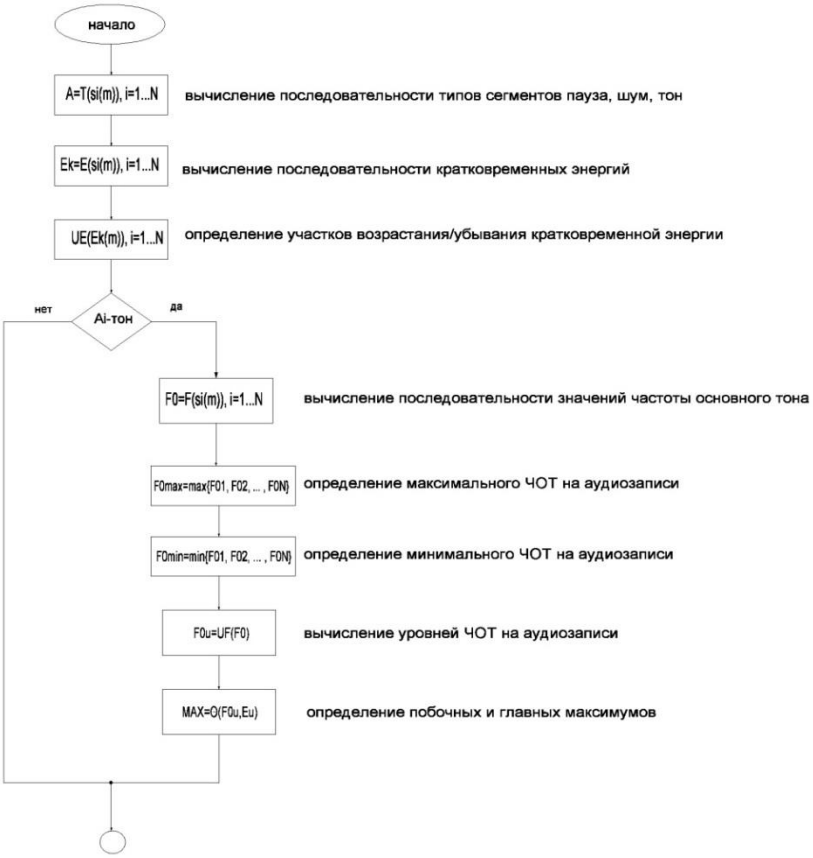

Рис. 1. Блок-схема алгоритма кодирования сегментов речевого сигнала

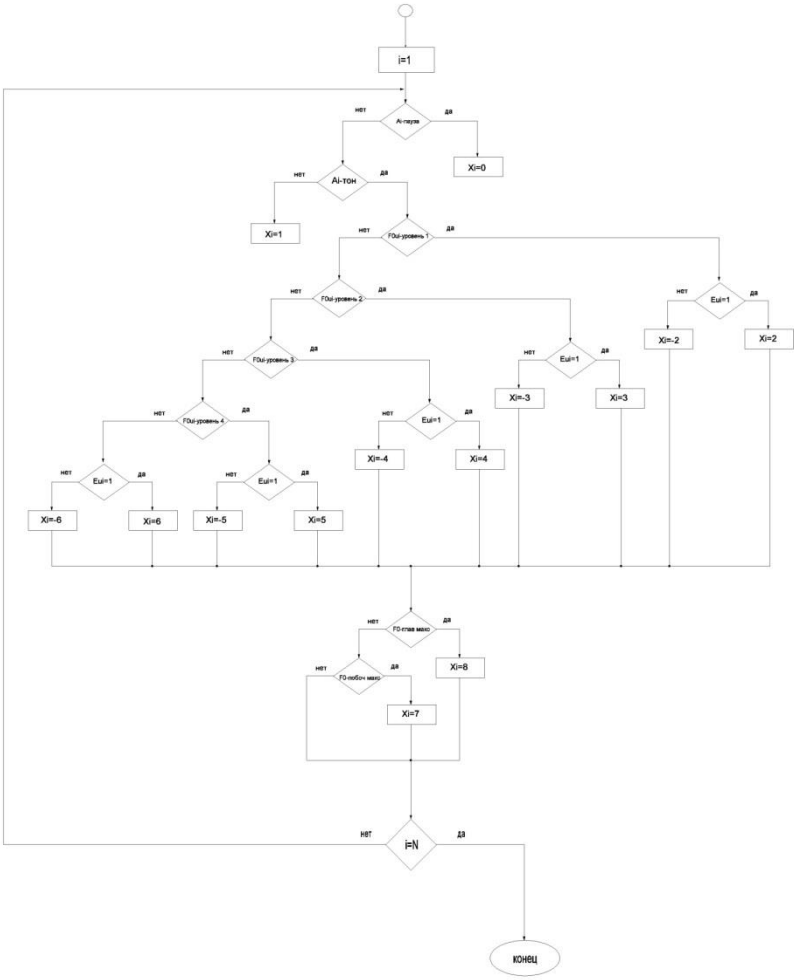

Рис. 2. Блок-схема алгоритма кодирования сегментов речевого сигнала (продолжение)

2.2. АЛГОРИТМ НА ОСНОВЕ КРОССКОРРЕЛЯЦИОННОЙ ФУНКЦИИ МЕЛОДИИ ОСНОВНОГО ТОНА И ПОСЛЕДОВАТЕЛЬНОСТИ КРАТКОВРЕМЕННЫХ ЭНЕРГИЙ

Для реализации просодической классификации предлагается использование кросскорреляционной функции мелодии основного тона и последовательности кратковременных энергий сигналов аудиозаписей. Аудиозапись разбивается на квазистационарные сегменты $s_i(m)$ длительностью K отсчетов, где i – номер сегмента речевого сигнала, $i = 1, 2, \ldots, P$, P – общее число сегментов в аудиозаписи речевого сигнала, $m = 1, \ldots, K - 1$. На каждом сегменте i вычисляется признак в соответствии с природой сегмента – вокализованный, невокализованный или пауза

(8) $A_i = T\left(s_i\left(m\right)\right), i = 1, 2, ..., P$,

где T – операция вычисления типа сегмента, а также кратковременная энергия сегмента

(9) $E_{k_i} = E\left(s_i\left(m\right)\right), i = 1, 2, ..., P$,

где E– операция вычисления кратковременной энергии сегмента. Соответственно формируются последовательности $\vec{A} = \left(A_1, A_2, ..., A_P\right)$ и $\vec{E}k = \left(Ek_1, Ek_2, ..., Ek_P\right)$. Если сегмент классифицирован как пауза, то $A_i = 0$, если классифицирован как невокализованный, то $A_i = 1$. На каждом вокализованном сегменте вычисляется частота основного тона (ЧОТ)

(10) $F0_i = F\left(s_i\left(m\right)\right), i = 1, 2, ..., P$,

где F – операция вычисления частоты основного тона, и формируется последовательность $\overrightarrow{F0} = \left(F0_1, F0_2, ..., F0_P\right)$.

При работе алгоритма без восстановления исходной формы речевого сигнала параметры A_i и Ek_i, $F0$i берутся из кадров вокодерной передачи.

По последовательности значений частоты основного тона и последовательности кратковременных энергий вычисляется их кросс-корреляционная функция

(11) $\vec{B} = \Phi(\overrightarrow{F0}, \overrightarrow{Ek})$,

где Ф – операция вычисления кросскорреляционной функции мелодии основного тона и последовательности кратковременных энергий. Вектор значений кросскорреляционной функции последовательности широких фонетических категорий подается на вход нейронной сети, которая принимается решение по отнесению данного вектора к какой-либо группе языков.

Алгоритм вычисления признаков представлен на рис. 3

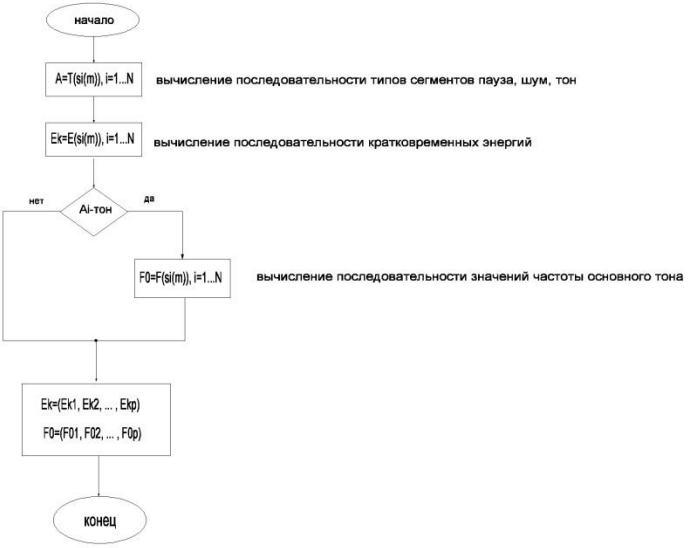

Рис. 3. Блок-схема алгоритма кодирования сегментов речевого сигнала

3. Методика применения алгоритмов интерпретации просодических признаков в задаче определения языка аудиосообщения

Для применения указанных алгоритмов была разработана следующая методика. Она заключается в последовательности ряда этапов.

Этап 1. Формирование обучающей речевой базы данных. Обучающая база данных должна удовлетворять следующим условиям: если N – общее число языков, d_m^i – число дикторов мужского пола языка i, d_f^i – число дикторов женского пола языка i, то $V_i(d_m^i, d_f^i) = V_j(d_m^j, d_f^j)$, где i, j – номера языков, $i, j = 1, \ldots, N$. То есть все возрастные группы должны быть представлены в равной пропорции дикторами мужского и женского пола, объемы речевых данных дикторов различных возрастных групп должны быть одинаковы. Объем речевых данных должен быть достаточен со статистической точки зрения для описания всех вариативностей произношения на данном языке. Общие объемы речевых баз по языкам должны быть равны.

Шаг 1. Получение от источника аудиосообщения в цифровом виде $S_t(f_d, m, p, f_r)$ с параметрами – формат f_r = «wav», частота дискретизации f_d = 8 кГц, режим m = моно, p = 16 бит, t – номер аудиосообщения.

Шаг 2. Фильтрация аудиосообщения $S_t(f_d, m, p, f_r)$ – удаление посторонних шумов. Получение фильтрованного аудиосообщения $S_t^f(f_d, m, p, f_r) = P[S_t(f_d, m, p, f_r)]$, где P – операция фильтрации.

Шаг 3. Формирование обучающих и тестовых данных. Для каждого языка L_i формируется база аудиосообщений $Z_{Li}\{S_1^f{}_{Li}(f_d, m, p, f_r), \quad S_2^f{}_{Li}(f_d, m, p, f_r), \ldots, S_{Mi}^f{}_{Li}(f_d, m, p, f_r)\}$, где Mi – общее число аудиосообщений языка Li.

Общая база аудиосообщений $Z = \{Z_{L1}, Z_{L2}, \ldots, Z_{LN}\}$.

Шаг 4. Обработка всех аудиосообщений всех языков заданным вокодером.

$Z^{vok} = \mathrm{VOK}(Z)$, где VOK – операция обработки базы аудиосообщений вокодером, $Z^{vok} = \{Z^{vok}_{L1}, Z^{vok}_{L2}, \ldots, Z^{vok}_{LN}\}$.

Шаг 5. Вычисление параметров из аудиосообщений в соответствии с разработанными алгоритмами – формирование базы параметров $Z^{vok}_{Li}{}^{Mod1} = \mathrm{Mod1}(Z^{vok}_{Li})$, $Z^{vok}_{Li}{}^{Mod2} = \mathrm{Mod2}(Z^{vok}_{Li})$, где Mod1, Mod2 –операции вычисления параметров в соответствии с разработанными алгоритмами описания просодических параметров речи.

Этап 2. Обучение искусственной нейронной сети, в процессе обучения происходит настройка различных параметров нейронной сети. Нейронные сети с различной топологией опи-

сываются различными математическими моделями, поэтому в каждом конкретном случае нейронная сеть будет описываться своей формулой. Для формирования групп языков строятся нейронные сети, число которых равно сочетанию из N по 2.

Этап 3. Тестовая оценка нейронной сети.

Шаг 1. Получение от источника аудиосообщения в цифровом виде $S_t(f_d, m, p, f_r)$ с параметрами: формат f_r = «wav», частота дискретизации f_d = 8 кГц, режим m = моно, p = 16 бит, t – номер аудиосообщения.

Шаг 2. Фильтрация аудиосообщения $S_t(f_d, m, p, f_r)$ – удаление посторонних шумов. Получение фильтрованного аудиосообщения $S_t^f(f_d, m, p, f_r) = P[S_t(f_d, m, p, f_r)]$, где P – операция фильтрации.

Шаг 3. Тестирование нейронных сетей. На вход нейронной сети для каждой пары языков L_i и L_j подаются аудиосообщения языков i и j, на выходе – оценка того, какому языку принадлежит данное аудиосообщение, t – номер аудиосообщения.

$$\hat{L}_t = NET\left(S_t^f\left(f_d, m, p, f_r\right)\right). \quad (12)$$

Шаг 4. Вычисление числа правильно распознанных аудиосообщений в каждой паре языков. Получение вектора $D = (d_{12}, d_{21}, d_{13}, d_{31}, \ldots, d_{N(N-1)}, d_{(N-1)N})$, где d_{ij} – число правильно определенных аудиосообщений для пары языков L_iL_j, $i \neq j$.

Шаг 5. Построение иерархического дерева языков на основе агломеративного иерархического алгоритма

$$\rho_{min}\left(\omega_i, \omega_j\right) = \min_{x_k \in \omega_i, x_l \in \omega_j} d\left(X_k, X_l\right), \quad (13)$$

где ω_i, ω_j – языки L_i и L_j, $\rho(\omega_i, \omega_j)$ – расстояние между L_i и L_j.

На основе иерархического дерева строятся группы языков.

4. Формирование речевой базы данных

Для проведения тестов в данной работе была сформирована база данных аудиозаписей, состав базы указан в таблице 2.

Источник аудиозаписей – каналы интернет вещания – телевидение и радио, т.е. речь, прошедшая обработку различными кодеками.

Таблица 2. Характеристики базы данных для проведения экспериментальной оценки эффективности моделей описания просодических признаков

Язык	Число дикторов	Суммарное время аудиозаписей на каждого диктора, мин	Пол диктора (м-мужской, ж-женский)	Процент обучающей/тестовой выборки, %
Китайский	10	100	5м/5ж	80/20
Английский	10	100	5м/5ж	80/20
Финский	10	100	5м/5ж	80/20
Французский	10	100	5м/5ж	80/20
Немецкий	10	100	5м/5ж	80/20
Японский	10	100	5м/5ж	80/20
Персидский	10	100	5м/5ж	80/20
Португальский	10	100	5м/5ж	80/20
Русский	10	100	5м/5ж	80/20
Испанский	10	100	5м/5ж	80/20

Для исключения влияния базы данных на эксперимент число дикторов по всем языкам выбрано одинаковым, суммарное время аудиозаписей выбрано одинаковым, также одинаков процент обучающей и тестовой выборок. Обучающая и тестовая выборки не перекрываются. Для проведения экспериментов все аудиозаписи обучающей и тестовой выборок разделялись на отрезки по 10 с.

Возрастной состав дикторов определить возможно приблизительно – мужчины и женщины от 20 до 50 лет, объем обучающей выборки – 80% от времени аудиозаписей каждого диктора, объем тестовой выборки – 20% от времени аудиозаписей каждого диктора. Деление аудиозаписей на обучающую и тестовую выборки произведено в случайном порядке.

5. Создание и настройка нейронной сети

Задача распознавания образов в большинстве случаев решается статистическими методами, но в случае речевых данных на различных языках достаточно сложно построить статистическое распределение рассматриваемых параметров, и поэтому в данной работе для классификации отрезков речи применены искусственные нейронные сети.

Как известно, для задач типа классификации число нейронов во входном слое вычисляется исходя из вектора признаков, который подается на вход [3], а число нейронов выходного слоя зависит от того, какая задача решается и какое применяется правило интерпретации выходных значений [3]. Для оценки числа нейронов в скрытых слоях применяют формулу [3]

$$(14)\quad \frac{N_y N_p}{1+log_2\left(N_p\right)} \le N_w \le N_y\left(\frac{N_p}{N_x}+1\right)\left(N_x+N_y+1\right)+N_y,$$

где N_y – размерность выходного вектора нейросети (НС), N_p – число элементов обучающей выборки, N_x – размерность входного вектора, N_w – общее число нейронов.

Выбор класса и архитектуры НС является нетривиальной задачей, для решения которой точных методов не существует [3]. Для выбора числа нейронов выделяют два метода: 1) чем больше нейронов, тем надежнее работа сети; 2) чем больше число нейронов, тем хуже создаваемая нейронная сеть аппроксимирует функцию.

Для реализации классификатора на базе нейронной сети был сделан выбор в пользу пакета MATLAB, который включает в себя функционал по нейронным сетям.

В работе экспериментальные исследования проводились со следующими сетями: сеть Кохонена, каскадная НС, сеть Элмана, многослойный персептрон, сеть Хопфилда, вероятностная сеть, сеть с радиальными базисными функциями RBF, НС встречного распространения – LVQ сети.

Алгоритмы, стандартные в MATLAB, использованные при обучении сетей [1]: квазиньютоновский алгоритм; алгоритм Левенберга-Марквардта с регуляризацией Байеса; метод сопряженных градиентов Флетчера-Ривса; метод сопряженных гради-

ентов Полака-Ривьера; метод сопряженных градиентов Пауэлла-Беаля; базовый метод градиентного спуска; метод градиентного спуска с переменным шагом обучения; алгоритм Левенберга-Марквартда, метод масштабированных сопряженных градиентов; метод градиентного спуска с моментом; метод градиентного спуска с моментом и переменным шагом обучения; метод «One Step Secant»; метод случайных приращений; эластичный алгоритм обратного распространения ошибки.

На первом этапе для построения сокращенных групп из 10 языков эксперименты проводились с отдельной сетью для каждой пары языков, то есть было построено 45 нейронных сетей.

Наилучшие показатели были получены при создании многослойного персептрона, поэтому было принято решение провести более точную настройку данного типа НС.

Но поскольку заранее неизвестно, какой язык подается на вход НС, было принято решение использовать единую архитектуру для сетей каждой пары языков.

6. Оценка алгоритма на основе широких фонетических категорий

Согласно формуле и исходным параметрам для тестирования НС: $N_y = 2$, $N_p = 600$, $N_x = 399$, число нейронов в скрытых слоях $117 \le N_w \le 2015$ для модели ШФК.

Поскольку Nw лежит в пределах от 117 до 2015, то при создании архитектуры НС число нейронов в слое варьировалось от 100 до 2000, соответственно число слоев от 1 (1 слой от 100 до 2000 нейронов) до 20 (20 слоев по 100 нейронов) в следующих конфигурациях: со 100 до 1000 нейронов с шагом в 10 нейронов в слое, с 1000 до 2000 с шагом в 100 нейронов. Максимальное число нейронов в слое 800

При построении различных архитектур многослойного персептрона для 45 пар языков формировался вектор целевых показателей достоверности распознавания $D = (d_{1,2}, d_{2,1}, d_{1,3}, d_{3,1}, d_{i,j}, d_{j,i}, d_{N,N-1}, d_{N-1,N})$, где N – общее число языков в САОЯ. Таким образом, длина вектора $D = 90$. Каждый элемент $d_{i,j}, d_{j,i} = 100$.

Вектор показателей достоверности распознавания $D_k = (d^k_{1,2}, d^k_{2,1}, d^k_{1,3}, d^k_{3,1}, d^k_{i,j}, d^k_{j,i}, d^k_{N,N-1}, d^k_{N-1,N})$ для текущей архитектуры НС имеет также длину 90 и расстояние между D и D_k определяется как

$$(15)\ D_r = \sqrt{\left(d_{1,2} - d^k_{1,2}\right)^2 + \left(d_{2,1} - d^k_{2,1}\right)^2 + \left(d_{i,j} - d^k_{i,j}\right)^2 + \ldots}$$
$$\ldots + \left(d_{j,i} - d^k_{j,i}\right)^2 \ldots + \left(d_{N,N-1} - d^k_{N,N-1}\right)^2 + \left(d_{N-1,N} - d^k_{N-1,N}\right)^2 .$$

Таким образом, тем меньше расстояние D_r, тем лучше настроена НС. В результате исследования D_r колебалась в пределах от 59,1861 до 532,4106. Наилучший показатель D_r = 72,5358 был получен для конфигурации НС – общее число нейронов 1400, 1 слой – 800 нейронов, 2 слой – 600 нейронов. Результаты определения языка представлены в таблице 3.

Таблица 3. Средние значения достоверности определения языка

	китайский	английский	финский	французский	немецкий	японский	персидский	португальский	русский	испанский
китайский		94,5	95,1	96,2	95,9	97,5	96,6	95,2	94,4	97,9
английский	93,8		97,4	92,8	93,8	93,6	98,1	94,5	94,0	97,8
финский	93,8	93,7		93,2	93,4	93,9	93,9	96,1	93,7	94,3
французский	94,2	93,6	93,2		93,9	93,4	94,0	94,8	93,8	94,4
немецкий	94,5	92,6	93,7	92,5		94,6	94,0	97,5	96,3	93,9
японский	83,6	94,1	74,0	98,3	93,3		94,0	84,9	94,4	98,0
персидский	84,4	94,0	74,6	93,3	93,8	83,6		92,7	84,3	93,2
португальский	94,2	93,6	93,5	93,9	94,2	94,5	93,5		93,9	98,4
русский	94,4	95,1	94,1	95,3	93,4	94,0	94,4	94,3		94,5
испанский	93,9	94,3	93,4	93,2	94,2	93,8	94,1	94,5	93,2	

Для группировки языков на группы применим агломеративный иерархический алгоритм. В качестве образов выступают пары языков, в качестве расстояния между образами – средние значения достоверности определения языка в паре при фиксированной вероятности ошибки первого и второго рода. В качестве расстояния между классами используем расстояние по принципу ближайшего соседа:

(16) $\rho_{min}\left(\omega_i,\omega_j\right)=\min\limits_{x_k\in\omega_i,x_l\in\omega_j} d\left(X_k,X_l\right)$,

где ω_i, ω_j – языки L_i и L_j, $\rho(\omega_i, \omega_j)$ - расстояние между L_i и L_j.

Таким образом, получаем граф иерархической классификации. Исходя из полученного графа иерархической классификации, получаем группы схожести языков.

7. Оценка алгоритма на основе кросскорреляционной функции мелодии основного тона и последовательности кратковременных энергий

Согласно формуле и исходным параметрам для тестирования НС: $N_y = 2$, $N_p = 600$, $N_x = 797$, число нейронов в скрытых слоях $117 \le N_w \le 2806$ для модели кросскорреляционной функции от последовательности значений основного тона и кратковременной энергии речевого сигнала.

Поскольку N_w лежит в пределах от 117 до 2806, то при создании архитектуры НС число нейронов в слое варьировалось от 100 до 3000, соответственно число слоев от 1 (1 слой от 100 до 3000 нейронов) до 20 (30 слоев по 100 нейронов) в следующих конфигурациях: со 100 до 1000 нейронов с шагом в 10 нейронов в слое, с 1000 до 3000 с шагом в 100 нейронов. Максимальное число нейронов в слое 800; D_r = 89,1449.

Результаты определения языка представлены в таблице 4.

Таблица 4. Средние значения достоверности определения языка

	китайский	английский	финский	французский	немецкий	японский	персидский	португальский	русский	испанский
китайский		97,7	94,7	92,8	97,8	97,9	93,8	91,7	93,1	92,1
английский	91,2		91,4	92,3	92,9	94,8	92,7	97,7	90,3	92,0
финский	90,9	91,5		95,8	94,7	94,6	95,4	90,9	93,6	95,9
французский	92,1	92,9	92,4		93,9	96,7	97,5	92,1	91,8	91,8
немецкий	92,5	90,2	91,4	90,4		91,8	92,2	92,4	93,0	95,4
японский	80,6	91,8	90,1	82,3	71,9		90,7	90,5	94,7	97,2
персидский	71,1	91,5	82,3	91,6	82,6	78,2		97,5	92,5	91,5
португальский	90,7	91,0	92,0	92,0	93,2	93,4	92,1		94,5	92,5
русский	91,0	91,7	90,6	92,6	92,3	92,4	91,7	91,6		96,2
испанский	90,5	92,9	90,9	92,8	91,2	93,1	91,4	92,1	93,6	

8. Заключение

Целью описанных в статье алгоритмов является комплексное описание просодических признаков речи для того, чтобы эти признаки можно было использовать при специальной обработке данных, в частности в задаче определения языка аудиосообщения. Как следует из приведенных таблиц, описание просодических параметров моделью ШФК дает большую достоверность определения языка, но незначительно в сравнении с кросскорреляционной функцией. Показатель близости текущих результатов определения языка к целевым D_r составил D_r = 72,5358 для модели АКФ от ШФК и D_r = 89,1449 для модели кросскорреляционной функции от последовательности значений основного тона и кратковременной энергии речевого сигнала.

Отличительной особенностью данных алгоритмов является то, что они применимы при определении языка аудиосообщения

по речи, преобразованной вокодерами, без восстановления исходной формы речевого сигнала.

Литература

1. ДЬЯКОНОВ В.П., КРУГЛОВ В.В. *Matlab 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики*. – М.: СОЛОН-ПРЕСС, 2006. – 456 с.
2. ИМАМВЕРДИЕВ Я.Н., СУХОСТАТ Л.В. *Подходы для оценки периода основного тона речевого сигнала в зашумлённой среде* // Речевые технологии. – 2014. – №1-2. – С. 84–102.
3. КОМАРЦОВА Л.Г., МАКСИМОВ А.В. *Нейрокомпьютеры: Учебное пособие для вузов*. – 2-е изд., перераб. и доп. – М.: изд-во МГТУ им. Н.Э. Баумана, 2004. – 400 с.
4. МИЛОШЕНКО А.А. *Разработка методики использования широких фонетических категорий в задачах верификации диктора*: Автореф. дис. канд. техн. наук. – Москва, 2010. – 94 с.
5. AMBIKAIRAJAH E., LI H., WANG L., YIN B. *Language Identification: A Tutorial* // IEEE Circuits and Systems Magazine. – 2011. – Vol. 11, Iss. 2. – P. 82–108.
6. BHATTACHARJEE U., SARMAH K. *Language identification system using MFCC and prosodic features* // Int. Conference on Intelligent Systems and Signal Processing (ISSP), Gujarat, 2013. – P. 194–197.
7. LEE R., LEUNG C.-C., MA B. *Spoken Language Recognition with prosodic features* // IEEE Trans. on Audio, Speech, and Language Processing. – 2013. – Vol. 21, Iss. 9. – P. 1841–1853.
8. MARTINEZ D., JEIDA E., ORTEGA A. *Prosodic features and formant modeling for an ivector-based language recognition system* // Proc. ICASSP, Vancouver, Canada, May 2013. – P. 6847–6851.
9. MARTÍNEZ D., BURGET L., FERRER L., SCHEFFER N. *iVector-based prosodic system for language identification* // IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 2012. – P. 4861–4864.

ALGORITHMS FOR INTERPRETATION OF PROSODIC FEATURES IN LOW-BITRATE SPEECH PROCESSING

Maxim Bessonov, «Russian peoples' friendship university», Moscow, graduate student (bessonovma@gmail.com).
Mais Pasha Farhadov, Institute of Control Sciences of RAS, Moscow, Doctor of Science, Senior Researcher, (Moscow, Profsoyuz naya st., 65, *mais@ipu.ru*).

Abstract: *We study the language identification problem using prosodic features. Prosodic features such as melody, rhythm, timbre and others are difficult to formalize mathematically. Two algorithms for a complex description of prosodic features are proposed in the paper. The first is based on the broad phonetic categories, and the second is based on the cross-correlation of the speech melody and the short-term energy sequence. The fundamental frequency was estimated by MELP algorithm. The performance of the proposed algorithms was evaluated experimentally on a database of speech recordings obtained from Internet and therefore encoded by low-bitrate vocoders. The database includes ten different languages. The proposed algorithms provide a feature description and a multi-layer neural network was used as a language classifier. Both algorithms show satisfactory classification performance, but the broad phonetic categories approach performs slightly better than the cross-correlation function. These algorithms can be applied to a speech signal processed by low-bitrate vocoders without decoding to the original signal.*

Keywords: language identification, neural networks, speech prosodic features, broad phonetic categories.

Статья представлена к публикации членом редакционной коллегии Н.И. Базенковым.

Поступила в редакцию 09.11.2016.
Опубликована 31.03.2017.

УДК 681.51+004.94
ББК 32.988-5

ИНФОРМАЦИОННАЯ СИСТЕМА С ПОНЯТИЙНОЙ МОДЕЛЬЮ ПРЕДМЕТНОЙ ОБЛАСТИ

Выхованец В. С.[1]
(*ФГБУН Институт проблем управления им. В.А. Трапезникова РАН, Москва*)

Описывается интеллектуальная информационная система, основанная на понятийном моделировании предметной области. Понятийная модель состоит из понятийной структуры и описания содержания понятий. Понятийная структура определена как множество понятий, на которых заданы четыре отображения абстрагирования: обобщение, типизация, ассоциация и агрегация. Описание содержания понятий осуществляется с помощью таблиц базы данных. Существенным отличием используемой понятийной модели от других является описание ассоциации как обычного понятия. Показано, что благодаря семантической инвариантности области понятийной интерпретации удается улучшить технологические и эксплуатационные характеристики информационной системы.

Ключевые слова: предметная область, информационная система, понятийная модель, понятийная структура, абстракции понятий, база знаний, репрезентация знаний.

1. Введение

Трудности управления современным крупным производством требуют внедрение информационных технологий, которые необходимы для ускоренного принятия решений на всех уровнях управления, адекватного реагирования на происходящие события, выработки и использования типовых рекоменда-

[1] *Валерий Святославович Выхованец, доктор технических наук, доцент* (*valery@vykhovanets.ru*).

ций и решений. Однако опыт внедрения современных информационных систем показывает, что реальная отдача от них оказывается значительно ниже, а сроки внедрения и связанные с ним затраты – существенно выше ожидаемых.

Последнее обусловлено несовершенством системы управления предприятием, неподходящей или отсталой инфраструктурой, недостаточной квалификацией персонала, размытой ответственностью, дублированием функций, слабым планированием и контролем, низкой корпоративной культурой [1]. Однако и современные информационные системы обладают рядом существенных недостатков, затрудняющих их эффективное использование по назначению.

Информационные системы строятся, как правило, по трехслойной архитектуре, которая включает в себя слой клиента, слой логики и слой базы данных. Основной особенностью таких систем является физическое разделение программ, отвечающих за хранение данных (слой базы данных), от программ, обрабатывающих данные (слой логики) и отображающих данные (слой клиента).

Усложнение решаемых задач привело к появлению информационных систем, имеющих четыре слоя: слой клиента, слой представления, слой логики и слой базы данных, где новый слой – слой представления – реализует информационную модель предметной области. Информационная модель предполагает использование некоторой формальной теории, позволяющей описывать объекты предметной области и связи между ними.

Все большую роль начинает приобретать концептуальное моделирование как направление в информационной технологии, с помощью которого строятся и встраиваются в информационные системы концептуальные модели различных предметных областей [14, 30]. Такие модели способствуют более быстрому построению и модификации приложений, решающих те или иные прикладные задачи.

Основная цель концептуального моделирования – формализация накопленных знаний о некоторой предметной области в форме, наиболее близкой пониманию пользователей и разработчиков информационной системы. С концептуальным подходом связывают ожидания эффективного описания сложных пред-

метных областей, повышения надежности и качества информационных систем, ускорения актуализации данных вслед за изменением в предметной области, обеспечения многократного использования моделей различных областей знания и т.п. [8]. Однако эти задачи еще не имеют законченной ни методологической, ни технологической проработки.

В последнее время активно развивается такое направление концептуального моделирования как онтологическое моделирование, где под онтологией понимается результат формализации некоторой области знаний с помощью концептуальной схемы. Концептуальная схема включает в себя единичные сущности предметной области и их классы (концепты), а также различного рода связи (отношения) между ними. Дополнительно к концептуальной схеме формулируются правила (аксиомы, теоремы, ограничения, функции интерпретации), принятые в предметной области [17].

Самыми распространенными типами отношений, используемых в онтологиях, является отношения категоризации (IS-A, KIND-OF, PART-OF, MEMBER-OF, INSTANCE-OF, CONTAINED-IN). Однако основную семантическую нагрузку в модели несет большое число специфических отношений между сущностями предметной области.

Правила онтологии задают утверждения, связывающие понятия и отношения между собой. Правила позволяют производить умозаключения и получать утверждения, которые не могут быть выражены посредством концептуальной схемы.

Для задания онтологий используются различные языки (OWL, CL, CycL и др.), различающиеся формальным аппаратом описания сущностей, концептов, связей и правил. Так, язык OWL использует дескрипционную логику[1], CL – *s*-выражения логики первого порядка, а CycL – исчисление предикатов.

Все перечисленные языки обладают одним существенным недостатком – высокой вычислительной сложностью вывода на знаниях. А языки, основанные на логике первого порядка и

[1] *В OWL в качестве одного из расширений используется также язык, эквивалентный исчислению предикатов.*

исчислении предикатов, к тому же еще и неразрешимы, что не гарантирует получения результата вывода за конечное время.

В свою очередь высокая вычислительная сложность языка дискрипционной логики связана с большим числом связей между концептами, вводимыми с помощью множества ролей (одноместных предикатов) для ограничения объемов концептов, а также с аксиомами вложенности концептов.

Настоящая статья посвящена описанию одной из реализаций четырехслойной корпоративной информационной системы, в основе которой лежит понятийное моделирование предметной области.[1]

Суть подхода заключается в том, что понятийные модели предметной области строятся в формализме понятийной структуры, которая определена как множество понятий, на которых заданы только способы их образования в виде отображений типизации, агрегации, обобщения и ассоциации одних понятий в другие.[2]

В понятийной структуре, в отличие от других известных формальных теорий, таких как семантическая и ассоциативная сеть [18, 21], исчисление предикатов [19], модальная и дискрипционная логика [17, 20], концептуальный анализ [15], теория концептуальной зависимости [22], формальный анализ понятий [28], концептуальные графы [31], ER-модель [26] и др., ассоциация является обычным понятием, а не именованным видом связи, задающим в модели некоторую роль или отношение.

Язык понятийной модели, как и язык дескрипционной логики, является разрешимым, так как эквивалентен одноместному исчислению предикатов. Однако, в отличие от дискрипционной логики, язык понятийной модели не содержит средств описания отношений и ролей, так как все необходимые для

---

[1] *Пример реализации информационной системы, представленный в настоящей статье, публикуется с разрешения ЗАО «Инженерные системы и сервис»* (*Группа компаний «ЛАНИТ»*).

[2] *Под отображением понимается правило, по которому элементу одного множества ставится в соответствие элемент* (*или элементы*) *другого* (*или того же*) *множества. В то время как отношение – это подмножество декартова произведения множеств.*

вывода правила непосредственно содержатся в понятийной структуре.

Это позволяет повысить уровень абстракции модели и разработать информационную систему, для функционирования которой требуется небольшое число общих алгоритмов с небольшой вычислительной сложностью. Эти алгоритмы не зависят от предметной области, так как формулируются в предельно общих операциях над понятиями.

Благодаря этому удается улучшить вычислительные, технологические и эксплуатационные характеристики информационных систем с понятийными моделями предметной области.

2. Конкретные понятия

Наиболее систематичное исследование понятия содержится в работе [4], где понятие характеризуется как форма (вид) мысли и является «результатом обобщения предметов некоторого класса по определенной совокупности общих для предметов этого класса и – в совокупности отличительных для них признаков».

Полагается, что понятия образуются (определяются) при абстрагировании [10, 12], где абстрагирование рассматривается как один из основных процессов умственной деятельности человека, позволяющий под воздействием некоторой проблематики мысленно вычленить и превратить в самостоятельный объект рассмотрения отдельные свойства, стороны или состояния сущностей обозреваемой предметной области. При этом выделяются четыре вида понятий: единичные, простые, конкретные и абстрактные.

Единичные понятия. При образовании (определении) единичных понятий абстракция проявляется в способности мысленного выделения в предметной области уникальных сущностей и присвоения им имен. Иными словами, образование единичного понятия – это замена сущности некоторым знаком, тождественным сущности в некотором смысле.

Простые понятия. При образовании (определении) простых понятий абстракция проявляется как некоторое сознатель-

ное неведение, позволяющее сосредоточиться на одной стороне сущностей и игнорировать другие их стороны.

Простые понятия образуются путем объединения сущностей, подобных в некотором смысле. Простым понятиям присваивается уникальное имя, мыслимое как единичное понятие, и задается область допустимых проявлений (значений), мыслимая как множество единичных понятий.

Пример выражения простого понятия и принадлежащих ему сущностей – единичных понятий – приведен на рис. 1.

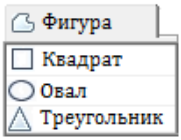

Рис. 1. Единичные и простые понятия

Конкретные понятия. Конкретные понятия образуются на основе объединения сущностей с одинаковыми признаками, позволяющими отличать сущности, принадлежащие понятию, от сущностей, понятию не принадлежащих. Выделяемые при этом признаки мыслятся как простые понятия, а имена понятий – как единичные понятия.

Определение 1. Конкретным понятием N называется упорядоченное множество из трех элементов: схемы shm N, интенсионала int N и экстенсионала ext N:

$$(1)\quad N=\begin{cases}\text{shm } N=(P_0,P_1,\ldots,P_{n-1}),\\ \text{int } N=\{(V_0^j,V_1^j,\ldots,V_{n-1}^j)\mid j=\overline{0,m-1}\},\\ \text{ext } N=\{E_0,E_1,\ldots,E_{m-1}\};\end{cases}$$

где N – имя понятия; P_i, i= 0, 1, …, n – 1, – понятия-признаки; $V_0^j, V_1^j, \ldots, V_{n-1}^j$, j = 0, 1, …, m – 1, – наборы значений понятий-

признаков, составляющие интенсионал понятия N; E_j, $j = 0, 1, \ldots, m-1$, – сущности, принадлежащие понятию N.[1]

Примеры выражения признаков конкретного понятия и принадлежащих ему сущностей приведены на рис. 2.

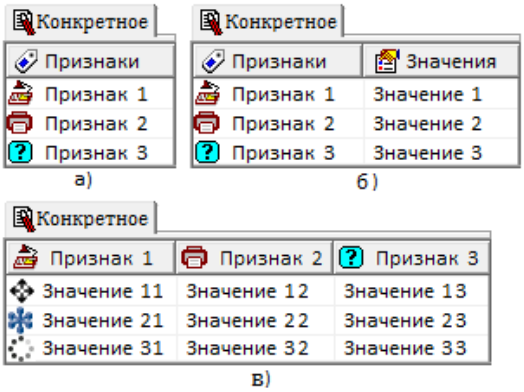

Рис. 2. Конкретное понятие:
а) признаки; б) одна сущность; в) множество сущностей

Из определения (1) следует, что между сущностями E_j, составляющими экстенсионал понятия N, и элементами $(V_0^j, V_1^j, \ldots, V_{n-1}^j)$ его интенсионала устанавливается взаимно однозначное соответствие. При этом сущности следует рассматривать как некоторые фрагменты предметной области, представляемые знаками, символами, образами и т.п. Для манипулирования сущностями в информационной системе их необходимо именовать и рассматривать как единичные понятия.

Следует заметить, что единичное понятие V (понятие-значение) и простое понятие P (понятие-признак) являются частными случаями конкретного понятия и в соответствии с (1) выражаются так:

---

[1] *Здесь и далее множества элементов заключаются в фигурные скобки, а наборы элементов (элементы мультимножеств или упорядоченные множества с повторением элементов) – в круглые.*

$$(2)\quad V=\begin{cases}\text{shm}\,V=(),\\ \text{int}\,V=\{V\},\\ \text{ext}\,V=\{V\};\end{cases}\qquad P=\begin{cases}\text{shm}\,P=(P),\\ \text{int}\,P=\{V_j\mid j=\overline{0,m-1}\},\\ \text{ext}\,P=\{V_0,V_1,\ldots,V_{m-1}\}.\end{cases}$$

Таким образом, конкретное понятие представляется множеством сущностей, которые образуют его экстенсионал или объем. Имя понятия – это знаковое выражение понятия, которому приписывается некоторый смысл. Схема или структура понятия задается набором признаков, характеризующих понятие. Интенсионал или содержание понятия рассматривается как наборы значений взаимосвязанных признаков, позволяющие распознать сущности, принадлежащие понятию.

3. Абстрактные понятия

При образовании (определении) абстрактных понятий используются более сложные формы абстрагирования, основанные на установлении между понятиями отношений независимости, дифференциации и интеграции признаков (рис. 3).

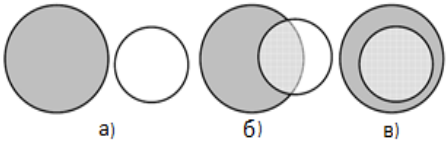

Рис. 3. Отношения признаков:
а) независимость; б) дифференциация; в) интеграция

Для образования абстрактных понятий применяются четыре абстракции: обобщение-специализация, типизация-конкретизация, агрегация-декомпозиция и ассоциация-индивидуализация [9, 11, 13, 25].

Обобщение и типизация, и обратные им абстракции – специализация и конкретизация, выражают общность понятий, проявляющуюся при дифференциации признаков. Ассоциация и агрегация, и обратные им – индивидуализация и декомпозиция, раскрывают структуру понятий, проявляющуюся при интеграции признаков.

Типизация-конкретизация. Типизация – порождение понятия на основе объединения экстенсионалов понятий с одинаковыми схемами.[1] При конкретизации понятия-типа выделяется одно из типизированных в нем понятий.

Для поиска сущностей в экстенсионале понятия-типа используется множество признаков, называемое ключом. Набор значений признаков, входящих в ключ, позволяет идентифицировать (отличать, именовать) сущности из экстенсионала понятия-типа.

Примеры выражения типизации-конкретизации приведены на рис. 4, где узел с именем «Ключ» используется для поиска и отображения в виде дочерних узлов сущностей понятия-типа путем ввода в наименование этого узла соответствующего ключа или его части. Если в ключ входит несколько признаков, то они разделяются знаком #.

Рис. 4. Типизация-конкретизация:
а) абстрактных понятий; б) конкретных понятий

Обобщение-специализация. При обобщении происходит порождение нового понятия на основе нескольких подобных понятий, когда новое понятие сохраняет все или часть общих признаков исходных понятий и игнорирует другие признаки.

[1] *В* [25] *типизация называется классификацией* (*англ. classification*).

Обобщение – порождение нового понятия на основе пересечения схем обобщаемых понятий и объединения их экстенсионалов. При специализации, наоборот, из понятия-обобщения выделяется одно из обобщенных в нем понятий.

