Математика программных систем

Mathematics of Program Systems

2010

Межвузовский сборник научных статей

Выпуск 7 (2010 г.)

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение высшего
профессионального образования
«Пермский государственный университет»

Государственное образовательное бюджетное учреждение
высшего профессионального образования
«Государственный университет – Высшая школа экономики»
(Пермский филиал)

Государственное образовательное учреждение высшего
профессионального образования
«Кубанский государственный университет»

МАТЕМАТИКА ПРОГРАММНЫХ СИСТЕМ

Межвузовский сборник научных статей

Выпуск 7 (2010 г.)

Пермь 2010

УДК 004.4+004.8+004.9
ББК 32.973
М34

Математика программных систем: межвуз. сб. науч. ст. / М34 под ред. А.И. Микова и Л.Н. Лядовой; Перм. гос. ун-т. – Пермь, 2010. – Вып. 7 (2010 г.). – 140 с.: ил.

ISBN 978-5-7944-1742-5 (Вып. 7)
ISBN 978-5-7944-1741-8

В сборник включены статьи, относящиеся к следующим направлениям научной работы: математические модели алгоритмов и программ; системы искусственного интеллекта; системы моделирования и машинная имитация; моделирование и технологии разработки информационных систем. Часть статей подготовлена при поддержке РФФИ (проект № 10-01-00794). В сборнике представлены результаты, полученные при выполнении проекта в 2010 г.

Материалы сборника могут представить интерес для научных работников, разработчиков информационных систем различного назначения и системных программистов, студентов вузов и аспирантов, интересующихся современными направлениями в области информационных технологий, а особенно тем категориям специалистов, которые работают в области имитационного моделирования, программной инженерии.

УДК 004.4+004.8+004.9
ББК 32.973

Печатается по решению редакционно-издательского совета Пермского государственного университета

Рецензенты: доктор физико-математических наук, профессор, директор учебного центра «Информатика» ***С.В. Русаков***;

кафедра информационных систем и математических методов в экономике Пермского государственного университета.

ISBN 978-5-7944-1742-5 (Вып. 7)
ISBN 978-5-7944-1741-8

ВВЕДЕНИЕ

Выпуск сборника 2010 г., в основном, включает статьи, подготовленные при финансовой поддержке РФФИ. Работы, результаты которых представлены в сборнике, выполнены в рамках реализации проекта № 10-01-00794 «Методы создания DSL-инструментария и моделирования предметных областей на основе многоуровневых онтологий и графовых грамматик», поддержанного грантом РФФИ.

Большое внимание в сборнике уделяется средствам создания адаптируемых информационных систем, базирующихся на предметно-ориентированных моделях, описываемых с помощью языков, создаваемых с использованием языковых инструментариев, а также их практическому применению при разработке информационных систем различного назначения.

В сборник включены также статьи, посвященные разработке ресурсно-эффективных алгоритмов.

Отдельный раздел сборника посвящен разработке систем искусственного интеллекта и их приложений для различных областей.

В сборнике представлены статьи сотрудников, аспирантов и студентов нескольких университетов Республики Беларусь, Российской Федерации, Украины.

МАТЕМАТИЧЕСКИЕ МОДЕЛИ АЛГОРИТМОВ И ПРОГРАММ

И.И. Блох, А.В. Дураков

Пермский государственный университет

ilyablokh@gmail.com, adurakov@gmail.com

АЛГОРИТМЫ ПОСТРОЕНИЯ МАРШРУТА ДЛЯ ГИС-ПРИЛОЖЕНИЯ ТУРИСТА

Введение

В наше время, в условиях глобализации непрерывно идёт процесс укрепления международных связей, что влечёт за собой постоянно растущий человеческий трафик между странами. Всё больше людей отправляется в туристические поездки, и часто перед ними встаёт вопрос о выборе подходящего для них туристического маршрута.