Как и при типизации-конкретизации для поиска сущностей в экстенсионале понятия-обобщения может использоваться ключ, который совпадает со схемой понятия-обобщения и может быть не уникальным.

Пример выражения обобщения-специализации приведен на рис. 5.

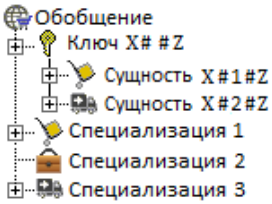

Рис. 5. Обобщение-специализация понятий

Следует заметить, что типизация-конкретизация является частным случаем обобщения-специализации. Однако при обобщении-специализации нельзя отобразить сущности понятия в виде таблицы (рис. 4б), так как обобщаемые понятия имеют разные схемы. В таблице возможно отображение только столбцов, принадлежащих схеме понятия-обобщения (принадлежащих общему набору признаков обобщаемых понятий).

Для указания на абстракцию обобщения-специализации связи между обобщаемыми понятиями проводят пунктирными линиями, для указания на абстракцию типизации-конкретизации – сплошными линиями (рис. 4, 5).

Ассоциация-индивидуализация. При ассоциации устанавливается взаимосвязь между сущностями одного и того же или разных понятий.[1] Ассоциация – порождение понятия на основе

[1] *Ассоциация, традиционно рассматриваемая как роль, связь или отношение понятий, – это такой же вид мысли, как и мысль о части* (*декомпозиция*)*, о целом* (*агрегация*)*, о частном* (*специализация*)*, об*

объединения схем ассоциируемых понятий и ограниченного декартова произведения их экстенсионалов. Ассоциация выражает специфическое соединение сущностей ассоциируемых понятий, позволяющее переходить от сущностей одних ассоциированных понятий к сущностям других ассоциированных понятий. При индивидуализации из понятия-ассоциации выделяются ассоциированные в нем понятия.

Для поиска сущностей в экстенсионале понятия-ассоциации используется набор признаков, называемый связью и задающий ключ. Для характеристики связи между ассоциированными понятиями используется запись вида $L_0 : L_1 : \ldots : L_{n-1}$, где n – число ассоциированных понятий, L_i – кардинальность вхождения i-го ассоциированного понятия, которая принимает одно из следующих значений: 1 (единичное вхождение), M (множественное вхождение).

Пример выражения ассоциации-индивидуализации приведен на рис. 6, где узел с именем «Ключ» используется для поиска и отображения в виде дочерних узлов сущностей понятия-ассоциации путем ввода в наименование этого узла признаков, входящих в связь (образующих ключ).

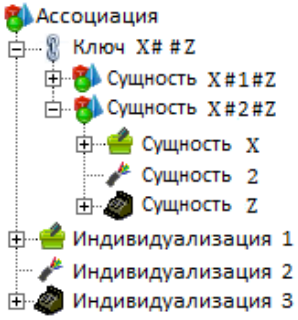

Рис. 6. Ассоциация-индивидуализация понятий

общем (*обобщение*), *о классе* (*типизация*), *об экземпляре* (*конкретизация*). *Отсюда следует, что ассоциация может быть представлена как обычное понятие или как вид мысли, закономерно возникающей между двумя понятиями, связанными в каком-либо смысле.*

Агрегация-декомпозиция. При агрегации понятие строится как соединение (композиция) других понятий. Агрегация – порождение понятия на основе объединения схем агрегируемых понятий и полного декартова произведения их экстенсионалов. При декомпозиции понятие-агрегат разделяется на входящие в него агрегированные понятия.

Пример выражения агрегации-декомпозиции приведен на рис. 7. В отличие от ассоциации, где между сущностями понятий устанавливаются только часть связей, в агрегации присутствуют все возможные связи. В этом случае в связь входит вся схема понятия-агрегата. Следует также заметить, что только понятие-агрегат может отображаться независимо по каждому агрегированному понятию (рис. 7б), для понятия-ассоциации это не имеет смысла.

Для указания на абстракцию ассоциации-индивидуализации связи между ассоциируемыми понятиями проводят пунктирными линиями, для указания на абстракцию агрегации-декомпозиции – сплошными линиями (рис. 6, 7).

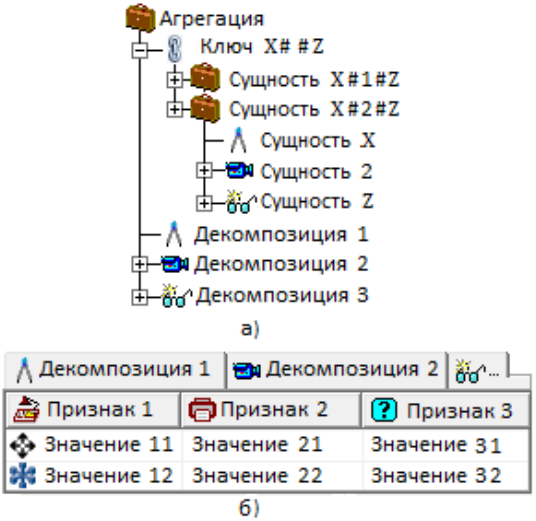

Рис. 7. Агрегация-декомпозиция:
а) абстрактных понятий; б) конкретных понятий

После описания абстракций понятий приведем их формальное определение [5].

Определение 2. Понятия N_j, $j = 0, 1, \ldots, n-1$, использованные для образования понятия N_G (N_T, N_C, N_A) путем обобщения (типизации, агрегации, ассоциации), называются обобщаемыми (типизируемыми, ассоциируемыми, агрегируемыми). При этом

$$(3)\quad \begin{cases} \operatorname{shm} N_G = \prod\limits_{j=0}^{n-1} \operatorname{shm} N_j, \\ \operatorname{int} N_G \supseteq \bigcup\limits_{j=0}^{n-1} \operatorname{int} N_j, \\ \operatorname{ext} N_G \supseteq \bigcup\limits_{j=0}^{n-1} \operatorname{ext} N_j; \end{cases} \qquad \begin{cases} \operatorname{shm} N_T = \prod\limits_{j=0}^{n-1} \operatorname{shm} N_j, \\ \operatorname{int} N_T = \bigcup\limits_{j=0}^{n-1} \operatorname{int} N_j, \\ \operatorname{ext} N_T = \bigcup\limits_{j=0}^{n-1} \operatorname{ext} N_j; \end{cases}$$

$$(4)\quad \begin{cases} \operatorname{shm} N_A = \coprod\limits_{j=0}^{n-1} \operatorname{shm} N_j, \\ \operatorname{int} N_A \subseteq \underset{j=0}{\overset{n-1}{\times}} \operatorname{int} N_j, \\ \operatorname{ext} N_A \subseteq \underset{j=0}{\overset{n-1}{\times}} \operatorname{ext} N_j; \end{cases} \qquad \begin{cases} \operatorname{shm} N_C = \coprod\limits_{j=0}^{n-1} \operatorname{shm} N_j, \\ \operatorname{int} N_C = \underset{j=0}{\overset{n-1}{\times}} \operatorname{int} N_j, \\ \operatorname{ext} N_C = \underset{j=0}{\overset{n-1}{\times}} \operatorname{ext} N_j; \end{cases}$$

где shm (int, ext) – схема (интенсионал, экстенсионал) понятия, $\coprod$ ($\prod$) – размеченное пересечение (объединение), выполняемое с повторением элементов, $\supseteq$ ($\cup$, $\times$) – включение (неразмеченное объединение, декартово произведение) множеств.

После определения абстракций понятий можно сделать следующие выводы:

– единичные понятия (понятия-значения) имеют пустую схему и именуются своим значением, а схема простого понятия (понятия-признака) состоит из имени этого понятия;

– сущности простого понятия (значения понятия-признака) являются единичными элементарными понятиями, а сами простые понятия (понятия-признаки) – объемными элементарными понятиями;

– простые понятия (понятия-признаки) образуются на основе абстракции обобщения, а конкретные понятия – на основе абстракции ассоциации.

4. Понятийная структура

Будем предполагать, что образование новых или выявление уже существующих понятий происходит в процессе изучения предметной области. При этом под углом зрения некоторой проблематики выделяются сущности, которые уже имеют или которым приписываются некоторые имена. Далее множество выявленных сущностей подвергается анализу на предмет установления их сходства и различия. Сходные сущности группируются, в результате чего происходит образование понятий, или наполнение уже имеющихся понятий проблемным содержанием.

В отличие от известных формализмов, где на понятиях задается множество отношений различной природы, будем использовать другой формализм – понятийную структуру, которую определим множеством понятий с четырьмя видами отображений, единственное назначение которых – показать способы образования понятий, способы их абстрагирования.[1]

Определение 3. Понятийной структурой $S = (N, T, G, C, A)$ называется конечное множество понятий N, на которых заданы четыре конечных множества отображений вида $N \to N$: типизации T, обобщения G, агрегации C и ассоциации A.

Понятийная структура призвана в формализованном виде отразить результаты понятийного анализа предметной области и выражает отображения одних понятий в другие. Используемые при этом абстракции рассматриваются как умственные операции, необходимые и достаточные для мысленного выделения и превращения в отдельные понятия тех представлений, которые накоплены относительно формализуемой предметной области.

---

[1] *Известные концептуальные модели обладают некоторой «концептуальной» незавершенностью. Они задаются множеством понятий и связей между ними, которые определяют смысловое содержание предметной области [24]. Однако любая такая связь также является понятием, с которым можно выполнять те же операции. Но в концептуальных моделях такие операции не предусмотрены.*

Пример понятийной структуры приведен на рис. 8. Понятие «Тип проекта» определено как результат типизации единичных понятий «Строительство», «Реконструкция» «Модернизация».

Понятие «Учетный проект» есть результат ассоциации простых понятий «Наименование» и «Дата», а также понятия-типа «Тип проекта».

В свою очередь понятие «Руководитель проекта» есть результат обобщения таких понятий как «Руководитель отдела», «Главный инженер» и «Директор департамента».

И, наконец, понятие «Назначение» определено как ассоциация понятий «Учетный проект» и «Руководитель проекта».

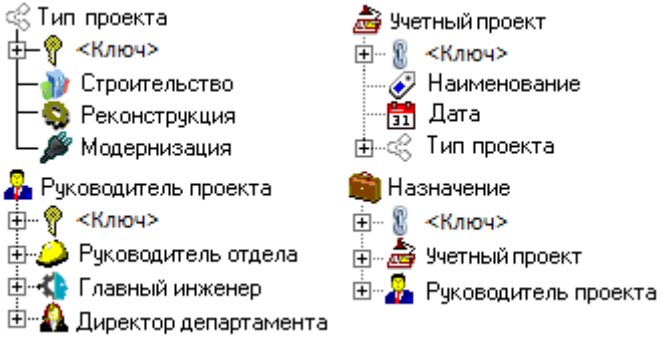

Рис. 8. Фрагмент понятийной структуры

Для поиска сущностей в экстенсионалах дифференциальных понятий используется специальный узел с пиктограммой «Ключ», а для поиска сущностей в экстенсионалах интегральных понятий – специальный узел с пиктограммой «Связь».

Схема понятия. Из определения понятия следует, что каждое понятие обладает схемой – набором признаков или простых понятий, на которых это понятие определено. Однако при задании понятийной структуры определению подлежат только отображения одних понятий в другие. Возникает задача определения схем понятий.

Для любого понятия из понятийной структуры S может быть найдена его схема по следующей рекуррентной процедуре:

– схема единичного понятия пуста;

– схема простого понятия состоит из имени этого понятия;

– схема понятия-дифференциации равна размеченному пересечению схем дифференцируемых понятий;

– схема понятий-интеграции равна размеченному объединению схем интегрируемых понятий;

– схема понятия, полученного в результате интеграции и дифференциации, равна размеченному объединению схем интегрируемых понятий, принадлежащая размеченному пересечение схем дифференцируемых понятий.[1]

Следует обратить внимание на то, что возможность определения схем абстрактных понятий позволяет отображать последние как конкретные понятия в виде списка сущностей, раскрываемых через первичные понятия-признаки предметной области.

Для верификации понятийной структуры требуется проверить вычислимость схем всех входящих в нее понятий.[2] Требование вычислимости схем понятий является распространением свойства регулярности (фундированности) множеств на понятия. В этом случае вычислимость схем понятий гарантирует отсутствие определений понятий через самих себя, что в любой формальной или содержательной теории, претендующей на адекватность, признается недопустимым [7].

Таким образом, в результате понятийного анализа предметной области может быть получено ее онтологическое описание, заданное понятийной структурой. Принципиальными отличиями понятийной структуры от других концептуальных схем является:

– отсутствие разделения терминов на значения, признаки, понятия, связи и роли, а использование одного общего термина – понятия, частными проявлениями которого являются значения, признаки, понятия, связи и роли;

---

[1] *Необходимость одновременного использования абстракций дифференциации и интеграции при определении понятий связана с тем, что на одном и том же наборе понятий может быть определено несколько обобщений путем различного ограничения набора обобщающих признаков.*

[2] *Задача определения вычислимости схем понятий является алгоритмически разрешимой из-за конечности используемых множеств.*

− возможность представления ассоциаций как самостоятельных понятий, что позволяет, например, выразить обобщение ассоциативных связей;

− определение понятий, которые одновременно могут быть как обобщением, так и ассоциацией других понятий;

− семантическая инвариантность описания, не требующая для своей интерпретации привлечения предметных знаний.

5. Понятийная модель

В понятийной структуре предметной области задаются понятия и способы их абстрагирования. По понятийной структуре может быть также вычислена схема понятия – набор простых понятий (понятий-признаков), позволяющих рассматривать абстрактные понятия как конкретные. Однако какие наборы значений простых понятий определяют ту или иную сущность конкретных и абстрактных понятий, остается неясным. Для этого предназначен интенсионал понятия, задающий правила соотнесения наборов значений простых понятий с той или иной сущностью из экстенсионала описываемого понятия.

Определение 4. Понятийной моделью M предметной области называется ее понятийная структура S, дополненная описанием интенсионалов D всех входящий в нее понятий, $M = (S, D)$.

Из определения 1 следует, что универсальной формой задания интенсионала является перечисление наборов признаков каждой сущности, принадлежащей понятию. Использование этой формы на практике приводит к трудностям, если объем понятия достаточно велик. Другая форма описания интенсионалов – использование процедур (формул, функций), разрешающих экстенсионал определяемого понятия.[1]

---

[1] *Проблема представления простых понятий восходит к известной проблеме задания множеств. Множество называется разрешимым, если существует алгоритм, позволяющий определить, принадлежит ли элемент этому множеству или нет. Множество называется перечислимым, если существует алгоритм, порождающий это множество, т.е. последовательно выдающий все его элементы. Известно, что всякое разрешимое множество перечислимо. Однако обрат-*

При использовании баз данных низкоуровневое представление понятийной модели предметной области может решаться только средствами, встроенными в систему управления базами данных [27]. Так единичные понятия задаются значениями простых типов данных: целыми числами, числами с плавающей запятой, датами, символами, строками с фиксированной и переменной длиной, двоичными данными.

Для задания простых понятий (понятий-признаков) могут использоваться ограничения целостности, сужающие области значений простых типов данных, или специальные таблицы, содержащие допустимые значения простого понятия. Конкретные понятия представимы таблицами, столбцы которых соответствуют простым понятиям из его схемы.

Сложнее обстоит дело с абстрактными понятиями. Представление абстрактного понятия-типа особых сложностей не вызывает и осуществляется с помощью запроса к базе данных, обрабатывающего записи одновременно из нескольких таблиц с одинаковым набором полей. Однако реализация абстрактного понятия-обобщения уже требует создания отдельной таблицы, где хранятся интенсионалы сущностей, отсутствующих в экстенсионалах обобщаемых понятий. Отсюда следует, что представление абстрактного обобщение осуществляется с помощью запроса к базе данных, в котором обработке подлежат все записи из таблицы понятия-обобщения и уникальные записи из таблиц обобщаемых понятий с набором полей, принадлежащих схеме понятия-обобщения.

Агрегация понятий представляется запросом, выполняющим произведение двух и более таблиц. Однако реализация абстрактного понятия-ассоциации уже не может быть сведена к запросу к базе данных и требует выделения отдельной таблицы, связывающей записи из двух и более таблиц ассоциируемых понятий.[1]

ное неверно, поскольку существуют перечислимые, но неразрешимые множества [*2*].

[1] *Следует заметить, что в реализованной информационной системе для представления понятий-агрегатов также используются таблицы. Это связано с тем, что на практике оказывается востребованным*

Как показано ранее, для отображения интенсионалов простых понятий используется табличная форма (рис. 9а). Табличные формы также могут быть сформированы для отображения интенсионалов конкретных и абстрактных понятий. В этом случае таблица создается на основе схемы раскрываемого понятия (рис. 9б).

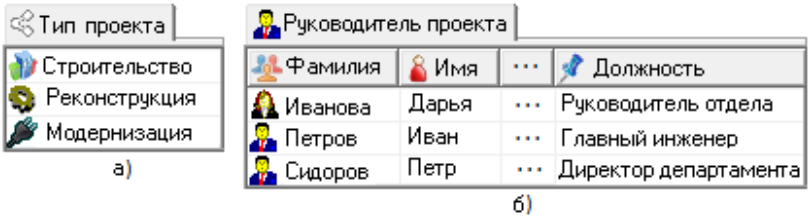

Рис. 9. Отображение интенсионалов понятий: а) простых; б) конкретных и абстрактных

Здесь важно заметить, что табличная форма (список) на рис. 9б в целом выражает некоторое понятие, все ее строки – интенсионал понятия, одна строка – конкретную сущность, а заголовки столбцов – схему понятия.

Таким образом, при использовании систем управления базами данных основным средством описания интенсионалов понятий является перечисление наборов взаимосвязанных признаков в виде физической или виртуальной таблицы. Физическая таблица непосредственно хранится в памяти, а виртуальная таблица формируется динамически в результате выполнения некоторого запроса к базе данных. В этом случае строка (запись) таблицы соответствует описываемой сущности из экстенсионала понятия, а столбец (поле) – признаку понятия.

небольшое число экземпляров понятия-агрегата. Для унификации процедур обработки понятий удобно такие экземпляры хранить в отдельной таблице.

6. Язык понятийной модели

Учитывая множество отображений абстрагирования, которые задаются между понятиями в понятийной структуре, для ее эффективного описания разработан специальный язык, формальная грамматика которого приведена на рис. 10.

Грамматика задана с точностью до пробельных знаков, которые могут появляться между терминальными и нетерминальными символами. Определяемые нетерминальные символы записаны в левой части правил вывода до знака →, а терминальные символы заключены в апострофы. Альтернативы в правой части правил вывода перечислены через разделитель |, а необязательные вхождения символов грамматики заключены в квадратные скобки.

1) Модель → Понятие [Модель]
2) Понятие → Значение | Свойство | Признак | Абстракция | Ссылка
3) Значение → Строка
4) Свойство → '=' Строка '#'
5) Признак → Имя '=' Значение '#'
6) Абстракция → Имя '=' '#' [Модель] '#'
7) Ссылка → '=' '#' [Путь] '#'
8) Путь → Имя '#' [Путь]
9) Имя → Строка
10) Строка → Символ [Строка]
11) Символ → '\ ' | '\#' | '\=' | '\\' | Буква | Цифра | Знак

Рис. 10. Формальная грамматика языка модели

Модель предметной области состоит из непустого перечисления ее понятий (правило 1). Понятие может выражаться значением или свойством, быть признаком или абстракцией, а также являться ссылкой (правило 2).

Значение выражается произвольной строкой (правило 3), а свойство – терминальным символом «равно», после которого следует произвольная строка (правило 4). Свойства используются для уточнения описания понятий, невыражаемого на языке

понятийной модели. В этом случае интерпретация значений свойств осуществляется принимающей модель программой.

Признак имеет имя, с которым связывается одно или несколько значений (правила 5 и 1). Абстракция также именуется, но с ней связывается некоторая подмодель (правило 6). Ссылка задается путем от корня модели к ранее описанному понятию (правила 7 и 8).

Имя, как и значение, выражается произвольной строкой (правило 9). Строка определена как произвольная последовательность символов (правило 10): букв, цифр и других знаков, допустимых в используемой кодировке (правило 11). Для обеспечения возможности использования в строке символов разметки (пробельные знаки, #, =, \) последние имеют специальное выражение (правило 11).

Ниже в качестве примера приведено описание понятийной структуры, показанной на рис. 8, где используются необходимые, но не показанные на рисунке, простые понятия «Строка» и «Дата».

```
Тип проекта = #
    = Типизация #
    Строительство #
    Модернизация #
    Реконструкция #
#
Учетный проект =#
    = Ассоциация #
    Наименование =# [1]
        =# Строка ##
    #
    =# Дата ##
    =# Тип проекта ##
#
Руководитель проекта = #
```

[1] *В случае необходимости на языке понятийной модели может быть выражена эквивалентность понятий. В примере понятие «Наименование» описано как эквивалентное понятию «Строка».*

```
    = Обобщение #
    =# Руководитель отдела ##
    =# Главный инженер ##
    =# Директор департамента ##
#
Назначение =#
    = Ассоциация #
    =# Учетный проект ##
    =# Руководитель проекта ##
#
```

7. Задачи предметной области

Использование понятийной структуры, задаваемой в виде дерева (рис. 8), и отображение сущностей из экстенсионалов понятий в виде списков (рис. 9) не являются достаточными с точки зрения функционирования информационной системы, призванной автоматизировать решения заданного класса прикладных задач.

Абстрагируясь от конкретного содержания действий и процедур, составляющих алгоритмы решения той или иной прикладной задачи, можно сделать вывод о том, что все такие действия сводимы к трем абстрактным операциям над понятиями, а именно: созданию понятия, изменению понятия и удалению понятия. Таким образом, семантически инвариантной формой описания решений прикладных задач в моделируемой предметной области является описание, состоящее всего из трех элементарных операций.

Создание понятий. Операция создания понятия возникает при усложнении понятийной модели предметной области и состоит в определении имени нового понятия и задания способа его абстрагирования. После создания нового понятия автоматически вычисляется его схема. В зависимости от вида понятия в базе данных создается или таблица – для простых и конкретных понятий, а также абстрактных понятий-обобщений и понятий-ассоциаций, или представление (запрос) – для абстрактных

понятий-типизаций и понятий-агрегаций, способные перечислить сущности, принадлежащие созданному понятию.[1]

Удаление понятий. Операция удаления понятия возникает в случае изменения представлений о предметной области и состоит в изменении описания всех понятий, в определения которых оно входит.

На рис. 11 показан пример отображения понятий «Объект» и «Проект» и доступных для них операций, активируемых через контекстное меню клиентского приложения.

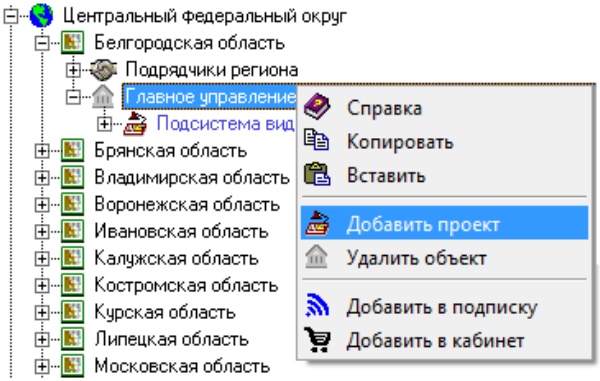

Рис. 11. Операции создания и удаления понятий

Изменение понятий. Операция изменения понятий используется при необходимости наполнить понятия конкретным предметным содержанием. Операции изменения выполняются над конкретными и простыми понятиями. В этом случае возможны три действия: редактирование существующей сущности (записи), удаление существующей сущности (записи) и добавление новой сущности (записи). Такие же действия выполняют-

[1] *Следует заметить, что современные системы управления базами данных помимо табличного хранения данных могут выполнять хранимые процедуры* [27]*, позволяющие формировать таблицы «на лету» путем вычисления как схем, так и содержания воспроизводимых ими понятий.*

ся и над абстрактными понятиями-обобщениями и понятиями-ассоциациями, имеющими свои таблицы.

На рис. 12 показан пример отображения экстенсионала понятия «Комплект» и его операций изменения.

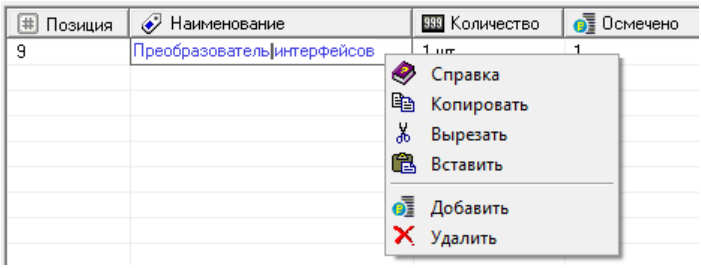

Рис. 12. Операции изменения понятия

Операции добавления и удаления сущностей понятия отображаются в контекстное меню приложения (рис. 11, 12), а операция редактирования сущности выполняется путем непосредственного изменения значения простого понятия или путем выбора из ниспадающего списка одного из его допустимых значений.

Для реализации операций создания, удаления и изменения понятий в системе управления базой данных могут быть предусмотрены специальные хранимые процедуры, обеспечивающие целостность и согласованность понятийной модели предметной области.

8. Информационная система

Многослойная структура информационной системы с понятийной моделью предметной области приведена на рис. 13.

Слой клиента реализован в виде клиентского приложения (рис. 14), функционирующего по принципу обозревателя (проводника, браузера): приложение получает данные от слоя знаний для отображения понятийной структуры в виде дерева (а), а интенсионалов и экстенсионалов понятий – в виде вкладок (б) и списка (в).

Рис. 13. Слои информационной системы

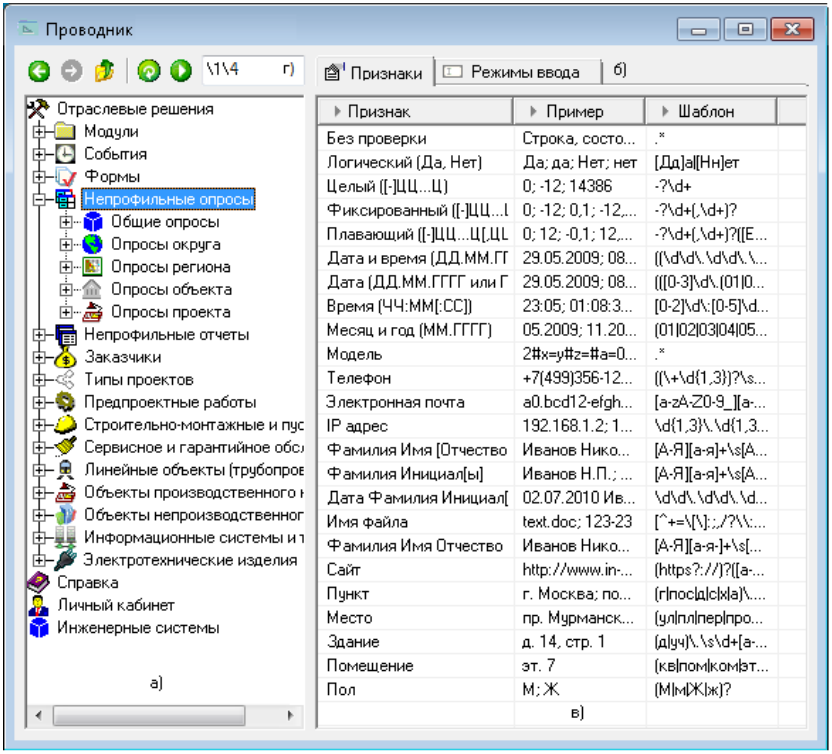

Рис. 14. Клиентское приложение:
а) дерево; б) вкладки; в) список; г) адресная строка

Для ускорения навигации по элементам формы используется кнопочное меню и адресная строка (г). Адресная строка также используется для поиска понятий по их именам.[1] Предусмотрена также возможность множественного и частичного отображение основной формы приложения, в которой отсутствует любой из перечисленных элементов.

Слой знаний служит для формирования данных в форматах, используемых клиентским приложением для отображения и манипулирования понятийной моделью.[2] Обмен данными осуществляется по двум интерфейсам: INode и IList. Интерфейс INode используется для отображения узла дерева и предоставляет его имя, пиктограмму, описание контекстного меню и вкладок, а также интерпретируемые тексты программ для изменения имени узла, обработки интерфейсных событий, отображения дочерних узлов. Интерфейс IList используется для отображения списка и предоставляет описание заголовка списка (столбцов), множество строк, состоящих из ячеек по числу столбцов, а также интерпретируемые тексты программ для изменения содержимого ячеек и обработки интерфейсных событий.

Для повышения эффективности информационной системы и сокращения времени реализации новых процедур предусмотрено использование унифицированным образом классов конкретных понятий (рис. 14), например:

– класса понятий, представляемых типовыми опросными формами;

– класса понятий, выражающих всевозможные отчеты, формируемые для анализа текущего состояния моделируемой предметной области;

– других классов понятий, имеющих типовое представление и однотипную обработку.

---

[1] *В связи с использованием длинных имен понятий возникают значительные неудобства при указании пути понятия в понятийной структуре. По этой причине в адресной строке приложения отображается иерархический адрес текущего понятия, состоящий из последовательности номеров понятий в текущем пути.*

[2] *Почему второй слой информационной системы назван слоем знаний, будет показано далее.*

Следует обратить внимание на то, что сама информационная система тоже имеет некоторую понятийную модель, работа с которой также происходит через клиентское приложение (рис. 14). В эту модель могут входить такие понятия как:

– модуль, подгружаемый в процессе работы клиентского приложения и служащий для реализации специфической функции отображения понятийной модели или решения специфической задачи предметной области;

– событие, регистрируемое в информационной системе и позволяющее задать обработчик для операций создания, удаления или изменения понятий;

– форма, создаваемая для реализации различных сценариев ввода и обработки данных пользователем;

– другие понятия, необходимые для реализации требований к модели конкретной предметной области.

Слой логики отвечает за выполнение операций над понятиями. Для реализации этого слоя могут использоваться как процедуры, запускаемые на выделенном для этого сервере (слой логики), так и хранимые процедуры системы управления базой данных (слой данных).

Для ограничения операций над понятиями, а также для формирования индивидуальных понятийных моделей, информационная система в слое логики реализует развитый механизм определения и наследования прав. Для любого понятия модели и группы пользователей (пользователя) могут быть заданы четыре вида прав: доступа, допуска, действия и владения (рис. 15).

Слой данных представлен системой управления базами данных. В случае развертывания больших информационных систем каждый слой может быть реализован не на одном, а на группе серверов, и содержать средства динамического распределения нагрузки на серверы следующего слоя. В этом случае слой данных реализуется в виде распределенной базы данных.

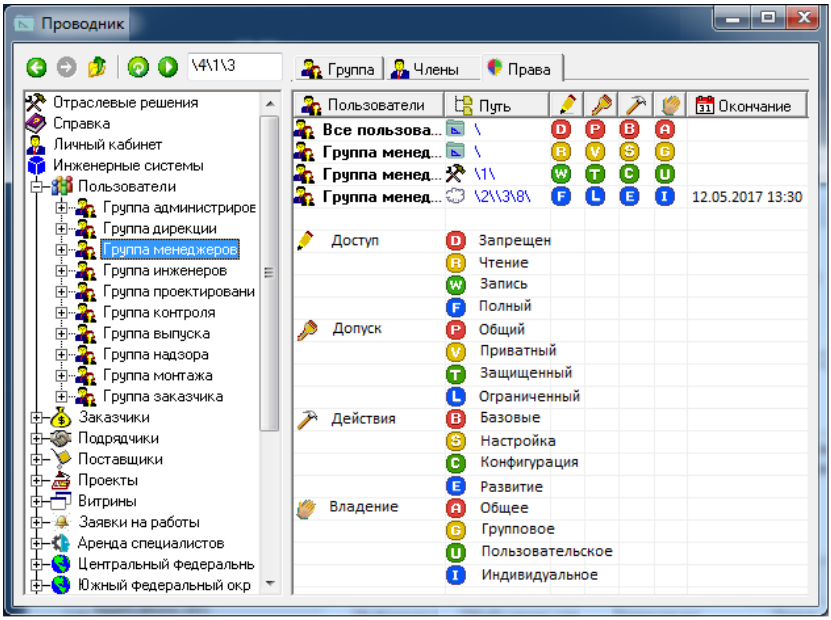

Рис. 15. Права

9. База знаний

В основе любой информационной системы, предназначенной для обработки знаний, лежит формальный аппарат представления знаний и манипулирования ими с целью имитации рассуждений человека для решения стоящих перед ним прикладных задач. В свою очередь под базой знаний понимается база данных, содержащая факты о некоторой предметной области, а также правила вывода, позволяющие автоматически выполнять умозаключения и получать новые утверждения об имеющихся или вновь вводимых фактах [3].

Информационная система с понятийной моделью предметной области $M = (S, D)$ может рассматриваться как база знаний. В этом случае понятийная структура S (понятия и способы их абстрагирования) задает правила вывода на знаниях, а интенсионалы понятий D – факты (суждения) о предметной области.

Суждения. Фактами (суждениями) являются высказываниями о принадлежности сущностей предметной области экстенсионалам понятий. В соответствии с формулой (1) сущность E принадлежит экстенсионалу понятия N, если и только если набор значений признаков сущности E, упорядоченный в соответствии со схемой понятия $\mathrm{shm}\,N$, принадлежит интенсионалу понятия $\mathrm{int}\,N$:

$$(5)\quad \left(\coprod_{(\forall P_i \in \mathrm{shm} N)} P_i[E] \right) \in \mathrm{int}\,N \leftrightarrow N(E),$$

где $\coprod$ – размеченное (упорядоченное) объединение, выполняемое с повторением элементов; $P_i[E]$ – функтор, возвращающий значение понятия-признака P_i сущности E; $\leftrightarrow$ – логическая связка двухстороннего следования; $N(E)$ – одноместный предикат принадлежности сущности E экстенсионалу понятия N, $N(E) \leftrightarrow E \in \mathrm{ext}\,N$.

Согласно формуле (5) выполнимость одноместного предиката $N(E)$ определяется по текущему состоянию интенсионалов D, т.е. информационная система реализует модель открытого мира.[1]

Умозаключения. Любое умозаключение может быть определено как переход от одного или нескольких суждений, составляющих посылку умозаключения, к утверждению – следствию умозаключения. Правила построения умозаключений задаются правилами вывода, принимаемыми в предметной области в качестве общезначимых, т.е. порождающих истинные утверждения при всех возможных посылках.

Правила построения умозаключений в рассматриваемом случае задаются правилами вывода, формализованными в понятийной структуре предметной области, а сама понятийная структура рассматривается как формальная теория, которая сохраняет истинность всех выводимых в ней следствий.

---

[1] *В модели открытого мира нарушается монотонность вывода, заключающаяся в том, что если в результате вывода получено некоторое утверждение, то при поступлении новых фактов выводимость этого утверждения не должна исчезнуть.*

Из формул (3) и (4) получаем логические высказывания, задающие правила вывода на знаниях:

(6) $\bigvee\limits_{(\forall N_i > N)} N_i(E) \to N(E)$; $\bigvee\limits_{(\forall N_i \triangleright N)} N_i(E) \leftrightarrow N(E)$;

(7) $N(E) \to \bigwedge\limits_{(\forall N_i \succ N)} N_i(E)$; $N(E) \leftrightarrow \bigwedge\limits_{(\forall N_i |\succ N)} N_i(E)$,

где $\vee$ ($\wedge$) – логическая связка «ИЛИ» («И»); $>$ ($\succ$, $\triangleright$, $|\succ$) – знак отображения обобщения (ассоциации, типизации, агрегации) понятий; $\forall$ – квантор всеобщности; $\to$ – логическая связка следования.

Обратное утверждение первого правила вывода из (6) верно только для абстракции типизации-конкретизации (второе правило), так как согласно (3) абстракция обобщения-специализации описывает большее число сущностей, чем их содержится в объединении экстенсионалов обобщаемых понятий.[1]

В свою очередь обратное утверждение первого правила вывода из (7) верно только для абстракции агрегации-декомпозиции (второе правило), так как согласно (4) абстракция ассоциации-индивидуализации описывает меньшее число сущностей, чем их содержится в декартовом произведении экстенсионалов ассоциируемых понятий.

Запросы. Для превращения информационной системы в полноценную базу знаний необходимо реализовать запросы для извлечения фактов (суждений) и вывода содержательных утверждений о моделируемой предметной области.

Факты (суждения) и утверждения представляют собой высказывания с логическим связками «И», «ИЛИ», «НЕ», в которых используются два типа предикатов:

– одноместные предикаты принадлежности сущности E понятию N вида $N(E)$;

---

[1] *В известных реализациях баз знаний не допускается пополнение понятия-обобщения новыми сущностями кроме как через пополнение обобщаемых понятий. Это делает понятие-обобщение в некотором смысле неполноценным понятием.*

– отношения вида $P[E] \circ V$, где $P[E]$ – функтор, возвращающий значение понятия-признака P сущности E, $\circ$ – знак отношения (=, ≠, >, ≥, <, ≤ и т.п.), V – некоторое понятие-значение.

На естественном языке предикат принадлежности может быть выражен следующими предложениями: «Строительство есть Тип проекта», «Руководитель проекта (Главный инженер)», «Проект 12 – Учетный проект», «Проект 12, Петров Иван – это Назначение», а отношения с признаками – «Дата > 12.11.2016», «Тип проекта = Модернизация» и т.п. (рис. 8).

На рис. 16 приведена форма для поиска сущностей понятий, удовлетворяющих условиям, задаваемым с помощью шаблона поиска. Шаблон поиска состоит из списка признаков понятия и отображается в левой панели формы при установке указателя на узел поиска сущностей с пиктограммой «Ключ». После задания ограничений на значения одного или нескольких признаков информационная система выполняет поиск сущностей, удовлетворяющих заданным условиям, и выводит последние в виде дочерних узлов узла поиска.

Поиск сущностей понятия по ключу выполняется аналогично и осуществляется после ввода ключа или его части в название узла поиска. Если в ключе несколько признаков, то они разделяются знаком # (рис. 4–7). Признаки, входящие в ключ, выводятся в шаблоне поиска жирным шрифтом (рис. 16).

Для более сложного поиска необходимо использовать запросы к базе знаний, для чего предусматривается соответствующий язык запросов и поддерживающая его машина вывода на знаниях, реализующая правила вывода вида (5), (6) и (7). Однако для многих прикладных запросов оказывается достаточным использования адресной строки клиентского приложения для нахождения в понятийной структуре понятий по их именам (рис. 14), с последующим поиском требуемых сущностей в экстенсионалах найденных понятий (рис. 16).[1]

[1] *Описание языка запросов и механизма логического вывода выходит за рамки настоящей статьи.*

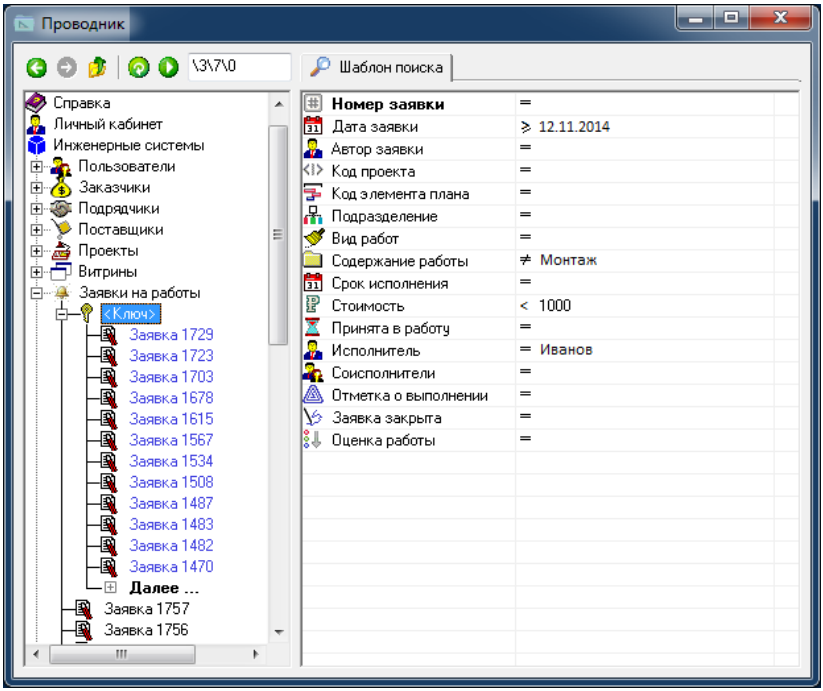

Рис. 16. Поиск в экстенсионалах понятий

10. Репрезентация знаний

Помимо задач представления, извлечения и актуализации знаний имеется другая важная задача – их репрезентация. Репрезентация знаний заключается в изменении формы представления знаний и осуществляется на основе построения понятийных подмоделей с последующей их визуализацией специальными программами.

Определение 5. Подмоделью M' понятийной модели $M = (S, D)$ называется такая понятийная модель $M' = (S', D')$, для которой выполняются следующие отношения: $S' \subseteq S$, $D' \subseteq D$, где S' – некоторая подструктура (фрагмент) понятийной структуры S, D' – описание интенсионалов понятий, входящих в понятийную структуру S'.

Для построения понятийной подмодели применяется следующая процедура. Вначале выделяется множество основных понятий, которые должны присутствовать в подмодели по условиям решаемой задачи. Затем итерационно строится понятийная подструктура, в которую включаются все понятия, имеющие связи с начальным, а далее – с текущим их множеством. Итерации завершаются, когда текущее множество понятий перестает пополняться. В завершении процедуры для полученной понятийной подструктуры создается описание интенсионалов входящих в нее понятий.

Построение понятийных подмоделей необходимо для создания данных, которые требуются для визуализации фрагментов предметной области сторонними программами. Например, для планирования проектов часто используется понятийная модель (рис. 17), в которой присутствуют такие понятия как задача (вкладка «Задачи»), ресурс (вкладка «Ресурсы»), а также связи между задачами и назначение задачам ресурсов (вкладка «План»).

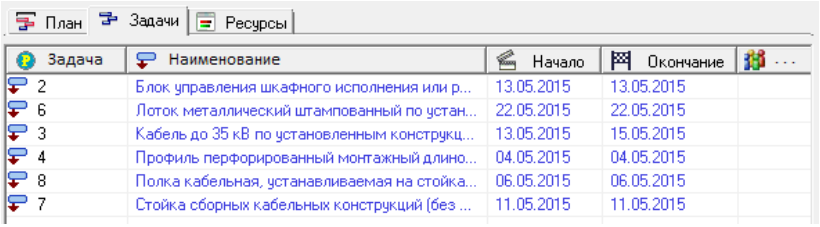

Рис. 17. Понятийная модель плана-графика

Для отображения такой подмодели могут использоваться формы, реализуемые соответствующими прикладными программами: диаграммы Ганта (рис. 18), ресурсные списки, графики использования ресурсов и т.п. Для этого в состав информационной системы включается модуль, выполняющий визуализацию планов-графиков на основе их понятийных моделей.

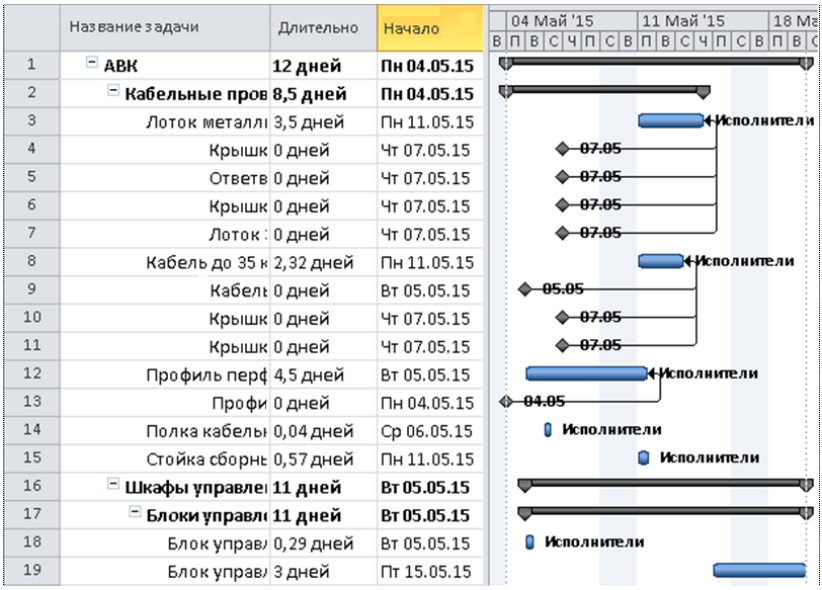

Рис. 18. Визуализация плана-графика (диаграмма Ганта)

Другим примером использования подмоделей является автоматическое создание различного рода документов (файлов) [6]. В этом случае подмодель дополняется правилами выражения (репрезентации) понятий в теле документа. Выразительные средства, используемые для такой репрезентации, будут зависеть от требуемой формы отображения (текст, графика, звук, анимация и др.).

Для отображения подмодели в текстовом виде правила репрезентации могут быть оформлены в виде шаблона документа. При создании шаблона используется специальный язык разметки, позволяющий задать формы выражения понятий в тексте. На рис. 19 приведена формальная грамматика языка текстовой репрезентации понятийных моделей.

Текст состоит из произвольных строк и операторов (правило 1). Для репрезентации понятийной модели используются операторы извлечения, вычисления, установки, выбора и итерации (правило 2).

1) Текст → Строка [Текст] | Оператор [Текст]
2) Оператор → Извлечение | Вычисление | Установка | Выбор | Итерация
3) Извлечение → '**[GET**' Путь ['**#FORMAT**' Формат] '**]**' | '**[$**' Путь ['**#**' Формат] '**]**'
4) Вычисление → '**[LET**' Выражение ['**#FORMAT**' Формат] '**]**' | '**[=**' Выражение ['**#**' Формат] '**]**'
5) Установка → '**[SET**' Путь ['**#VALUE**' Выражение] '**]**' | '**[@**' Путь ['**#**' Выражение] '**]**'
6) Выбор → '**[IF**' Выражение '**#THEN**' Текст ['**#ELSE**' Текст] '**]**' | '**[?**' Выражение '**#**' Текст ['**#**' Текст] '**]**'
7) Итерация → '**[FOR**' Сущность ['**=**' Путь] '**#WHILE**' Выражение '**#REPEAT**' Текст '**]**' | '**[***' Сущность ['**=**' Путь] '**#**' Выражение '**#**' Текст '**]**'

Рис. 19. Формальная грамматика языка репрезентации

Оператор извлечения позволяет получить и вставить на место своего нахождения отформатированное значение, извлекаемое по заданному пути в понятийной модели (правило 3). Оператор вычисления используется для репрезентации в виде текста отформатированного значения некоторого вычисляемого выражения (правило 4). Синтаксис и семантика выражений – как у языков высокого уровня. В реализованной информационной системе в качестве языков для задания выражений используется интерпретируемые языки VBScript и JScript.

Оператор установки служит для изменения значений в понятийной модели и может использоваться, в том числе, для создания временных простых понятий (переменных) (правило 5). Оператор выбора необходим для реализации текстового ветвления в процессе репрезентации модели (правило 6), а оператор итерации – для репрезентации составных понятий (правило 7). Операторы могут быть вложены друг в друга, так как все части операторов представляют собой обычный текст.

На рис. 20 показан шаблон документа, а на рис. 21 – сам документ, полученный в результате репрезентации некоторой понятийной подмодели.

Поз.	Наименование и техническая характеристика	Тип, марка, обозначение документа, опросного листа
1	2	3
[*prt=.Specificatio	[@prti#".Specification." & prt [$$prtj]	
[*cpl=$prti# #1[@ cplj #	[$$cplj.Title]? [$cplj.Components > 0 #, в составе]	[$$cplj.Model]
[*cmp=[$cplj.Compon	[@ cmpk # cplj & ".Components." & cmp [$$cmpk].Title]	[$$cmpk].Model]
[*tbn=TableNote##[=Mid("****",1,Note("TableNote",TableNote(tbn)))] [$TableNote($tbn)]]		

Рис. 20. Шаблон документа

Поз.	Наименование и техническая характеристика	Тип, марка, обозначение документа, опросного листа
1	2	3
	Оборудование	
1	Кондиционер полупромышленный, сплит, Q=9,8 кВт, в составе:	FHQ100B / RR100BW
1.1	блок внутренний подпотолочного исполнения, однопоточный	FHQ100B
1.2	блок внешний с низкотемпературным комплектом «Иней»	RR100BW
2	Пульт управления индивидуальный, инфракрасный	ARC433A73

Рис. 21. Текстовый документ

Аналогичным образом создаются и используются подмодели других устойчивых фрагментов предметных областей и соответствующие им формы визуализации, например:

– графики и диаграммы (графическое представление данных линейными отрезками или геометрическими фигурами) [16];

– инфографика (графическое представление графов, карт, рисунков, формул и т.п.) [29];

– техническая графика (графическое представление схем, чертежей, аксонометрий) [6];

– динамические модели бизнес-процессов в различных нотациях (графическое представление процессов и их текущих состояний) [23, 32].

11. Заключение

В настоящей статье описано построение информационной системы с понятийной моделью предметной области. Модели предметной области названы понятийными, чтобы отличать их от известных концептуальных моделей. В концептуальных моделях задаются понятия (концепты) и разного рода связи (отношения) между ними, несущие часть семантической нагрузки модели. Другая часть семантики концептуальной модели содержится в дополнительных данных, доопределяющих связи между понятиями в виде логических выражений, формул, функций и т.п.

В понятийных моделях связи между понятиями сами являются понятиями, а модель строится на основе выявления и описания абстракций, послуживших образованию (определению) понятий. Отказ от описания ассоциаций в виде связей с различной семантической разметкой делает понятийную структуру предметной области представимой в виде дерева и более наглядной.

Предметная семантика полностью задается интенсионалами понятий, а абстрагирование понятий, формализованное в понятийной структуре предметной области, определяет не более чем структурированность интенсионалов. В этом случае не требуется задавать логические высказывания (формулы, функции), характеризующие понятия и являющиеся правилами вывода. Все, что необходимо для вывода на знаниях, содержится в понятийной структуре предметной области и интенсионалах понятий.

Таким образом, коренное отличие рассмотренного подхода заключается в использовании помимо логики, еще одного семантического инварианта – правил образования и выражения

понятий.[1] Это потребовало определения ассоциаций (связей) между понятиями в виде самостоятельных понятий, а также рассмотрения обобщения как полноценного понятия, которому могут принадлежать сущности, не принадлежащие обобщаемым понятиям.

В других подходах ассоциация понятием не является, выражается связью между понятиями и определяется как единица смысла, посредством которой производится описание предметной семантики. В свою очередь обобщение не предполагает наличие собственных сущностей, которые не являются сущностями обобщаемых понятий.

Информационная система с понятийной моделью предметной области относится к классу интеллектуальных систем и предназначена для хранения и обработки знаний. Она позволяет пользователю описать предметную область: задать понятия, способы их абстрагирования, определить предметную семантику в виде интенсионалов понятий, установить соответствие между предметной областью и сущностями их экстенсионалов описываемых понятий, а также выполнять содержательные запросы к формируемой и постоянно изменяющейся базе знаний и репрезентацию накопленных знаний в других формах.

При таком подходе для актуализации модели при изменениях в предметной области не требуется решения задач, связанных с программированием. Более того, как актуализация модели, так и прикладные задачи предметной области решаются посредством семантически инвариантных для всех предметных областей операций создания, изменения и удаления понятий.

Другими немаловажными достоинствами информационных систем с понятийным моделированием предметной области являются:

---

[1] *Очевидно, что процесс абстрагирования не зависит ни от какой области интерпретации, а определяется только способностями самого познающего субъекта. Следовательно, формализация способов образования и выражения понятий может рассматриваться как теория, претендующая, как и исчисление предикатов, на семантическую инвариантность во всех «мыслимых мирах».*

– прозрачность – использование предельно общих и естественных методов анализа предметной области, унификация обследования предприятия перед внедрением информационной системы;

– настраиваемость – возможность учета отраслевой специфики предприятий, применимость на предприятиях любого размера и сферы деятельности, быстрота и поэтапность внедрения;

– адаптируемость – возможность формирования понятийных подмоделей для конкретных пользователей и их групп, использование единого унифицированного интерфейса пользователя, широкие возможности по настройке прав доступа к понятийной модели или ее части;

– гибкость – быстрое реагирование на изменения в предметной области, простая актуализация понятийной модели в соответствии с изменяющимися внешними условиями, легкая модифицируемость информационной системы;

– открытость – небольшое число унифицированных и устойчивых межслойных интерфейсов, способность взаимодействовать с другими информационными системами на основе семантически инвариантного языка понятийной модели;

– масштабируемость – возможность создания и использования сложных и многоаспектных понятийных моделей, расширяемость информационной системы путем увеличения числа серверов в каждом слое и динамического распределения нагрузки на серверы нижележащих слоев;

– интегрированность – легкий перенос данных от других информационных систем на основе простого и эффективного языка понятийной модели, репрезентация знаний с помощью сторонних программных средств.

Основные трудности, которые имеют место при использовании информационных систем с понятийными моделями – это необходимость освоения новой методологии и технологии моделирования предметной области и репрезентации накопленных знаний, а также отказ от устоявшихся узкоспециализированных форм пользовательского интерфейса.

В итоге, информационная система с понятийным моделированием предметной области является представителем нового поколения информационных систем в методологическом, тех-

нологическом и функциональном плане. Использование понятийной модели создает предпосылки для улучшения прозрачности бизнес-процессов предприятия, способствует оптимизации затрат и повышению инвестиционной привлекательности, уменьшает риски владения информационной системой, а именно:

– проектные риски, связанные с созданием информационной системы;

– технологические риски, связанные с потерей или искажением данных в процессе актуализации модели;

– эксплуатационные риски, связанные с поддержанием информационной системы в работоспособном состоянии и обеспечением независимости от поставщика;

– риски сопровождения, связанные с изменчивостью предметной области.

Литература

1. АБРАМОВА Н.А. *О проблеме рисков из-за человеческого фактора в экспертных методах и информационных технологиях* // Проблемы управления. – 2007. – №2. – С. 11–21.
2. БАРВАЙС ДЖ. *Теория множеств*. Справочная книга по математической логике. – Ч. 2. – М.: Наука, 1982. – 392 с.
3. ВАГИН В.Н., ГОЛОВИНА Е.Ю., ЗАГОРЯНСКАЯ А.А., ФОМИНА М.В. *Достоверный и правдоподобный вывод в интеллектуальных системах* / Под. ред. В.Н. Вагина, Д.А. Поспелова. – М.: Физматлит, 2004. – 704 с.
4. ВОЙШВИЛЛО Е.К. *Понятие как форма мышления: логико-гносеологический анализ*. – М.: Изд-во МГУ, 1989. – 239 с.
5. ВЫХОВАНЕЦ В.С. *Исчисление понятий* // Труды VII Международной конференции «Когнитивный анализ и управление развитием ситуаций» (CASC'2007). – М.: Институт проблем управления, 2007. – С. 31–35.
6. ВЫХОВАНЕЦ В.С. *Репрезентация знаний* // Материалы Международной конференции и выставки «Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта» (CAD/CAM/PDM'2007). – М.: Институт проблем управления, 2007. – С. 49–52.

7. ВЫХОВАНЕЦ В.С. *Прикладной понятийный анализ* // Труды VIII Международной конференции «Когнитивный анализ и управление развитием ситуаций». – М.: Институт проблем управления, 2009. – С. 62–65.
8. ВЫХОВАНЕЦ В.С. *Методы анализа крупномасштабного производства. Понятийный анализ и моделирование* // Труды III Международной конференции «Управление развитием крупномасштабных систем». – М.: Институт проблем управления, 2009. – С. 308–316.
9. ВЫХОВАНЕЦ В.С. *Мультипроблемный анализ и многоаспектное моделирование социальных систем* // Труды V Международной конференции «Управление развитием крупномасштабных систем» (MLSD'2011). – М.: Институт проблем управления, 2011. – С. 158–170.
10. ВЫХОВАНЕЦ В.С. *О понятии понятия* // Труды IX Международной конференции «Когнитивный анализ и управление развитием ситуаций» (CASC'2011). – М.: Институт проблем управления, 2011. – С. 39–42.
11. ГАСКАРОВ Д.В. *Интеллектуальные информационные системы*. – М.: Высшая школа, 2003. – 431 с.
12. ГОРСКИЙ Д.П. *Вопросы абстракции и образования понятий*. – М.: Изд-во АН СССР, 1961. – 352 с.
13. *Макетирование, проектирование и реализация диалоговых информационных систем* / Под ред. Е.И. Ломако. – М.: Финансы и статистика, 1993. – 320 с.
14. КОГАЛОВСКИЙ М.З., КАЛИНИЧЕНКО Л.А. *Концептуальное и онтологическое моделирование в информационных системах* // Программирование. – 2009. – Т. 35, №5. – С. 3–25.
15. НИКАНОРОВ С.П. НИКИТИНА Н.К., ТЕСЛИНОВ А.Г. *Введение в концептуальное проектирование АСУ: Анализ и синтез структур*. – М.: Концепт, 2007. – 236 с.
16. ПАКЛИН Н.Б., ОРЕШКОВ В.И. Бизнес-аналитика: от данных к знаниям. – СПб.: Питер, 2013. – 706 с.
17. РАССЕЛ С., НОРВИГ П. *Искусственный интеллект: современный подход*. – М.: Вильямс, 2006. – 1408 с.
18. СОЛСО Р. *Когнитивная психология*. – СПб: Питер, 2002. – 592 с.
19. СТОБО ДЖ. *Язык программирования Пролог*. – М.: Радио и связь, 1993. – 368 с.
20. ФЕЙС Р. *Модальная логика*. – М.: Наука, 1974. – 516 с.

21. ЦЕЙТИН Г.С. *Программирование на ассоциативных сетях* // ЭВМ в проектировании и производстве. – Л.: Машиностроение, 1985. – Вып. 2. – С. 16–48.
22. ШЕНК Р. *Обработка концептуальной информации*. – М.: Энергия, 1980. – 360 с.
23. ЯЦУТКО А.В., ВЫХОВАНЕЦ В.С. *Динамическое управление бизнес-процессами на основе совмещенных сетей управления и данных* // Инженерный журнал: Наука и инновации. – 2013. – №2 (14). – URL: http://engjournal.ru/catalog/mathmodel/social/530.html (дата обращения: 09.06.2016).
24. BORGIDA A.T., CHAUDHRI V.K., GIORGINI P., et al. *Conceptual Modeling: Foundations and Applications*. – Springer-Verlag, 2009. – 527 p.
25. BRODIE M.L. *On the Development of Data Models* // In: On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages / Eds: M.L. Brodie, J. Mylopoulos, J.W. Schmidt. – New York: Springer-Verlag, 1984. – P. 19–48.
26. CHEN P. *The Entity-Relationship Model – Toward a Unified View of Data* // ACM Transactions on Database Systems. – 1976. – Vol. 1, No. 1. – P. 9–36.
27. *Encyclopedia of Database Systems* / Eds: L. Liu, M.T. Ozsu. – Springer, 2009. – 3752 p.
28. GANTER B., WILLE R. *Formal Concept Analysis: Mathematical Foundations*. – Berlin: Springer, 1999. – 284 c.
29. LANKOW J., RITCHIE J., CROOKS R. *Infographics: The Power of Visual Storytelling*. – Wiley, 2012. – 264 p.
30. OLIVE A. *Conceptual Modeling of Information Systems*. – Springer, 2007. – 455 p.
31. SOWA J.F. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. – Pacific Grove: Brooks Cole Publishing, 2000. – 594 p.
32. VYKHOVANETS V., YATSUTKO A. *Dynamic business process management based on the combined control and data networks* // Preprints of the 2013 IFAC Conference on Manufacturing Modelling, Management, and Control. – Saint Petersburg: Saint Petersburg State University, 2013. – P. 672–677.

INFORMATION SYSTEM USING CONCEPTUAL DOMAIN MODEL

Valeriy Vykhovanets, Institute of Control Sciences of RAS, Moscow, Dr. Sc., Leading research associate (valery@vykhovanets.ru).

Abstract: Conceptual modeling is used for formal specification of domain knowledge and its integration into an information system. We describe an intelligent information system based on the conceptual domain modeling. The system consists of four layers: client, knowledge, logic and data. The knowledge layer is based on a conceptual model which consists of a conceptual structure and a content of concepts. A conceptual structure is specified as a set of concepts, which contains four abstractions: generalization, typification, association and aggregation. A content of each concept is specified by database tables. The significant difference between the proposed conceptual model and others is the description of the association not as a relationship but as an ordinary concept. This implies that the conceptual structure can be represented as a tree. And the information needed for knowledge-based inference is contained in the concept structure itself. We show that the semantic invariance of the conceptual interpretation improves technological and operational characteristics of the information system.

Keywords: subject area, information system, conceptual model, conceptual structure, abstraction of concepts, knowledge base, knowledge representation.

Статья представлена к публикации членом редакционной коллегии Н.И. Базенковым.

Поступила в редакцию 05.07.2016.
Опубликована 31.03.2017.

УДК 621.391
ББК 32.81

РАСПРЕДЕЛЕНИЕ ОГРАНИЧЕННЫХ РЕСУРСОВ В СИСТЕМЕ С УСТОЙЧИВОЙ ИЕРАРХИЕЙ (НА ПРИМЕРЕ ПЕРСПЕКТИВНОЙ СИСТЕМЫ ВОЕННОЙ СВЯЗИ)

Кузнецов А. В.[1],
(*Воронежский государственный университет, Воронеж*)

Статья посвящена формализации задачи распределения ограниченных ресурсов в иерархической социальной системе, причем ресурсы выделяются агенту в соответствии с ролью агента в системе. Рассматривается применение предлагаемой математической модели к организации системы связи специального назначения. Предлагается алгоритм автоматического разбиения системы связи на радиосети в соответствии с предназначением групп абонентов системы связи. При этом ни один абонент заранее не обладает полным знанием обо всей системе связи, а неизвестные заранее сведения о предназначении абонентов устанавливаются в процессе обмена маячками.

Ключевые слова: задача распределения ресурсов, мера сходства графов, абстрактная структура данных, планирование радиосетей, протокол обмена данными, когнитивная сеть.

Введение

Задача распределения ограниченных ресурсов и построения оптимального расписания в сетевых структурах, в том числе и иерархических структурах транспортного типа, достаточно хорошо известна, и каждый год появляется большое количество статей по этой теме. Например, можно упомянуть работу [9], посвя-

[1] *Александр Владимирович Кузнецов, кандидат физико-математических наук, доцент* (*avkuz@bk.ru*).

щенную планированию железнодорожных перевозок, в [13] исследуются задачи распределения ресурсов в иерархических системах, в [14] рассматривается задача согласования входных и выходных параметров участка газотранспортной системы и др. Обычно в таких задачах необходимо распределить ресурсы между вершинами некоторого графа, причем поток ресурсов через ребра графа так или иначе ограничен. Решением таких задач будет, например, оптимальное в каком-либо смысле расписание или распределение ресурсов (не являющихся элементами самого графа) по вершинам графа.

В данной статье, однако, будет описан принципиально иной тип задач распределения ресурсов, часто встречающийся в задачах планирования военной связи (и, в меньшей степени, в гражданской связи), точнее, в задаче распределения и назначения частот. В описываемой задаче ресурсом являются сами ребра некоторого графа (соответствующие каналам связи), которые необходимо распределить между парами вершин в соответствии с заданным шаблоном. Сам шаблон, который далее будет именоваться графом потребностей, аналогичен сети потребностей и возможностей из работы [2]. Вершинами – потребителями ресурсов являются пользователи системы связи или/и их технические средства.

При этом характер взаимоотношений между агентами определенных типов в системе намного более постоянен, чем конкретный состав агентов. Решением такой задачи является вариант распределения каналов, отвечающий некоторым требованиям. Будет предложен подход к решению таких задач, ориентированный на работу не с конкретными агентами системы, а с абстрактной структурой системы агентов, «отвязанной» от самих агентов.

Также стоит отметить, что несмотря на большое количество литературы по частотному планированию (см., например, монографию [1]), в ней основное внимание уделяется вопросу физической совместимости частот и оптимального расположения радиостанций. Вопросы распределения и перераспределения уже заве-

домо электромагнитно-совместимых частот по нескольким тысячам мобильных радиостанций (а это обычная ситуация для воинского формирования тактического уровня) затрагиваются довольно мало. Обычно принято формулировать задачу оптимального назначения частотных каналов как задачу минимальной раскраски графа, в которой, в отличие от классической постановки задачи, смежность вершин определяют ограничения на использование не только одного цвета, но и некоторых их комбинаций (см., например, [3, 10]). Однако такая постановка обычно не учитывает ни иерархии агентов, которым надо назначить частоты, ни возможных изменений в составе агентов, ни случая недостатка частот. При этом подготовка списка электромагнитно-совместимых частот – довольно трудная вычислительно задача и решать ее каждый раз заново при каждом изменении состава радиостанций, как иногда предлагается в более специализированных изданиях, – дело крайне непростое.

Возможны и другие подходы к управлению спектром, применяемые в основном в сетях когнитивного радио и основанные, например, на теории игр [12], но ориентированные также на существенно более простую систему связи (фактически состоящую из равноправных средств, соединяющихся по типу «точка–точка»), чем рассматриваемая в настоящей статье.

1. Постановка задачи

Для дальнейшего изложения нам необходимо дать математическое описание системы связи и «настройки системы связи» (распределения канального ресурса), воспользовавшись понятиями, предложенными автором в работе [5]. Задача распределения ресурсов, упомянутая во введении, в самом простом виде обычно формулируется следующим образом.

1.1. ОБЪЕКТЫ

Даны такие объекты:

1) задано множество классов средств

$$CComm = \{ccomm_1, \ldots, ccomm_s\};$$

2) задано множество средств

$$Comm = \{(comm_1; ccomm_{i_1}), \ldots, (comm_n; ccomm_{i_n})\},$$

где $comm_i$ – идентификатор i-го средства, а $ccomm_{i_j}$ – класс i_j-го средства, $1 \leqslant i \leqslant s$;

3) задано множество агентов

$$Ag = \{ag_1, \ldots, ag_m\};$$

4) между множествами Ag и $Comm$ задано отношение принадлежности

$$f_{have} : Comm \to Ag,$$

и если для $comm \in Comm$ $f_{have}(comm) = ag$, то будем говорить, что агент ag обладает средством $comm$;

5) задано множество классов каналов

$$CF = \{cf_1, \ldots, cf_l\}.$$

В реальных задачах класс соответствует частоте или пропускной способности или подобным характеристикам канала;

6) задана функция возможностей средств

$$f_{comm} : CComm \times CF \to \mathbb{Z},$$

определяющая, сколько максимально каналов заданного типа может образовать средство заданного типа одномоментно;

7) задано множество каналов

$$F = \{(f_1; cf_{i_1}), \ldots, (f_p; cf_{i_p})\},$$

где f_i – идентификатор i-го канала, а cf_{i_j} – класс i_j-го канала, $1 \leqslant i \leqslant p$;

8) задан не содержащий петель маркированный мультиграф потребностей (пример такого графа показан на рис. 1)

$$\mathrm{Req} = (Ag, E_{\mathrm{Req}}, \varphi_{\mathrm{Req}}), \quad E_{\mathrm{Req}} \subseteq Ag^2 \times CF,$$

$\varphi_{\mathrm{Req}} : Ag \times E_{\mathrm{Req}} \to \mathbb{Z}$ – функция инцидентности, которая показывает, какие именно агенты каким количеством каналов какого класса должны быть соединены между собой; в качестве меток ребер выступают классы каналов;

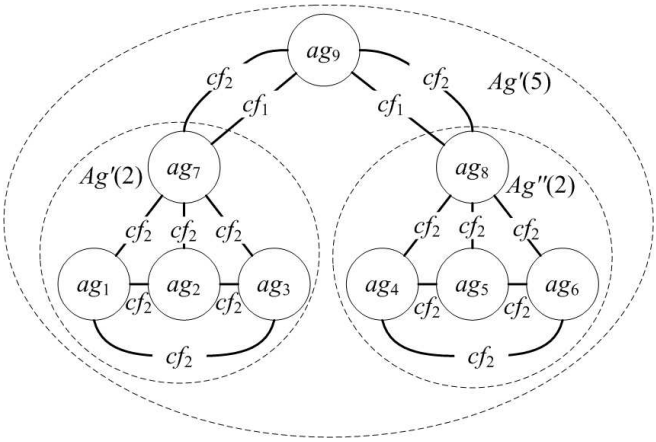

Рис. 1: Граф потребностей Req

9) задана функция емкости класса канала

$$f_e : CF \to \mathbb{N},$$

определяющая, сколько максимально агентов могут одновременно использовать канал данного класса.

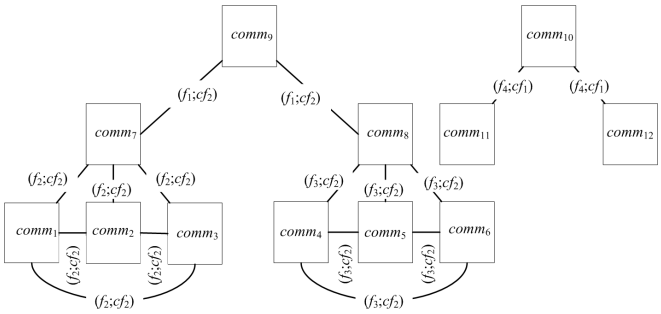

Рис. 2: Граф связи Γ

1.2. УСЛОВИЯ

Необходимо распределить каналы из F по всем средствам из $Comm$, получив в результате граф связи

$$\Gamma = (Comm, E, \varphi), \quad E \subseteq F,$$

$\varphi : Comm \times E \to \{0, 1\}$ – функция инцидентности, так чтобы

1) каждому ребру графа Req соответствовало одно ребро графа Γ или непрерывный путь в графе Γ (случай ретрансляции);

2) каждой вершине ag графа Req соответствовала бы вершина $comm$ графа Γ, такая что $f_{have}(comm) = ag$;

3) количество ребер e графа Γ, маркированных одним и тем же каналом $(f, cf) \in F$, не превышает емкости канала данного класса $f_e(cf)$;

4) количество ребер e графа Γ, маркированных каналом (f, cf) и инцидентных вершине $comm$, не превзойдет возможности средства $f_{comm}(ccomm, cf)$.

Пример графа Γ, соответствующего графу Req, показанному на рис. 1, изображен на рис. 2.

Подытожим, что в исследуемой модели искомой переменной является матрица инцидентности графа Γ, неформально иногда

называемая «вариант распределения радиочастот». Известными параметрами модели являются каналы F, средства $Comm$, агенты Ag, отношения f_{have}, f_e, f_{comm} и граф Req.

Процесс распределения каналов из множества F между агентами и принадлежащими им средствами часто неформально называется «настройкой системы связи». Для упрощения мы будем считать, что все каналы из F заведомо совместимы электромагнитно, иначе необходимо будет добавлять еще метки длин ребер Γ и ограничения на возможные назначения каналов для близких друг к другу средств, после чего применять уже известные методы типа метода координационных колец.

1.3. НЕДОСТАТКИ МОДЕЛИ

Хотя автору и не приходилось видеть статьи с описанием задачи распределения ресурсов такого типа, но она вполне типична в определенных сферах деятельности и решается обычным перебором. Применительно к военной связи (в которой агентами являются должностные лица Вооруженных сил) такой алгоритм был реализован автором в составе большого коллектива в виде разработанного в АО «Концерн «Созвездие» программного обеспечения ЕСВП [7]. При традиционном подходе к обсуждаемой задаче распределения наблюдаются следующие недостатки:

1. Граф Req в буквальном смысле рисуется оператором в течение достаточно продолжительного времени, и при каждом изменении состава множеств $Comm$, Ag необходимо перерисовывать его заново, что при мощности множеств $Comm$, Ag в сотни элементов может затянуться на неопределенный срок.

2. Более того, в реальности множество каналов F не постоянно: обычно каналы выбывают из него со временем непредсказуемым образом в силу преднамеренных или непреднамеренных помех. При существенном изменении множества каналов граф Γ, разумеется, необходимо конструировать заново. При этом отношения между новыми агентами будут иметь примерно такую же структуру, что и между их предшественниками.

3. В реальности агенты из Ag, как правило, неравноправны, и для каждого $ag_i \in Ag$ может быть задан вес α_i, определяющий,

какую долю имеющихся в F каналов следует выделить данному агенту в ситуации нехватки каналов на всех агентов.

4. Граф Г доводится до средств (радиостанций) с помощью запоминающих устройств разного типа. Эта методология восходит к временам, когда средств связи было относительно немного и сгруппировать их было несложно.

5. Граф Г может быть послан и по каналу связи, но возникает вопрос: ведь именно Г и задает настройку канала связи (частоту, как минимум) и как послать Г, когда каналы вообще еще никак не настроены?

6. Наконец, при случайном выходе из строя центра планирования, способного рассчитывать графы Г, вся система вообще постепенно выходит из строя.

Цель настоящей статьи двояка:

1. Дать расширенную формулировку ранее сформулированной задачи, учитывающую вышеупомянутые недостатки.

2. Описать систему связи (безразлично, гражданскую ли, военную ли, хотя первоначально автор работал именно с военными системами связи), в которой каналы назначаются не централизовано, а выбираются каждым средством исходя из характеристик самого средства и положения этого средства в общей системе, продолжив, тем самым, работу, начатую в [8].

Отметим, что в разрабатываемых сейчас за рубежом перспективных системах военной связи [17] проблемы 1–6 активно решаются с помощью технологии «когнитивной радиосети» (cognitive network), в которой узлы отслеживают изменения частотного спектра (spectrum sensing), как и в предлагавшейся ранее методологии «когнитивного радио», и обладают информацией о виде, предназначении и состоянии узлов сети (knowledge plane). В числе прочих технологий применяется динамическое управление спектром, в ходе которого узлы согласованно занимают или освобождают те или иные частотные каналы в зависимости от своего положения в иерархии узлов, изменений помеховой обстановки или других обстоятельств. В настоящей работе, по

сути, рассматривается частная задача динамического управления спектром в когнитивной радиосети – начальное выделение частот в соответствии с ролями агентов.

Ранее автором в работе [8] рассматривалась организация настройки радиосети по заранее не организованному каналу связи посредством обмена маячками настройки (МН) и маячками ответа (МО) между центром управления связью (ЦУС) и настраиваемыми радиостанциями. Для этого все радиостанции снабжались одинаковой сеткой технических каналов, ЦУС рассылал на одном из выбранных из предустановленной сетки каналов МН, тогда как радиостанции сканировали вышеупомянутую сетку каналов и при обнаружении МН отправляли МО. После получения МО ЦУС направлял радиостанции сгенерированные заблаговременно настроечные данные, включающие рабочую частоту и другие параметры. Предполагалось, что настроечные данные рассчитываются исходя из сведений о составе и структуре настраиваемой системы связи. Вопросы закрытия передаваемых данных решались с помощью асимметричной криптосистемы типа RSA, исключающей необходимость предварительного распространения ключей (автор сознательно не будет касаться возможности использования RSA в России для такой цели, поскольку это вопрос более политический, чем технический).

Стоит отметить, что алгоритмы автоматической кластеризации известны достаточно давно (см., например [18]), однако они применяются в основном для организации сенсорных сетей, в которых основным принципом построения кластеров является энергетическая эффективность, а не положение средства в какой-либо иерархии.

2. Распределение ресурсов в зависимости от роли агента и в условиях недостатка ресурсов

Ранее уже упоминалась неравноправность агентов из Ag. Вопросы конкретной организации агентов, которая может определяться графом воинской иерархии или иной социальной структурой, выходят далеко за рамки настоящей статьи. Достаточно

сказать, что каждому агенту $ag \in Ag$ однозначно соответствует вектор признаков агента

$$\mathbf{I} : ag \leftrightarrow (ag^1, \ldots, ag^q),$$

и задан «вес» агента $\alpha(ag^1, \ldots, ag^q) = \alpha(\mathbf{I}(ag)) = \alpha \circ \mathbf{I}(ag)$, $\alpha \circ \mathbf{I}(ag) \in [0, 1]$.

При фактической реализации системы, которая будет далее описываться в настоящей статье, надо считать, что вектор признаков агента хранится непосредственно в каждом агенте. Можно сказать, что каждый агент еще до начала распределения ресурсов «знает» свои (и только свои) признаки. В системах военной связи такими признаками может быть уровень воинской иерархии, к которой относится агент, тип воинского формирования агента, порядковый номер и тому подобные сведения. Отмечу, что в военных системах такие сведения становятся секретными лишь в совокупности и то, что они содержатся в агенте, точнее, в технических средствах, принадлежащих агенту, никак не скомпрометирует всю систему.

Множество всех векторов признаков агентов обозначим как $\mathcal{S}$. Таким образом, сущности подраздела 1.1 дополнятся сущностями:

10) определена функция важности агента

$$\alpha : \mathcal{S} \to [0, 1];$$

11) определена биективная функция

$$\mathbf{I} : Ag \to \mathcal{S},$$

$$\sum_{ag \in Ag} \alpha \circ \mathbf{I}(ag) = 1.$$

В случае нехватки ресурсов (каналов) условие 1 подраздела 1.2 заменится на

1)
$$\sum_{ag\in Ag}\left|\alpha\circ\mathbf{I}(ag)-\frac{|F(ag)|}{|F|}\right|\to\min,$$

где $F(ag)$ – множество ребер Γ, инцидентных хотя бы одному средству из множества $Comm(ag)\subset Comm$, $Comm(ag)=\{comm\in Comm|f_{have}(comm)=ag\}$.

В результате задача распределения каналов в условиях нехватки канального ресурса превратится в задачу комбинаторной оптимизации.

Когда вручную задается граф Req, фактически происходит следующее. Составитель графа имеет в своем сознании ряд предикатов $\mathfrak{P}_i:\mathcal{S}^{|Ag|}\to\{0,1\}$, позволяющих ему сгруппировать агентов в группы, которые должны быть связаны на основании неких признаков:

(1) $\mathfrak{P}_i(\mathbf{I}(ag_1),\ldots,\mathbf{I}(ag_m))\to$
$\varphi_{\mathrm{Req}}(ag_{j_i},cf_{k_i})=n_i\wedge j_i\in J_i\subset\{1,\ldots,m\}\wedge k_i\in K_i\subset\{1,\ldots,l\}$,
т.е. если выполнено некоторое утверждение $\mathfrak{P}_i$ о агентах ag_s, $s=\overline{1,m}$, то некоторые из них должны быть соединены $1\leqslant n_i\leqslant p$ каналами класса cf_{k_i}.

Разумеется, эти признаки формализуемы, и весь граф Req подраздела 1.1 целиком заменяется

8) семейством предикатов $\mathrm{PReq}=\{\mathfrak{P}_i\}$.

Таким образом, возможно заменить сложность постоянного (при каждом распределении ресурсов) конструирования графа Req на сложность первоначальной классификации объектов и выделения признаков их объединения.

При реализации системы связи, которая будет описываться в статье далее, необходимо полагать, что множество PReq также «содержится внутри» агента.

Как кажется, введенными понятиями можно было бы и ограничиться при формализации задачи, но в задачах формирования сетей военной связи (да и вообще в мобильных сетях) состав

множества Ag непостоянен – агенты выбывают со временем, прибывают подкрепления и т.п., хотя структура взаимоотношений между агентами остается по большей части неизменной. Наконец, возможна такая ситуация, когда необходимо последовательно распределить каналы для нескольких одинаковых по организационной структуре и по оснащению, но отличных по составу воинских формирований.

3. Структурное сходство

3.1. ОПРЕДЕЛЕНИЯ

Для полного описания поставленной в статье задачи осталось только определить, какие агенты можно считать сходными. С этой целью можно применить подход из [16]: два графа считаются сходными, если они изоморфны или имеют изоморфные подграфы. Для упрощения будем сперва полагать, что граф Req описывает потребности в связи, например, некоего идеального воинского формирования, состоящего из совершенно одинаковых более мелких единиц.

Определение 1. *Назовем изоморфизмом маркированных одинаковыми метками графов* $\mathrm{Req} = (Ag, E_{\mathrm{Req}}, \varphi_{\mathrm{Req}})$ *и* $\mathrm{Req}' = (Ag', E_{\mathrm{Req}}, \varphi'_{\mathrm{Req}})$ *взаимно однозначное отображение множеств вершин графов*

$$\mathcal{H} : Ag \to Ag',$$

переводящее смежные вершины с меткой β *в смежные вершины с той же меткой* β.

На множестве всех подграфов графа Req (обозначим его 2^{Req}) можно с помощью изоморфизма графов ввести отношение эквивалентности: два подграфа эквивалентны, если они гомоморфны в смысле вышеприведенного определения. Таким образом возможно построить фактор-множество $(2^{\mathrm{Req}}/\sim)$. В вышеприведенном примере идеального воинского формирования классами эквивалентности Req, содержащими более одного подграфа, будут подграфы, соответствующие потребностям в связи

взводов, рот, батальонов и т.п. однотипных воинских формирований.

Будем говорить, что два различных эквивалентных подграфа – максимальные эквивалентные подграфы, если никакие содержащие их подграфы уже не эквивалентны никакому подграфу, кроме себя. Вершины ag_1, ag_2, $\mathcal{H}(ag_1) = ag_2$ двух максимальных эквивалентных подграфов можно отождествить друг с другом, введя отношение эквивалентности уже на множестве агентов, полагая, что $ag_1 \sim ag_2$, если $\mathcal{H}(ag_1) = ag_2$. Обозначим $\mathfrak{Ag} = (Ag/\sim)$.

Определение 2. *Назовем структурой графа* $\mathrm{Req} = (Ag, E_{\mathrm{Req}}, \varphi_{\mathrm{Req}})$ *граф* $\mathrm{Req}' = (\mathfrak{Ag}, E_{\mathrm{Req}}, \tilde{\varphi}_{\mathrm{Req}})$*, в котором каждая вершина* $ag \in Ag$ *заменена соответствующим классом эквивалентности* $\mathfrak{ag} \in \mathfrak{Ag}$*,* $ag \in \mathfrak{ag}$.

Для наиболее полного выделения структуры из Req необходимо пользоваться следующим алгоритмом построения изоморфизма:

Алгоритм 1 (Структурирование)**.**

1) Задается минимальный диаметр выделяемого подграфа $\delta > 0$.

2) Устанавливается $\mathrm{Req}' := \mathrm{Req}$, $i := 0$.

3) Пока $\mathrm{diam}(\mathrm{Req}') > \delta$:

   а) из Req' выделяется максимальный подграф Req_i, изоморфный с изоморфизмом $\mathcal{H}^i$, как минимум, еще одному отличному от себя подграфу Req';

   б) устанавливается $\mathrm{Req}' := \mathrm{Req}_i$ (способ выделения подграфа будет описан в подразделе 3.2);

   в) устанавливается $i := i + 1$.

4) Устанавливается $m := i$.

5) Конструируется общий изоморфизм $\mathcal{H}_\delta$, $\mathcal{H}_\delta(ag) = \mathcal{H}^j(ag)$, если ag является вершиной Req_j и не является вершиной Req_{j+1}.

Очевидно, что построение фактор-множества классов агентов можно производить и через предикаты из PReq, полагая эквивалентными два агента ag_1, ag_2, на которых совпадают значения предикатов из PReq при подстановки в них всех возможных значений других агентов из Ag.

Таким образом, возможно рассматривать при распределении ресурсов не все признаки $(ag^1, \ldots, ag^q)$ агента ag, а лишь те, которые одинаковы для всех агентов класса $\mathfrak{ag}$, $ag \in \mathfrak{ag}$, и распределять ресурсы не между агентами, а между классами агентов, принимая во внимание лишь общее количество агентов. Однократно выделив структуру по какому-либо графу PReq и распределив по данной структуре каналы, можно использовать этот вариант распределения и для других графов потребностей, которые полностью или в части какого-либо своего достаточно большого подграфа можно представить как вариант реализации исходной структуры.

В этом случае возможно добиться как масштабируемости и переносимости решения задачи распределения ресурсов на другие множества агентов с той же структурой, так и ликвидировать зависимости решения задачи от текущего состава агентов.

Остался еще один вопрос: а что если граф потребностей состоит не из в точности изоморфных между собой подграфов, а из в некоторой мере сходных подграфов? В этом случае зададим меру сходства подграфов следующим образом:

Определение 3. *Назовем ε-сходством маркированных одинаковыми метками графов $\mathrm{Req}' = (Ag', E_{\mathrm{Req}}, \varphi'_{\mathrm{Req}})$ и $\mathrm{Req}'' = (Ag'', E_{\mathrm{Req}}, \varphi''_{\mathrm{Req}})$, $Ag' \subseteq Ag$, $Ag'' \subseteq Ag$ взаимно однозначное отображение множеств вершин графов*

$$\mathcal{H}_\varepsilon : Ag' \setminus BAg' \to Ag'' \setminus BAg'',$$

переводящее смежные вершины из $Ag' \setminus BAg'$ с меткой β в смежные вершины из $Ag'' \setminus BAg''$ с той же меткой β, такое что мера несходства графов

$$\mu(\mathrm{Req}', \mathrm{Req}'') = \frac{|BAg'| + |BAg''|}{|Ag'| + |Ag''| - |Ag' \setminus BAg'|} < \varepsilon.$$

Очевидно, что так определенная мера несходства изоморфных графов равна нулю. Данная мера, в сущности, совпадает с биотопическим расстоянием, которое, в свою очередь, является частным случаем расстояния Штейнхауса.