Задача выбора подходящего маршрута очень актуальна, как ввиду широкого круга заинтересованных в её решении лиц, так и ввиду отсутствия действующих приложений, способных эту задачу решить.

На сегодняшний день эта проблема решается двумя способами. Первый – это использование стандартных маршрутов, предлагаемых туристическими агентствами. Однако такой вариант часто не подходит людям по одному или нескольким из следующих критериев: цена, посещение всех интересуемых достопримечательностей, продолжительность экскурсии и т.д. К тому же есть риск, что экскурсовод окажется непрофессионалом. Альтернативный путь – использование аудиогида,

с которым можно самостоятельно передвигаться по городу и слушать запись. Однако такие гиды никак не помогают человеку именно с выбором маршрута, который больше всего ему бы подошёл.

В данной статье предлагаются и исследуются различные методы построения наиболее подходящего для человека туристического маршрута.

Постановка задачи построения маршрута

Всего выделено два ключевых фактора, определяющих то, насколько маршрут подходит человеку. Первый – это время, необходимое для его прохождения.. Второй – это содержание маршрута, а именно совокупность всех находящихся на нём достопримечательностей и иных интересных мест с учётом их релевантности для конкретного туриста. Под релевантностью понимается соответствие пожеланиям человека.

Для нахождения подходящего маршрута необходимо последовательно решить следующие задачи:

1. Необходимо предложить способ вычисления релевантности каждого отдельного объекта. Каждому объекту в городе соответствует несколько тегов, всё множество которых разбито на тематические словари. Пользователь имеет возможность указать несколько интересующих его тегов. Таким образом, вычисление релевантности будет основываться на поиске подходящих объектов при помощи тегов. Также на значение релевантности будет оказывать влияние рейтинг объекта.

2. Необходимо предложить способ определения полезности маршрута. Полезность маршрута – некоторое численное выражение его содержания, основанное на совокупности релевантностей входящих в него объектов.

3. Необходимо разработать алгоритм нахождения подходящего маршрута. Дополнительную сложность в задачу вносит то обстоятельство, что неизвестно, сколько объектов должно входить в такой маршрут, то есть в принципе их может оказаться от 0 до N, где N – количество объектов на карте.

В качестве исходных данных выступает карта города с отмеченными на ней различными местами интереса, достопримечательностями, отелями и т.д., а также информация обо всех объектах карты: текстовое описание, координаты, рейтинг и т.д.

Для всей работы с картой используется платформа Microsoft Bing Maps.

Подходы к решению задачи

Для решения поставленной задачи выполнена ее декомпозиция на три задачи, описанные выше. Рассмотрим их решения.

Релевантность объекта

Как ранее уже было установлено, на значение релевантности объекта влияют его рейтинг и то, насколько близкими оказались теги пользователя к его собственным (далее близость).

Определим способ измерения близости. Всего возможны три ситуации, в которых каждый тег, введённый пользователем:

1. Является тегом объекта, то есть полное совпадение.
2. Не является тегом объекта, однако принадлежит тому же словарю или словарям тегов, что и некоторые из принадлежащих объекту. Данную ситуацию можно рассматривать как косвенную близость, при которой объект тематически пользователю подходит, однако полного совпадения нет.
3. Не находится среди тегов объектов и не принадлежит ни одному из словарей, в который входят теги объекта.

За каждый случай полного совпадения (ситуация 1) значение близости увеличивается на некоторую величину k_1, при косвенной близости (ситуация 2) на величину $k_2 < k_1$ за попадание в один словарь, а в случае ситуации 3 значение близости не увеличивается. Таким образом, если пользователь ввел N тегов, из которых n_1 подходят к случаю 1, n_2 к случаю 2 и n_3 к случаю 3, при этом для n_2 тегов из второго случая имеет место n_2^* попаданий в словари тегов, то значение близости будет равно

$$B = n_1 \cdot k_1 + n_2^* \cdot k_2 + n_3 \cdot 0.$$

Рейтинг объектов R измеряется от 0 до 100%. Предлагается следующий способ учёта рейтинга, при котором релевантность будет равна:

$$RL = B \cdot 0.01 \cdot R.$$

Полезность маршрута

Полезность является показателем того, насколько объекты, содержащиеся в маршруте, в своей совокупности подходят человеку. Таким образом, полезность будет считаться как некоторая функция от входящих в маршрут релевантностей. Дальнейшей задачей будет построение такого маршрута, у которого полезность будет максимальна.