Если задано $\varepsilon > 0$, то можно определить максимальные ε-сходные подграфы аналогично максимальным изоморфным подграфам. То есть два ε-сходные подграфа максимальные, если никакие содержащие их подграфы графа Req не являются ε-сходными ни с одним подграфом кроме себя самих. Отображение $\mathcal{H}_\varepsilon$ порождает отношение эквивалентности $ag_1 \sim ag_2 \equiv ag_1 = \mathcal{H}_\varepsilon(ag_2)$, и справедливы все вышеупомянутые рассуждения о построении структуры графа Req.

Подытоживая, можно сказать, что задача о распределении каналов между агентами по заданному образцу заменяется на задачу о распределении каналов между классами эквивалентности агентов по заданному образцу с последующей заменой классов агентов на попарно различных произвольных представителей этих классов, имеющихся в данный момент функционирования системы в наличии. Иначе говоря, граф Req подраздела 1.1 целиком заменяется

8) структурой графа потребностей $\mathrm{Req}' = (\mathfrak{Ag}, E_{\mathrm{Req}}, \tilde{\varphi}_{\mathrm{Req}})$ или эквивалентным ей набором предикатов.

3.2. СТРУКТУРИРОВАНИЕ ГРАФА ПОТРЕБНОСТЕЙ КАК ЗАДАЧА КЛАСТЕРИЗАЦИИ

Описанное в предыдущем подразделе структурирование графа близко по своей сути к задаче кластеризации, известной в машинном обучении [15]. Для работы с такой задачей определим, что в пространстве векторов признаков агентов $\mathcal{S}$ задана функция $\rho : \mathcal{S}^2 \to \mathbb{R}$, $\rho(sag_1, sag_2) > 0$, $\rho(sag_1, sag_2) = \rho(sag_2, sag_1)$, sag_1, $sag_2 \in \mathcal{S}$, $sag_1 \neq sag_2$, $\rho(sag, sag) = 0$, $sag \in \mathcal{S}$, которая показывает, насколько близки типы двух агентов.

Если бы кластерная структура графа Req была бы известна заранее (в случае военной системы связи это означает, что сразу полностью известно, из каких полков, батальонов и т.п. подраз-

делений состоит воинское формирование, для которого мы хотим распределить частоты), то необходимо было бы лишь проверить кластера примерно одинакового размера (различающиеся не более чем на $\mathcal{E}_0 \geqslant 0$) на изоморфность или на ε-сходство. Хотя существует довольно много методов построения изоморфизмов графов, нам, очевидно, подойдет не любой из них. Для выбора нужного отображения рассмотрим приведенный ниже алгоритм.

Алгоритм 2 (Построение изоморфизма или ε-сходства)**.**

1) Если Ag', $Ag'' \subset Ag$ – два кластера, $0 \leqslant |Ag''| - |Ag'| < \mathcal{E}_0$, то отождествим с вершиной $ag' \in Ag'$ вершину
(2) $$ag'' \in \arg \min_{ag \in Ag''} \rho(\mathbf{I}(ag'), \mathbf{I}(ag)).$$
Для того чтобs запретить выбор «за неимением лучшего» в качестве близких по типу слишком уж несходных агентов, можно ввести дополнительно условие
(3) $$\rho(\mathbf{I}(ag'), \mathbf{I}(ag'')) < \mathcal{E}_1.$$
Получим семейство отображений $\mathfrak{H}_1 = \{\mathcal{M}^i : Ag' \to Ag''\}$.

2) Отберем из $\mathfrak{H}_1$ только инъективные отображения $\mathcal{H}_\varepsilon : D' \to Ag''$, $D' \subseteq Ag'$, получив в результате семейство отображений $\mathfrak{H}_2$. Отметим, что в силу (3) отображение $\mathcal{H}_\varepsilon$ может быть определено не на всем Ag'

3) Обозначим как $B(\mathcal{H}_\varepsilon)$ количество ребер, таких что если ag_1, $ag_2 \in Ag'$ соединены ребром с меткой β, то и $\mathcal{H}_\varepsilon(ag_1)$, $\mathcal{H}_\varepsilon(ag_2) \in Ag''$ соединены ребром с меткой β.

4) Из $\mathfrak{H}_2$ выберем отображение с наибольшей по мощности областью определения D' и с наибольшим значением $B(\mathcal{H}_\varepsilon)$. Таким образом, при выборе $\mathcal{H}_\varepsilon$ необходимо максимизировать величину $|D'| + B(\mathcal{H}_\varepsilon)$.

Если кластерная структура Req заранее не известна, а агенты представлены только своими векторами признаков, поскольку мы заведомо знаем, что агенты организованы иерархически, может быть применен алгоритм иерархической кластеризации с построением дендрограммы (dendrogram). После этого останется лишь

просмотреть дендрограмму и применять к кластерам, оказавшимся на одном уровне дендрограммы, алгоритм 2.

Таблица 1: Значения функции **I** для агентов из рис. 1

Агент	Вектор признаков
ag_1	$(1,1,1,0)$
ag_2	$(1,1,2,0)$
ag_3	$(1,1,3,0)$
ag_4	$(1,2,4,0)$
ag_5	$(1,2,5,0)$
ag_6	$(1,2,6,0)$
ag_7	$(1,1,7,1)$
ag_8	$(1,2,8,1)$
ag_9	$(2,1,9,1)$

Проиллюстрируем все изложенное на максимально упрощенном примере графа Req, изображенного на рис. 1. Пусть представленные на графе агенты имеют вектора признаков, приведенные в таблице 1, т.е. $\mathcal{S} \subset \mathbb{Z}^4$. Здесь если

$$\mathbf{I}(ag) = sag = (ag^1, ag^2, ag^3, ag^4),$$

то ag^1 – это уровень агента в иерархии, ag^2 – номер некоторой общности агентов внутри уровня иерархии (например, «1 мотострелковый взвод 1 мотострелкового батальона»), ag^3 – идентификатор агента, ag^4 – признак того, является ли агент ведущим (например, командиром взвода). В качестве меры сходства объектов ag_1 и ag_2, $\mathbf{I}(ag_1) = sag_1 = (ag_1^1, ag_1^2, ag_1^3, ag_1^4)$, $\mathbf{I}(ag_2) = sag_2 = (ag_2^1, ag_2^2, ag_2^3, ag_2^4)$ примем функцию

$$\rho(sag_1, sag_2) = \sum_{i=1}^{3} [ag_1^i - ag_2^i] + \frac{1}{2}[ag_1^4 - ag_2^4], \tag{4}$$

где $[x-y] = 0$, если $x - y = 0$ и $[x-y] = 1$ во всех иных случаях.

На рис. 1 пунктирной линией показано, что, например, при формировании кластеров из агентов с мерой несходства ρ, строго

меньшей чем 2, получатся кластера $Ag'(2)$ и $Ag''(2)$ (и, возможно, другие), а при формировании кластеров из агентов с мерой несходства ρ, меньшей чем 5, – только один кластер $Ag'(5)$.

Если выбирать в качестве соответствующих пары агентов из $Ag'(2)$ и $Ag''(2)$, для которых ρ минимальна (шаг 1 алгоритма 2), то получим изоморфизм $\mathcal{H} : Ag'(2) \to Ag''(2)$, указанный в таблице 2. Вообще возможно несколько вариантов построения изоморфизма, но в силу конструкции функции ρ (4) «ведущий» агент в одном кластере всегда будет переходить в «ведущего» агента в другом, что принципиально важно в приложениях.

Таблица 2: Конструкция изоморфизма $\mathcal{H} : Ag'(2) \to Ag''(2)$

Агент в $Ag'(2)$	ag_1	ag_2	ag_3	ag_7
Агент в $Ag''(2)$	ag_6	ag_5	ag_4	ag_8
Значение ρ	2	2	2	2

В результате применения алгоритма структурирования с минимальным диаметром выделяемого подграфа 2 будет получен граф, изображенный на рис. 3, $\mathfrak{ag}_1 = \{ag_1, ag_6\}$, $\mathfrak{ag}_2 = \{ag_2, ag_5\}$, $\mathfrak{ag}_3 = \{ag_3, ag_4\}$, $\mathfrak{ag}_4 = \{ag_7, ag_8\}$, $\mathfrak{ag}_5 = \{ag_9\}$. Таким обра-

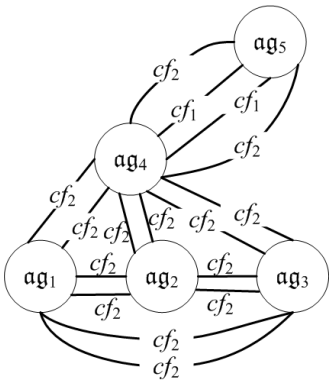

Рис. 3: Структура Req' графа потребностей из рис. 1

зом вместо графа потребностей с девятью вершинами получен граф с пятью вершинами (а если взять меньший минимальный

диаметр выделяемого подграфа, то возможно построить структуру и с тремя вершинами). При построении графа связи Γ это может существенно ускорить процесс распределения частот, так как нужно будет просматривать практически вдвое меньше вершин графа потребностей.

Можно отметить, что в разобранном случае правила построения графа Req из множества PReq, упомянутого в разделе 2, формулируются следующим образом

1. Если $ag_1 \neq ag_2$ и $\rho(\mathbf{I}(ag_1), \mathbf{I}(ag_2)) < 2$, то вершины ag_1 и ag_2 должны быть соединены ребром, помеченным cf_2.

2. Если $(\mathbf{I}(ag_1))^1 = 1$ и $(\mathbf{I}(ag_1))^4 = 1$ и $(\mathbf{I}(ag_2))^1 = 2$, то вершины ag_1 и ag_2 должны быть соединены ребром, помеченным cf_1.

3. Если $(\mathbf{I}(ag_1))^1 = 1$ и $(\mathbf{I}(ag_1))^4 = 1$ и $(\mathbf{I}(ag_2))^1 = 2$, то вершины ag_1 и ag_2 должны быть соединены ребром, помеченным cf_2.

Если задано, что $F = \{(f_1; cf_2), (f_2; cf_2), (f_3; cf_2), (f_4; cf_1)\}$, $f_{have}(comm_i; ccomm_2) = ag_i$, $1 \leqslant i \leqslant 9$, $f_{have}(comm_{10}; ccomm_1) = ag_9$, $f_{have}(comm_{11}; ccomm_1) = ag_7$, $f_{have}(comm_{12}; ccomm_1) = ag_8$, а также что $fcomm(ccomm_1, cf_1) = fcomm(ccomm_2, cf_2) = 16$, $fcomm(ccomm_2, cf_1) = fcomm(ccomm_1, cf_2) = 0$, то из графа Req, показанного на рис. 1, или из структуры Req' графа потребностей (рис. 3) в соответствии с правилами раздела 1.2 легко получается граф связи, приведенный на рис. 2.

4. Пример системы связи, основанной на структурном подходе

Мы привели формальное описание задачи распределения каналов по системе агентов в соответствии с заданным шаблоном, исправляющее практически все недостатки обычного подхода, приведенные в начале работы. Остался лишь один вопрос – можно ли построить систему связи, вообще исключающую единый центр распределения канального ресурса, но при этом способную занимать каналы связи в соответствии с предназначением

каждого агента, а не хаотично? Предложим в качестве ответа на этот вопрос систему связи (рис. 4), которая является дальнейшим развитием системы, описанной в [8].

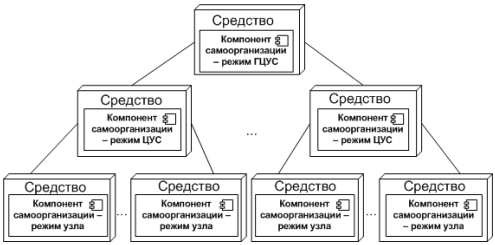

Рис. 4: Схема иерархической самоорганизующейся сети

Пусть каждый агент $ag \in Ag$ «знает» вектор своих признаков (дескриптор) $\mathbf{I}(ag) = (ag^1, \ldots, ag^n)$ и семейство предикатов PReq. Далее, каждый агент может иметь три роли:

1) главный центр управления связи (ГЦУС);

2) центр управления связи (ЦУС);

3) узел сети (УС).

Агенты могут менять свои роли по мере необходимости (в качестве прототипа реального средства связи с такой функциональностью могут быть предложены устройства типа «когнитивное радио», описанные в [6, 11]). Для всех агентов задана одна общая технологическая сетка каналов $F' \subset F$. Функционирование системы связи вкратце выглядит так:

Алгоритм 3 (Функционирование системы)**.**

1. $ag_M \in Ag$ с ролью «ГЦУС» выбирает лучший по неким признакам канал $f_t \in F'$ и начинает на нем передачу своего дескриптора $d_M = \mathrm{I}(ag_M)$ в составе маячка настройки (МН), который также содержит канал ответа $f_r \in F'$. Агенты с другими ролями последовательно циклически перебирают каналы из F' и ожидают получение какого-либо дескриптора. Одновременно «ГЦУС» ждет ответов на канале $f_r \in F'$

2. Если агент ag_1 с ролью «ЦУС» или «УС» и семейством предикатов PReq в процессе перебора канала обнаружил МН с дескриптором $d_2 \in \mathcal{S}$ агента ag_2, такой, что из предикатов в

соответствии с (1) следует, что ag_1 и ag_2 должны быть соединены, то ag_1 отправляет маячок ответа (МО) по каналу f_r, содержащий дескриптор $d_1 = \mathrm{I}(ag_1)$ и идентификатор передатчика $comm \in Comm$.

3. Когда «ГЦУС» получает достаточное количество МО, он строит по полученным из них дескрипторам агентов и соответствующим идентификаторам передатчика фрагмент графа Req и решает по имеющимся внутри него правилам PReq для него задачу распределения канального ресурса. Полученный в результате фрагмент графа связи Γ отсылается всем учтенным отправителям МО в виде пакета настроечных данных (НД). Таким образом, «ГЦУС» разделяет частотный ресурс F между агентами ag_i с ролью «ЦУС» на подмножества F_i, $\cup_i F_i = F$.

4. Агент ag_i с ролью «ЦУС» после получения ПНД настраивает в соответствии с ним свои средства и начинает на некоторой частоте $f_t^i \in F_i$ передачу МН, аналогично тому, как это ранее делал «ГЦУС».

5. Если агент ag_1 с ролью «УС» и семейством предикатов PReq в процессе перебора канала обнаружил МН от «ЦУС» с дескриптором $d_2 \in \mathcal{S}$ агента ag_2, такой, что из предикатов в соответствии с (1) следует, что ag_1 и ag_2 должны быть соединены, то ag_1 отправляет маячок ответа (МО) по каналу f_r, содержащий дескриптор $d_1 = \mathbf{I}(ag_1)$ и идентификатор передатчика $comm \in Comm$.

6. Когда «ЦУС» получает достаточное количество МО, он строит по полученным из них дескрипторам агентов и соответствующим идентификаторам передатчика фрагмент графа Req и решает по имеющимся внутри него правилам PReq для него задачу распределения канального ресурса. Полученный в результате фрагмент графа связи Γ отсылается всем учтенным отправителям МО в виде пакета настроечных данных (НД). Таким образом, «ЦУС» распределяет частотный ресурс F между агентами ag_i с ролью «УС».

Работа алгоритма может быть проиллюстрирована следующей диаграммой UML (рис. 5):

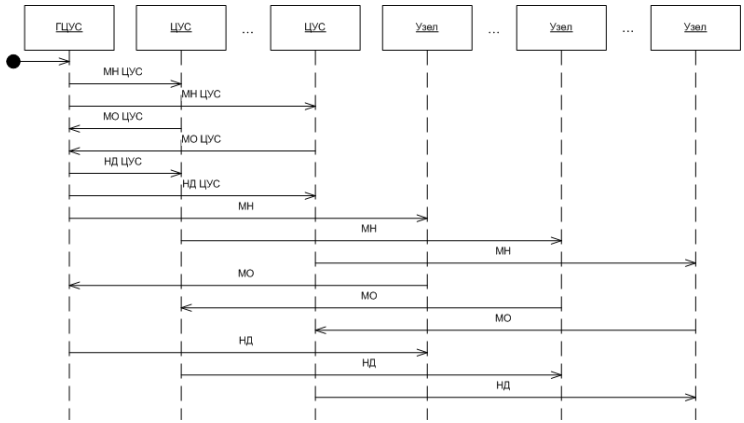

Рис. 5: Обмен маячками

Как упоминалось ранее, при необходимости закрытие и удостоверение подлинности маячков может производиться с помощью асимметричной криптосистемы типа RSA.

5. Выводы и перспективы

В статье предложено математическое описание особого вида задачи распределения ресурсов в соответствии с заданным шаблоном. Также предложен алгоритм структурирования множества потребителей ресурсов. Надо отметить, что эта задача не обязательно связана с системами радиосвязи и распределением частот. «Каналам» могут соответствовать, например, информационные каналы разной пропускной способности между неравноправными пользователями глобальной вычислительной сети с заданными QoS. Также «каналами» могут быть варианты прокладки железнодорожных путей между населенными пунктами при заданной матрице связности населенных пунктов и с ограничением на общую длину рельсов.

Выбор приведенной в статье постановки задачи, использующей понятийный аппарат теории графов, связан с тем, что автор использовал именно такие конструкции при построении имитационной модели движения и связи иерархически организованных

агентов, в том числе и для визуализации изменений состояния системы связи. Вышеуказанная модель основана на клеточном автомате, симулирующем перемещение групп агентов по пересеченной местности [4]. При этом вершины графа Г в каждый момент времени поставлены в соответствие клеткам автомата, в которых находятся агенты. В процессе движения агенты теряют друг друга из видимости из-за особенностей проходимого ландшафта, в результате чего возникает необходимость в перестройке графа связи Г.

Дальнейшее развитие может лежать в направлении совершенствования мер сходства графов потребностей в связи, например, в представлении графа потребностей связи в виде модели состояний и переходов (state transition model) и в замене изоморфизма подграфов в определении сходства подграфов на отношения взаимной симуляции (simulation) или бисимуляции (bisimulation) и в переходе при задании структуры системы связи к сущностям типа коданных (codata).

Литература

1. БЫХОВСКИЙ М.А., ВАСИЛЬЕВ А.В., ЛАШКЕВИЧ А.В. и др. *Основы управления использованием радиочастотного спектра: Частотное планирование сетей телерадиовещания и подвижной связи. Автоматизация управления использованием радиочастотного спектра. Т. 3*. – М.: Красанд, 2012. – 361 с.
2. ВИТТИХ В.А., СКОБЕЛЕВ П.О. *Мультиагентные модели взаимодействия для построения сетей потребностей и возможностей в открытых системах* // Автоматика и телемеханика. – 2003. – №1. – С. 177–185.
3. ГЕНИАТУЛИН К.А., НОСОВ В.И. *Применение алгоритмов вершина-краска и краска-вершина для модификации метода координационных колец при частотно-пространственном планировании системы спутниковой связи с зональным обслуживанием* // Вестник СибГУТИ. – 2014. – №3. – С. 23–36.

4. КУЗНЕЦОВ А.В. *Модель совместного движения агентов с трехуровневой иерархией на основе клеточного автомата* // ЖВМ и МФ. – 2017. – Т. 57, №2. – С. 339–349.
5. КУЗНЕЦОВ А.В., БЕССОНОВ В.В. *Описание математической модели системы связи* // Теория и техника радиосвязи, 2010. – №2. – С. 58–64.
6. КУЗНЕЦОВ А.В., БЕССОНОВ В.В. *Технология создания самоорганизующейся радиосети с функциями когнитивного радио на основе принципов программно-определяемого радио* // Кибернетика и высокие технологии XXI века. XV Международная научно-техническая конференция. – Воронеж: ООО НПФ «Саквоее», 2014. – С. 176–187.
7. КУЗНЕЦОВ А.В., БЕССОНОВ В.В., КРУЧИНИН С.В. и др. *Объектно-лингвистическая модель единой среды визуального проектирования и формат хранения и переноса данных о системе связи на основе XML* // Вестник Воронежского государственного технического университета, 2011. – Т. 7, № 8. – С. 56–60.
8. КУЗНЕЦОВ А.В., ЖАРКОВ С.Н. *Настройка беспроводной сети специального назначения по защищенному радиоканалу* // Электросвязь. – 2016. – №12. – С. 28–35.
9. ЛАЗАРЕВ А.А., МУСАТОВА Е.Г., ГАФАРОВ Е.Р. и др. *Теория расписаний. Задачи железнодорожного планирования. Научное издание.* – М.: ИПУ РАН, 2012. – 92 с.
10. НОСОВ В.И. *Методы обеспечения электромагнитной совместимости в сетях радиосвязи* // Вестник СибГУТИ. – 2007. – №1. – С. 52–56.
11. ОБЕЛЬЧЕНКО М.В., БЕССОНОВ В.В., КУЗНЕЦОВ А.В. и др. *Средство передачи данных телекоммуникационной сети и телекоммуникационная сеть* // Патент России №2549120, опубл. 20.04.2015.
12. ОШМАРИН Д.В. *Распределение канальных ресурсов в сетях когнитивного радио на основе теории игр* // Бизнес-информатика. – 2010. – №4(14). – С. 38–45.

13. ПРИЛУЦКИЙ М.Х., АФРАЙМОВИЧ Л. Г. *Многоиндексные задачи распределения ресурсов в иерархических системах* // Автоматоматика и телемеханика. – 2006. – №6. – С. 194–205.
14. ПРИЛУЦКИЙ М.Х., ДИКАРЕВ К.И. *Распределение ресурсов в иерархических системах с активными элементами* // Вестник ННГУ. – 2012. – №5-2. – С. 181–189.
15. JAIN A., MURTY M., FLYNN P. *Data clustering: A review* // ACM Computing Surveys. – 1999. – Vol. 31, No. 3. – P. 264–323.
16. PELILLO M. *Replicator equations, maximal cliques, and graph isomorphism* // Neural Computation. – 1999. – Vol. 11(8). – P. 1933–1955.
17. REDI J., RAMANATHAN R. *The DARPA WNaN network architecture* // Proc. of the Military Communications Conference (MILCOM'2011), 2011. – P. 2258–2263.
18. SUMAIYA BEGUM D., NITHYA R., PRASANTH K. *Energy Efficient Hierarchical Cluster Based Routing Protocols In WSN - A Survey* // Int. Journal for Innovative Research in Science & Technology, 2014. – Vol. 1, Iss. 7. – P. 261–266.

ALLOCATION OF LIMITED RESOURCES IN A SYSTEM WITH A STABLE HIERARCHY (ON THE EXAMPLE OF PROSPECTIVE MILITARY COMMUNICATIONS SYSTEM)

Alexander Kuznetsov, Voronezh State University, Voronezh, Cand.Sc., associate professor (avkuz@bk.ru).

Abstract: The article is devoted to the resource allocation problem in a hierarchical social system where resources are distributed according to an agent's role in the system. We propose a general mathematical formulation and apply it to the problem of channel allocation in a special-purpose communication network. Each channel belongs to a certain class and the agents' hierarchy impose constraints on the allocation such that a pair of agents with given roles should be assigned with channels from given classes. These constraints are represented as a structure which is called needs graph. An algorithm of the automatic partitioning of a radio communication system in accordance with the roles of the nodes is proposed. None of the nodes have complete knowledge about the entire communication system so an information about roles of the nodes is unknown in advance and is established by a beacon exchange process. We illustrate the proposed approach by an example of a special-purpose military communication networks.

Keywords: resource allocation problem, graph similarity measures, abstract data structures, radio network planning, data exchange protocols, cognitive networks.

Статья представлена к публикации
членом редакционной коллегии В.М. Вишневским.

Поступила в редакцию 27.10.2016.
Дата опубликования 31.03.2017.

УДК 519.687.1/4
ББК 32.988-5

СОВРЕМЕННОЕ СОСТОЯНИЕ И ПЕРСПЕКТИВЫ ИНДУСТРИАЛЬНЫХ ПРИМЕНЕНИЙ МНОГОАГЕНТНЫХ СИСТЕМ

Городецкий В. И.[1], Бухвалов О. Л.[2]
(*Санкт-Петербургский институт информатики и автоматизации РАН, Санкт-Петербург*)
Скобелев П. О.[3],
(*Институт проблем управления сложными системами РАН, Самарский аэрокосмический университет, Самара*)
Майоров И. В.[4]
(*Научно-производственная компания «Разумные решения», Самарский государственный технический университет, Самара*)

Рассматриваются основные тенденции и перспективы развития индустриальных приложений многоагентной технологии, анализируются недавние прогнозы и реальное состояние ее практического применения. Анализируются внешние и внутренние причины, препятствующие широкому промышленному внедрению многоагентных систем и технологий, а также анализируются уроки, которые следует извлечь из этого анализа. Описываются свойства и типы современных и будущих приложений, в реализации которых многоагентные технологии имеют неоспоримые преимущества. Показывается, что многоагентным системам и технологиям в настоящее время нет альтернативы при управлении крупномасштабными объекта-

[1] *Владимир Иванович Городецкий, доктор технических наук, профессор* (*gor@iias.spb.su*).
[2] *Олег Леонидович Бухвалов* (*psyhoveter@gmail.com*).
[3] *Петр Олегович Скобелев, доктор технических наук, профессор* (*petr.skobelev@gmail.com*).
[4] *Игорь Владимирович Майоров* (*imayorov@smartsolutions-123.ru*).

ми сетевой структуры, построенными на принципах самоорганизации.

Ключевые слова: многоагентные системы, индустриальные приложения, самоорганизация, объекты сетевой структуры.

1. Введение

В обзоре Gartner, вышедшем в свет в октябре 2015 года, многоагентные системы (МАС) и технологии (МАС-технологии) включены в список наиболее перспективных информационных технологий (ИТ) следующего десятилетия [26]. Такой точки зрения придерживаются и многие специалисты в области информационных технологий. Однако несколько неожиданным является тот факт, что ИТ-индустрия не спешит с использованием многоагентной технологии, хотя последняя и в среде специалистов числится в списке перспективных уже давно. Естественно встает вопрос о том, почему многочисленные и уверенные предсказания относительно хороших перспектив технологии МАС пока не оправдываются и когда можно ожидать ее успехов на индустриальном уровне.

В настоящей работе дается краткое описание истории развития этого направления, анализируются проблемы, которые существенно тормозят внедрение МАС в практику, и что необходимо сделать, по мнению авторов, для полноправного выхода МАС-технологий на рынок индустриальных внедрений.

Концепция МАС была впервые предложена в середине 1980-х годов. Она сразу была высоко оценена как научным, так и индустриальным сообществами. В исследования и разработки в области МАС и технологий в 1990-е годы были вовлечены достаточно большие научные силы ведущих университетов и ИТ-компаний мира. В результате уже за первые два десятилетия были построены базовые теоретические основы МАС, начались активные разработки в области технологии и инструментальных средств ее поддержки. К началу 2000-х годов было разработано несколько хорошо продуманных методологий создания МАС, началась разработка инструментальных программных средств их поддержки. В 1996 г. была создана общественная организа-

ция FIPA (от англ. *Foundation for Intelligent Physical Agents*), главной задачей которой было научное обоснование стандартов в области агентов и МАС, а уже в 2005 г. она стала одним из комитетов IEEE по стандартизации. В это время ожидалось, что МАС и соответствующая технология готовы занять место лидирующей принципиально новой парадигмы проектирования и технологии разработки современных распределенных интеллектуальных систем индустриального уровня практически любой сложности, причем для самого широкого спектра приложений.

Тогда казалось, что оснований для такой точки зрения вполне достаточно. Действительно, эта концепция выглядела очень привлекательной и естественной для понимания и применения. С самого начала она позиционировалась как парадигма создания сложных систем, построенная на биологических принципах (англ. *bio-inspired* – вдохновляемая биологией, живыми системами), которая предлагает строить системы и решать задачи в том же стиле, в каком они решаются в живой природе и человеческом сообществе, в частности, путем взаимодействий, лежащих в основе самоорганизации. Основной принцип создания концептуальной модели МАС-приложений использует разбиение сложной задачи с множеством взаимодействующих сущностей на относительно простые законченные подзадачи, понятные специалисту. Решение этих задач поручается программными агентам, которые разрабатываются и программируются практически автономно, работают асинхронно и параллельно и взаимодействуют с помощью простой техники обмена сообщениями на языке, близком к естественному, т.е. аналогично тому, как это делается при решении задач в сообществе живых существ, в частности, в человеческом сообществе. Это взаимодействие, реализуемое с помощью диалогов и протоколов, может быть достаточно разнообразным. При этом агенты могут генерировать события и посылать сообщения другим агентам, вырабатывать и согласовывать варианты решений, передавать входные и выходные данные, оценивать результаты решения своих подзадач, формировать задания для других агентов, поддерживать синхронизацию коллективных действий, передавать сигналы обратной связи и т.п. Эта концепция представляется естественной для приложений, в которых участвует много раз-

ных участников с собственными интересами или любых других относительно автономных сущностей. Например, это относится к задачам транспортной логистики, где объектами планирования являются отдельные заказы и грузы, а исполнителями плана являются транспортные средства, водители, станции ремонта и т.п. То же самое относится и к производственной логистике, аналогичными объектами которой являются (на нижнем уровне) заказы, отдельные технологические производственные операции, рабочие и станки, выполняющие эти операции, материалы и т.д.

Особенно привлекательными для МАС-технологий были и остаются до настоящего времени задачи индивидуальной и коллективной робототехники. Специалистам в области робототехники эта концепция с самого начала представлялась идеальной для моделирования коллективного поведения автономных роботов в различных миссиях [33]. Можно указать и ряд других классов приложений, для которых МАС-парадигма вплоть до настоящего времени воспринимается как наилучший, а иногда и просто единственно возможный вариант концептуализации, моделирования и программной реализации. Это прежде всего касается приложений, управляющих сложными крупноразмерными объектами сетевой структуры, а приложений подобного рода на практике становится с каждым днем все больше не только в транспорте и производстве, но и в энергетике, здравоохранении, военном деле и многих других приложениях. Наступающая эра Интернета вещей, в приложениях которой центральным аспектом является именно взаимодействие распределенных автономных сущностей (как «вещей» и их удаленных пользователей, так и «вещей» между собой для более сложных запросов) порождает новый широкий класс приложений, для которых МАС-технология представляется приоритетной технологией.

Одной из самых привлекательных сторон МАС-парадигмы является ее способность естественно и эффективно решать самую трудную задачу разработки сложных программ, а именно программирование взаимодействий множества компонент программы. В концепции и технологии МАС, по существу, эта задача отделяется от программирования агентов и реализуется с помощью диалогов и протоколов их взаимодействия. Важно от-

метить, что в своей базовой формулировке парадигма МАС особо акцентирует внимание на этом факте: она формулируется как *парадигма вычислений на основе взаимодействий* (англ. *computation as interactions*) [35]. Концепция обмена сообщениями с использованием диалогов и протоколов для реализации взаимодействий оказалась очень привлекательной и продуктивной на практике. Не случайно она в последующем получила широкое распространение и в других архитектурах и технологиях разработки сложных интеллектуальных систем.

Убедительным показателем уровня зрелости разработок в области теории и практики МАС был проект *Agentlink III* Европейской комиссии FP-6 (2004-2005), основным результатом которого стала дорожная карта *RoadMap: «Agent Technology: Computing as Interaction*» [35]. Этот документ фактически подвел краткие итоги двадцатилетнего развития парадигмы, модели и технологии МАС, дал оценку практических перспектив МАС и, что самое важное, дал предельно оптимистический прогноз перспектив индустриальных применений МАС до 2015+ г.

Однако уже в начале 2000-х годов в развитии теории и технологии МАС что-то пошло не так, как ожидалось. Например, в программе Европейской комиссии FP5 исследования, посвященные непосредственно развитию теории и инструментальных средств технологической поддержки процессов разработки МАС почти не финансировались. В [38] по этому поводу отмечается, что с середины 2000-х годов публичное восприятие работ в области МАС стало менее значимым. Финансирование проектов по программам FP5 и FP7 Европейской комиссии по направлению ICT (англ. *Information and Communication Technologies*) было сфокусировано на других направлениях, таких как сервис-ориентированные вычисления, ГРИД-вычисления, автономные вычисления и др. Удачные разработки (англ. *success stories*) этого времени в области интеллектуальных приложений, которые были выполнены в этот период ведущими ИТ-компаниями мира, в частности, Apple, Facebook, Google, SAP, совсем не были связаны с МАС или с МАС-технологиями, по крайней мере, в общественном восприятии. В программах FP5 и FP7 поддерживались отдельные проекты, в которых присутствовали компоненты, реализованные с использованием МАС-

архитектур и технологий, но проекты в целом не ассоциировались с исследованиями в интересах развития теории и практики агентов или МАС. На конференции AAMAS'2007 в приглашенном докладе вице-президента и директора автономной лаборатории компании Моторола Дж. Стресснера (John Strassner) одной из ключевых тем был анализ состояния индустриальных разработок в области многоагентных приложений. Автор этого доклада с удивлением говорил, что он обнаружил не более шести МАС-приложений, разработанных за 20-летнюю историю МАС, которые, хотя и приближенно, но могли бы квалифицироваться как индустриальные [45].

Скотт Делоч (Scott A. DeLoach), профессор Канзасского университета, один из ведущих ученых в области МАС-технологий, под руководством которого разработаны методология *O-MaSE* и инструментальное средство a*gentTool*, в работе [24] явно обозначил отсутствие прогресса в широком промышленном применении МАС. По его мнению, несмотря на более чем 20-летние усилия по разработке агентских технологий, сама эта область все еще находится на ранней стадии развития и не достигла достаточной зрелости, например, зрелости объектно-ориентированного подхода (ООП) в программировании, который появился в конце 1960-х годов (Simula в 1967) и в начале 90-х годов стал общепринятым. В [24] выделяется также ряд существенных недоработок в области теории и технологии МАС, которые, по его мнению, требуют уточнения. Они обсуждаются далее в разделе 4 данной работы. Автор [24] считает, что разработчики МАС в первую очередь должны продемонстрировать способность МАС-концепции и технологи создавать сложные адаптивные и самоорганизующиеся распределенные системы промышленного уровня, и именно это для МАС остается главным исследовательским вопросом до сих пор.

В 2013 году была опубликована заметная работа [38], в которой авторы тщательно проанализировали реальное состояние прикладных разработок в области МАС на тот момент. Общая идея этой весьма своевременной, а, возможно, даже несколько запоздавшей работы, сформулирована авторами уже в ее первых строках: «*В то время как имеются убедительные свидетельства важности МАС и технологий как исследовательской об-*

ласти, остается неясным, какой практический эффект от нее имеется к настоящему времени.»[1]

Эта работа интересна в нескольких аспектах. Во-первых, она дает детальный обзор прикладных МАС, которые были разработаны к 2012–2013 гг., что позволяет оценить качество прогноза, данного на этот же срок в работе [35]. Во-вторых, в ней имеется интересная статистика, которая позволяет более глубоко проанализировать реальное состояние МАС-разработок различного уровня зрелости на это совсем недавнее время, а также выявить некоторые причины неудовлетворительного состояния таких разработок.

Обобщая сказанное выше о современном состоянии и ближайших перспективах в области индустриальных применений МАС и технологий, можно утверждать, что в этой области в настоящее время имеются негативные тенденции, тормозящие использование больших потенциальных возможностей МАС. По этой причине представляется важным выявить и проанализировать эти тенденции, а также оценить реальные перспективы МАС как информационной технологии индустриального уровня в ближайшем будущем.

Этот анализ и составляет главную цель данной работы.

Далее в разделе 1 даются краткие сведения о ключевых свойствах МАС и технологий, а также уточняются задачи данной работы. В разделе 2 анализируется состояние и оцениваются перспективы прикладных разработок в области МАС на 2005 г. [35] с прогнозом их динамики на период (2005–2015+) гг. В разделе 3 на основании фактического материала работ [34, 38] и других источников информации дается характеристика современного состояния разработок в области прикладных МАС различного уровня зрелости и приводится сравнение достигнутых результатов с их прогнозом на 2015+ гг., приведенным в [35]. В разделе 4 анализируются негативные тенденции в области теории и практики МАС, а также причины, которые, по мнению

[1] "*While there is ample evidence that Multiagent Systems and Technologies are vigorous as a research area, it is unclear what practical application impact this research area has accomplished to date.*"

авторов данной работы, привели к определенному кризису в области индустриальных приложений МАС и технологий. В разделе 5 намечаются возможные варианты преодоления этого кризиса, которые могли бы способствовать скорейшему и более эффективному использованию практически неограниченного потенциала МАС и как парадигмы концептуализации, и как методологии разработки, и как технологии программирования самых сложных распределенных интеллектуальных систем. В этом же разделе приводятся также примеры удачных разработок МАС-приложений, выполненных под руководством и при участии авторов данной работы и формулируются особенности использованных моделей и технологий, которые сделали возможным практическую реализацию потенциала многоагентных систем и технологий. В заключении резюмируются основные результаты работы.

2. Многоагентные системы как парадигма вычислений на основе взаимодействий

Агентом принято называть автономную компьютерную программу (систему), которая способна к целенаправленному поведению в динамической, непредсказуемо изменяющейся внешней среде. В этом определении агента в качестве его ключевых свойств выделяются *автономность* и *целенаправленность* поведения. *Автономность* агента понимается как его способность функционировать в интересах достижения поставленной цели *без вмешательства* человека или других систем и при этом осуществлять самоконтроль над своими действиями и внутренним состоянием.

Многоагентная система определяется как *сеть слабо связанных решателей частных проблем* (агентов), которые существуют в общей среде и взаимодействуют между собой для достижения тех или иных целей системы. Взаимодействие может осуществляться агентами либо прямым образом – путем *обмена сообщениями*, либо некоторым *косвенным образом*, когда одни агенты воспринимают присутствие других агентов через изменения во внешней среде, с которой они взаимодействуют. МАС может содержать несколько однотипных или разнотипных аген-

тов, которые могут иметь общие и/или различные цели, могут быть распределенными по компьютерной сети, могут быть написаны на различных языках программирования и работать на различных операционных платформах. Существуют различные взгляды на то, какими свойствами должны обладать агенты и существуют различные их классификации. Однако для целей данной работы вполне достаточно введенных описаний агентов и МАС. Отметим, что именно так сформулированы понятия агентской программы и МАС в базовом документе [35].

Взаимодействие рассматривается в МАС как основной способ вычислений и координации поведения множества автономных программных или физических агентов. В соответствии с существующими стандартами агенты взаимодействуют между собой на языке высокого уровня, используя протоколы. Взаимодействие агентов, а значит, и их результирующее совместное поведение, может иметь различные цели. Агенты МАС могут взаимодействовать с целью *кооперативного решения* некоторой общей сложной или крупномасштабной задачи. В этом случае задача разбивается, например, пользователем на более простые относительно автономные задачи, которые поручаются разным агентам. В таком варианте взаимодействие агентов имеет целью координацию локальных решений для достижения некоторого требуемого качества решения задачи в целом. Эта координация может достигаться либо в полностью распределенном варианте, либо с помощью управления с той или иной степенью централизации, реализуемого агентом, специально выделенным для этих целей. Качество решения исходной большой задачи обычно оценивается с помощью некоторой глобальной функции полезности (англ. *global utility function*), значение которой зависит от локальных решений агентов.

Другой характер взаимодействия агентов реализуется в случае, когда каждый агент имеет свои цели, однако он по каким-либо причинам не в состоянии решить задачу самостоятельно, а потому вынужден прибегать к помощи других агентов. Это взаимодействие агентов тоже имеет целью кооперацию, однако, в отличие от предыдущего случая, в этой ситуации агенту в кооперации может быть *отказано*, что определяется дополнительными соглашениями между агентами, которые принято

называть *взаимными обязательствами* агентов (англ. *commitments*).

Если взаимные обязательства агентов относительно слабы и они «помогают» друг другу не в ущерб собственным интересам, то такое объединение агентов в МАС называется *альянсом*. Если агенты объединяются в группы с достаточно *сильными* взаимными обязательствами, как правило, с тем чтобы помочь друг другу выстоять в конкурентной борьбе с другими агентами и/или их группами, то такое объединение агентов принято называть *коалицией*. Агенты коалиции всегда имеют четко оговоренные условия, определяющие, в каких условиях и каким образом они помогают друг другу. Условия, при которых агенты коалиции прекращают помогать друг другу, также обычно четко оговариваются, и эти условия называются *соглашениями* (англ. *conventions*). Если группа агентов решает общую задачу и при этом она действует как один агент, то такое объединение агентов называется *командой*, и в случае такого объединения локальные цели агентов, формируемые динамически, всегда должны быть направлены на достижение общей цели команды.

Еще один тип взаимодействия агентов имеет место тогда, когда агенты не кооперируются, а, наоборот, *конкурируют* друг с другом. В этом случае каждый агент имеет собственную цель и является, как принято говорить, *самозаинтересованным* (от англ. *self-interested*), или эгоистичным. Примеры таких моделей МАС дает электронная коммерция, процессы создания и функционирования виртуальных предприятий и др. При определенных условиях агенты могут как конкурировать, так и кооперироваться друг с другом, при этом они могут легко переходить от конкуренции к кооперации, и наоборот, что называется отношением *коопетиции* (coopetition – от англ. *cooperation* + *competition*).

Например, сотрудники виртуальной организации могут конкурировать за мелкие заказы, но сразу же объединяться, если появляется сложный заказ, который не может быть выполнен ни одним из них поодиночке.

МАС как одно из направлений в современных информационных технологиях формирует *область исследований и разработок*, в которой, с прагматической точки зрения, принято раз-

личать три основных направления с общей теоретической основой (рис. 1) [35]:

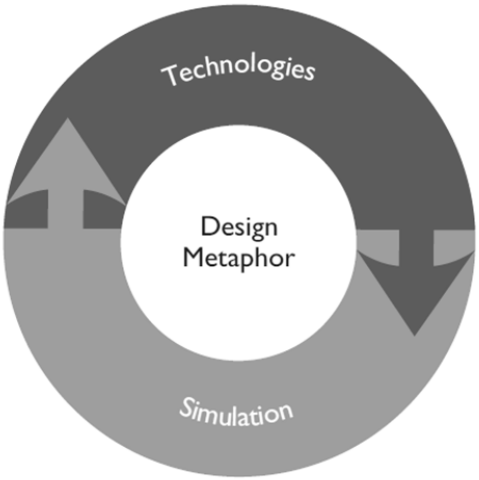

Рис. 1. Прагматика использования понятий агентов и МАС (*рисунок взят из общедоступного ресурса* [35])

– агенты и МАС как *метафора концептуального проектирования* (с акцентом на модульность);

– агенты и МАС как *источник технологий* (с акцентом на *взаимодействия* как принцип вычислений и принятия согласованных решений);

– агенты и МАС как *средство имитационного моделирования* (с акцентом на *автономное поведение* компонент, *взаимодействующих* на основе *протоколов*).

Что касается теоретической основы этих трех направлений, то к настоящему времени предложены различные теории агентов и МАС, и каждая из них обладает своими выразительными возможностями, определяет сложность концептуальной модели агента и МАС. Обычно каждая теория агентов требует разработки специализированной технологии и инструментов для поддержки процессов проектирования приложений и их программирования. К числу наиболее распространенных моделей аген-

тов и МАС относятся реактивные модели, логические модели, *BDI*-модели (от англ. *Belief-Desire-Intension*), поведенческие, био-инспирированные, самоорганизующиеся и другие модели. Естественно, что каждая из них имеет свои достоинства и недостатки, а самое главное – свою область приложений. Как хорошо известно, адекватный выбор модели по отношению к конкретному классу приложений является основным фактором, определяющим потенциальную успешность конкретной прикладной разработки[1]. В разделе 4, посвященном анализу причин негативных тенденций в области индустриальных применений агентов и МАС, этот аспект обсуждается более предметно и детально.

3. 2005 год: состояние прикладных разработок в области МАС и прогноз на 10+ лет

Предваряя анализ прикладных разработок в области МАС и технологий, авторы [35] пишут, что исследования и разработки в этой области к 2005 г. еще не достигли уровня зрелости, необходимого для их использования в индустриальных разработках (возраст примерно 20 лет на 2005 г.). В качестве аргумента они сравнивают возраст МАС-технологий и ООП, которое приобрело практическую значимость более чем через 30 лет. Например, ООП-язык *C++* был создан через 32 года, а *JAVA* – через 39 лет после первых работ по ООП. Авторы также сетуют на слабость методологий для проектирования МАС, разработанных к тому времени, хотя с этим трудно согласиться.

Действительно, к этому времени было создано и уже достаточно длительно тестировалось несколько глубоко проработанных методологий МАС, например, *Gaia* [52], *Tropos* [37], *MaSE* [22], *ADELFE* [16], *MESSAGE* [19], *Prometheus* [39] и ряд других.

[1] *Это естественное и как будто совершенно тривиальное утверждение; тем не менее, как показывает многолетний опыт, оно постоянно нуждается в напоминании*.

Другое дело, что на то время они еще не были поддержаны адекватными инструментальными средствами. Заметим, что отсутствие таких средств было не случайным: оно было следствием ряда принципиальных ошибок (об этом позже), допущенных в области теории агентов и МАС, о которых документ [35] ничего не говорит. Обратим внимание на то, что перечисленные методологии относятся к числу самых ранних разработок в этой области (все они были созданы до 2003 г.). Следует заметить, что активность разработок в области методологий разработки агентов и МАС, а также инструментальных средств для поддержки их проектирования и программной реализации была достаточно высокой до 2005 года и в последующие несколько лет, когда было выполнено более десятка новых разработок. Однако большинство из них к настоящему времени постигла одинаковая участь: их разработки и тестирование прекращены, и только небольшое число методологий и инструментов демонстрирует свою жизнеспособность и в настоящее время (в списке выше они не указаны), но о них речь пойдет позже и особо.

В таблице 1 приведены данные динамики прогноза разработок промышленных образцов МАС на период до 2015+ г. по различным прикладным областям [35]. Из данных этой таблицы видно, что в качестве наиболее перспективных областей использования МАС в то время рассматривались телекоммуникационные системы и сети, производственные системы, транспорт, здравоохранение, финансовая сфера, программное обеспечение и т.п. Общее число МАС промышленного уровня к концу периода прогнозировалось в количестве около 250 экземпляров.

Однако, хотя в период после 2005 г. активность исследований и разработок в области МАС-технологий и средств их инструментальной поддержки не снижалась, практика показала, что этот прогноз оказался слишком оптимистичным. Реальные достижения в этой области оказались намного скромнее.

Таблица 1. Прогноз количества индустриальных разработок прикладных МАС на период до 2015+ г.

	Наименование области приложения	Прогноз числа разработанных приложений по годам		
		2010	2015	2015+
1	Телекоммуникационные системы и сети	17	28	35
2	Производственные системы	15	27	35
3	Транспортная логистика	15	25	32
4	Здравоохранение	12	25	34
5	Электричество, газ, вода	10	17	25
6	Программное обеспечение	8	17	24
7	Финансы, страхование, недвижимость	10	17	24
8	Административные системы	6	12	21
9	Аппаратное обеспечение	3	9	12
10	Строительство	1	6	11
…	…	…	…	…

4. Прогноз и реальность на 2015+

В литературе можно найти сведения о нескольких сотнях МАС-приложений, разработанных в период 2005-2015 гг. в различных прикладных областях. Анализ состояния исследований и разработок в области индустриальных и других применений МАС и технологий с акцентом на тенденции в этой области проведен в ряде работ, среди которых наиболее заметными являются обзоры [38] и [34]. Далее используются в основном сведения и мнения, приведенные в первом из этих обзоров.

Работа [38] анализирует 152 различных приложения, сведения о которых были либо предоставлены авторами разработок, либо почерпнуты из научной литературы. Важно отметить, что более половины проанализированных разработок в последующем, к сожалению, не были подтверждены авторами, так что их практическое использование или продолжение их разработок остается под вопросом [38].

Наиболее заметные и удачные разработки МАС-приложений разного уровня зрелости при различном соотношении автоматической генерации кода и ручного программирования были выполнены в области управления производством, транспортной логистики, в области аэрокосмических приложений и в энергетике. Сведения о некоторых из них приведены в таблицах 2, 3 и 4, соответственно [34, 38].

Анализ уровня зрелости этих и других разработок этого периода показал, что прогноз 2005 г. оказался излишне оптимистичным, причем в нескольких аспектах. В нем, прежде всего, сильно переоценено общее число созданных промышленных образцов МАС, прогнозируемых к 2015 г.

По-иному распределились эти разработки и в зависимости от областей приложений, что, однако, неудивительно, поскольку после 2005 г. появились новые классы актуальных приложений, например, мобильные приложения и мобильные сервисы. Но, наверное, самым неожиданным оказалось то, что темпы появления новых разработок МАС-приложений стали постепенно замедляться. Этот факт отражает реальное падение интереса индустриальных компаний к МАС-технологии. Практика показала, что многие приложения, для которых МАС рассматривались как наиболее перспективная технология программной реализации [35], к 2013 г. были успешно реализованы с помощью других технологий. Среди них наиболее конкурентоспособными оказались сервис-ориентированные технологии, ГРИД-вычисления, автономные, повсеместные, облачные, туманные вычисления и др. Следует заметить, что эти технологии появились значительно позже, однако смогли быстро потеснить МАС-технологии.

Таблица 2. Примеры МАС для управления производством

Назначение/ Заказчик	**Разработчик**	**Предметная область**	**Уровень зрелости**
Production 2000+	Daimler Chrysler, Schneider	Управление производством	Промышленный образец
Car body painting	Daimler-Benz	Управление производством	Программный прототип
BHP Billiton	Rockwell	Управление технологическим процессом	Промышленный образец
Chilled Water System	Rockwell	Распределенное управление	Программный прототип
Cambridge packing cell	U. Cambridge	Управление производством	Лабораторный стенд
FABMAS	Technical U. of Ilmenau	Управление производством	Промышленный образец
PS-Bikes	Universita de Genova	Управление производством	Программный прототип
Axion-Holding	СПИИРАН-НПК «Разумные решения»	Планирование производства	Промышленный образец
Shop Modelarna Liaz	Certicon, Gerstner Laboratory	Планирование производства	Промышленный образец
SkodaAuto	Gedas, Certicon, Gerstner Lab.	Планирование производства	Промышленный образец
Agent Steel System	Saarstahl AG, DFKI Gmbh	Планирование производства	Промышленный образец
SDM Laboratory	Yokogawa	Управление оборудованием	Лабораторный стенд
NovaFlex	Uninova	Управление производством	Лабораторный стенд
ADACOR	Polytechnic Institute of Braganca	Управление производством	Лабораторный стенд
ABAS	Tampere U. of Technology, Schneider Electric	Управление производством	Лабораторный стенд
OntoReA	TU Wien, Rockwell Automation, COPA-DATA	Управление производством	Лабораторный стенд

Таблица 3. Примеры МАС разного уровня зрелости, созданные для управления логистикой

Назначение / Заказчик	**Исполнитель**	**Предметная область**	**Применение**
Air Liquide America	NuTech	Оптимизация логистики	Промышленный образец
Tankers International	Magenta	Планирование логистики	Промышленный образец
Airport ground service operations	Airbus, EADS, Cologne University of Applied Sciences, Группа компаний «Генезис знаний»	Управление работой наземных служб аэропорта, служб питания на борту, расписанием полетов авиакомпании и др.	Программный прототип и лабораторный стенд
Taxi scheduling / Addison Lee	Magenta	Планирование в реальном времени	Промышленный образец
Rent-a-car scheduling and optimization / Avis	Magenta	Планирование и оптимизация в реальном времени	Промышленный образец
Transportation orders consolidation, routing and scheduling / GIST	Magenta	Консолидация, маршрутизация и планирование паллет в реальном времени	Промышленный образец

Таблица 3. Примеры МАС разного уровня зрелости, созданные для управления логистикой (*продолжение*)

Trucks schedul-ing/Prologics	НПК «Разумные решения»	Планирование грузоперевозок с (в реальном времени)	Промышленный образец
Southwest Airlines	BiosGroup	Оптимизация работы наземных служб	Промышленный образец
ABX Logistics	Whitestein	Транспортная логистика в реальном времени	Промышленный образец
MAST	Rockwell	Динамическая маршрутизация това-ров	Моделирование, Лабораторный стенд
MAS-RFiD	U. of Castilla-La Mancha	Управление логистикой	Моделирование
MASDIMA	TAP, LIACC	Адаптация работы авиалинии	Лабораторный стенд

Таблица 4. Примеры МАС для аэрокосмической отрасли и энергетики

Назначение/ клиент	**Разработчики**	**Предметная область**	**Применение**
NASA airspace satellites	NASA	Управление запросами спутников	Промышленный образец
РКК «Энергия»	НПК «Разумные решения»	Динамическое перепланирование полетов и грузопотока МКС, поддержка принятия решений по управлению сменами ГОГУ, нештатным и аварийным ситуациям, программами научных экспериментов	Промышленные образцы
Aerogility	LostWax	Интеллектуальная поддержка принятия решений	Моделирование
Turkey energy forecast	KKK Per. Bsk.	Прогнозирование потребности в энергии	Моделирование
California Energy Commission	AESC, Acronymics	Координирование и планирование	Промышленный образец
Large urban area	Rockwell	Обработка воды	Моделирование

Работа [38], в которой впервые явно прозвучали тревожные ноты относительно перспектив МАС-технологий в конкуренции с другими современными интеллектуальными информационными технологиями, появилась очень своевременно. Она, хотя и в излишне мягкой форме, отражала реальное отношение индустриального сообщества к перспективам МАС-технологий, и это отношение было в 2013 г. и остается сейчас явно не в пользу МАС. Заметим, что работа [38] отражает позицию специалиста в

области MAC, который понимает, что к этому времени потенциал MAC пока не используется в полной мере. Но если бы она отражала позицию индустриальных разработчиков, то критика перспектив MAC-технологий была бы, скорее всего, гораздо более жесткой.

Излишняя мягкость оценок перспектив MAC-технологий в работе [38] обусловлена еще и тем, что основными источниками информации о разработанных и развернутых MAC-приложениях были сами авторы этих разработок. Таких было 103 из 202 приложений, данные о которых рассматривались в качестве исходного материала для анализа в [38], хотя понятно, что в реальности авторы, в силу субъективизма, всегда склонны несколько переоценивать качество и уровень собственных разработок.

Кроме того, нельзя не отметить, что ряд научных разработок в литературе был заявлен авторами как полноценные промышленные системы, хотя на деле они представлял собой всего лишь первые пилоты или исследовательские прототипы.

Обратимся к результатам анализа прикладных разработок в области MAC, приведенным в [38]. После предварительного изучения исходного множества MAC-приложений, информация о которых была предоставлена разработчиками или найдена авторами работы [38] в научной литературе, для анализа было отобрано 152 разработки. Эти разработки анализировались с различных точек зрения. Рассмотрим некоторые из них.

1. *Зрелость разработки*. По этому свойству все разработки разделены на 3 группы:

- *промышленные* системы или близкие к ним (их оказалось 46 из 152);
- исследовательские *программные прототипы промышленного уровня*, которые тестировались на реальных данных, но не были использованы в реальной работе (55 из 152), а также
- *лабораторные исследовательские прототипы* (*пилотные проекты*), которые использовались в учебных исследовательских, и других аналогичных целях (46 из 152).

  Про остальные 5 разработок достоверная информация у авторов [38] отсутствовала.

2. *Тип коллектива разработчиков*. По этому свойству все разработки разделены также на 3 группы:

– *индустриальные* компании (43 из 152, из них 26 разработок отвечают промышленному уровню зрелости);
– *университетское сообщество* (37 из 152, из них только 4 разработанные системы отвечают промышленному уровню зрелости) и
– *смешанные коллективы*, когда в разработках принимали участие одновременно сотрудники обоих сообществ, названных выше (58 из 152, из которых 16 разработок соответствуют промышленному уровню зрелости).

Про остальные 14 разработок достоверная информация у авторов [38] отсутствует.

3. *Тип агентского приложения*. По этому свойству выделено 3 типа разработок:

– *MAC* (таких 125 из 152);
– *автономные* агенты (14 из 152);
– *интерфейсные* агенты (11 из 152).

Для двух разработок тип агентского приложения авторам [38] неизвестен.

Результаты анализа, приведенные в [38], показывают, что активность использования МАС-технологий в реальной жизни оказалась намного ниже прогноза, данного в [35]. Действительно, вместо ожидаемых примерно 200–250 приложений промышленного уровня, к 2015 г. в реальности было разработано всего лишь 46, и при этом по некоторым из них информация о том, что они практически используются после 2013 г., не была подтверждена публикациями. Индустриальное сообщество формирует меньше трети общего интереса к МАС-приложениям (43 из 152 приложений), а основная активность в этой части формируется научным сообществом, т.е. специалистами в области многоагентных систем, *субъективно* заинтересованными в расширении области применимости МАС-технологий.

Важным показателем тенденций этого периода является активное использование в это время методологий и инструментальных средств разработки МАС-приложений. Примерно в 72% случаев авторы разработок использовали ту или иную методологию и/или программный инструментарий. Это косвенно

отражает тот факт, что МАС-приложения в этот период разрабатывались главным образом для тестирования методологий и инструментальных программных систем, и, скорее всего, не были инициированы потребностями индустриальных разработчиков. Это подтверждается тем фактом, что в большинстве случаев разработкой методологии и разработкой приложений с использованием соответствующих методологий занимались одни и те же научные коллективы. Это как раз отражает активность исследований и разработок в области методологий МАС и инструментальных программных систем в период 2005–2015 гг., что уже отмечалось в начале данного раздела.

Что касается предпочтительных языков программирования, то в 82% случаев в качестве базового языка программирования использовались такие стандартные языки как *Java* (55%), *C/C++/C#* (15,6%), *PHP* (7,3%) и *Python* (4,1%).

Среди стран, которые вели и ведут наиболее активные исследования и практические разработки промышленных МАС, лидерами являются США (12 приложений промышленного уровня из 46), Великобритания (6), Испания (5), Германия (4), Италия (4), Чехия (3), Австралия (2), Россия (2), Швеция (2), Швейцария (2), Франция (1), Голландия (1), остальные страны (2). Важно, однако, заметить, что разработчики из США, Австралии и России, на которых приходится треть всех успешных индустриальных разработок МАС (они находятся в активной эксплуатации вплоть до настоящего времени), не используют модели, методологии и программные инструменты, воспринимаемые мировым научным сообществом как наиболее передовые и перспективные. Они используют самые простые модели, например, реактивные и поведенческие, не используют стандартный достаточно сложный *ACL*-язык коммуникаций агентов, принятый FIPA [14], не используют стандартную FIPA-архитектуру агентской платформы [25], а также опираются на собственные механизмы принятия решений агентами.

Интересно отметить, что прогноз по типам приложений, которые, по мнению авторов документа [35], будут представлять наибольший интерес для разработчиков приложений, в целом оправдался. Как и предсказывалось, лидером в области использования МАС-приложений индустриального уровня остаются

транспортная и производственная логистика (9 разработанных приложений), телекоммуникации (9 приложений), электронная коммерция (4), аэрокосмические (4) и военные приложения (3), энергетика (2) и управление бизнес-процессами (2). Неожиданными аутсайдерами в этом аспекте оказались системы в области здравоохранения (2), робототехники (2), в финансовой сфере (0), в области административного управления (0), и в ряде других классов приложений, которым в [35] предсказывались гораздо лучшие перспективы.

Основные выводы, которые можно сделать из материалов данного раздела, состоят в следующем:

1. МАС-технологии развиваются гораздо сложнее, медленнее и труднее, чем это хотелось бы научному сообществу и чем это необходимо индустриальному сообществу. Немало способствует этому и сложившаяся система финансирования научных грантов в Европе, которая требует от исследователей не «стоять» долго в области конкретных технологий, но развивать новые направления и решать конкретные прикладные задачи общеевропейского масштаба. Поэтому одинаково конкурентными оказываются как технологии широкого применения, например, МАС, нейросети (англ. *neuronets*), повсеместные вычисления (англ. *ubiquitous computing*), туманные вычисления (англ. *fog computing*) интеллектуальное окружение (англ. *ambient intelligence*) и т.п., так и весьма частные технологии типа нечеткой логики (англ. *fuzzy logic*), генетических алгоритмов (англ. *genetic algorithms*), муравьиных алгоритмов (*ant colony*) и т.п.

2. Основные исследования и разработки ведутся в основном в научном сообществе. Индустриальные компании, которые на начальном этапе были, по сути, инициаторами МАС-разработок и ранее играли ведущую роль в стимулировании и финансировании разработок в области МАС, в настоящее время фактически полностью ушли от поддержки этих разработок. К ним относятся Motorola, Siemens и др. компании. Некоторые компании, например, IBM, Daimler, NASA, Google активно используют агентов как часть своих разработок, но не называют их агентскими и не акцентируют внимание на агентских компонентах своих разработок. Ряд компаний (British Telecom, например) снизил объемы агентских разработок [38]. Вместе с тем, на рын-

ке появляются новые промышленные компании (Rockwell, DHL и др.), которые пытаются активно использовать МАС-технологии.

3. Интерес индустриального сообщества к использованию МАС и МАС-технологий к настоящему времени значительно снизился. При этом наблюдается высокая скрытая турбулентность, перегруппировка и перестройка зарождающегося рынка интеллектуальных информационных систем и технологий. На этом рынке МАС и МАС-технологии реально имеют много конкурентов, и эти конкуренты активно предлагают свои решения для приложений, которые изначально относились к компетенции МАС, вытесняя тем самым последних с этого рынка или не давая им быстро захватить новые сегменты.

Очевидно, что в настоящее время МАС и технологии во многом еще пока проигрывают своим конкурентам на рынке интеллектуальных информационных технологий индустриального уровня. Но, с другой стороны, очевидно также и то, что в последние годы постоянно расширяется *число новых ниш и новых классов приложений*, в которых имеются большие перспективы для агентов и МАС. Поэтому представляется важным выяснить причины, которые в настоящее время тормозят практическое использование огромного потенциала, которыми МАС и МАС-технологии, бесспорно, обладают. Это вопрос рассматривается в следующем разделе.

5. Что мешает практическому использованию огромного потенциала МАС?

Наверное, самое ценное, что в настоящее время реально предложила теория и практика МАС, это концептуализация сложных систем и задач, которые ими решаются. Именно естественная и понятная концептуализация модели и архитектуры программной реализации систем практически любой сложности привлекает внимание и исследователей, и разработчиков приложений. Если проанализировать литературу по многоагентным системам, а также тематику докладов на ведущих конференциях по МАС, то можно видеть, что подавляющая часть их посвящена концептуальным моделям приложений и архитектур их про-

граммной реализации. Следует заметить, что эта тенденция наблюдается и в настоящее время, хотя и не так очевидно, как это было примерно до 2010 г. На этом этапе разработки приложений МАС предлагают понятный и привлекательный вариант проектирования и здесь все смотрится весьма перспективно.

Однако уже на следующем этапе, когда на основании концептуальной модели нужно построить формальную модель агента, построить архитектуру программно-коммуникационной среды, с помощью которой взаимодействуют агенты, а также описать язык общения агентов, все оказывается, мягко говоря, несколько сложнее, и причины этого нельзя назвать объективными. Сложность шага формализации модели и архитектуры МАС многократно возрастает. Решения, которые были предложены[1] для этого этапа специалистами в области формальных моделей, практически отторгаются теми, кто заинтересован в их практическом использовании.

Рассмотрим этот и другие аспекты теории МАС аспект несколько подробнее.

5.1. ОТСУТСТВИЕ ОБЩЕПРИНЯТОГО ПОНИМАНИЯ КЛЮЧЕВЫХ ПОНЯТИЙ МАС

На это указано еще в работе [24]. Отсутствие четких определений и соглашений по основным понятиям в области МАС сильно затрудняет взаимопонимание между исследователями и разработчиками. Например, большинство профессионалов в области компьютерных технологий согласны с определениями основных концепций ООП, такими как классы, объекты, наследование, инкапсуляция. Они легко оперируют этими понятиями на практике. В то же время специалисты в области МАС имеют различное понимание таких понятий как агент, роль, переговоры, план, возможность и другие. Реальная проблема в области МАС состоит в том, что необходимо уточнить содержание этих базовых понятий и согласовать их взаимоотношения со сходны-

[1] *Возможно, правильнее было бы сказать, что эти решения были навязаны прикладникам.*

ми концепциями ООП, которые также используются в области программирования агентов и МАС.

5.2. ОТСУТСТВИЕ ОБЩЕПРИНЯТОЙ НОТАЦИИ ДЛЯ ПРЕДСТАВЛЕНИЯ МОДЕЛЕЙ И КОМПОНЕНТ МАС

Поскольку еще не сформулировано даже содержание общепринятых определений агентских понятий и их взаимосвязей [24], то отсутствие общей нотации для их описания и описания отношений на их множестве не позволяет исследовать различные модели МАС на практике. Например, после перевода в общую нотацию, удалось найти много схожих свойств в методологиях *O-MasE* [23] и *Prometheus* [39]. Стандартизация моделей и способов их представления очень желательны, поскольку при отсутствии согласованных концепций существует опасность неадекватных оценок новых многообещающих подходов в области МАС-технологий.

5.3. КОНЦЕПТУАЛЬНАЯ И ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ ЛОГИЧЕСКОЙ ФОРМАЛИЗАЦИИ BDI-МОДЕЛИ АГЕНТА И МАС

Почти два десятилетия основные усилия исследователей в области теории МАС были направлены на разработку моделей *интеллектуальных* агентов. Уже в самом начале этих исследований был сформулирован ряд свойств, которыми должен обладать агент. В этом контексте каждый отдельный агент рассматривался как интеллектуальная сущность с собственной развитой базой знаний, или, по крайней мере, моделью знаний, средствами целеполагания и механизмами планирования целенаправленного поведения в непредсказуемой внешней среде. Эта точка зрения достаточно активно пропагандировалась в течение многих лет. Можно сказать, что специалисты в области описания формальных моделей «соревновались» в том чтобы обеспечить агента все новыми и новыми интеллектуальными возможностями в части автономного поведения вплоть до способности определять намерения других агентов. Естественно, что простыми средствами столь мощные интеллектуальные возможности агента описать и реализовать невозможно, что приводило к постоянному усложнению формальной модели агента и МАС.

С самого начала развития теории агентов и МАС в качестве базовой формальной модели интеллектуального агента была выбрана *BDI*-модель (*BDI* от англ. *Belief–Desire–Intention, Убеждение–Желание–Намерение*) [50], в которой знания, убеждения, намерения и механизмы рассуждений описываются в терминах исчисления предикатов, расширенного модальными и темпоральными операторами.

Большинство специалистов и в настоящее придерживаются понятий *BDI*-модели агента в МАС и их логической формализации. Однако эта модель, с одной стороны, определенно сложна для понимания и является существенным барьером в интерпретации базовых понятий МАС [24], включая понимание самого термина *BDI*. По мнению автора работы [24], необходимо сосредоточиться на интерпретации тех понятий МАС, которые акцентируют внимание не на способах формального представления модели агента и МАС, а на понятиях, существенных для описания агентов и МАС как средств концептуализации и технологии разработки распределенных систем, использующих принципы самоорганизации. Такими понятиями для *BDI*-архитектуры являются, например, поведенческие и мотивационные свойства агентов.

Важно отметить, что сама по себе *концептуальная основа BDI-модели* достаточно естественна и убедительна. Суть ее состоит в том, что эта концепция оперирует двумя типами понятий: ментальными и поведенческими. К *ментальным* понятиям относятся *события*, *убеждения*, *цели* и *взаимные убеждения*. *События* отражают (представляют формально) то, что происходит в реальном времени с агентами или объектами внешнего мира. Они имеют в качестве атрибутов *тип события* и *имя агента*, генерирующего событие, или агента, с которым событие ассоциировано. *Убеждения* – это знания агента о *внешнем мире*, а именно утверждения, в истинности которых агент убежден в текущий момент времени. Но они, во-первых, могут изменяться во времени и, во-вторых, могут быть неверными с самого начала или стать таковыми в некоторый момент времени. В этом их отличие от знаний. *Цели* – это некоторые состояния агента, которые он стремится достичь. Эти состояния называются целевыми состояниями агентов. Поскольку агенты могут обмени-

ваться знаниями и часто должны координировать свое поведение, то важным понятием рассматриваемой модели является понятие взаимных убеждений агентов. *Взаимные убеждения* агентов складываются из убеждений группы агентов. Поскольку разные агенты могут иметь различные убеждения, то взаимные их убеждения – это та часть индивидуальных убеждений, в которых сходятся все агенты группы.

К поведенческим понятиям *BDI*-модели агента относятся *индивидуальные обязательства* и *соглашения* агента, его *индивидуальные намерения*, а также *общие* (коллективные) *обязательства* и *намерения* команды агентов. *Индивидуальные обязательства* агента – это цели (состояния, подцели дерева целей), которые он должен достигнуть в соответствии теми задачами из общего множества задач агентов, которые агент взял на себя. *Индивидуальное намерение* (англ. *intention*) агента – это обычно конкретное *действие*, которое он собирается выполнить для достижения индивидуальной цели. *Соглашениями* называются условия, при выполнении которых агент может отказаться от своих индивидуальных обязательств. В зависимости от типа кооперации (альянс, коалиция или команда), эти соглашения могут быть более жесткими или менее жесткими. Например, если МАС является командой агентов, то эти соглашения определяются однозначно и являются достаточно жесткими [20]. Перечисленные понятия могут быть положены в основу верхнего (предметно-независимого) уровня онтологии МАС. Один из примеров онтологии поведенческих понятий BDI-модели агента и МАС можно найти в работе [8].

Если обратиться к истории развития искусственного интеллекта, то можно вспомнить, что в нем в течение достаточно долгого периода господствовали логические языки представления знаний и логический вывод в качестве механизма рассуждений[1]. Можно легко увидеть большую аналогию этой истории с тем,

[1] *Одним из примеров является печально знаменитый японский проект вычислительных машин пятого поколения, который опирался на логический язык Пролог и который полностью провалился уже в 1980-е годы.*

что происходило и, к сожалению, все еще происходит в настоящее время в области теории агентов и МАС.

Действительно, были попытки использовать логическую формализацию понятий, представляющих *BDI*-концепцию агента и МАС, т.е. описать ментальные понятия агента и модель их распределенного взаимодействия в терминах исчисления предикатов первого порядка (ИП-1). Однако достаточно быстро было обнаружено, что выразительные возможности ИП-1 недостаточны для описания моделей ментальных понятий агентов. Приведем простой пример.

Пусть выражение $[q/p]\varphi$ означает, что формула $[q/p]\varphi$ получена из формулы φ путем подстановки в нее подформулы q вместо подформулы p (в каком-то числе случаев ее вхождения в формулу φ), и при этом подформулы p и q имеют одинаковые истинностные значения во всех интерпретациях. В классической логике всегда будет справедливо

$$|= (p \leftrightarrow q) \rightarrow \varphi \leftrightarrow [q/p]\varphi,$$

где символ $|=$ трактуется как отношение, сохраняющее истинностное значение. Это свойство принято называть экстенсиональностью[1]. Содержательно оно означает, что истинность формулы φ не изменится при использовании любого количества подстановок подформулы q вместо подформулы p, если обе они имеют одно и то же истинностное значение. Однако это свойство делает невозможным адекватное описания свойств ментальных понятий и их взаимосвязей. Покажем это на простом примере из области футбола роботов [49].

Пусть в убеждениях агента истинны такие факты:

1. p: «*мяч находится у своего игрока*»,
2. c: «*мячом владеет своя команда*» и
3. q: «*у меня мяча нет*».

Очевидно, что истинной является также формула «*с поскольку p*», где бинарный предикат «*поскольку*» является одним

[1] *Термин происходит от английского слова extension, который переводится как расширение. Здесь он применяется в смысле расширяемости логики за счет использования подстановок указываемого далее типа.*

из конструктивов, который может использоваться в схемах рассуждений с ментальными понятиями. Если в последней формуле использовать подстановку истинной подформулы q вместо истинной подформулы p, то в результате получится формула «*с поскольку q*» («*мячом владеет своя команда*» поскольку «*у меня мяча нет*»), про истинность которой уже ничего определенного утверждать нельзя. Таким образом, предикат «*поскольку*» не является экстенсиональным, а потому не может быть представлен в языке классической логики. Таких примеров можно привести много [50]. Заметим, что описываемая проблема имеет семантический характер.

Слабые выразительные возможности ИП-1 были преодолены за счет его обогащения модальными и темпоральными операторами. А далее все получилось по аналогии с продвижением логической модели в области искусственного интеллекта. Аналогия эта состоит в активном продвижении логической модели, обогащенной модальными и темпоральными операторами в качестве базовой модели агента и МАС. Именно эта модель и получила название *BDI*-модели.

Но эта модель даже теоретически намного сложнее ИП-1, и вряд ли можно надеяться на ее практически приемлемую эффективность. Заметим, что в модели *BDI*-агента взаимодействие агентов предполагается, в основном, на уровне простых диалогов. Это означает, что агенты должны быть настолько «интеллектуальными», чтобы самостоятельно строить модели других агентов и модель внешней среды для принятия решений, что на практике оказалось невозможным. Кроме того, такой подход означает отказ от сути многоагентной парадигмы, которая формулируется как вычисления на основе взаимодействий.

В итоге результат активного продвижения логической формализация *BDI*-модели получился ожидаемым: как и логическая модель искусственного интеллекта, она породила много новых математических проблем и интересных задач в области неклассических логических исчислений, что сильно способствовало развитию соответствующих разделов математики. Но это совсем не способствовало созданию практически реализуемых моделей и технологий в области МАС. Более того, это затормозило практическое использование МАС технологий на десятилетие, как

минимум. Действительно, в настоящее время имеется глубоко разработанная логическая теория *BDI*-модели агента, которая теоретически позволяет строить агентов высокого уровня интеллектуальности, способных к планированию целенаправленного поведения и автономному принятию решений, к распределенной координации поведения в достаточно сложных ситуациях. Но эти возможности отвечают уровню теорем существования в математике, так что при попытках практического использования логической модели *BDI*-агента уже для относительно простых приложений возникают серьезные проблемы, главным образом проблемы вычислительной сложности. Даже для простых МАС-приложений модели знаний агентов получаются чрезвычайно громоздкими, а механизмы рассуждений, использующие вывод в логических исчислениях с модальными и темпоральными операторами, – совершенно неподъемными. Ярким примером подобной ситуации являются *BDI*-модели коллективного поведения роботов, которые активно финансировались DARPA в течение почти десятилетия. В конце 1990-х годов они рассматривались сообществом специалистов в области многоагентных систем как самые значительные (англ. *influental*) достижения [46, 47]. Эти работы достаточно детально проанализированы в [10]. Однако на практике эти результаты демонстрировались авторами на очень простых примерах. В начале 2000-х гг. оба эти проекта были закрыты. Можно сказать, что хорошую модель коллективного поведения агентов в автономной миссии погубила логическая модель *BDI*-агента с использованием модальных и темпоральных операторов.

Однако в это время, хотя и на вторых ролях, развивались и другие модели агентов (о них речь пойдет позже). И именно эти модели, главным образом, были использованы в большей части тех приложений, которые рассматриваются как успешные в [34], [38] и в ряде других обзоров по практическому использованию МАС и соответствующих технологий. Два таких примера приводятся в конце раздела 6.

5.4. СТАНДАРТЫ FIPA (FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS)

Поскольку парадигма и технология МАС выросли из научного направления, которое называлось распределенный искусственный интеллект и которое имело целью решение практических задач, то вопрос о стандартах стал темой исследований еще в середине 1990-х годов, когда была создана общественная организация FIPA. В задачи этой организации входило научное обоснование стандартов в области МАС-технологий. Но поскольку FIPA была создана тем же научным сообществом, которое продвигало в практику логическую модель *BDI*-агента и МАС, то эта же модель рассматривалась FIPA в качестве базовой модели МАС и при разработке стандартов.

Например, язык коммуникации агентов *ACL* (англ. *Agent Communication Language*) [14] в этом стандарте использует весьма сложный язык для описания содержания сообщений, которыми обмениваются агенты. Предшественником этого языка был язык *KQML* (*Knowledge Query and Manipulation Language*). Разработка этого языка началась еще в 1970-е годы, когда он позиционировался как язык представления знаний в системах искусственного интеллекта с исчислением предикатов в своей основе. Современная его версия, получившая название *ACL*, представляет собой достаточно мощный и выразительный язык интерпретирующего типа, который манипулирует понятиями онтологии и способен представлять содержание сообщений, которыми обмениваются агенты, на языке, близком к естественному языку. Но он привносит в стандарт все черты логической модели *BDI*-агента со всеми вытекающими отсюда последствиями из-за проблем вычислительной сложности и большой загрузки каналов связи. С другой стороны, он труден для понимания и использования разработчиками приложений, а свобода, которую язык *ACL* предоставляет разработчикам в модификации компонент синтаксической оболочки этого языка (так называемых перформативов), используется главным образом в исследовательском сообществе специалистов.

На практике же в большинстве случаев оказывается возможным обойтись значительно более простыми специализированными языками. Примером специализированного языка явля-

ется язык обмена сообщениями, принятый в сервере RoboCup [41]. Он использует только необходимые и достаточные средства общения агентов и поэтому достаточно эффективен. Например, для двух команд симуляционного футбола по 11 агентов каждая оказывается достаточным обычного персонального компьютера выпуска 2004 года для того, чтобы реализовать интенсивный обмен сообщениями между агентами команд в режиме реального времени. Совсем другой, более простой и прагматичный язык используется в модели МАС, предложенной сначала в инструментах компании *Magenta*, а затем развитый (теми же разработчиками) в инструментах Группы компаний «*Генезис знаний*» и НПК «*Разумные решения*» [5, 13], где для представления текущих знаний активно используется понятие *сцены* (*модели ситуации*) как хранилища сведений о сети потребностей и возможностей (ПВ-сети) в окружающем мире.

Иная модель обмена сообщениями по сравнению со стандартом FIPA используется также в программном инструментарии *Cougaar* [21], который разработан в США по проекту DARPA для его использования в военных приложениях. В нем обмен сообщениями поддерживается архитектурой доски объявлений, на которой агенты предлагают свои сервисы (доступные другим агентам как подключаемые модули) и ищут необходимые им сервисы по мере необходимости. При этом доска объявлений значительно расширяет множество доступных сервисов за счет доступности веб-сервисов в соответствии с *UDDI*-протоколом [48]. Эта возможность реализуется специальной компонентой доски объявлений, которая называется *сервлет*. Она поддерживает коммуникации с объектами Интернет по *http*-протоколу. Этот сервис доступен для использования всеми подключаемыми модулями узла. Отметим, что сервлет может возвращать данные в форматах *HTML*, *XML*, серии *Java*-объектов и бинарных данных.

Важно подчеркнуть, что модели агентов и МАС, предложенные в инструментах компании *Magenta* и *Cougaar*, оказались наиболее успешными в части индустриальных разработок. Можно также заметить, что все перечисленные инструменты не используют стандартную платформу FIPA.

К большому недостатку FIPA-стандартов следует отнести также и то, что спецификации FIPA вообще не рассматривают проблемы *параллельного программирования*, хотя МАС – это концепция, изначально ориентированная на параллельные вычисления.

5.5. ОТСУТСТВИЕ ГИБКИХ ПРОМЫШЛЕННЫХ МЕТОДОВ И ТЕХНОЛОГИЙ ДЛЯ РАЗРАБОТКИ МАС-ПРИЛОЖЕНИЙ

В работе [24] отмечается, что разработчики промышленных многоагентных систем сталкиваются с множеством агентских методологий и в большинстве случаев – с отсутствием программных инструментов их поддержки. Это, возможно, связано с тем, что новые методологии являются недостаточно гибкими и трудными для распространения на широкие области применений. В большинстве инструментов агенты служат только для дополнительной переупаковки объектов при помощи ООП, что дает некоторые преимущества, но значительно снижает возможности агентского подхода. Поэтому насущной задачей является интеграция существующих методологий разработки и инструментов их поддержки в единую хорошо определенную технологию.

Одной из причин современного кризиса в области МАС и технологий является ошибочная стратегия в области методологий и средств разработки МАС. С самого начала исследователи в области МАС придавали большое значение созданию методологий проектирования и программной реализации МАС, причем в период 2005–2010 гг. эти исследования и разработки велись наиболее активно. В этот период общее мнение состояло в том, что хорошая методология разработки и мощное средство инструментальной поддержки ее программной реализации позволят в значительной степени автоматизировать процесс создания прикладных МАС промышленного уровня, так что их создание можно будет поставить на поток.

В период до 2010 г. было разработано более десяти методологий проектирования и разработки МАС, если не считать большого количества других менее значимых разработок в области методологий, которые привнесли мало нового. К числу наиболее перспективных и наиболее обоснованных методологий

следует отнести такие методологии как *Gaia* [52], *Tropos* [37], *MaSE* [22], *ADELFE* [16], *MESSAGE* [19], *Prometheus* [39], *SADDE* [42] и ряд других. Большинство разработок в области методологий МАС сопровождалось также созданием инструментальных программных средств их поддержки (см., например, [36]). Эти средства использовали либо авторскую методологию разработки, либо одну из списка приведенного выше, которые в сообществе специалистов считались передовыми. Например, к этому времени были разработаны такие средства как *agentTool* [27], *Zeus* [31], *agentBuilder* [15], *PASSI* [18], MASDK [28] и ряд других.

Все эти инструменты для своей разработки потребовали длительных усилий больших коллективов высокопрофессиональных исследователей, разработчиков, программистов и тестеров. Обычно на разработку и тестирование методологии и поддерживающего ее программного инструментария уходило не менее 10 лет.

Как правило, передовые методологии этого периода использовали концепцию разработки, управляемую моделью (англ. *model-driven engineering*). В методологиях такого типа концептуальная модель прикладной МАС, формальные модели ее стандартных компонент (модели агентов, онтологии, протоколов, сообщений и пр.), а также модели взаимодействия компонент и архитектура целевого программного продукта описывались на некотором формальном языке, при этом предпочтение отдавалось графическим языкам.

Такой формальный язык должен был автоматически поддерживать непротиворечивость моделей компонент системы разного уровня абстракции и генерацию протоколов взаимодействия агентов системы. Предполагалось, что формальная модель приложения, построенная таким способом, далее будет компилироваться в код программы на языке высокого уровня (например, в код *C++*, *Java* или в код другого языка аналогичного уровня). После этого система должна была дополняться компонентами, для которых код может быть написан только вручную (это касалось «не агентских» компонент). Полученный высокоуровневый код должен был после этого компилироваться в исполняемый код.

Нетрудно видеть, что инструментальная поддержка подобной методологии проектирования и программной реализации МАС-приложений оказывалась достаточно «тяжелой» и сложной, особенно в тех случаях, когда использовалась логическая модель *BDI*-агента и МАС. Как правило, методологии и инструментальные средства опирались на стандарты FIPA. По этой, а также по ряду других причин добиться желаемой эффективности методологии разработки и программной реализации приложения, так же, как и вычислительной эффективности целевого приложения, не удавалось. В итоге такой методологический и инструментальный подход, за некоторыми редкими исключениями, себя не оправдал. Только отдельные программные прототипы удавалось с его помощью довести до экспериментальных образцов.

Эта стратегия разработки прикладных МАС активно пропагандировалась и практически использовалась в период (2005–2015+) гг. В литературе можно найти сведения о нескольких сотнях МАС-приложений, разработанных в этот период в различных прикладных областях (см. также таблицы 2–4 в разделе 3 данной работы). Но среди этих разработок большинство не получило последующего развития и применения на практике, и их разработка была прекращена.

Та же судьба к настоящему времени постигла и большинство методологий и поддерживающих их инструментальных программных средств: разработка и поддержка большей части из них в настоящее время прекращена, поскольку эти разработки не привели к прорыву в области индустриальных применений МАС и технологий.

Повторим, что основной причиной этого, наряду с рядом других причин, является неверный выбор базовой модели агента и МАС, который был поддержан «тяжелыми» стандартами FIPA.

5.6. ПОЗИЦИОНИРОВАНИЕ МАС И МАС-ТЕХНОЛОГИЙ В ПРИКЛАДНЫХ ОБЛАСТЯХ

Как уже отмечалось ранее, существует много классов приложений, которые ранее позиционировались как приложения, для которых модель, архитектура и технология МАС являются

наиболее перспективными по сравнению с другими технологиями. Однако практика показала, что многие из них уже давно имеют удачную программную реализацию индустриального уровня с помощью других технологий, в то время как их многоагентные реализации либо явно проигрывают таким реализациям, если они имеются, либо эти приложения к настоящему времени вообще не имеют агентской реализации. Причиной такой ситуации является то, что с самого начала МАС рассматривалась как достаточно универсальная ИТ-парадигма и технология, однако практика показала, что это явно не так.

К настоящему времени область применений МАС и технологий, в которых они имеют неоспоримые преимущества, фактически не определена и еще только складывается. Перечисление классов потенциальных приложений в [35] базировалось в основном на поверхностных ассоциациях и потому было неубедительно. В этом документе, скорее, перечисляются те прикладные области, в которых агентские технологии могут быть использованы, причем наряду с другими областями. В результате более зрелые технологии оказались в ряде случаев более успешными, что способствовало утверждению негативных мнений о МАС-технологиях и снижению интереса индустриального сообщества относительно их возможностей.