Предлагается следующий вариант расчета полезности:

$$P = \sum_i RL_i.$$

Полезность маршрута равняется сумме релевантностей его объектов. При таком способе вычисления полезности предполагается, что наиболее важным критерием является именно высокий уровень соответствия интересам пользователя на всём протяжении маршрута, то есть максимальной должна быть сумма релевантностей. При этом самый релевантный объект может быть и не включен в маршрут вообще.

Алгоритм, основанный на решении задачи коммивояжера

Отсортируем список объектов по убыванию их релевантностей. Наиболее подходящий маршрут будет строиться следующим образом:

1. Из упорядоченного списка объектов выбирается первые *M*.
2. Для выборки решается задача коммивояжера
3. Затраченное на весь маршрут время сравнивается с временем, выделенным пользователем для прохождения маршрута
4. Если алгоритм решения задачи коммивояжера потратил больше времени, чем требуется, то величина *M* будет уменьшена. Если же алгоритм потратил меньше времени, то *M* будет увеличена. Изменение величины *M* проводится по принципу дихотомии, то есть каждое изменение в два раза меньше предыдущего. В первый раз *M* увеличивается или уменьшается также в два раза.
5. Алгоритм завершает работу тогда, когда происходит стабилизация значения *M*, то есть при очередном прохождении четвертого шага изменения *M* равняются 0.

В результате работы алгоритма получается маршрут, проходящий через наиболее релевантные для туриста объекты, и при этом занимающий время, максимально близкое к требуемому.

Приведённый выше алгоритм не гарантирует получение самого полезного маршрута из всех возможных. Это объясняется тем, что маршрут строится из самых релевантных объектов, а полезность считается как сумма релевантностей. Однако может оказаться так, что большее значение полезности будет достигаться при включении в маршрут большего числа объектов, не являющихся наиболее релевантными среди всех. Например, самый релевантный объект, согласно алгоритму, будет всегда включен в маршрут. Однако могут найтись такие объекты, чья суммарная релевантность выше, при тех же временных затратах на посещение. Предложенный алгоритм не сможет построить такой маршрут, который включал бы в себя набор менее релевантных по отдельности объектов вместо одного, более релевантного, чем любой из набора.

Генетический алгоритм

Для построения подходящего маршрута предлагается использовать классический генетический алгоритм.

Предлагается следующая функция приспособленности в решаемой задаче:

$$fit = \begin{cases} P, t \leq t_{max} \\ \varepsilon, t > t_{max} \end{cases},$$

где P – полезность маршрута; t – время, необходимое для прохода маршрута (вычисляется при помощи жадного алгоритма решения задачи коммивояжера); t_{max} – время, которым ограничен пользователь ($\varepsilon > 0, \varepsilon \approx 0$).

Способ кодирования решения следующий: каждая особь кодирует набор объектов, которые содержатся в маршруте. Таким образом, количество генов совпадает с числом всех рассматриваемых объектов. *i*-й ген может принимать значения 0 или 1 (0 означает, что *i*-й объект не включен в маршрут, а 1 – объект в маршрут включен).

Для скрещивания используется одноточечный кроссовер. Однако при вышеописанном способе кодирования особи присутствует проблема одноточечного кроссовера, которая может в значительно степени снизить эффективность алгоритма. Если кодировать хромосому таким образом, что *i*-й ген соответствует *i*-му объекту, то для получения хороших результатов потребуется большее количество итераций.