Вместе с тем можно привести много различных прикладных задач в самых разных областях, в которых МАС-технологии являются единственным вариантом технологии разработки на множестве всех существующих технологий. К ним относятся задачи, которые решаются в распределенном варианте с применением методов распределенного решения задач на базе принципов самоорганизации и эволюции. В таких задачах классические модели, методы и средства оптимизации быстро «захлебываются» в переборе вариантов, а МАС-технологии оказываются способными строить хотя бы допустимые решения. И эту область приложений следует очертить заранее.

Примером такой задачи в производственной логистике может быть построение согласованных производственных планов и сменно-суточных заданий рабочих в десятках цехов крупного машиностроительного предприятия с миллионами сборочных операций в день. Аналогичной задачей является составление

железнодорожных расписаний на участках интенсивного движения сотен поездов. К такому же классу относятся задачи управления цепочками поставок для тысяч наименований товаров, движущихся с разной скоростью, и т.д. Примеры этих и других подобных промышленных приложений МАС с измеримыми результатами в части повышения эффективности работы предприятий подробно описаны в [1, 28]. Заметим, что формальные модели, используемые в них, свободны от большинства недостатков, перечисленных в данном разделе.

Можно утверждать, что для МАС-технологий пришла пора извлекать уроки и сузить возможный спектр применения от общесистемного применения в распределенных системах – к созданию действительно сложных адаптивных систем, реализуемых на основе принципов самоорганизации, для решения сложных, многокритериальных, плохо формализованных и вычислительно трудоемких задач в реальном времени, как, например, управление ресурсами, распознавание образов, извлечение знаний и т.д. [11, 12, 40, 43].

6. Перспективы развития индустриальных МАС-технологий

Несмотря на описанные негативные аспекты истории развития теории и технологии МАС, она вполне укладывается в классическую схему зарождения и выхода новой технологии из лабораторий в промышленные применения, а этот процесс на практике никогда не бывает линейным (рис. 2).

В начале своего развития в 1990-е годы на стыке объектного программирования, параллельных вычислений, телекоммуникаций и искусственного интеллекта (вспомним синие экраны Norton Commander в DOS и первые интернет-модемы в то время) МАС стали вызовом в программировании и потребовали больших усилий выдающихся ученых и программистов для первых успешных разработок.

Несложно предсказать, что процесс развития любой технологии, тем более такой революционной по своей сути, как МАС-технология, будет и в дальнейшем весьма нелинейным. В этом процессе потребуется сделать еще много попыток его транс-

формации, успешных и безуспешных, прежде чем удастся сделать МАС-технологию продуктивной и коммерческой.

Рис. 2. Предыстория и перспективы развития МАС и МАС-технологий в контексте стандартного цикла развития новых технологий

Помимо рассмотренных выше концептуальных и алгоритмических проблем, разработчики МАС часто встречаются и с не менее значимыми специфическими технологическими проблемами, которые свойственны системам, использующим *локальные* механизмы самоорганизации и эволюционные вычисления. Заметим, что именно эти механизмы являются базовыми при поиске решений в крупномасштабных задачах с оптимизацией управления объектами сетевой структуры, т.е. в той прикладной области, которая является для МАС-технологий наиболее перспективной. Так, в работах [40, 43] отмечается, что при создании МАС, решающих задачи распределения, планирования, оптимизации и контроля использования ресурсов на множестве объектов, формирующих сеть, зачастую возникают специфические эффекты и проблемы, затрудняющие поиск решения, которые перечисляются ниже:

– В условиях постоянных модификаций управления динамическим объектом, работающим в реальном времени, бывает очень трудно оценить, насколько текущее решение далеко от динамически изменяющегося «оптимального» решения, так что алгоритм управления обычно пытается лишь «догнать» этот оптимум и всегда опаздывает.

– Решение зависит от предыстории, например, от порядка наступления тех или иных событий, что обычно обусловлено динамикой объекта управления и нелинейностью как самого управляемого объекта, так и нелинейностью управления («чувствительностью ко времени»).

– Зачастую появляется «эффект бабочки», при котором малые изменения на входе системы приводят к неожиданным для наблюдателя большим изменениям на выходе, что обычно связано с переходом самоорганизующейся системы из одного локального оптимума (устойчивого состояния) в другой.

– Реакция системы может непредвиденно замедляться для наблюдателя в случае возникновения длинной цепочки изменений, вносимых в управление объектом; это обычно бывает обусловлено большим числом вариантов и экспоненциальным характером роста сложности вычислений на локальном участке или неконтролируемым расширением волны переговоров в зависимости от длины цепочки агентов, затрагиваемых изменениями.

– При повторном запуске системы с теми же самыми входными данными решение может оказаться другим; это обычно бывает обусловлено недетерминизмом параллельных асинхронных процессов и малыми флуктуациями решений или наличием в алгоритмах работы системы случайных механизмов выбора в эвристиках, используемых в процессе поиска «наилучшего» решения, что приводит к тому, что разные компоненты программы в разных реализациях получают иные входные данные.

– В силу динамики объекта управления решение, полученное с помощью эволюционного подхода, невозможно «откатить» назад, поскольку ситуация, как правило, при такой попытке будет уже иной.

– При попытках доработки управления вручную, например, при попытках изменить ручным способом расписание использо-

вания ресурсов, случаются непредвиденные для пользователя существенные изменения расписания как в калейдоскопе и т.д. При возникновении каждой такой проблемы разработчикам на практике приходиться искать собственные решения, во многом зависящие от предметной области.

Наконец, как отмечается в [5, 11, 12], разработчиков МАС подстерегают и серьезные проблемы организационного, коммерческого и тому подобного характера, обусловленные конкуренцией на рынке информационных технологий. В качестве примеров можно привести следующие:

– Разработки МАС в сфере управления часто вторгаются в самую «запретную» область принятия решений, критическую для бизнеса, где «все уже занято» (хотя в наличии обычно имеются только учетные системы) или процессы принятия решений совершенно не формализованы.

– Процесс поиска решений в МАС является распределенным, причем с акцентом на самоорганизацию, что на практике противоречит традициям иерархического управления и, что немаловажно, традициям проектирования систем по принципу «сверху–вниз», поскольку самоорганизующиеся системы разрабатываются по принципу «снизу–вверх» [9].

– Продажа инновационных разработок МАС требует глубокого участия предметных специалистов, а не только разработчиков, и занимает много времени (от 3 до 24 месяцев).

– Разработка МАС приложений, важных для бизнеса (по опыту крупных проектов), требует затрат сил и времени примерно в 3–5 раза больше, чем ожидается вначале.

– Объем разработки многоагентной системы управления («движка») занимает не более 25% общего времени, в то время как все остальное время тратится на другие вопросы, связанные с учетом, базами данных, интерфейсом пользователя, интеграцией и т.д.

– Разработка первой версии многоагентной системы наиболее трудоемка и занимает от 3 до 6 месяцев (минимум) даже при наличии опыта использовании МАС-технологии.

– Внедрение МАС часто занимает больше времени, чем сама разработка, поскольку требует выявления и отработки правил

принятия и согласования решений, а также ее интеграции с уже существующими информационными системами заказчика.

– Примерное соотношение затрат труда (%) по основным фазам проекта МАС разработки: проектирование – 10, разработка – 20, тестирование – 15, поставка, внедрение и обучение – 35, поддержка – 20.

– Разработанная система должна «выживать» в условиях постоянных ошибок пользователей, наличия неполных данных для проектирования, поступления неточных данных и т.д.

– Пользователи должны быть мотивированы на внедрение системы и в идеале премироваться как по результатам внедрения, так и по величине достигаемого повышения эффективности в работе, чтобы работать на одну цель с разработчиками.

– Пользователи должны иметь возможность вручную дорабатывать расписания, поскольку всегда остаются факторы, которые не представляется возможным учесть при автоматическом принятии решений в системе, и т.п.

Преодоление указанных проблем, безусловно, требует значительных усилий, но в случае успеха результаты окупают все вложения, обеспечивая на практике следующие преимущества МАС-технологий:

– Позволяют создавать интеллектуальные системы нового класса для решения сложных проблем и дают результаты, сопоставимые с результатами работы специалистов.

– Повышают эффективность использования ресурсов, качество обслуживания, снижают затраты денег и времени, риски и штрафы в экономике реального времени.

– Решают сложные задачи и предлагают рациональные, пригодные на практике решения за счет перехода от полного перебора - к поиску конфликтов и их разрешению путем достижения компромиссов.

– Поддерживают работу в реальном времени с быстрой и гибкой реакцией на события.

– Обеспечивают индивидуальный подход для каждого участника процесса использования МАС, включая заказчиков продукции и услуг, исполнителей, субподрядчиков и т.д.;

– Помогают снизить зависимость от персоналий в принятии решений.

– Снижают затраты на разработку за счет повторного использования кода при переходе к новым сферам применений и усложнении решения.

– Дают возможность моделирования типа «Если–то» для оптимизации и прогнозирования развития ситуации.

– Создают надежную и масштабируемую платформу для роста сложности решаемых задач и развития бизнеса без роста численности управленческого персонала.

Можно утверждать, что в настоящее время в области МАС идет активная работа над первыми ошибками и преодоление недостатков, что станет залогом новых достижений.

Вместе с тем, полноценный выход МАС на плато продуктивности на уровне отраслевых стандартов на наш взгляд можно ожидать не ранее 2020–2030 гг.

К числу наиболее перспективных областей использования МАС-технологий относятся такие *новые* прикладные области, как

– *Аэрокосмическая отрасль* – коллективное самоорганизующееся поведение беспилотных космических и летательных аппаратов (БПКА и БПЛА), управление группировками малых спутников, тренажеры для летчиков и авиадиспетчеров, космическая логистика и др.

– *B2B-сети производственных и транспортных* предприятий, стратегическое планирование и оперативное управление производством, сетевая логистика (транспортная и др.) – это одна из основных областей, где уже активно используются агентские решения и технологии и где все еще остается большое поле для использования сетевых МАС-технологий [7].

– *Военные приложения* – имеется много косвенных признаков и свидетельств активности индустриальных разработок в этой области за рубежом, однако по большей части эта информация закрытая.

– *Коллективная робототехника*, *автономные миссии роботов* – пока находится в стадии изучения возможностей, однако специалистами всего мира рассматривается как одна из самых перспективных областей для многоагентных приложений.

– *Смарт грид*, *виртуальные электростанции* и другие приложения в области *энергетики*. Эти разработки уже начаты 4–

5 лет тому назад, и перспективы МАС в этой области подтверждаются ростом активности соответствующих разработок.

– *Здравоохранение* – решения в области окружающей среды для поддержки здоровья населения (англ. *ambient assisted living, personal healthcare и др.*).

– Распределенные системы *наблюдения и обеспечения безопасности*, сенсорные сети, интеллектуальные пространства, и туманные вычисления (англ. *fog computing*), например, в интересах задач Интернета вещей [17, 51].

– *Мобильные приложения* – это совсем новая область МАС-приложений, задачи которой по большей части только формулируются. Важно отметить, что по оценкам компании Gartner до 40% будущих мобильных приложений в течение ближайших 10 лет будут построены на основе многоагентных технологий.

Однако уже в настоящее время имеются МАС-приложения, которые демонстрируют свои преимущества на практике. Приведем два примера таких приложений, которые разработаны под руководством и при участии авторов данной работы. Они реализованы с помощью авторских технологий, которые во многом избегают тех недостатков, которые в разделе 4 отнесены к главным причинам того, что многоагентные технологии к настоящему времени пока еще не заняли среди других интеллектуальных технологий ведущего места.

6.1. САМООРГАНИЗУЮЩИЕСЯ B2B-СЕТИ ПРЕДПРИЯТИЙ

B2B-сетями называют вид экономического и информационного взаимодействия множества предприятий, имеющий целью координацию планирования и исполнения распределенных бизнес-процессов *в интересах самого бизнеса*. Это означает, что потребитель конечного продукта, производимого предприятиями сети, отсутствует в ней. Узлами сети являются предприятия, которые на долгосрочной основе кооперируются в выполнении потока заказов, поступающих в произвольные узлы сети в режиме реального времени. Полагается, что B2B-сеть является открытой, т.е. к ней могут присоединяться новые узлы, некоторые узлы могут выходить из сети навсегда или возвращаться в нее снова. Такая сеть является сетью парных взаимодействий (англ. *peer-to-peer*, p2p-сетью), в которой как коммуникации, так

и содержательные взаимодействия поддерживаются с помощью p2p-протоколов.

В общем случае полагается, что заказы могут поступать в сеть через произвольные ее узлы в режиме реального времени. Заказ может состоять из сложно структурированного множества подзаказов, которые по тем или иным причинам не могут быть выполнены одним узлом сети. Причины могут быть технологическими (предприятие не имеет возможности выполнить все компоненты заказа), ресурсными (предприятия перегружено заказами и может выделить на новый заказ лишь ограниченные ресурсы) или экономическими (распределенный процесс выполняется специализированными предприятиями и это обходится дешевле) и др. Более подробное описание существа B2B-сетей, основных особенностей организации их функционирования, а также алгоритмов распределенного планирования и составления расписаний исполнения взаимосвязанных заказов, поступающих в сеть, имеется в [43]. Далее рассматривается только одна из наиболее сложных и трудоемких задач, решаемых в B2B-сетях в режиме реального времени, а именно задача координации расписаний распределенного выполнения заказов.

Многоагентная архитектура узла B2B-сети представлена на рис. 3.

Алгоритм распределенной координации локальных расписаний узлов B2B-сети строится на основе обмена метаданными между ее узлами в процессе составления ими локальных расписаний. В составлении локальных расписаний узла принимают агенты трех типов:

– *Агент узла* или узел некоторого подразделения предприятия (англ. *Department Agent, DA*) ответственен за составления локальных расписаний узла и за взаимодействие с *DA*-агентами других узлов при решении задачи распределенной координации на метауровне.

– *Агент ресурса* участвует в выполнении *Протокола контрактных сетей* [44] при составлении локального расписания.

– *Локальный агент заказа* участвует в составлении локального расписания узла, отвечая за соблюдение технологии выполнения локальной части заказа.

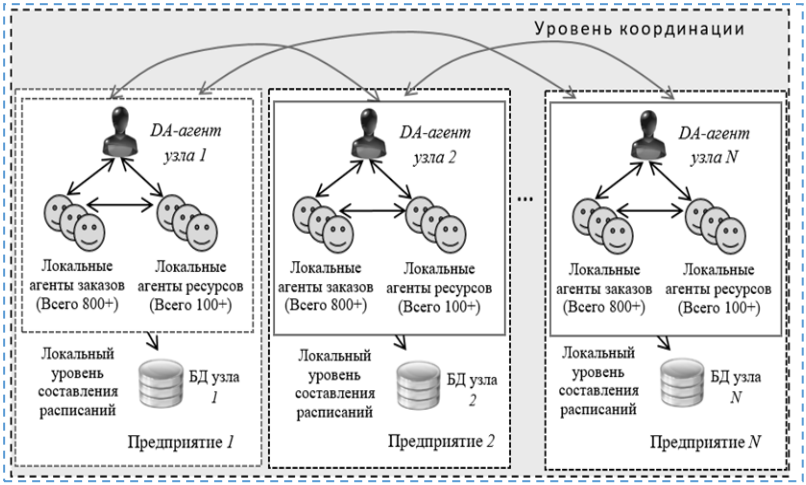

Рис. 3. Архитектура МАС распределенного составления расписаний выполнения взаимосвязанных заказов в производственной B2B-сети

Детали взаимодействия этих агентов при составлении *локального* расписания описаны в [2]. В системе имеется также Агент заказа (англ. *Order-agent*, *OA*), который является агентом мета-уровня. Он владеет информацией о заказе в целом, например, о самом раннем допустимом времени начала исполнения заказа и о самом позднем допустимом времени его завершения. Важно отметить, что в этой архитектуре информация о ресурсах и технологиях узлов доступна только агентам соответствующих узлов, что решает задачу конфиденциальности

Задача координации локальных расписаний решается на основе взаимодействия *DA*-агентов узлов, а также их взаимодействия с мета-агентами *OA*. Протокол из взаимодействия описан в [3, 6]. Поддержка этого взаимодействия выполняется с помощью распределенной программно-коммуникационной платформы, которая реализует традиционные сервисы платформы, минимальный набор которых включает в себя сервисы белых и желтых страниц, а также сервис адресации сообщений на основе того или иного p2p-протокола [6, 7, 29].

Однако архитектура и реализация этой платформы (среды исполнения агентов и среды их взаимодействия) значительно

отличатся от той, которая диктуется стандартами FIPA. В принятой архитектуре (рис. 4) средой исполнения всех агентов является *Akka System*, которая играет роль *локальной* компоненты p2p-платформы агентов. Эта компонента реализует функции управления потоками, диспетчеризацию сообщений, управляет жизненным циклом акторов (агентов) [30], является каркасом для дополнительных модулей системы и отвечает за ряд других важных сервисных задач. Все узлы B2B-сети разделяют общую онтологию *Ontology*, т.е. одну модель данных.

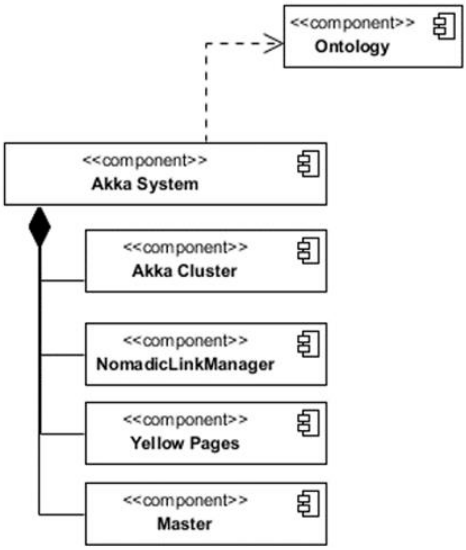

Рис. 4. Архитектура информационно-коммуникационной платформы

Akka Cluster – это модуль расширения платформы *Akka*, предоставляющий возможности объединения отдельных *Akka System* в единую p2p-сеть. Специальная компонента *NomadicLinkManager* отвечает за управление ссылочной целостностью распределенных структур, таких как распределенный технологический процесс, где каждая операция должна ‘знать”, какой элемент бизнес-процесса ей предшествует и для каких из его элементов предшественником является она сама. Компонент

Yellow Pages – это агент, управляющий поиском и распространением информации о сервисах, предоставляемых отдельными узлами B2B-сети. Под сервисами в данном случае понимаются возможности по исполнению тех или иных типов бизнес-операций, а также атрибуты исполнителей, необходимые для определения возможности назначения бизнес-операции тому или иному исполнителю. Агент *Master* выполняет ряд важных инфраструктурных задач, таких как управление стартом узла, координация компонент одного узла между собой, супервайзинг и контроль других компонент узла, а также предоставление интерфейсов для корпоративной информационной системы и внешних систем, являющихся клиентами B2B-сети. Отметим, что в данной архитектуре отсутствует агент белых страниц, который предусматривается стандартом FIPA. Это связано с тем, что *Akka* по имени актора (агента) всегда может определить его адрес.

Эффективная реализация рассматриваемой системы (см., например, [2, 3]) с помощью многоагентной архитектуры и технологии стала возможной благодаря тому, что в ней используется простая модель агента, в основе которой лежит понятие актора [30], и эта модель намного проще *BDI*-модели. В этой реализации взаимодействие распределенных компонент поддерживается нестандартной агентской платформой, которая намного «легче», чем распределенная p2p-агентская платформа, предусматриваемая стандартом FIPA. Технология *Akka* естественно подходит для реализации многопоточных и распределенных систем. Она предлагает хорошую модель организации параллельных/распределенных вычислений, основанную на модели акторов, которая избавляет разработчиков от ряда острых проблем, не решенных в классических FIPA-платформах, таких как *JADE* [32]. Взаимодействие агентов основано на передаче некоторых событий, а не сложных *ACL*-сообщений, что также повышает вычислительную эффективность многоагентной реализации рассматриваемого весьма сложного приложения, сохраняя при этом все концептуальные преимущества парадигмы МАС.

6.2. АДАПТИВНОЕ УПРАВЛЕНИЕ МОБИЛЬНЫМИ БРИГАДАМИ

Рассмотрим задачу управления крупной региональной сетью аварийно-ремонтных бригад. Организационно система управления включает в себя центр приема заявок от населения по телефону (далее для краткости «*колл-центр*» от англ. *call center*), центральную диспетчерскую и специально оснащенные мобильные бригады техников по ремонту и обслуживанию объектов в сетях газоснабжения, водоснабжения, водоотведения, телекоммуникации и др. в районе с сотнями тысяч жителей.

Заявки от населения («аварийные заявки») характеризуются срочностью устранения аварии (средняя, важная и критическая), местом проведения работ и видом работ. Помимо заявок от населения на устранение аварий, существуют также планово-профилактические работы, которые проводятся этой же службой бригад согласно утвержденному ежемесячному плану графику. Чем сложнее работа, тем более высокой квалификации требуются специалисты для ее выполнения. Для некоторых видов работ требуется наличие у бригады дополнительного оборудования, например, сварочного. Каждая бригада имеет в распоряжении специальный автомобиль для выездов по заявкам, оснащенный необходимым оборудованием. В любой момент времени местоположение бригады определяется датчиком GPS/Глонасс навигации специального автомобиля бригады. Каждый бригадир имеет мобильный терминал, через который он «видит» план выполнения заявок, назначенных на его бригаду, и может делать фактические отметки о выезде, о начале и об окончании работ по каждой заявке. Аварийные заявки имеют горизонт планирования на текущий день, плановые – на месяц. Невыполненные заявки плана текущего дня переносятся на следующий день.

Содержание задачи управления мобильными бригадами состоит в следующем.

1. Автоматически распределить и выполнить наискорейшим образом открытый пул заявок, имеющийся на текущий день, в порядке срочности их выполнения в соответствии с приоритетами заявок – *критическая*, *важная*, *средняя* и *плановая* заявка. Сложность задачи планирования работ достаточно высокая вви-

ду большого количества ежедневных заявок и ограниченным количеством аварийных бригад. Например, в Средневолжской газовой компании г. Самара обычно бывает порядка 50 заявок в день при наличии 25 бригад, а в Водоканале г. Волгоград – порядка 64 заявок, при том что общее количество бригад – 15. При этом большую роль играет фактор времени.

2. Дополнительно необходимо оперативно, гибко и эффективно реагировать на поток заявок (событий) реального времени, а именно на поступление более срочных заказов или на внезапную поломку одного из специальных автомобилей бригад в пути, что приводит недоступности той или иной бригады. В частности, при поступлении более срочной заявки нужно адаптивно определить бригаду, которая может ее выполнить и сможет прибыть на место быстрее других с учетом своих уже имеющихся в плане работ, а также оперативно перестроить расписание работ этой бригады.

3. Управлять маршрутом движения бригады к месту работ с учетом текущей дорожной ситуации (дорожные знаки, разметка, пробки) с тем, чтобы сократить время движения.

4. Сократить простой квалифицированных и хорошо оснащенных бригад, которые должны гибко маневрировать расписанием, переходя от срочных заявок – к профилактике и наоборот.

5. Обеспечить индивидуальный подход к каждой поступившей заявке и ресурсу, назначаемому для ее выполнения. За выполнение каждой заявки бригаде должен начисляться бонус в зависимости от сложности выполненных работ согласно регламенту работы сервисной службы. В качестве основного показателя эффективности работы бригады используется фактическое время выполнения работы по заявке и отсутствие повторных выездов на следующий день.

Решение задачи управления мобильными бригадами реализуется с помощью адаптивного MAC-планировщика, который способен планировать выполнение задач обслуживания практически в реальном времени, обеспечивая минимизацию общего пробега автотранспорта обслуживающих бригад, время, необходимое для выполнения заявок на обслуживание и общее число невыполненных заявок клиентов в текущий день.

Рассмотрим архитектуру разработанного MAC-планировщика, которая включает в себя виртуальный мир программных агентов, каждый из которых может обмениваться p2p-сообщениями с другими агентами для того, чтобы сообщить им о своих возможностях и своих предложениях по изменению параметров своего состояния (времени, стоимости, координат, статусов и др.).

Поток внешних событий генерируется заявками, которые поступают из колл-центров и от диспетчеров, сообщениями от ресурсов мобильных бригад о статусах заявок, сообщениями сервисов геопозиционирования GPS/Глонасс и карт о координатах, а также сообщениями диспетчеров о недоступности ресурсов и пробках на дорогах.

Сущность работы MAC-планировщика состоит в скользящей адаптивной обработке сообщений о событиях, в ходе которой план гибко корректируется без остановов системы. При отсутствии внешних событий и нагрузки по выполнению аварийных заявок бригады выполняют плановые работы по поддержке/обслуживанию сетей предприятия.

Логическая архитектура системы представлена на рис. 5.

Основной оценкой состояния и возможности выполнения заявки является суммарное время поездки бригады по места выполнения заявки и время ее выполнения.

В отличие от сложных формализованных BDI-моделей агентов, в разработанном MAC-планировщике используются простые агенты, которые вовлечены в интенсивные переговоры. Сам MAC-планировщик представляет собой распределенную систему активно взаимодействующих агентов следующих типов:

– *Агент заявки*, целью которого является поиск наиболее подходящей специализированной бригады, способной выполнить работу ее как можно скорее и с нужным качеством.

– *Агент региона* помогает балансировать распределение машин между районами города с учетом прогноза поступления заявок.

– *Агент маршрутов перемещений*, чья цель состоит в том, чтобы предложить бригаде маршрут движения к месту аварии с минимальным временем в пути между заявками и таким обра-

зом способствовать увеличению числа заявок, выполняемых бригадами за смену.

– *Агент бригады*, целью которого является подбор и согласование заявок и маршрутов с целью повышения продуктивности работы бригады.

– *Агент предприятия*, стремящийся улучшить наиболее проблемные фрагменты расписания мобильных бригад компании.

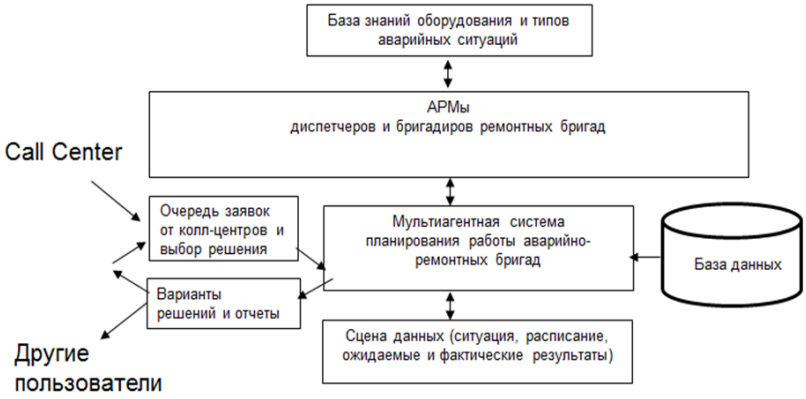

Рис. 5. Логическая архитектура системы адаптивного управления аварийно-ремонтными бригадами

Все решения по распределению и планированию работы бригад, которые формируются с помощью переговоров агентов по соответствующим протоколам и самоорганизации, предлагаются диспетчеру, который может пи необходимости интерактивно перестроить расписание любой бригады (рис. 6).

Типичный протокол взаимодействия агентов при поступлении заявки на аварийное обслуживание с описанием аварийной ситуации через интерфейс колл-центра выглядит следующим образом:

– *Агент предприятия* создает *агента заявки* с атрибутами местоположения, статуса срочности, оценки сложности, требуемой специализации и квалификации бригады, оценки времени выполнения и предельного срока, а также стоимости работ.

– *Агент заявки* рассылает сообщение *агентам региона* с указанием места аварии, а также агентами регионов, которые граничат с регионом, где произошла авария.

– *Агенты регионов* запрашивают *агентов бригад*, способных по типу работ и географическому местоположения аварии выполнить заявку с ориентировочным временем начала работ;

– *Агенты бригад* запрашивают *агентов маршрутов* об оценке времени на подъезд к месту аварии с учетом текущей дорожной ситуации.

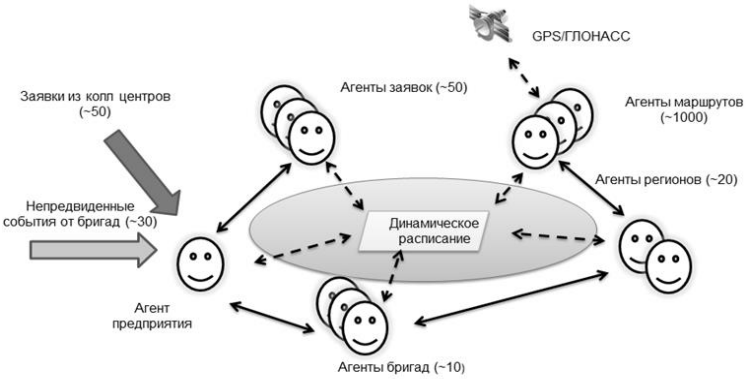

Рис. 6. Сцена мира агентов адаптивного управления аварийно-ремонтными бригадами (штрих линия – чтение из сцены и изменения в сцене, сплошная линия – обмен сообщениями и переговоры агентов)

При получении отклика от агента маршрутов *агент каждой бригады* строит изменения в своем текущем плане с учетом приоритетов запланированных заявок.

– При наличии свободных временных окон в своем расписании работ *агент бригады* отправляет *агенту предприятия* скорректированный план выполнения заявки;

– При наличии конфликтов в расписании *агент бригады* рассылает сообщения *агентам* запланированных *заявок* с оценкой возможных временных смещений и сброса менее приоритетных заявок.

– *Агенты* сдвинутых *заявок* в случае согласия уведомляют *агента бригады* и *агента предприятия* об изменении расписания. В случае, если новая заявка получила отказ на обслуживание в нужные сроки, *агент предприятия* обращается к *агентам* соседних *регионов* с запросом на возможность обслуживания. При отсутствии предложений со стороны соседних регионов *агент предприятия* принудительно планирует заявку подходящей бригаде в конце ее текущего расписания с учетом приоритета.

– *Агенты* менее приоритетных *заявок*, которые оказались вне расписания, повторно обращаются к *агентам регионов* и цикл такой цикл пересчета расписания повторяется;

– При выполнении бригадой очередной заявки *агент предприятия* отмечает фактическое время и корректирует статистику времен исполнения работ данного типа.

– В каждом новом цикле планирования *агент предприятия* оценивает время выполнения каждой заявки, загруженность бригад и прибыльность выполнения, направляя *агенту диспетчера* сообщение с текущими характеристиками расписания.

В результате такого цикла обмена сообщениями текущее расписание по событиям постоянно адаптивно подстраивается под вновь поступающие заявки и текущую дорожную ситуацию. Время, необходимое для оценки вариантов размещения заявки составляет обычно не больше 4 с.

Первое внедрение такого планировщика было выполнено в Средневолжской газовой компании, которая уже несколько лет успешно эксплуатирует описанную систему.

Использование адаптивного планировщика для обработки поступающих заявок в реальном времени позволило достичь повышения продуктивности обслуживающих бригад на 40%, поскольку каждая бригада газовой службы в настоящее время выполняет в среднем 12 задач (ранее эта цифра была равна 7 задач в день). В результате внедрения описанного МАС-планировщика удалось добиться заметного улучшения различных показателей работы:

– сокращение времени реакции на непредвиденные события до 30%;

– повышение заинтересованности бригад в увеличении продуктивности работы;

– поддержка гибкого планирования в реальном времени;

– сокращение трудоемкости и числа ошибок диспетчеров в 2 раза;

– сокращение времени обучения новых диспетчеров в 3–5 раз;

– развития бизнеса компании без роста численности управленческого персонала.

Ведутся работы по другим практическим внедрениям описанного МАС-планировщика, в частности, на предприятии Водоканал в г. Волгограде.

7. Заключение

Чрезмерно высокая оптимистичность прогнозов по широкому промышленному внедрению МАС была обусловлена различными причинами. МАС-технология с самого начала своего становления рассматривалась как достаточно универсальная интеллектуальная информационная технология, способная успешно конкурировать с другими технологиями в широком классе приложений. Однако это позиция оказалась не вполне оправданной. В результате в значительном числе прикладных задач более зрелые технологии оказались более успешными, что подорвало доверие к МАС-технологиям в глазах индустриального сообщества и, соответственно, интерес к их использованию и финансированию.

Большие негативные последствия для авторитета МАС-технологий обусловлены также существенной недооценкой сложности разработки МАС-приложений, которая в ее современной форме требует от разработчиков не только высокой алгоритмической квалификации и хороших навыков объектного программирования, но также и навыков программирования параллельных и асинхронных вычислений, привлечения моделей искусственного интеллекта и активного включения в процесс разработки достижений в области телекоммуникаций, включая мобильные планшеты и сотовые телефоны.

Серьезным внутренним препятствием для развития МАС стала ориентация базовой модели агента и многоагентной системы на использование логического языка, дополненного модальными и темпоральными операторами, для формального описания *BDI*-модели поведения агентов. Эта модель казалась весьма привлекательной из-за ее богатых выразительных возможностей и математической корректности. Но на деле она оказалось совершенно бесполезной для создания промышленных систем. Особенности грантовой системы финансирования современной науки за рубежом и в России также сказались на результатах, в особенности, для тех исследователей, кто торопился сделать МАС своим новым «маркетинговым флайером».

Не менее важными факторами охлаждения первого «горячего» интереса к МАС в индустриальном сообществе стали значительные трудности промышленных внедрений, связанные со сложностью инструментальных программных средств для поддержки технологии МАС, с высокой стоимостью критических для бизнеса разработок и необходимостью решения множества других системных задач, не связанных напрямую с МАС-технологиями.

Вместе с тем, число успешных промышленных применений МАС продолжает неуклонно расти и захватывает все новые области. И хотя темпы этого роста пока слишком далеки от желаемых, а также от тех прогнозов, которые делались в этом отношении в начале 2000-х годов, агентские модели и технологии по-прежнему остаются перспективными для большого количества новых и будущих приложений, в частности, для тех приложений, которые характеризуются сетевой структурой автономных сущностей, высокой неопределенностью и динамикой поведения, большой размерностью, а также используют принципы самоорганизации в качестве базовых принципов управления и решения сложных задач.

Можно утверждать, что многоагентные технологии – это новый перспективный технологический базис для обеспечения «Grand Future Vision» для многих «сдвигов парадигм» в современных отраслях промышленности, в частности, в таких как

– Управление предприятиями: реформирование больших, централизованных, монолитных организаций в сети бизнес-

центров и центров знаний, работающих на виртуальном рынке, вплоть до «организаций» отдельных сотрудников;

– Авиация и космос: самоорганизующиеся группировки дешевых малых космических аппаратов и беспилотных летательных аппаратов как альтернатива большим и дорогостоящим спутникам, и аналогичным летательным аппаратам;

– Энергетика: переход от больших централизованных электростанций к распределенным сетям малых электростанций и других генераторов электроэнергии на местах, более дешевых и учитывающих индивидуальные нужды потребителей и т.д.

Очевидно, что новое поколение систем такого рода следует изначально ориентировать на Интернет вещей, где каждая вещь рано или поздно должна стать «умной». Она должна обладать не только датчиками и устройствами воздействия на объекты внешнего мира и коммуникации, но и принимать решения, причем, по возможности, согласованно с другими вещами. И тут МАС-технологии, без сомнения, помогут интеллектуализации как самих приложений, так и интеллектуализации их взаимодействия и кооперации.

Вместе с тем, для продуктивного развития МАС становится необходимой смена базовой парадигмы формализации модели агента и информационно-коммуникационной инфраструктуры, обеспечивающей поддержание его жизненного цикла, взаимодействия его с внешней средой и облачными сервисами для того, чтобы добиться вычислительной эффективности этой модели в режиме реального времени. В новой парадигме формализации МАС рационально строить как множество простых агентов с богатой компонентой взаимодействия и широким использованием принципов самоорганизации и эволюции, присущих живым системам.

Потребуются новые усилия ученых и специалистов-практиков для осознания этих новых принципов и для разработки качественно новых инструментов создания МАС в парадигме, основанной на самоорганизующейся поведенческой модели агентов и их взаимодействии.

Результаты этих разработок будут востребованы для решения многих сложных задач и создания нового поколения интеллектуальных систем в парадигме Интернета вещей, более от-

крытых, гибких и эффективных для ежедневного использования потребителями.

Благодарности. Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проекты 16-01-00759 и 14-07-00493). Часть исследований выполнена в рамках проекта № 214 подпрограммы «Интеллектуальные информационные технологии и системы» Программы Президиума РАН I.5П. Частичное финансирование проекта выполнялось за счет средств проекта № 0073-2015-0003, выполняемого в СПИИРАН по бюджетной тематике.

Литература

1. АНДРЕЕВ В.В., БАТИЩЕВ С.В. ВИТТИХ В.А. и др. *Методы и средства создания открытых мультиагентных систем для поддержки процессов принятия решений* // Известия Академии Наук. Теория и системы управления. – 2003. – №1. – С. 126–137.
2. БУХВАЛОВ О.Л., ГОРОДЕЦКИЙ В.И., КАРСАЕВ О.В. и др. *Производственная логистика: Стратегическое планирование, прогнозирование и управление конфликтами* // Известия ЮФУ. – 2012. – №3. – С. 209–218.
3. БУХВАЛОВ О.Л., ГОРОДЕЦКИЙ В.И., КАРСАЕВ О.В. и др. *Распределенная координация в B2B- производственных сетях* // Известия Южного федерального университета. Технические науки. – 2013. – №3. – С. 193–203.
4. ВИТТИХ В.А., СКОБЕЛЕВ П.О. *Метод сопряженных взаимодействий для управления распределением ресурсов в реальном времени* // Автометрия. – 2009.– Т. 45, №2. – С. 78–87.
5. ВИТТИХ В.А., СКОБЕЛЕВ П.О. *Мультиагентные модели взаимодействия для построения сетей потребностей и возможностей в открытых системах* // Автоматика и телемеханика. – 2003. – №1. – С. 177–185.

6. ГОРОДЕЦКИЙ В.И., КАРСАЕВ О.В., САМОЙЛОВ В.В. и др. *Инструментальные средства для открытых сетей агентов* // Известия РАН. Теория и системы управления. – 2008. – №3. – С. 106–124.
7. ГОРОДЕЦКИЙ В.И. *Многоагентная самоорганизация в B2B-сетях* // XII Всероссийское совещание по проблемам управления ВСПУ-2014, Москва, 16–19 июня 2014 г.: Труды. – М.: Институт проблем управления им. В.А. Трапезникова. [Электронный ресурс]. – URL: http://vspu2014.ipu.ru/proceedings/vspu2014.zip (дата обращения 11.02.2016 г.). – С. 8954 – 8965.
8. ГОРОДЕЦКИЙ В.И., САМОЙЛОВ В.В., ТРОЦКИЙ Д.В. *Базовая онтология коллективного поведения автономных агентов и ее расширения* // Известия РАН: Теория и системы управления. – 2015. – №5. – С. 102–121.
9. ГОРОДЕЦКИЙ В.И. *Самоорганизация и многоагентные системы. II. Приложения и технология разработки* // Известия РАН. Теория и системы управления. – 2012. – №3. – С. 102–123.
10. ГОРОДЕЦКИЙ В.И**.** *Теория, модели, инфраструктуры и языки спецификации командного поведения автономных агентов.* Ч. 1 // Искусственный интеллект и принятие решений. – 2011. – №2. – С. 19–30.
11. СКОБЕЛЕВ П.О. *Интеллектуальные системы управления ресурсами в реальном времени: принципы разработки, опыт промышленных внедрений и перспективы развития* // Приложение к теоретическому и прикладному научно-техническому журналу «Информационные технологии». – 2013. – №1. – С. 1–32.
12. СКОБЕЛЕВ П.О. *Мультиагентные технологии в промышленных применениях: к 20-летию основания Самарской научной школы мультиагентных систем* // Мехатроника, автоматизация, управление. – 2010. – №12. – С. 33–46.
13. СКОБЕЛЕВ П.О. *Открытые мультиагентные системы для оперативной обработки информации в процессах принятия решений* // Автометрия. – 2002. – №6. – С.45–61.

14. *ACL – Agent Communication Language*. – URL: http://fipa.org/specs/fipa00061/SC00061G.pdf (дата обращения: 19.01.2016).
15. *AgentBuilder – an integrated software toolkit that allows software developers to quickly develop intelligent software agents and agent-based applications*. – URL: http://www.agentbuider.com (дата обращения: 19.01.2016).
16. BERNON C. et al. *ADELFE: A Methodology for Adaptive Multi-Agent Systems Engineering* // Proc. of the 3th International Workshop on Engineering Societies in the Agents World. – 2002. – P. 156–169.
17. BONOMI F. et al. *Fog computing and its role in the internet of things* // Proc. of the First Edition of the MCC Workshop on Mobile Cloud Computing, (MCC 2012), New York, NY. USA, 2012. – P. 13–16.
18. BURRAFATO P., COSSENTINO M. *Designing a multi-agent solution for a bookstore with PASSI methodology* // Proc. of International Conference on Agent-Oriented Information Systems. (Eds. Giorgini P., Lesperance Y., Wagner G., Yu E.), 2002. – P. 119–135.
19. CAIRE G. et al. *Agent Oriented Analysis using MESSAGE/UML* // Agent-Oriented Software Engineering II, Springer-Verlag, Lecture Notes in Computer Science. – 2002. – Vol. 2222. – P. 119–125.
20. COHEN P., LEVESQUE H. *Teamwork* // Nous. – 1991. – Vol. 25. – P. 487–515.
21. *Cougaar Agent Architecture*. – URL: http://cougaar.org/wp/documentation/tutorials/ (дата обращения: 19.01.2016).
22. DELOACH S. *Analysis and Design using MaSE and agentTool* // Proc. of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), Miami University Press, 2001.
23. DELOACH S., GARCIA-OJEDA J. *The O-MaSE Methodology* // Eds.: M. Cossentino et al. Handbook on Agent-Oriented Design Processes, Springer-Verlag, Berlin–Heidelberg, 2014. – P. 253–285.

24. DELOACH S.A *Moving multi-agent systems from research to practice* // Int. J. Agent-Oriented Software Engineering. – 2009. –Vol. 3, No. 4. – P. 378–382.
25. *FIPA Agent Management System*. – http://www.fipa.org/specs/fipa00023/XC00023H.html (дата обращения: 19.01.2016).
26. *Gartner. Top Strategic Predictions for 2016 and Beyond: The Future Is a Digital Thing*. – URL: https://www.gartner.com/doc/3142020?refval=&pcp=mpe (дата обращения: 19.01.2016).
27. GARCIA-OJEDA J. et al. *AgentTool Process Editor: Supporting the Design of Tailored Agent-based Processes* // Proc. of the 24th Annual ACM Symposium on Applied Computing, March 8–12, 2009, Honolulu, Hawaii, USA.
28. GORODETSKY V., KARSAEV O., SAMOYLOV V. et al. *Support for Analysis, Design and Implementation Stages with MASDK* // Eds.: Luck M., Gomez-Sanz J.J. – Springer, Heidelberg, LCNS. – 2009. – Vol. 5386. – P. 272–288.
29. GORODETSKY V., KARSAEV O., SAMOYLOV V.et al. *P2P Agent Platform: Implementation and Testing* // Lecture Notes in Artificial Intelligence. – 2009. – Vol. 5319. – P. 41–54.
30. HEWITT C., BISHOP P., STEIGER R.A *Universal Modular Actor Formalism for Artificial Intelligence* // Proc. of International Joint Conference on Artificial Intelligence (IJCAI-1973). – P. 235–245.
31. HYACINTH S. et al. ZEUS: A *Toolkit for Building Distributed Multi-Agent Systems* // Proc. of the 3rd annual conference on Autonomous Agents (AGENTS'99), New York, NY, 1999. – P. 360–361.
32. *JADE*. – URL: http://sharon.cselt.it/projects/jade/ (дата обращения: 19.01.2016).
33. KAMINKA G. *Robots Are Agents, Too!* // Keynote Lecture. International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2007), Honolulu, Hawaii, May 2007.
34. LEITAO P., VRBA P. *Recent Developments and Future Trends of Industrial Agents* // Proc. of «Holonic and Multi-Agent Systems for Manufacturing» (HoloMAS-2011), Vol. 6867 of the series Lecture Notes in Computer Science. Springer Verlag, 2011–2012. – P. 15–28.

35. LUCK M. et al. *Agent Technology: Computing as Interaction* (*A Roadmap for Agent Based Computing*) // AgentLink. – 2005. – URL: http://www.agentlink.org/roadmap/ (дата обращения 19.01.2016).
36. LUCK M., GOMEZ-SANZ J. *Agent-oriented Software Engineering IX* // Lecture Notes in Computer Science. Springer. – 2009. – Vol. 5386.
37. MYLOPOULUS J., CASTRO J. *Tropos: A Framework for Requirements-Driven Software Development* // Proc. of 12th Conference on Advanced Information System Engineering (CAISE), 2000.
38. MŰLLER J., FISHER K. *Application Impact of Multiagent Systems and Technologies: A Survey* // In «Agent-Oriented Software Engineering» book series. – Springer, 2013. – P. 1–26.
39. PADGHAM L., WINIKOFF M. *Prometheus: A Paradigm Methodology for Engineering Intelligent Agents* // Proc. of the Workshop on Agent-Oriented Methodologies, 2002. – P. 97–108.
40. RZEVSKI G., SKOBELEV P. *Managing complexity.* – WIT Press, London–Boston, 2014. – 156 p.
41. RoboCup Soccer Server. http://dangermouse.brynmawr.edu/pyrobot/tars/rcsoccersim.pdf (дата обращения: 19.01.2016).
42. SIERRA C. et al. SADDE: *Social Agents Design Driven by Equations* // In: Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook. – Kluwer Academic Publishers, NY, 2004. – P. 1–24.
43. SKOBELEV P. *Multi-Agent Systems for Real Time Adaptive Resource Management* // Industrial Agents: Emerging Applications of Software Agents in Industry / Eds.: P. Leitão, S. Karnouskos. – Elsevier, 2015. – P. 207–230.
44. SMITH G. Contract Net Protocol: *High-level Communication and Control in a Distributed Problem Solver* // *IEEE Transactions on Computers*. – 1980. – C-29(12). – P. 1104–1113.

45. STRASSNER J. *Using Agents and Autonomic Computing to Build Next Generation Seamless Mobility Services* // Keynote Lecture. International Conference on Autonomous Agents and Multi-agent Systems (AAMAS 2007), Honolulu, Hawaii, May 2007.
46. SYCARA K., SUKTHANKAR G. *Literature Review of Teamwork Models* // CMU-RI-TR-06-50, November 2006, Robotics Institute Carnegie Mellon University. http://www.ri.cmu.edu/pub_files/pub4/sycara_katia_2006_1/sycara_katia_2006_1.pdf (дата обращения: 19.01.2016).
47. TAMBE M. *Towards Flexible Teamwork* // Journal of Artificial Intelligence Research. – 1997. – No. 7. – P. 83–124.
48. *Technology Reports, Universal Description, Discovery, and Integration (UDDI)*. – URL: http://xml.coverpages.org/uddi.html (дата обращения: 19.01.2016).
49. VAN DER HOEK W. *Logical Foundations of Agent-based Computing*. // In: Multi-agent systems and applications. Lecture Notes in Artificial Intelligence / Eds.: M. Luck, V. Marik, O. Stepankova, R. Trappl. – Vol. 2086. – Springer Verlag, 2001.
50. WOOLDRIDGE M. *An Introduction to MultiAgent Systems.* – John Wiley & Sons, 2009. – 368 p.
51. YI SH. et al. *Security and Privacy Issues of Fog Computing: A Survey* // Wireless Algorithms, Systems and Applications. – Vol. 204 of the series Lecture Notes in Computer Science. – Springer International Publishing, Switzerland, 2015. – P. 685–695.
52. ZAMBONELLI F., JENNINGS N., WOOLDRIDGE M. *Developing Multiagent systems: The GAIA methodology* // ACM Transactions on Software Engineering and Methodology. – 2003. – No. 12(3). – P. 417–470.

INDUSTRIAL APPLICATIONS OF MULTI-AGENT SYSTEMS: CURRENT STATE AND PROSPECTS

Vladimir Gorodetsky, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, Doctor of Science, professor (gor@iias.spb.su).

Oleg Bukhvalov, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, (psyhoveter@gmail.com).
Petr Skobelev, Institute for the Control of Complex Systems of the Russian Academy of Sciences and Samara Aerospace University, Samara, Doctor of Science, professor (petr.skobelev@gmail.com).
Igor Mayorov, Software Engineering Company "Smart Solutions", and Samara State Technical University, Samara, (imayorov @smartsolutions-123.ru).

Abstract: *Multi-agent system are perceived as one of the most promising technologies for distributed systems development. However current evidences show a lack of industry-level applications. The paper examines the main trends and prospects in development of industrial applications of multi-agent systems and technology, as well as analyzes the recent forecasts and the current state-of-the art with its practical applications. A survey of currently present researches, techniques, frameworks and standards is provided and supported with the most recent statistical data. The paper also analyzes external and internal obstacles for broad commercialization of multi-agent systems and technologies and the lessons learnt through these analyses. The paper describes properties and types of current and future applications for which multi-agent technology has unquestionable advantages including enterprise management, aviation and space, energetics, transportation etc. The paper also shows that multi-agent systems and technologies currently have no alternatives in self-organized control of large-scale objects of network structure.*

Keywords: multi-agent systems, industrial applications, self-organization, networked objects.

Статья представлена к публикации членом редакционной коллегии Д.А. Новиковым.

Поступила в редакцию 17.03.2017.
Опубликована 31.03.2017.

УДК 519.86
ББК 22.18

ОПТИМИЗАЦИЯ СТРУКТУРЫ БАЗЫ ДАННЫХ РЕАЛЬНОГО ВРЕМЕНИ

Мирошник С. Н.[1, a], Гончар Д. Р.[2, a, b]
Фуругян М. Г.[3, a, c]
([a]*Вычислительный центр им. А.А. Дородницына ФИЦ «Информатика и управление» РАН,*
[b]*Московский физико-технический институт (государственный университет),*
[c]*Московский государственный университет им. М.В. Ломоносова, Москва*)

Исследуется задача оптимизация структуры базы данных реального времени. Разработаны эвристические алгоритмы решения данной задачи и алгоритм, основанный на сведении ее к задаче булевого программирования.

Ключевые слова: системы управления базами данных, системы реального времени, эвристические алгоритмы, оптимизация.

1. Введение

Одной из основных проблем проектирования баз данных (БД) реального времени является проблема минимизации избыточности информации. Теоретические основы создания и оптимизации БД содержатся в [1, 6]. Настоящая работа является

[1] *Сергей Николаевич Мирошник, научный сотрудник, кандидат физико-математических наук* (*rtsccas@ya.ru*).
[2] *Дмитрий Русланович Гончар, старший научный сотрудник, кандидат технических наук* (*trpl@ya.ru*).
[3] *Меран Габибуллаевич Фуругян, заведующий сектором, кандидат физико-математических наук, доцент* (*rtsccas@ya.ru*).

продолжением работ [2–4, 5], посвященных методам создания соответствующей структуры БД, в которой избыточность информации сведена к минимуму. Этой проблеме посвящено большое количество публикаций, в которых основное внимание уделено технической части создания подобных БД, но не связанных с использованием информации в БД для решения задач в реальном времени. В данной работе основное внимание уделено математической стороне проблемы. В работах [2–4, 5] используется понятие избыточности информации трех типов, а именно межфайловая, внутрифайловая и внутримодульная избыточности. В работах [2, 4] подробно даны определения этих понятий, а также формулы, с помощью которых эти избыточности вычисляются. В зависимости от типов задач разработаны эвристические алгоритмы, минимизирующие разные типы избыточностей. Так, например, в работе [4] рассматриваются задачи, в которых неиспользуемая информация сравнима с используемой. Предлагается алгоритм создания БД, в которой минимизируется внутримодульная избыточность информации. В данной работе, так же как в работе [2], основное внимание уделено минимизации межфайловой и внутрифайловой избыточностей. Предлагается принципиально новый, более совершенный, алгоритм создания БД по сравнению с аналогичными алгоритмами, предложенными в [2]. Эти результаты используются в разрабатываемой авторами инструментальной системе автоматизации проектирования систем реального времени, в которой эта БД является одной из составляющих элементов. Данная САПР СРВ может быть использована при решении задач, возникающих при лётных экспериментах, экологическом или экономическом мониторинге и в ряде других областях.

2. Постановка задачи

Имеется n независимых друг от друга программных модулей $i = 1, ..., n$. Для их выполнения требуются данные, содержащиеся в файле Ф, который состоит из полей $Ф_1, Ф_2, \ldots, Ф_r$. При выполнении каждого модуля используется часть полей $Ф_j$ файла Ф. Для заданного числа K строятся файлы $F_1, F_2, \ldots, F_k$,

$k \le K$, каждый из которых содержит некоторые поля Φ_j файла Φ, причем для каждого модуля i некоторый файл $F_{j(i)}$ должен содержать все поля Φ_j, необходимые для его работы. Суммарная длина файлов F_i не должна превышать заданной величины W. Внутрифайловая избыточность входной информации определяется как величина $I_1 = \sum_{i=1}^{n}(|F_{j(i)}| - |\overline{F}_{j(i)}|)$, где $|F_{j(i)}|$ – длина файла $F_{j(i)}$, $|\overline{F}_{j(i)}|$ – суммарная длина полей файла $F_{j(i)}$, используемых работой i. Межфайловая избыточность базы данных определяется как величина $I_2 = \sum_{j=1}^{r}(K_j - 1)|\Phi_j|$, где K_j – число файлов F_i, в которых содержится поле Φ_j, $|\Phi_j|$ – длина поля Φ_j. Задача заключается в создании для заданных K и W такого набора файлов $F_1, F_2, \ldots, F_k$, $k \le K$, для которого

(1) $\sum_{j=1}^{k}|F_j| \le W$

и величина I_1 минимальна. Впервые авторами эта постановка была сформулирована в работе [2]. Если решений более одного, то из их числа выбирается такое, при котором величина I_2 минимальна. В качестве критерия оптимальности можно взять линейную комбинацию величин I_1 и I_2, например, $I_1 + I_2$.

Отметим, что решение задачи в разрабатываемой авторами инструментальной системе автоматизации проектирования систем реальном времени разделяется на два больших этапа.

A. Предварительный этап (не в реальном времени), на котором в том числе осуществляется формирование групп близких модулей.

B. После этого решается задача в реальном времени, используя выполненную на предварительном этапе оптимизацию расположения полей в файлах.

3. Сведение к задаче булевого программирования

Пусть v_{pj}, $p = 1, 2, \ldots, n$, $j = 1, 2, \ldots, r$, – массив, состоящий из нулей и единиц, причем $v_{pj} = 1$, если для выполнения модуля p требуется поле Φ_j, и $v_{pj} = 0$ в противном случае.

Введем переменные x_{ij}, $i = 1, 2, \ldots, k$; $j = 1, 2, \ldots, r$, и y_{ip}, $i = 1, 2, \ldots, k$; $p = 1, 2, \ldots, n$, принимающие значения 0 и 1, причем $x_{ij} = 1$, если файл F_i содержит поле Φ_j, и $x_{ij} = 0$ в противном случае; $y_{ip} = 1$, если для работы модуля p требуется файл F_i, и $y_{ip} = 0$, в противном случае.

Сформулированная задача может быть записана в виде следующей задачи булевого программирования [2]:

$$(2)\quad \min_{x_{ij}, y_{ip}} \sum_{p=1}^{n}\sum_{i=1}^{k}\left\{ y_{ip}\left[\sum_{j=1}^{r}(x_{ij} - v_{pj})\cdot\left|\Phi_j\right|\right]\right\};$$

$$(3)\quad \min_{x_{ij}} \sum_{j=1}^{r}\left[\left(\sum_{i=1}^{k} x_{ij}\cdot\left|\Phi_j\right|\right) - \left|\Phi_j\right|\right];$$

$$(4)\quad \sum_{i=1}^{k}\sum_{j=1}^{r} x_{ij}\cdot\left|\Phi_j\right| \le W;$$

$$(5)\quad \sum_{i=1}^{k} y_{ip} = 1, \quad p = 1, 2, \ldots, n;$$

$$(6)\quad \begin{array}{l} y_{ip}(x_{ij} - v_{pj}) \ge 0; \quad i = 1, 2, \ldots, k; \\ p = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, r. \end{array}$$

Здесь x_{ij}, y_{ip} принимают значения 0 и 1. Число переменных в данной задаче равно $k(r + m)$, а число ограничений – $(nkr + 1)$.

Условие (2) соответствует минимизации величины i_1, условие (3) – минимизации величины i_2, условие (4) – условию (1), условие (5) приписывает каждому модулю $p \in n$ некоторый файл f_i. Благодаря условию (6) у каждого модуля $p \in n$ соответствующий ему файл f_i будет содержать все необходимые поля Φ_j. Сначала должна быть решена задача минимизации (2) при ограничениях (4)–(6). Затем на полученном множестве решений (если решение не однозначно) решается задача минимизации (3).

4. Эвристический алгоритм создания структуры БД

Будем предполагать, что все поля пронумерованы натуральным рядом чисел 1, …, r. Запись модуля M_i есть набор полей $\{\phi\}_i$, идущих подряд из набора ϕ_1, …, ϕ_r. Таким образом, в записи модуля M_i есть номер первого поля a_i и номер последнего поля b_i. Длина записи модуля M_i есть $l_i = b_i - a_i + 1$. Записи разных модулей могут использовать одинаковые поля. Природа полей и их длины не рассматриваются. В данной работе в записи модулей входят только используемые поля, неиспользуемые поля не рассматриваются, так как они не влияют на работу алгоритма. Основная идея минимизации избыточной информации состоит в том, чтобы определенным образом объединить модули в различные группы, в дальнейшем оформляемых в виде файлов со своими атрибутами с последующим вычислением I_1 и I_2 (см. [3]). Это объединение осуществляется так, чтобы каждый модуль попал только в один файл. Длина L записи файла F определяется количеством различных полей всех модулей, образующих этот файл. В записи файла F есть первое поле A со своим номером и последнее B, $L = B - A + 1$. Теперь каждый модуль M для своей работы обращается ко всем полям записи своего файла F. В частности, различные модули могут использовать одинаковые поля файла. Вследствие разности длины l модуля M и длины L файла F образуются неиспользуемые этими модулями поля, $L \geq 1$. Эти неиспользуемые поля всех модулей файла F образуют внутрифайловую избыточность $I_1(F)$ файла F. Эта избыточная информация вычисляется по формуле

$$(7) \quad I_1(F) = L \cdot t - \sum_{j=1}^{t} l^j ,$$

где t – число модулей в файле F.

Естественно, полная внутрифайловая избыточность БД определяется по формуле

$$(8) \quad I_1 = \sum_{i=1}^{k} (L_i \cdot t_i - \sum_{j=1}^{t} l_i^j)$$

для всех файлов F_i, …, F_k.

Далее, среди полей записей файлов F_i есть повторяющиеся поля. Эти поля образуют межфайловую избыточность I_2 и вычисляются по формуле

$$(9) \quad I_2 = \sum_{i=1}^{k} L_i - r.$$

4.1. ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

Предложенные в [3] алгоритмы так же, как и в данной работе, основаны на различных способах объединения модулей в файлы. Главным в этом объединении есть определение понятия близости двух модулей. Для того чтобы сравнить алгоритмы из [3] и из настоящей работы, приведено описание краткой схемы алгоритма из [3].

Алгоритм из [3] имеет существенные недостатки. Модули объединяются в файлы на основе формальных признаков близости модулей, не учитывающих образующиеся внутрифайловые и межфайловые избыточности информации.

Схема алгоритма состоит в следующем. Строится распределение P повторяемости полей файла Ф. Вводится понятие опорного модуля, содержащее поле максимальной повторяемости ϕ^* и максимальной длиной l^*. Близкий к опорному модуль M длиной l должен содержать поле ϕ^* и $l \le l^*$. Перебором находим все модули, удовлетворяющие этим двум условиям. Найденные таким образом модули исключаются из $M_1, \ldots, M_n$ вместе со всеми значениями используемых ими полей. После этого строится новое распределение P повторяемости полей. Если найденное ранее поле ϕ^* в новом распределении P не изменилось, увеличиваем длину l^* на величину Δ и ищем новые модули длиной $l \le l^* + \Delta$ и содержащие поле ϕ^*. Увеличение длины l^* продолжается до тех пор, пока в очередном распределении полей P не изменится поле ϕ^*. Найденные таким образом модули образуют файл F. Процедура повторяется, пока все модули не будут распределены по файлам $F_1, \ldots, F_k$. Только после этого вычисляются I_1 и I_2 по формулам (7)–(9).

В данной работе предлагается принципиально другой алгоритм объединения модулей $M_1, \ldots, M_n$ в файлы $F_1, \ldots, F_k$. Этот

алгоритм использует другое понятие близости пары модулей M_i и M_j, а также близость модуля M к набору модулей $\{M\}$.

Введем понятие близости набора модулей $\{M\}_s$ и модуля M_{s+1}, где s – число модулей в наборе. Будем считать, что $\{M\}_s$ и M_{s+1} являются близкими, т.е. M_{s+1} может быть включен в набор модулей $\{M\}_s$, если будет выполнено определенное условие. Чтобы определить это условие, необходимо вычислить два вида избыточности информации.

1. Набору $\{M\}_s$ соответствует избыточность информации I_1^s. Включение M_{s+1} в $\{M\}_s$ изменяет внутрифайловую избыточность информации на величину $\Delta I_1^{s+1} = I_1^{s+1} - I_1^s$, где I_1^{s+1} соответствует избыточности информации набора модулей $\{M\}_{s+1}$.

2. Для модуля M_{s+1} и набора модулей $\{M\}_s$ вычислим количество совпадающих полей. Для этого воспользуемся модифицированной формулой (3). Обозначим I_2^{s+1} – количество совпадающих полей пары M_{s+1} и $\{M\}_s$.

Замечание. Формула (3) используется для вычисления избыточной информации набора файлов, а именно вычисления количества совпадающих полей разных файлов. В данном случае эта формула вычисляет значение I_2^{s+1} для определения близости пары M_{s+1} и $\{M\}_s$.

Определение. Модуль M_{s+1} и набор модулей $\{M\}_s$ являются близкими и M_{s+1} может быть включен в набор модулей $\{M\}_s$, если $\Delta I_1^{s+1} \le I_2^{s+1}$.

4.2. АЛГОРИТМ СОЗДАНИЯ СТРУКТУРЫ БД

Алгоритм использует определение близости модулей. Пусть приведенным ниже алгоритмом построен набор близких модулей $\{M\}_s$, s – число модулей. Рассмотрим модуль M_{s+1}, l_{s+1} – его длина. Относительно этого модуля требуется определить: $M_{s+1} \subset \{M\}_s$ или $M_{s+1} \not\subset \{M\}_s$.

Внутрифайловая избыточная информация I_1^s в наборе модулей $\{M\}_s$ вычисляется с помощью формулы (7): $I_1^s = L_s \cdot s - \sum_{i=1}^{s} l_i$, где L_s длина набора модулей $\{M\}_s$.

1. Вычислим ΔI_1^{s+1}. Для набора модулей $\{M\}_{s+1}$ избыточная информация есть

$$I_1^{s+1} = L_{s+1} \cdot (s+1) - \sum_{i=1}^{s+1} l_i ,$$

где L_{s+1} длина набора модулей $\{M\}_{s+1}$. Дополнительная избыточная информация ΔI_1^{s+1} есть $\Delta I_1^{s+1} = I_1^{s+1} - I_1^s$, или

$$\Delta I_1^{s+1} = L_{s+1} \cdot (s+1) - (\sum_{i=1}^{s} l_i + l_{s+1}) - L_s \cdot s + \sum_{i=1}^{s} l_i .$$

Окончательно

(10) $\Delta I_{s+1} = (L_{s+1} - L_s) \cdot s + L_{s+1} - l_{s+1}$.

Здесь $L_s = b_s^* - a_s^* + 1$, $l_{s+1} = b_{s+1} - a_{s+1} + 1$, $L_{s+1} = b_{s+1}^* - a_{s+1}^* + 1$, $b_{s+1}^* = \max(b_s^*, b_{s+1})$, $a_{s+1}^* = \min(a_s^*, a_{s+1})$, $b_s^* = \max\{b\}_s$, $a_s^* = \min\{a\}_s$.

2. Определим I_2^{s+1}. Для пары M_{s+1} и $\{M\}_s$ вычислим по формуле (9) избыточную информацию I_2^{s+1}. Получаем

(11) $I_2^{s+1} = L_s + l_{s+1} - (b_{s+1}^* - a_{s+1}^*)$.

Теперь согласно определению условие $\Delta I_1^{s+1} \le I_2^{s+1}$ о вхождении $M_{s+1} \subset \{M\}_s$ записывается так:

(12) $(L_{s+1} - L_s) \cdot (s+1) \le 2 \cdot l_{s+1} - (b_{s+1}^* - a_{s+1}^*)$.

Если условие (12) не выполняется, то M_{s+1} вместе с модулями, не вошедшими в набор $\{M\}_s$, могут образовывать другие группы похожих модулей.

4.3. ВЫЧИСЛИТЕЛЬНАЯ ЧАСТЬ АЛГОРИТМА

Для поиска близких модулей с целью формирования группы модулей $\{M\}$ требуется многократный перебор всех модулей. Существует одна особенность при реализации перебора. На некотором шаге перебора может оказаться, что длина L группы

$\{M\}$ увеличилась. Отсюда следует, что ранее отвергнутые модули могут оказаться близкими к $\{M\}$. Поэтому для поиска всех модулей, близких к набору $\{M\}$, требуется многократный перебор модулей. Естественно, из перебора исключаются те модули, которые уже попали в группу $\{M\}$.

Поиск первой группы близких модулей реализуется с помощью нескольких вычислительных шагов.

1. Путем перебора находим модуль M, для которого $l = \max(l_1, \ldots, l_n)$.

2. Перебором ищем модуль, близкий к $\{M\}$. Заметим, что при этом переборе каждый следующий рассматриваемый модуль должен быть близок к уже найденной ранее группе модулей.

3. Если после первого перебора модулей увеличилась длина L группы модулей $\{M\}$, то перебор модулей для поиска близких к набору $\{M\}$ повторяется.

4. Перебор модулей для поиска близких к набору $\{M\}$ заканчивается, если после очередного перебора модулей длина L построенной группы $\{M\}$ не изменилась.

5. Построенная группа модулей оформляется как файл F с соответствующими характеристиками: длиной L, списком модулей и их количеством, внутрифайловой избыточностью $I_1(F)$.

6. Модули файла F исключаются из набора модулей $M_1, M_2, \ldots, M_n$ и не используется повторно для построения последующих файлов.

7. Начиная с шага 1 строятся все файлы $F_1, F_2, \ldots, F_k$.

Алгоритм заканчивает работу, когда все модули будут распределены по файлам.

Замечания

1. В результате объединения модулей в файлы могут остаться модули, не вошедшие ни в один файл. Эти «одиночные» модули образуют отдельные файлы. В частности, «одиночным» модулем может оказаться модуль, использующий только одно поле. Этот модуль также может быть оформлен как отдельный файл.

2. Следует отметить возможную неоднозначность выбора первого модуля, а также неоднозначность выбора последующих близких модулей. Отсюда может быть построено множество

различных вариантов файлов и соответствующих им БД. В данной работе предлагается построить только один вариант набора файлов.

Вычисление внутрифайловой избыточности осуществляется по формуле $I_1 = \sum_{i=1}^{k} I_1(F_i)$.

Для найденного набора файлов $\{F\}_k$ определяется внутрифайловая избыточность I_1 и межфайловая I_2. Число $I = I_1 + I_2$ является показателем качества построенной БД. Вообще, учитывая замечание 2, из множества чисел I для каждого варианта БД можно выбрать наилучшую БД.

Вычислительная сложность алгоритма 2 есть $O(n^3r)$, вычислительная сложность алгоритма 1 есть $O(n^2r^3)$.

Для конкретной задачи значение для I_1 и I_2 можно сравнить с их наибольшими значениями. Пусть все модули объединены в один файл. Тогда $I_2 = 0$, а I_1 принимает максимальное значение. Если каждый модуль образует отдельный файл, то $I_1 = 0$ и I_2 принимает максимальное значение.

4.3. ВЫЧИСЛИТЕЛЬНАЯ ЧАСТЬ АЛГОРИТМА

На приводимом ниже примере показывается работа описанных в этом разделе алгоритмов.

Пример задан в виде таблицы.

Таблица. Пример работы описанных в разделе алгоритмов

№ модуля	№ поля								
	1	2	3	4	5	6	7	8	9
1	1	1	1	1	1	1	1		
2	1	1	1						
3			1	1	1				
4			1	1	1	1	1	1	
5					1	1	1	1	
6		1	1	1					
7						1	1	1	1

В этом примере число модулей $n = 7$, число полей $r = 9$. Модули не содержат неиспользуемые поля, так как они не влияют на работу алгоритма и, если нужно, их можно учесть на последней стадии работы алгоритма.

Алгоритм 1.

1.1. Из распределения P повторяемости полей получаем $ф^* = 3$. Опорный модуль M_1. F_1 : 1, 2, 3, 6, $L_1 = 7$, $I_1(F_1) = 12$.

Исключим модули 1, 2, 3, 6.

1.2. Из нового распределения P получаем $ф^* = 5$. Опорный модуль M_4. F_2 : 4, 5, $L_2 = 6$, $I_1(F_2) = 2$.

1.3. F_3 : 7, $L_3 = 4$, $I_1(F_3) = 0$.

Получаем $I_1 = 14$, $I_2 = 8$, $I = 22$.

Алгоритм 2.

2.1. F_1 : 1, 4, $L_1 = 8$, $I_1(F_1) = 3$.

2.2. F_2 : 5, 7, $L_2 = 5$, $I_1(F_2) = 2$.

2.3. F_3 : 2, 6, $L_3 = 4$, $I_1(F_3) = 2$.

2.4. F_4 : 3, $L_4 = 3$, $I_1(F_4) = 0$.

Получаем

$I_1 = 7$, $I_2 = 11$, $I = 18$.

Значение I для второго алгоритма меньше первого, и потому БД, построенная вторым алгоритмом, лучше. Для сравнения: $\max I_1 = 33$, $\max I_2 = 21$.

Литература

1. ДЕЙТ К. ДЖ. *Введение в системы баз данных*. – 8-е изд. – М.: Вильямс, 2006.
2. МИРОШНИК С.Н., ФУРУГЯН М.Г. *Оптимизация структуры базы данных реального времени* // Некоторые алгоритмы планирования вычислений в многопроцессорных системах. – М.: ВЦ РАН, 2012. – С. 24–37.
3. МИРОШНИК С.Н. *Алгоритм оптимизации базы данных реального времени для двух файлов* // Некоторые алгоритмы планирования вычислений и оптимизации баз данных в многопроцессорных системах. – М.: ВЦ РАН, 2013. – С. 41–48.
4. МИРОШНИК С.Н. *Алгоритм оптимизации базы данных реального времени для неиспользуемых полей модулей* // Некоторые алгоритмы планирования вычислений и методы многокритериальной оптимизации для многопроцессорных систем. – М.: ВЦ РАН, 2014. – С. 32–40.
5. МИРОШНИК С.Н. *Алгоритм оптимизации структуры базы данных реального времени с минимальной избыточностью информации* // Некоторые алгоритмы составления расписаний в многопроцессорных системах. – М.: ВЦ РАН, 2015. – С. 25–34.
6. ULMAN L. *PHP6 and MYSQL for dynamic web sites*. – Peachpit Press, 2007.

REAL-TIME DATABASE STRUCTURE OPTIMIZATION

Sergey Miroshnik, Dorodnicyn Computing Centre of FRC «Informatics and Control» of RAS, Moscow, Cand. Sc. (rtsccas@ya.ru).
Dmitry Gonchar, Dorodnicyn Computing Centre of FRC «Informatics and Control» of RAS, Moscow, Moscow Institute of Physics and Technology (State University), Moscow, Cand. Sc. (trpl@ya.ru).
Meran Furugyan, Dorodnicyn Computing Centre of FRC «Informatics and Control» of RAS, Moscow, Lomonosov Moscow State University, Moscow, Cand. Sc. (rtsccas@ya.ru).

Abstract: *The problem of real-time database structure optimization is considered. The objective is to minimize the information redundancy with respect to real-time usage. There is a set of active processes, each process uses several data fields from a given database. The fields should be divided into a set of files such that each process need not use more than one file for normal operation. Different measures are proposed for intra-file, inter-file and inter-process redundancies. We focus on the problem of inter- and intra-file redundancy minimization which can be reduced to a Boolean programming problem. We propose two algorithms for the database structure optimization problem. The first algorithm is based on the Boolean programming reduction and the second one is heuristics with a polynomial computational complexity. An illustrative example is provided. The algorithms are included into a CAD for real-time systems which can be used for flight experiments, ecological monitoring and other fields.*

Keywords: database management system, real-time systems, heuristic algorithms, optimization.

Статья представлена к публикации
членом редакционной коллегии А.А. Лазаревым.

Поступила в редакцию 28.01.2016.
Опубликована 31.03.2017.

УДК 004.942 + 004.272.43
ББК 32.973.202

ФРАГМЕНТНЫЙ ПОДХОД К ДИАГНОСТИРОВАНИЮ КОМПОНЕНТОВ ЦИФРОВЫХ СИСТЕМ СО СТРУКТУРОЙ МИНИМАЛЬНОГО КВАЗИПОЛНОГО ГРАФА (НА ПРИМЕРЕ ГРАФА РАЗМЕРА 7×7)

Ведешенков В. А.[1]
(*ФГБУН Институт проблем управления им. В.А. Трапезникова РАН, Москва*)

Разработано два варианта фрагментного подхода, различающиеся составом диагностируемых фрагментов. В первом варианте ЦС логически разделяется на 7 фрагментов одинакового состава: коммутатор, 4 абонента и 8 линий связи. Во втором варианте 7 абонентов проверяются по одному, каждый фрагмент включает коммутатор и 8 линий связи. Исправный диагностический монитор (ДМ) поочередно проверяет компоненты фрагмента, раскодирует полученный синдром проверок, передает диагноз обслуживающему персоналу для ремонта неисправных компонентов. Представленный фрагментный подход можно использовать для диагностирования компонентов ЦС аналогичной структуры с другими параметрами.

Ключевые слова: цифровая система, абонент, коммутатор, линия связи, фрагментный подход, раскодирование.

1. Введение

В настоящей работе под цифровыми системами (ЦС) понимаются модели многомашинных или многопроцессорных

[1] *Виктор Алексеевич Ведешенков, доктор технических наук, старший научный сотрудник* (*vva@ipu.ru, Москва, ул. Профсоюзная, д.* 65, *тел.* (495) 334-75-90).

вычислительных систем, отражающие необходимые диагностические свойства и параметры анализируемых вычислительных систем.

Минимальный квазиполный граф образуется на основе однородного двудольного графа, одну долю которого составляют коммутаторы $m \times m$, а другую – m-портовые абоненты. Значение m выбирается минимальным, при котором любые два узла в одной доле связаны σ путями длины два через разные узлы в другой доле. В одной доле имеется n коммутаторов, а в другой – n абонентов. Каждый такой путь проходит через один коммутатор, и разные пути проходят через разные коммутаторы. Для рассматриваемых топологий параметры n и m связаны соотношением $n = m(m-1)/\sigma + 1$ и не могут быть взяты произвольно [6, 7, 8]. Пример такого графа размера 7×7 с двумя путями между двумя абонентами приведен на рис. 1 для $n = 7$, $m = 4$, $\sigma = 2$.

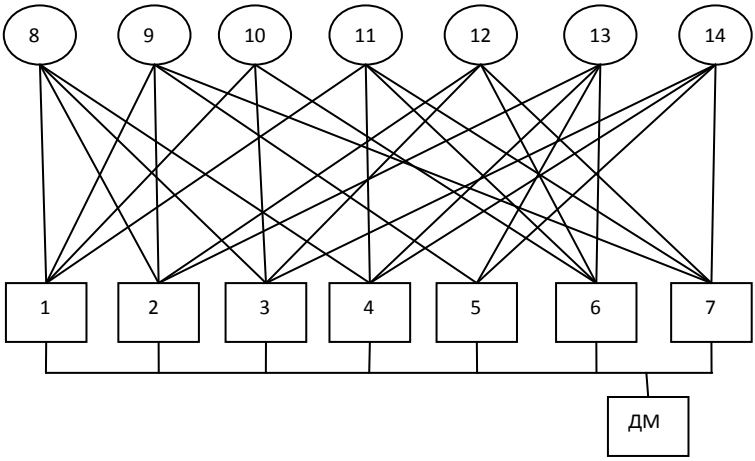

Рис. 1. Схема цифровой системы со структурой минимального квазиполного графа $S_{7,4,2}$ размера 7×7 с двумя путями между двумя абонентами: 1-7 – абоненты; 8-14 – коммутаторы; ДМ – диагностический монитор

В числе возможных областей применения графов с подобной новой структурой называют отказоустойчивые многомашинные вычислительные системы (МВС) реального времени, где, например, подмножество вершин одной доли представляет совокупность процессорных элементов или вычислительных машин, а подмножество вершин другой доли – коммутаторы [9, 10].

Экспериментально установлено, что предложенный в [2, 3] подход к диагностированию неисправных компонентов ЦС со структурой минимального квазиполного графа размера 7×7 с двумя путями между двумя абонентами обеспечивает достоверность однократного (без ремонта) диагностирования не больше, чем:

– двух неисправных абонентов и двух неисправных коммутаторов при исправном состоянии всех линий связи, или

– одного неисправного абонента и трёх неисправных коммутаторов при исправном состоянии всех линий связи, или

– двух неисправных линий связи при исправном состоянии всех абонентов и коммутаторов.

Одной из основных причин, определяющих значения установленных оценок достоверных результатов, является то, что диагностический тест построен для одновременной идентификации (без ремонта) неисправных компонентов всей анализируемой ЦС.

Альтернативным для подхода из [2, 3] является подход, в котором диагностические процедуры выполняются над частью (фрагментом) всей ЦС. В дальнейшем предполагается, что неисправности компонентов ЦС таковы, что прекращают работу тех компонентов, в которых они возникли, и не влияют на работоспособность других компонентов. Такая предпосылка гарантирует достоверность результатов диагностирования – подозреваются в неисправности только те компоненты, которые находятся в составе тестируемого фрагмента, так как компоненты других фрагментов не влияют на работоспособность компонентов этого фрагмента.

В статье разработан фрагментный подход к диагностированию неисправных компонентов ЦС рассматриваемой структуры, позволяющий увеличить

количество однозначно диагностируемых неисправных компонентов.

2. Постановка задачи

Исходная МВС со структурой минимального квазиполного графа представлена диагностическим графом, у которого n вершин одной доли представляют абоненты (процессоры) системы, n вершин другой доли – коммутаторы.

Будем считать, что допускаются устойчивые кратные отказы компонентов (абонентов, коммутаторов, линий связи), причем неисправности компонентов таковы, что прекращают работу тех компонентов, в которых они возникли, и не влияют на работоспособность других компонентов.

Организацией процессов диагностирования в различных фрагментах ЦС и обработкой полученных результатов занимается исправный диагностический монитор (ДМ) (в терминологии А. Авиженисса [1, стр. 13] – «обслуживающая мини-ЭВМ, исполняющая контрольные программы с целью наблюдения за работой остальной части системы»). ДМ находится в модуле, внешнем по отношению к диагностируемой ЦС [1].

Требуется разработать фрагментный подход к диагностированию (с ремонтом) неисправных компонентов анализируемой ЦС, однозначно идентифицирующий хотя бы один неисправный компонент.

3. Фрагментный подход к диагностированию неисправных компонентов ЦС со структурой минимального квазиполного графа (вариант 1)

Для проверки работоспособности компонента с предполагаемым характером функционирования (работа–отказ) со стороны проверяющего модуля достаточно послать к нему запрос, на который проверяемый компонент должен дать ответ в течение заданного интервала времени $\Delta\tau$. Отсутствие

ответа в течение интервала $\Delta\tau$ является признаком проявления (обнаружения) неисправности компонента.

Предположение о том, что возникшие неисправности компонентов не влияют на работоспособность других компонентов, позволяет считать, что неисправные компоненты обнаруживаются только на тех проверках, которые построены для тестирования этих компонентов. Функции контролирующего теста состоят в том, чтобы проверить каждый компонент анализируемой системы хотя бы один раз.

Еще один постулат теории диагностики: длина диагностического теста, однозначно идентифицирующего неисправные компоненты, уменьшается с уменьшением общего количества диагностируемых компонентов. Но для достоверного диагностирования даже одиночных неисправных компонентов необходимо, чтобы каждый компонент тестировался не менее чем в двух проверках.

Для реализации фрагментного подхода разделим ЦС со структурой минимального квазиполного графа размера $n \times n$ на фрагменты одинаковой структуры и состава. В качестве такого фрагмента выберем подсистему, включающую 1 коммутатор, m абонентов, связанных с данным коммутатором, и $2m$ линий связи между данным коммутатором и m абонентами. Поскольку в ЦС n коммутаторов, то ЦС разделяется на n фрагментов, связанных через общие абоненты. Таким образом, процесс диагностирования всей ЦС разделяется на n этапов диагностирования одного (очередного) фрагмента.

Контроль и диагностирование компонентов каждого фрагмента ЦС организуем следующим образом. Вначале выполняется контролирующий тест фрагмента. Если этот тест «проходит» успешно – не обнаруживает неисправностей, то выполняется переход к контролю следующего фрагмента. Если же тест не «проходит», то выполняется переход к диагностированию неисправных компонентов во фрагменте, на котором не «прошел» контролирующий тест. Если в результате диагностирования однозначно идентифицируется хотя бы один компонент, подозреваемый в неисправности, то выполняется его восстановление. После восстановления – замены компонента, подозреваемого в неисправности, исправным компонентом –

продолжается контроль остальных фрагментов ЦС. Если результат диагностирования – неоднозначен, то возможны различные действия, которые будут рассмотрены ниже.

Организацию процессов диагностирования в различных фрагментах ЦС и обработку полученных результатов выполняет исправный диагностический монитор (ДМ), имеющий доступ к каждому абоненту.

Одинаковая структура и состав фрагментов позволяют использовать метод и средства диагностирования, разработанные для одного фрагмента, для диагностирования компонентов других фрагментов после соответствующей замены номеров компонентов в таблицах проверок.

Для последующих построений введем в рассмотрение фрагмент той же структуры, что и фрагменты анализируемой ЦС, компоненты которого обозначим номерами 1, …, 8 – линии связи, 9 – коммутатор, 10, 11, 12, 13 – абоненты. Назовем этот фрагмент унифицированным.

Для контроля и диагностирования технического состояния компонентов этого фрагмента используем 12 проверок, вариант которых показан в таблице 1. По существу каждая проверка $p_{j,i,k}$ контролирует работоспособность начального абонента A_j, коммутатора c_i, конечного абонента A_k и двух соединяющих их линий связи: l_{j-i} и l_{j-i}. Нижний индекс (j, i, k), используемый в клетках заголовка второго, третьего и четвертого столбцов таблицы 1, состоит из индексов начального абонента, коммутатора и конечного абонента, участвующих в составе проверки данной строки. В клетках четвертого столбца «Оценка $r_{j,i,k}$» показаны символы r_i, которые заменяются нулем или единицей после выполнения проверки i-й строки таблицы 1.

Например, для проверки $p_{10,9,12}$ в клетке на пересечении первой строки и третьего столбца «Проверяемые подсистемы $U_{j,i,k}$» записаны номера начального абонента 10, коммутатора 9, конечного абонента 12 и линий связи 1 и 2, входящих в состав подсистемы $U_{10,9,12}$. (Отметим, что в клетках таблицы 1 номера индексов линий связи показаны без символа l.)

Таблица 1. Таблица проверок унифицированного фрагмента

№№ п./п.	Проверка $p_{j,i,k}$	Проверяемые подсистемы $U_{j,i,k}$	Оценка $r_{j,i,k}$
1	$p_{10,9,12}$	10, 1, 9, 2, 12	r_1
2	$p_{12,9,13}$	12, 3, 9, 4, 13	r_2
3	$p_{13,9,11}$	13, 5, 9, 6, 11	r_3
4	$p_{11,9,12}$	11, 7, 9, 2, 12	r_4
5	$p_{12,9,10}$	12, 3, 9, 8, 10	r_5
6	$p_{10,9,11}$	10, 1, 9, 6, 11	r_6
7	$p_{11,9,13}$	11, 7, 9, 4, 13	r_7
8	$p_{13,9,10}$	13, 5, 9, 8, 10	r_8
9	$p_{10,9,13}$	10, 1, 9, 4, 13	r_9
10	$p_{13,9,12}$	13, 5, 9, 2, 12	r_{10}
11	$p_{12,9,11}$	12, 3, 9, 6, 11	r_{11}
12	$p_{11,9,10}$	11, 7, 9, 8, 10	r_{12}

Проверки 9–12 образуют контролирующий тест для унифицированного фрагмента. Нетрудно убедиться, что на этих проверках каждая линия связи этого фрагмента тестируется только один раз, каждый абонент – дважды, а коммутатор – четыре раза, что неизбежно, так как коммутатор входит в каждую проверку.