Для решения проблемы одноточечного кроссовера, предлагается иначе подойти к кодированию решений. Для этого список объектов переформировывается таким образом, чтобы расположенные близко друг к другу объекты были расположены близко друг к другу в новом списке. Новый список, обладающий необходимым свойством, может быть представлен как порядок обхода всех объектов жадным алгоритмом для задачи коммивояжера.

Таким образом, решение будет кодироваться иначе: *i*-му гену хромосомы будет соответствовать тот объект, который является *i*-м в обходе всех объектов жадным алгоритмом для задачи коммивояжера.

Сравнение алгоритмов

Были запрограммированы и сопоставлены три алгоритма поиска подходящего пути для туриста. Первый – алгоритм, основанный на составлении упорядоченного по релевантностям списка объектов и на задаче коммивояжера. Второй – классический генетический алгоритм с порядковым кодированием хромосом. Третий алгоритм – модификация

второго генетического алгоритма с устранённой проблемой одноточечного кроссовера.

Использовалось два критерия сравнения:

1) итоговая полезность маршрута (считается, что чем выше полезность маршрута, тем лучше работает алгоритм);

2) разница между временем, заданным пользователем, и временем, вычисленным алгоритмом (эта разница должна быть как можно ближе к 0).

Было проведено несколько сравнительных тестов. В каждом из них фиксировались данные, получаемые от пользователя, и выбирались различные параметры алгоритмов. За каждый тест алгоритм решал поставленную задачу фиксированное число раз.

Было установлено, что на простых тестах, в которых пользователь вводил мало информации в систему о своих предпочтениях, первый алгоритм работает лучше (рис. 1).

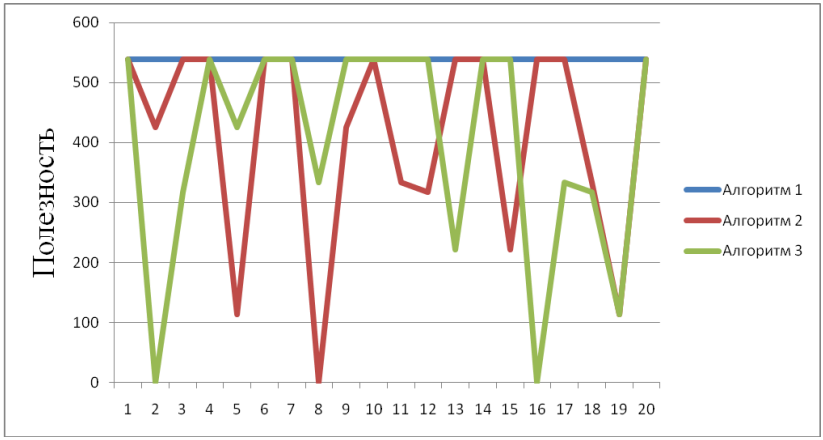


Рис. 1. График полезности маршрутов, составленных с помощью различных алгоритмов при простом тесте

Однако на сложных тестах, с большим объёмом входных данных, генетические алгоритмы показали значительно лучшие результаты (рис. 2).

Самые лучшие результаты показал генетический алгоритм со специальным способом кодирования особей. Этот алгоритм может использоваться в дальнейшем при создании ГИС-приложения туриста.

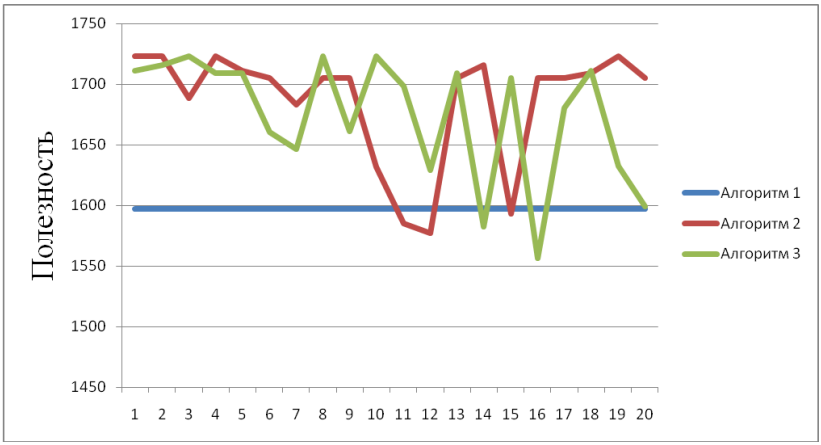


Рис. 2. График полезностей маршрутов, составленных различными алгоритмами при сложном тесте