Двенадцать проверок, показанных в таблице 1, образуют полный тест для контроля работоспособности 12 путей, которые возможны между любой парой из четырех абонентов, связанных с коммутатором 9 с помощью восьми прилегающих линий связи.

Каждая проверка выполняется следующим образом. ДМ передает очередному абоненту A_j команду «Переслать абоненту A_k через коммутатор c_i сообщение о проверке $p_{j,i,k}$». Абонент A_k, получив такое сообщение, подтверждает его получение ДМ, который заносит в клетку столбца «Оценка $r_{j,i,k}$» оценку 0. Если в течение интервала $\Delta\tau$ такого подтверждения от абонента A_k не приходит, то ДМ заносит в эту же клетку столбца «Оценка $r_{j,i,k}$» оценку 1. После записи оценки (0 или 1) ДМ передает

следующему абоненту команду на выполнение очередной проверки.

После выполнения всех 12 проверок в столбце «Оценка $r_{j,i,k}$» таблицы 1 будет записан 12-разрядный двоичный код, называемый синдромом проверок R_k [11]. Преобразование синдрома R_k в диагноз – номера неисправных и исправных компонентов анализируемой системы – называется раскодированием.

Для построения таблиц проверок, аналогичных таблице 1, для конкретных фрагментов ЦС сформируем таблицы соответствия номеров компонентов унифицированного фрагмента и фрагментов ЦС с коммутаторами 8–14 (показаны в таблицах 2_1 и 2_2).

Для примера в таблице 3 показана таблица проверок для фрагмента с коммутатором 8, построенная на основании записей таблицы 1 и таблицы 2_1.

Таблица 2_1. Таблица соответствия для коммутаторов 8,9,10,11

№ компонента унифиц. фрагмента	Номер компонента фрагмента (комм. 8)	Номер компонента фрагмента (комм. 9)	Номер компонента фрагмента (комм. 10)	Номер компонента фрагмента (комм. 11)
1	1–8	1–9	1–10	1–11
2	8–3	9–5	10–5	11–6
3	3–8	5–9	5–10	6–11
4	8–4	9–7	10–6	11–7
5	4–8	7–9	6–10	7–11
6	8–2	9–2	10–3	11–4
7	2–8	2–9	3–10	4–11
8	8–1	9–1	10–1	11–1
9	ком. 8	ком. 9	ком. 10	ком. 11
10	аб. 1	аб. 1	аб. 1	аб. 1
11	аб. 2	аб. 2	аб. 3	аб. 4
12	аб. 3	аб. 5	аб. 5	аб. 6
13	аб. 4	аб. 7	аб. 6	аб. 7

Таблица 2_2. Таблица соответствия для коммутаторов 12,13,14

Номер компонента унифиц. фрагмента	Номер компонента фрагмента (комм. 12)	Номер компонента фрагмента (комм. 13)	Номер компонента фрагмента (комм. 14)
1	2–12	2–13	3–14
2	12–6	13–5	14–5
3	6–12	5–13	5–14
4	12–7	13–6	14–7
5	7–12	6–13	2–14
6	12–3	13–4	14–4
7	3–12	4–13	4–14
8	12–2	13–2	14–3
9	ком. 12	ком. 13	ком. 14
10	аб. 2	аб. 2	аб. 3
11	аб. 3	аб. 4	аб. 4
12	аб. 6	аб. 5	аб. 5
13	аб. 7	аб. 6	аб. 7

Таблица 3. Таблица проверок (*для коммутатора 8*)

№№ п/п	Проверка $p_{j,i,k}$	Проверяемые подсистемы $U_{j,i,k}$	Оценка $r_{j,i,k}$
1	$p_{1,8,3}$	1, (1–8), 8, (8–3), 3	r_1
2	$p_{3,8,4}$	3, (3–8), 8, (8–4), 4	r_2
3	$p_{4,8,2}$	4, (4–8), 8, (8–2), 2	r_3
4	$p_{2,8,3}$	2, (2–8), 8, (8–3), 3	r_4
5	$p_{3,8,1}$	3, (3–8), 8, (8–1), 1	r_5
6	$p_{1,8,2}$	1, (1–8), 8, (8–2), 2	r_6
7	$p_{2,8,4}$	2, (2–8), 8, (8–4), 4	r_7
8	$p_{4,8,1}$	4, (4–8), 8, (8–1), 1	r_8
9	$p_{1,8,4}$	1, (1–8), 8, (8–4), 4	r_9
10	$p_{4,8,3}$	4, (4–8), 8, (8–3), 3	r_{10}
11	$p_{3,8,2}$	3, (3–8), 8, (8–2), 2	r_{11}
12	$p_{2,8,1}$	2, (2–8), 8, (8–1), 1	r_{12}

4. Способ раскодирования результатов проверок компонентов ЦС со структурой минимального квазиполного графа (на примере графа размера 7×7)

Для раскодирования результатов проверок и формирования диагноза воспользуемся алгебрологическим методом из [4], являющимся формализацией известного в технической диагностике метода пересечений.

Напомним основные положения этого метода.

А. Обозначим переменной $\overline{m_n}$ исправное состояние компонента с номером n, а переменной m_n – неисправное состояние компонента с номером n, т.е. для этих переменных справедливы логические соотношения:

– компонент n исправен – $\overline{m_n} = 1,\ m_n = 0,$

– компонент n неисправен – $\overline{m_n} = 0,\ m_n = 1.$

Функцию технического состояния компонентов с номерами $n_1, n_2, \ldots, n_k$:

– при нулевом результате их проверки тестом p_j (оценка $\overline{r_j}$) запишем так:

$\overline{F_j} = \overline{m_{n1}} \cdot \overline{m_{n2}} \cdot \ldots \cdot \overline{m_{nk}}$ (все компоненты исправны),

– при единичном результате их проверки тестом p_j (оценка r_j) запишем так:

$F_j = m_{n1} \vee m_{n2} \vee \ldots \vee m_{nk}$ (хотя бы один из компонентов неисправен).

Б. Предлагаемый метод раскодирования результатов тестирования ЦС на множестве проверок $p_1, n_2, \ldots, p_k$ состоит из следующих этапов.

1. Для каждого результата $\overline{r_j}$ (r_j) записать функцию $\overline{F_j}$ (F_j), в которую подставить номера компонентов, взятые из состава подсистемы, проверяемой p_j.

2. Построить функцию $\overline{\Phi}$ как конъюнкцию всех функций $\overline{F_j}$.

3. Упростить дизъюнктивные члены функций F_j, вычеркнув те переменные, для которых в функции $\overline{\Phi}$ есть одноименные переменные с отрицанием (с чертой сверху).

4. Построить функцию $\Psi = \Phi \bigcap \overline{\Phi}$ путем логического умножения оставшихся частей функций F_j и функции $\overline{\Phi}$, выполнить поглощение членов большей длины членами меньшей длины. При этом будут полезны известные формулы:

$$m_n \wedge \overline{m_n} = 0,\ m_n \vee \overline{m_n} = 1,\ m_n(m_n \vee m_p) = m_n .$$

5. Выделить из полученного логического выражения члены минимальной длины. Они определяют диагноз – подмножество компонентов, подозреваемых в неисправности по результатам выполненных проверок.

Приведем несколько примеров использования алгебрологического метода раскодирования результатов проверок. (С целью сокращения объема текста некоторые из производимых операций будем опускать.)

Пусть после выполнения 12 проверок таблицы 1 получен единичный синдром проверок (поскольку здесь и в дальнейшем синдром проверок будет всегда 12-разрядным, то дополнительно разрядность упоминать не будем):

$$R_x = 1_1 1_2 1_3 1_4 1_5 1_6 1_7 1_8 1_9 1_{10} 1_{11} 1_{12},$$

где номер проверки записан в качестве нижнего индекса полученной оценки: 1 или 0. Так как синдром R_x содержит только единичные оценки, то для последующего анализа нужны только функции F_j (показаны ниже):

$$F_1 = (10 \vee 1 \vee 9 \vee 2 \vee 12),\ F_2 = (12 \vee 3 \vee 9 \vee 4 \vee 13),$$

$$F_3 = (13 \vee 5 \vee 9 \vee 6 \vee 11),\ F_4 = (11 \vee 7 \vee 9 \vee 2 \vee 12),$$

$$F_5 = (12 \vee 3 \vee 9 \vee 8 \vee 10),\ F_6 = (10 \vee 1 \vee 9 \vee 6 \vee 11),$$

$$F_7 = (11 \vee 7 \vee 9 \vee 4 \vee 13),\ F_8 = (13 \vee 5 \vee 9 \vee 8 \vee 10),$$

$$F_9 = (10 \vee 1 \vee 9 \vee 4 \vee 13),\ F_{10} = (13 \vee 5 \vee 9 \vee 2 \vee 12),$$

$F_{11} = (12 \vee 3 \vee 9 \vee 6 \vee 11)$, $F_{12} = (11 \vee 7 \vee 9 \vee 8 \vee 10)$.

Поскольку номер 9 входит во все функции F_j, то при логическом умножении функций F_j их общий член (номер 9) выйдет из скобок. Сделаем это сразу при построении функции Ψ_x:

$$\Psi_x = 9 \vee (10 \vee 1 \vee 2 \vee 12) \wedge (12 \vee 3 \vee 4 \vee 13) \wedge$$
$$\wedge (13 \vee 5 \vee 6 \vee 11) \wedge (11 \vee 7 \vee 2 \vee 12) \wedge (12 \vee 3 \vee 8 \vee 10) \wedge$$
$$\wedge (10 \vee 1 \vee 6 \vee 11) \wedge (11 \vee 7 \vee 4 \vee 13) \wedge (13 \vee 5 \vee 8 \vee 10) \wedge$$
$$\wedge (10 \vee 1 \vee 4 \vee 13) \wedge (13 \vee 5 \vee 2 \vee 12) \wedge (12 \vee 3 \vee 6 \vee 11) \wedge$$
$$\wedge (11 \vee 7 \vee 8 \vee 10).$$

После логического умножения, поглощения членов большей длины членами меньшей длины и сокращения подобных членов получим начальную часть функции Ψ_x в следующем виде (здесь и в последующем знаки конъюнкции из начального вида функции заменены точкой):

(1) $\Psi_x = 9 \vee 10 \cdot 11 \cdot 12 \vee 10 \cdot 11 \cdot 13 \vee 10 \cdot 12 \cdot 13 \vee 11 \cdot 12 \cdot 13 \vee$
$\vee 1 \cdot 3 \cdot 5 \cdot 7 \vee 2 \cdot 4 \cdot 6 \cdot 8 \vee 1 \cdot 3 \cdot 5 \cdot 2 \cdot 4 \cdot 8 \vee 1 \cdot 3 \cdot 7 \cdot 2 \cdot 6 \cdot 8 \vee$
$\vee 1 \cdot 5 \cdot 7 \cdot 4 \cdot 6 \cdot 8 \vee 3 \cdot 5 \cdot 7 \cdot 2 \cdot 4 \cdot 6.$

Каждый член функции Ψ_x определяет один из вариантов диагноза: комбинацию номеров компонентов, неисправность которых приводит к единичному синдрому R_x. Для пояснения причин полученных результатов раскодирования заменим номера компонентов в (1) символами соответствующих компонентов с теми же номерами. Тогда вместо функции Ψ_x получим диагноз для компонентов унифицированного фрагмента Dz_9:

$Dz_9 = c_9 \vee a_{10} \cdot a_{11} \cdot a_{12} \vee a_{10} \cdot a_{11} \cdot a_{13} \vee a_{10} \cdot a_{12} \cdot a_{13} \vee a_{10} \cdot a_{11} \cdot a_{13} \vee$

(2) $\vee l_1 \cdot l_3 \cdot l_5 \cdot l_7 \vee l_2 \cdot l_4 \cdot l_6 \cdot l_8 \vee l_1 \cdot l_3 \cdot l_5 \cdot l_2 \cdot l_4 \cdot l_8 \vee l_1 \cdot l_3 \cdot l_7 \cdot l_2 \cdot l_6 \cdot l_8 \vee$
$\vee l_1 \cdot l_5 \cdot l_7 \cdot l_4 \cdot l_6 \cdot l_8 \vee l_3 \cdot l_5 \cdot l_7 \cdot l_2 \cdot l_4 \cdot l_6.$

Диагноз Dz_9, показанный в (2), содержит 11 подмножеств компонентов, подозреваемых в неисправности:

– коммутатор 9;
– или три из четырех абонентов (4 подмножества);
– или четыре входные линии связи: 1, 3, 5, 7
– или четыре выходные линии связи: 2, 4, 6, 8;

– или три входных и три выходных линии связи (6 подмножеств).

Нетрудно найти другие комбинации неисправных абонентов и линий связи, проверка которых дает единичный синдром. Например, продолжение функции Ψ_x из (1) будет содержать такие конъюнкции:

$\Delta\Psi_x = 10{\cdot}11{\cdot}4{\cdot}5 \vee 10{\cdot}12{\cdot}4{\cdot}5 \vee 10{\cdot}13{\cdot}2{\cdot}3 \vee$

$\vee\ 11{\cdot}12{\cdot}4{\cdot}5 \vee 11{\cdot}13{\cdot}2{\cdot}8 \vee 12{\cdot}13{\cdot}1{\cdot}7.$

Способ подбора подобных комбинаций состоит в том, чтобы отбирать очередной член из состава тех проверяемых подсистем, в которых нет ни одного из ранее отобранных членов данной комбинации. Подбор заканчивается тогда, когда в составе всех 12 проверяемых подсистем из таблицы 1 будет хотя бы один компонент из данной комбинации.

Состав конъюнкций из функций Ψ_x и $\Delta\Psi_x$ определяет границы количества неисправных компонентов разных типов, при достижении которых данный способ диагностирования не «работает»: не дает достоверного диагноза хотя бы для одного неисправного компонента.

В таблице 4 приведены результаты диагностирования некоторых характерных отказовых ситуаций, полученные с использованием алгебрологического метода раскодирования синдромов для проверок из таблицы 1.

В левом столбце таблицы 4 показан номер N_k отказовой ситуации, для которой в последующих клетках этой строки приведены соответствующие данные. В столбце «Отказовая ситуация» показаны номера компонентов ситуации N_k (в скобках – количества неисправных абонентов и линий связи в этой ситуации). В следующем столбце записан синдром R_k, полученный после выполнения проверок из таблицы 1, причем (как и ранее) номер проверки записан в качестве нижнего индекса оценки: 1 или 0. Результаты раскодирования синдрома R_k показаны в столбце «Диагноз», краткие примечания к этому диагнозу показаны в последнем столбце.

Анализ записей таблицы 4 позволяет сделать следующие выводы.

Таблица 4. Таблица результатов диагностирования 1

N_k	Отказ. ситуация	Синдром проверок R_k	Диагноз	Примечание
1	10 (1, 0)	$1_1 0_2 0_3 0_4 1_5 1_6$ $0_7 1_8 1_9 0_{10} 0_{11} 1_{12}$	$10 \vee 1 \cdot 8$	неоднозн. без общ. части
2	1·8 (0, 2)	$1_1 0_2 0_3 0_4 1_5 1_6$ $0_7 1_8 1_9 0_{10} 0_{11} 0_{12}$	$10 \vee 1 \cdot 8$	неоднозн. без общ. части
3	10, 12 (2, 0)	$1_1 1_2 0_3 1_4 1_5 1_6$ $0_7 1_8 1_9 1_{10} 1_{11} 1_{12}$	$(10 \vee 1 \cdot 8) \cdot$ $(12 \vee 2 \cdot 3)$	неоднозн. без общ. части
4	13, 6, 7 (1, 2)	$0_1 1_2 1_3 1_4 0_5 1_6$ $1_7 1_8 1_9 1_{10} 1_{11} 1_{12}$	$(13 \vee 4 \cdot 5) \cdot$ $(6 \cdot 7 \vee 11)$	неоднозн. без общ. части
5	10,·12, 4,·6 (2, 2)	$1_1 1_2 1_3 1_4 1_5 1_6$ $1_7 1_8 1_9 1_{10} 1_{11} 1_{12}$	Ψ_x	неоднозн. без общ. части
6	9 (комму-татор)	$1_1 1_2 1_3 1_4 1_5 1_6$ $1_7 1_8 1_9 1_{10} 1_{11} 1_{12}$	Ψ_x	неоднозн. без общ. части
7	1, 2, 3, 4 (0, 4)	$1_1 1_2 0_3 1_4 1_5 1_6$ $1_7 0_8 1_9 1_{10} 1_{11} 0_{12}$	$1 \cdot 2 \cdot 3 \cdot 4 \cdot$ $\bar{5} \cdot \bar{6} \cdot \bar{7} \cdot \bar{8} \cdot \bar{9} \cdot$ $\overline{10} \cdot \overline{11} \cdot \overline{12} \cdot \overline{13}$	однозн. полный
8	2, 4, 5, 7 (0, 4)	$1_1 1_2 1_3 1_4 0_5 0_6$ $1_7 1_8 1_9 1_{10} 0_{11} 1_{12}$	$2 \cdot 7 \cdot (4 \cdot 5 \vee 13)$	неоднозн. общ/часть
9	2, 4, 6, 8 (0, 4)	$1_1 1_2 1_3 1_4 1_5 1_6$ $1_7 1_8 1_9 1_{10} 1_{11} 1_{12}$	$9 \vee 1 \cdot 3 \cdot 5 \cdot 7$ $\vee\ 2 \cdot 4 \cdot 6 \cdot 8$	неоднозн. без общ. части

1. Результаты раскодирования синдрома R_k существенно зависят от числа нулей в его составе. Как отмечалось, раскодирование 12-разрядного единичного синдрома дает большое число вариантов комбинации компонентов, подозреваемых в неисправности. Как видно из (1) и (2), характерной чертой вариантов диагноза после раскодирования

такого синдрома является отсутствие однозначной общей части – одного или нескольких компонентов, входящих во все варианты диагноза.

Примеры отказовых ситуаций, приводящих к неоднозначному диагнозу без общей части, показаны в строках 1, 2, 3, 4, 5, 9 таблицы 4. Важной особенностью отказовых ситуаций, приведенных в строках 1, 3, 4, 5, является наличие в их составе неисправных абонентов 10, 12, 13.

Дело в том, что неисправность абонента 10 (ситуация 1) даст единичную оценку на проверках 1, 5, 6, 8, 9, 12, а после раскодирования – диагноз $(10 \vee 1 \cdot 8)$, показанный в строке 1. Одновременная неисправность двух линий связи 1 и 8, соединяющих абонент 10 с коммутатором 9, приведет к тому же синдрому (в строке 2) и тому же диагнозу, что и для неисправного абонента 10 (в строке 1).

По той же причине – последовательное соединение абонента 13 и примыкающих линий связи 4, 5 – в диагнозе для ситуаций 4, 8 появляется двучлен $(13 \vee 4 \cdot 5)$, означающий неоднозначность диагноза без общей части.

2. Для получения более определенного диагноза синдром R_k должен содержать несколько нулевых оценок. Из таблицы проверок унифицированного фрагмента (таблица 1) нетрудно заключить, что нулевая оценка любой проверки подтверждает исправность пяти компонентов: двух абонентов, коммутатора и двух линий связи. Знание исправных компонентов при раскодировании позволяет уменьшить состав некоторых функций F_j, вычеркнув переменные тех компонентов, которые признаны исправными проверкой с нулевой оценкой.

В строках 7, 8 таблицы 4 приведены примеры отказовых ситуаций, для которых формируется более определенный диагноз: однозначный, полный (строка 7) и неоднозначный с общей частью (строка 8). Такие виды диагноза удовлетворяют требованиям постановки задачи и могут быть использованы для восстановления исправности диагностируемого фрагмента.

Для повышения разрешающей способности диагноза в отказовых ситуациях, включающих один или несколько неисправных абонентов, выполним процедуры контроля и

диагностирования других фрагментов ЦС. Так как каждая линия связи и соответствующий коммутатор тестируются в составе только одного фрагмента, а каждый абонент тестируется в составе четырех фрагментов, то продолжение процедур контроля и диагностирования ЦС позволит получить информацию, уточняющую техническое состояние компонентов ЦС.

5. Фрагментный подход к диагностированию неисправных компонентов ЦС со структурой минимального квазиполного графа (на примере графа размера 7х7) – вариант 2

Первый вариант фрагментного подхода к диагностированию неисправных компонентов анализируемой ЦС использует унифицированные процедуры для поочередного диагностирования фрагментов, включающих разные коммутаторы и связанные с ними абоненты. Анализ результатов диагностирования компонентов унифицированного фрагмента, показанных в таблице 4, выявил недостаток такого подхода: увеличение количества отказовых ситуаций с неоднозначным диагнозом без общей части. Одной из причин таких диагнозов является наличие неисправного абонента в составе фрагмента.

Перспективным подходом к устранению этого недостатка является разделение во времени процедур диагностирования абонентов и примыкающих к ним линий связи. Для реализации такого подхода разделим процесс диагностирования ЦС на два этапа: проверка абонентов и проверка коммутаторов и линий связи в составе уменьшившихся фрагментов.

На первом этапе ДМ посылает очередному абоненту A_j, $j = 1, \ldots, 7$, запрос о его состоянии. Если абонент A_j исправен, то он отправляет ответ ДМ, который фиксирует оценку $r_j = 0$. Если в течение интервала $\Delta\tau$ ответ не придет, то ДМ фиксирует оценку $r_j = 1$. Поскольку каждый абонент проверяется поодиночке, то фиксируемая оценка ($r_j = 0$ или 1) однозначно определяет техническое состояние проверенного абонента: исправен или нет. После проверки всех абонентов ДМ передает

полученные оценки обслуживающему персоналу для замены на исправные тех абонентов, для которых получены оценки $r_j = 1$.

Второй этап состоит из n подэтапов (в последующем тексте $n = 7$, $m = 4$). На каждом подэтапе ДМ с помощью m исправных абонентов проверяет очередной фрагмент системы, включающий коммутатор и присоединенные к нему m входных и m выходных линий связи (для последующих ссылок назовем его фрагментом 2).

Во втором варианте фрагментного подхода используются те же диагностические процедуры (проверки, раскодирование синдрома с целью формирования диагноза), что и первом варианте. Поэтому при изложении второго варианта основное внимание будет уделено описанию различий в получаемых результатах.

Для диагностирования технического состояния унифицированного фрагмента 2 ДМ использует 12 проверок, показанных в таблице 5, которая построена на основе таблицы проверок универсального фрагмента (таблица 1). Исправные абоненты A_j и A_k, $j \neq k = 10, 11, 12, 13$, вынесены в отдельные столбцы. Поэтому в столбце «Проверяемые подсистемы $U_{j,i,k}$» таблицы 5 записаны номера коммутатора 9 и линий связи, входящих в состав подсистем $U_{j,i,k}$, проверяемых абонентом A_j. Отметим, что в клетках строк 1, …, 12 таблицы 5 (как в таблице 1) номера индексов линий связи показаны без символа l.

После выполнения всех 12 проверок в столбце «Оценка $r_{j,i,k}$» таблицы 5 будет записан синдром проверок R_x, раскодирование которого даст диагноз – номера неисправных и исправных компонентов анализируемого фрагмента системы.

В таблице 6 приведены результаты диагностирования некоторых характерных отказовых ситуаций, которые получены с использованием алгебрологического метода раскодирования. При построении таблицы 6 записи в столбцах имеют тот же смысл, что и в одноименных столбцах таблицы 4. Исключением является столбец «Отказовая ситуация», в котором показаны номера неисправных линий связи N_k, а в скобках – количества неисправных входных и выходных линий связи.

Таблица 5. Таблица проверок для унифицированного фрагмента 2

№№ п/п	Проверка $p_{j,i,k}$	Абонент A_j	Проверяемые подсистемы $U_{j,i,k}$	Абонент A_k	Оценка $r_{j,i,k}$
1	$p_{10,9,12}$	10	1, 9, 2	12	r_1
2	$p_{12,9,13}$	12	3, 9, 4	13	r_2
3	$p_{13,9,11}$	13	5, 9, 6	11	r_3
4	$p_{11,9,12}$	11	7, 9, 2	12	r_4
5	$p_{12,9,10}$	12	3, 9, 8	10	r_5
6	$p_{10,9,11}$	10	1, 9, 6	11	r_6
7	$p_{11,9,13}$	11	7, 9, 4	13	r_7
8	$p_{13,9,10}$	13	5, 9, 8	10	r_8
9	$p_{10,9,13}$	10	1, 9, 4	13	r_9
10	$p_{13,9,12}$	13	5, 9, 2	12	r_{10}
11	$p_{12,9,11}$	12	3, 9, 6	11	r_{11}
12	$p_{11,9,10}$	11	7, 9, 8	10	r_{12}

Анализ записей таблицы 6 позволяет сделать следующие выводы.

1. В постановке задачи написано, что разрабатываемый способ диагностирования должен однозначно идентифицировать хотя бы один неисправный компонент.

Для ситуаций N_1, N_2, приводящих к единичному синдрому, после раскодирования получим диагноз

(3) $\Psi_x^* = 9 \vee 1\cdot3\cdot5\cdot7 \vee 2\cdot4\cdot6\cdot8 \vee 1\cdot3\cdot5\cdot2\cdot4\cdot8 \vee$

$\vee\ 1\cdot3\cdot7\cdot2\cdot6\cdot8 \vee 1\cdot5\cdot7\cdot4\cdot6\cdot8\cdot \vee 3\cdot5\cdot7\cdot2\cdot4\cdot6.\cdot$

Нетрудно заметить, что этот диагноз можно получить из функции Ψ_x из (1) после вычеркивания членов, включающих переменные абонентов (ведь абоненты исправны).

Таблица 6. Таблица результатов диагностирования 2

N_k	Отказовая ситуация	Синдром проверок R_k	Диагноз	Примечание
1	1, 3, 5, 7 (4, 0)	$1_1 1_2 1_3 1_4 1_5 1_6$ $1_7 1_8 1_9 1_{10} 1_{11} 1_{12}$	Ψ_x^*	неоднозн. без общ. части
2	1, 2, 9 (1, 1, коммутатор)	$1_1 1_2 1_3 1_4 1_5 1_6$ $1_7 1_8 1_9 1_{10} 1_{11} 1_{12}$	Ψ_x^*	неоднозн. без общ. части
3	1, 3, 7 (3, 0)	$1_1 1_2 0_3 1_4 1_5 1_6$ $1_7 0_8 1_9 0_{10} 1_{11} 1_{12}$	$1 \cdot 3 \cdot 7 \cdot 4 \cdot \bar{2} \cdot \bar{5} \cdot \bar{6} \cdot \bar{8} \cdot \bar{9}$	однозн. не полный
4	1, 2, 3, 7 (3, 1)	$1_1 1_2 0_3 1_4 1_5 1_6$ $1_7 0_8 1_9 1_{10} 1_{11} 1_{12}$	$1 \cdot 2 \cdot 3 \cdot 7 \cdot 4 \cdot \bar{5} \cdot \bar{6} \cdot \bar{8} \cdot \bar{9}$	однозн. не полный
5	1, 2, 3, 4 (2, 2)	$1_1 1_2 0_3 1_4 1_5 1_6$ $1_7 0_8 1_9 1_{10} 1_{11} 0_{12}$	$1 \cdot 2 \cdot 3 \cdot 4 \cdot \bar{5} \cdot \bar{6} \cdot \bar{7} \cdot \bar{8} \cdot \bar{9}$	однозн. полный
6	1, 2, 3, 6, 7 (3, 2)	$1_1 1_2 1_3 1_4 1_5 1_6$ $1_7 0_8 1_9 1_{10} 1_{11} 1_{12}$	$2 \cdot 3 \cdot 6 \cdot 7 \cdot \bar{9}$ $(1 \vee 4) \cdot \bar{5} \cdot \bar{8}$	неоднозн. общ/часть

Диагноз Ψ_x^* из (3) содержит 7 дизъюнктивных членов, которые определяют возможные варианты неоднозначного диагноза без общей части:

– неисправен коммутатор 9

– или четыре входные линии связи: $1 \cdot 3 \cdot 5 \cdot 7$,

– или четыре выходные линии связи: $2 \cdot 4 \cdot 6 \cdot 8$,

– или три входных и три выходных линии связи (4 варианта).

Такой диагноз не удовлетворяет требованиям постановки задачи. Он показывает границы количества неисправных компонентов разных типов, при достижении которых данный способ диагностирования не «работает». Индикатором,

предупреждающим о получении такого диагноза, является единичный синдром проверок *R*, показанный в строках 1, 2.

2. В таблице 6 показаны примеры нескольких отказовых ситуаций, синдромы проверок *R* которых содержат один или несколько нулей, что приводит к диагнозам, удовлетворяющим требованиям постановки задачи:

– для ситуации 5 – диагноз однозначный и полный;

– для ситуации 3, 4 – диагноз однозначный, но неполный: осталось не идентифицированным техническое состояние компонента 4, над номером которого поставлен знак «тильда»;

– для ситуации 6 – диагноз неоднозначный, но с общей частью: подозреваются в неисправности компоненты 2, 3, 6, 7, а компонент 1 показан в скобках (в числе неоднозначно диагностируемых).

Суммируя вышесказанное, можно заключить, что второй вариант фрагментного подхода гарантирует однозначное диагностирование:

– от одного до семи неисправных абонентов

– или не больше четырех неисправных линий при исправном коммутаторе и четырех исправных абонентах каждого фрагмента.

3. Таблицы проверок (таблица 5) и результатов диагностирования (таблица 6) построены в номерах компонентов унифицированного фрагмента. Чтобы использовать их для диагностирования компонентов очередного фрагмента анализируемой ЦС, нужно воспользоваться соответствующей таблицей сопоставления – таблицей 2_1 или таблицей 2_2. Как и ранее в варианте 1, с помощью одной из этих таблиц преобразуем таблицу проверок унифицированного фрагмента 2 (таблица 5) в таблицу проверок анализируемого фрагмента. Диагноз, полученный после раскодирования 12-разрядного синдрома проверок анализируемого фрагмента с помощью универсальной программы, будет записан в номерах компонентов унифицированного фрагмента 2. Для получения диагноза в номерах компонентов анализируемого фрагмента эти номера нужно преобразовать в номера данного фрагмента с помощью той же таблицы сопоставления. Подобные действия

нужно выполнить для каждого фрагмента анализируемой системы.

В [4] представлена модификация варианта фрагментного диагностирования, разработанная для фрагмента 3, содержащего коммутатор и 4 линии связи, соединяющие данный коммутатор с четырьмя абонентами. Уменьшение числа линий связи с восьми до четырех позволило уменьшить число проверок в таблице, аналогичной таблице 5, с двенадцати до шести при обеспечении диагноза, удовлетворяющего требованиям постановки задачи.

6. Особенности восстановления ЦС со структурой минимального квазиполного графа (на примере графа размера 7×7)

В соответствии с записями в столбце «Примечание» таблиц 4 и 6 приведенные результаты диагностирования отказовых ситуаций можно объединить в две группы:

1) диагноз однозначный или неоднозначный, но с общей частью;

2) диагноз неоднозначный, без общей части.

Получив диагноз, относящийся к первой группе, обслуживающий персонал должен заменить исправными компоненты общей части, подозреваемые в неисправности, и повторить процедуры диагностирования для подтверждения исправности восстановленного фрагмента или идентификации оставшихся неисправных компонентов.

Неоднозначный диагноз без общей части получается, прежде всего для отказовых ситуаций, синдром проверок которых содержит только единицы. В таблицах 4 и 6 показаны примеры отказовых ситуаций, синдромы которых содержат один или несколько нулей, но после их раскодирования также получается неоднозначный диагноз без общей части.

Для восстановления отказовых ситуаций с диагнозом из второй группы нужно сократить количество неисправных компонентов в составе анализируемой отказовой ситуации, заменив некоторые компоненты, подозреваемые в неисправности, на исправные. При этом право выбора «первого

хода» – первоначальных замен до получения приемлемого синдрома – остается за обслуживающим персоналом, который может использовать дополнительные критерии.

Одним из таких критериев может быть выбор из формулы диагноза членов наименьшей длины, поскольку вероятность отказа меньшего числа компонентов больше, чем вероятность отказа большего числа компонентов. Такое вероятностное обоснование справедливо для компонентов одного типа и может быть ошибочным для компонентов разных типов. В качестве примеров можно назвать ситуации 5 и 6 в таблице 4, синдромы проверок которых – единичные коды. Диагноз таких синдромов – функция Ψ_x из (1) – имеет член наименьшей длины: коммутатор 9, которого нет в составе ситуации 5. Другой пример возможной ошибки – ситуация 2 в таблице 4 (одновременная неисправность линий связи 1 и 8), синдром проверок которой неразличим от синдрома проверок для ситуации 1 (абонент 10). Как видно из табл. 4, диагноз для этих ситуаций – одинаков ($10 \vee 1 \cdot 8$), и выбор для замены члена меньшей длины (компонента 10) означает ошибку, если во фрагменте имеет место отказовая ситуация 2.

Еще более неопределенным представляется выбор заменяемых компонентов в ситуации, для которой варианты диагноза имеют равное число компонентов одного типа, подозреваемых в неисправности. В этом случае в качестве критерия выбора очередного заменяемого компонента можно использовать число вхождений каждого компонента в состав подсистем, контролируемых на двенадцати проверках из таблицы 1 (или таблицы 5 для второго варианта). Подсчет показывает, что коммутатор имеет 12 вхождений, каждый из четырех абонентов – по 6 вхождений, каждая из восьми линий связи – по 3 вхождения. В соответствии с этими числами в первом варианте фрагментного подхода предлагается в каждом фрагменте заменять сначала коммутатор, затем поочередно – по одному случайно выбранному абоненту, затем поочередно – по одной случайно выбранной линии связи. После каждой замены повторяются проверки из таблицы 1. Замены проводятся до

получения приемлемого диагноза: однозначно идентифицирующего хотя бы один неисправный компонент.

Второй вариант имеет 2 этапа диагностирования: сначала поочередно проверяются и заменяются неисправные абоненты. На втором этапе предлагается в каждом фрагменте заменить сначала коммутатор, затем поочередно – по одной линии связи.

И в первом, и во втором вариантах можно заменять не один, а несколько компонентов. Это позволит сократить время восстановления фрагмента за счет возможного использования большего числа исправных резервных компонентов.

Например, пусть неисправны 4 входные линии связи (ситуация N_1 в таблице 6). Тогда проверки из таблицы 5 дадут единичный синдром и неоднозначный диагноз без общей части. Заменим входную линию связи 5, тогда исходная ситуация N_1 перейдет в ситуацию N_3 с однозначным, но неполным диагнозом, определяющим однозначное продолжение действий по восстановлению фрагмента. Неизбежное увеличение времени диагностирования за счет нескольких прогонов проверок из таблиц 1 или 5 представляется не слишком большой задержкой по сравнению со временем замены неисправных компонентов.

7. Заключение

Для увеличения количества однозначно диагностируемых компонентов цифровых систем со структурой минимального квазиполного графа предложен фрагментный подход к диагностированию компонентов анализируемых ЦС. Разработано 2 варианта реализации фрагментного подхода, различающиеся составом диагностируемых фрагментов. В первом варианте ЦС разделяется на n фрагментов одинакового состава: коммутатор, m абонентов, m входных и $2m$ выходных линий связи коммутатора с абонентами. Во втором варианте n абонентов выделяются в отдельный блок, а фрагменты получаются меньшего состава: коммутатор, m входных и $2m$ выходных линий связи коммутатора с исправными абонентами.

Организацию процессов диагностирования в различных подсистемах ЦС и обработку полученных результатов

выполняет исправный диагностический монитор (ДМ), имеющий доступ к каждому абоненту.

Применительно к каждому фрагменту и блоку абонентов ДМ реализует последовательный способ диагностирования, включающий тестирование компонентов фрагмента (блока) и замену идентифицированных неисправных компонентов на исправные (с участием обслуживающего персонала).

В первом варианте фрагментного подхода наличие неисправных абонентов в составе проверяемых фрагментов ограничивает возможные однозначные диагнозы одиночными неисправными компонентами каждого фрагмента.

Второй вариант фрагментного подхода гарантирует достоверное диагностирование

– от одного до семи неисправных абонентов

– или не больше четырех неисправных линий при исправном коммутаторе и четырех исправных абонентах каждого фрагмента.

Сравнение этих цифр с величинами диагностируемости ЦС [2, 3], приведенными во введении, позволяет заключить, что эти цифры существенно больше величин из [2, 3]. Такое различие обусловлено использованием фрагментного подхода, диагностирующего отдельные фрагменты, включающие только часть компонентов всей ЦС, и возможностями восстановления (ремонта) идентифицированных неисправных компонентов.

Эти преимущества второго варианта фрагментного подхода прогнозируют возможности его практического применения, прежде всего, для диагностирования ЦС на стадии первичной отладки или после больших перерывов в использовании по назначению. Что же касается первого варианта фрагментного подхода, то его целесообразно использовать для диагностирования одиночных неисправных компонентов в процессе использования ЦС по назначению.

Представленные варианты фрагментного подхода к диагностированию компонентов анализируемых ЦС состоят из последовательности проверок и восстановления компонентов очередного фрагмента. Основным параметром, определяющим количество необходимых проверок для каждого фрагмента, является значение m – количество портов одного абонента.

Длина диагностического теста для отдельного фрагмента определяется формулой $[L = m(m-1)]$, в рассмотренном примере $L = 12$ (в таблицах 1 и 5). С увеличением значения m длина теста L достаточно быстро растет. Можно уменьшить число проверок в диагностическом тесте, но при этом уменьшатся количество и кратность однозначно диагностируемых неисправных компонентов. Если окажется, что длина выбранного теста $L < L^* = 2m$, то такой тест не будет различать даже некоторые неисправные одиночные компоненты.

Количество и продолжительность выполняемых процедур пропорциональна числу модулей n в одной доле. Что же касается параметра σ – количества путей между двумя абонентами, проходящих через σ коммутаторов, – то этот параметр не влияет на процедуры проверок одного фрагмента, так как указанные пути проходят через разные коммутаторы.

Представленный фрагментный подход к диагностированию компонентов цифровых систем со структурой минимального квазиполного графа является достаточно общим и может быть распространен на ЦС аналогичной структуры с другими параметрами.

В качестве примера подобного распространения в [5] представлен вариант фрагментного диагностирования неисправных компонентов цифровой системы со структурой минимального квазиполного графа с параметрами $n = 13$, $m = 4$, $\sigma = 1$.

Литература

1. АВИЖЕНИС А. *Отказоустойчивость – свойство, обеспечивающее постоянную работоспособность цифровых систем* // ТИИЭР. – 1978. – Т. 66, №10. – С. 5–25.
2. ВЕДЕШЕНКОВ В.А., КУРАКО Е.А., ЛЕБЕДЕВ В.Н. *О диагностировании цифровых систем со структурой минимального квазиполного графа размера 7×7* // Проблемы управления. – 2014. – №6. – С. 68–76.

3. ВЕДЕШЕНКОВ В.А., КУРАКО Е.А., ЛЕБЕДЕВ В.Н. *О диагностируемости компонентов цифровых систем со структурой минимального квазиполного графа размера 7×7 с 2 путями между 2 абонентами* // Управление большими системами. – Вып. 58. – М: ИПУ РАН, 2015. – С. 90–114.
4. ВЕДЕШЕНКОВ В.А.; *Подход к фрагментному диагностированию компонентов цифровых систем со структурой минимального квазиполного графа* (*на примере графа размера 7×7*) // Проблемы управления. – 2016. – №6. – С. 53–58.
5. ВЕДЕШЕНКОВ В.А. *О диагностировании неисправных компонентов цифровых систем со структурой минимального квазиполного графа (на примере графа размера 13×13* // Труды XXIV Международной научной конференции «Проблемы управления безопасностью сложных систем», Москва, 21 декабря 2016 г. – Издательский центр РГГУ, 2016. – С. 188–192.
6. КАРАВАЙ М.Ф., ПОДЛАЗОВ В.С**.** *Распределенный полный коммутатор как «идеальная» системная сеть для многопроцессорных вычислительных систем* // Управление большими системами. – Вып. 34. – М:. ИПУ РАН. 2011. – С. 92–116.
7. КАРАВАЙ М.Ф., ПАРХОМЕНКО П.П., ПОДЛАЗОВ В.С. *Комбинаторные методы построения двудольных однородных минимальных квазиполных графов (симметричных блок-схем)* // Автоматика и телемеханика. – 2009. – №2. – С. 153–170.
8. КАРАВАЙ М.Ф., ПАРХОМЕНКО П.П., ПОДЛАЗОВ В.С. *Простые методы построения квазиполносвязных графов (симметричных блок-схем)* // Сб. тр. IV Междунар. конф. «Параллельные вычисления и задачи управления (PACO'2008). – М.: ИПУ РАН, 2008. – CD-ROM. – С. 232–249.
9. ALVERSON B., FROESE E., KAPLAN L., ROWETH D. *Cray XC™ Series Network, WP-Aries01-1112.* – Cray Inc, 2012. – 28 p. – URL: http://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf (дата обращения: 16.03.2017).

10. ALVERSON R., ROWETH D., KAPLAN L. *CRAY INC. The Gemini System Interconnect* // 18th IEEE Symposium on High Performance Interconnects. – 2009. – P. 83–87.

A FRAGMENTATION APPROACH TO DIAGNOSIS OF DIGITAL SYSTEMS COMPONENTS WITH MINIMAL QUASICOMPLETE GRAPH STRUCTURES (AN EXAMPLE OF 7 × 7 GRAPH)

Victor Vedeshenkov, Institute of Control Sciences of RAS, Moscow, Doctor of Science, Senior research associate (*vva@ipu.ru*).

Abstract: The problem of fault diagnosis in a multi-processor computing system is considered. The model of a multi-processor system is called a digital system (DS). A system consists of switches, end-nodes and communication links. The communication graph is bipartite where one part is the switches and the other part is the end-nodes. We propose two methods of a digital system diagnosis based on the fragmentation approach. The first method divides the system into 7 unified fragments, each of which contains a switch, 4 end-nodes and 8 communication links. In the second method, each of the 7 end-nodes is tested separately and every fragment includes a switch and 8 communication links. A reliable diagnosis monitor (DM) sequentially tests the components of a fragment, decodes the received syndrome and then sends the diagnosis to the maintenance staff for a recovery of the broken components. The proposed methods can be applied for diagnosis of another system with similar structure and different parameters.

Keywords: digital systems, communication graph, switch, distributed multi-processor system, fragmentation, decoding.

Статья представлена к публикации членом редакционной коллегии В.В. Мазаловым.

Поступила в редакцию 16.02.2016.
Опубликовано 31.03.2017.