Библиографический список

1. *Вороновский Г.К., Махотенко К.В., Петрашев С.Н., Сергеев С.А.* Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. – Х.: ОСНОВА, 1997.
2. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ / Пер. с англ. под ред. Шеня А. – М.: МЦНМО: БИНОМ. Лаборатория знаний, 2004.
3. *Яминов Б.* Генетические Алгоритмы. [Электронный ресурс] [*http://rain.ifmo.ru/cat/view.php/theory/unsorted/genetic-2005*].
4. Bing Maps Platform Overview.URL [Электронный ресурс] [*http://www.microsoft.com/maps/product/overview.aspx*].

Е.Ю. Данилова

Пермский государственный университет

ket-eref@yandex.ru

ПРОМЕЖУТОЧНОЕ ПРЕДСТАВЛЕНИЕ ПРАВИЛЬНОЙ РАСКРАСКИ ГРАФА В ГЕНЕТИЧЕСКОМ АЛГОРИТМЕ

Введение

Пусть дан граф G, описываемый двумя множествами: U – множество вершин и V – множество ребер, $U = \{u_1, u_2, \ldots, u_n\}$, $V = \{(u_i, u_j)\}_{i,j\in[1, n],\ i\neq j}$. *Раскраска графа* – это функция f, преобразующая множество вершин U в отрезок натурального ряда $\{1, 2, 3, \ldots, K\}$:

$$f: U \to \{1, 2, 3, \ldots, K\}.$$

Если при этом выполняется условие, что для любых $(u_i, u_j) \in V$, $f(u_i) \neq f(u_j)$, то раскраска называется *правильной*, а граф G – *K-раскрашиваемым* [1]. Если K – минимальное число, при котором граф является K-раскрашиваемым, то K называется хроматическим числом графа.

Задача нахождения хроматического числа графа является NP-полной – для ее точного решения необходимо совершить полный перебор всех возможных вариантов раскрасок. Алгоритмы полного перебора для больших графов (или любых других больших систем) требуют значительных вычислительных ресурсов. Верхней оценкой сложности полного перебора для n-вершинного графа является величина $O(n!)$. Поэтому для получения ответа за приемлемое время используются стохастические и эвристические методы. Одним из примеров эвристических методов могут служить генетические алгоритмы.

Генетические алгоритмы, построенные для одной задачи, различаются, прежде всего, способами кодирования. Из способа кодирования, т.е. вида генотипа, проистекают виды скрещивания и мутации. Естественным является желание использовать тот из алгоритмов, написанных для решения одной задачи, который дает более точные результаты за меньшее или, по крайней мере, приемлемое время.

В предыдущих работах были рассмотрены три генетических алгоритма для решения задачи коммивояжера, [2]. Первый – с классическим кодированием гамильтоновых циклов, второй – основанный на промежуточном представлении, и третий – комбинированный из первых двух. В результате проведенных исследований выяснилось, что комбинированный генетический алгоритм значительно превосходит классический и модифицированный алгоритмы по вероятности получения точного решения.

Было выдвинуто предположение, что комбинация различных генетических алгоритмов для решения одной задачи увеличивает вероятность получения точного решения. Для проверки этого предположения было решено применить метод комбинирования ГА для другой NP-полной задачи – задачи нахождения хроматического числа графа.

Для комбинированного генетического алгоритма характерный три задачи:

1) переход от классического ГА к модифицированному;
2) переход от модифицированного ГА к классическому;
3) выбор критериев смены генетических алгоритмов.

При решении первой задачи возникла проблема, заключающаяся в том, что обратное преобразование (т.е. переход от классической особи к модифицированной) не может быть однозначно осуществлен при применении того «жадного» алгоритма, который обычно используется в таком случае. Таким образом, было необходимо заменить этот «жадный» алгоритм на такой, который обладал бы двумя свойствами:

- не ухудшал полученное решение (т.е. при его применении найденное хроматическое число не было бы больше, чем найденное при применении обычного «жадного» алгоритма);
- имел обратное преобразование.

Обычный «жадный» алгоритм

Пусть дан граф G размерности n и перестановка $s = \begin{pmatrix} 1 & 2 & \cdots & n \\ p_1 & p_2 & \cdots & p_n \end{pmatrix}$ из n элементов. Представим ее в виде $p_1, p_2, \ldots, p_n$. Исходя из определения перестановки: $p_i \in [1; n]$ $\forall$ i и $p_i \neq p_j$, $\forall$ i, j. Таким образом, перестановкой можно представлять порядок обхода графа.

Будем обходить граф в соответствии с перестановкой s. Для каждой новой вершины по порядку проверяем цвета. Красим вершину в

первый же подходящий цвет. Если не в один из наличествующих цветов покрасить вершину нельзя, то добавляем еще один цвет, и красим в него. Блок-схема алгоритма показана на рис. 1.

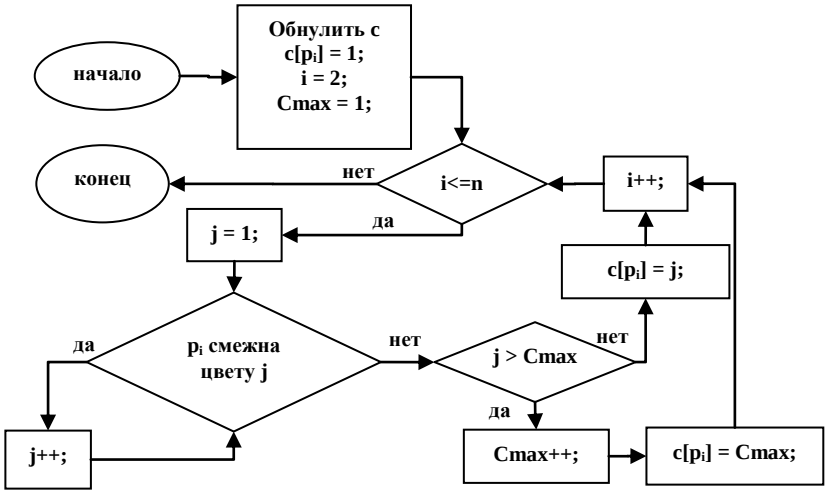


Рис. 1. Блок-схема обычного «жадного алгоритма»

Измененный «жадный» алгоритм

Введем понятие смежности вершины и цвета.

Определение. ◄Вершина p_i называется смежной цвету c_j, если существует такая вершина p_k, которая покрашена в цвет c_j и при этом является смежной вершине p_i.►

Пусть дан граф G размерности n и порядок обхода вершин графа $s = p_1, p_2, \ldots, p_n$. Вершина с номером p_1 красится в первый цвет. Далее для каждой последующей вершины p_i проверяется, нельзя ли ее покрасить в тот же цвет, что и предыдущую вершину. Если это можно сделать, то вершина красится в тот же цвет, что и предыдущая. Иначе ищется минимальный цвет, несмежный вершине. Если такой цвет найден, то вершина красится в этот цвет, иначе вершина красится в новый цвет.

Блок схема измененного «жадного» алгоритма представлена на рис. 2. В результате работы алгоритма будет получена раскраска графа в массиве c и номер максимального цвета в переменной *Cmax*.

Преобразование, проводимое по измененному «жадному» алгоритму, называется прямым преобразованием.

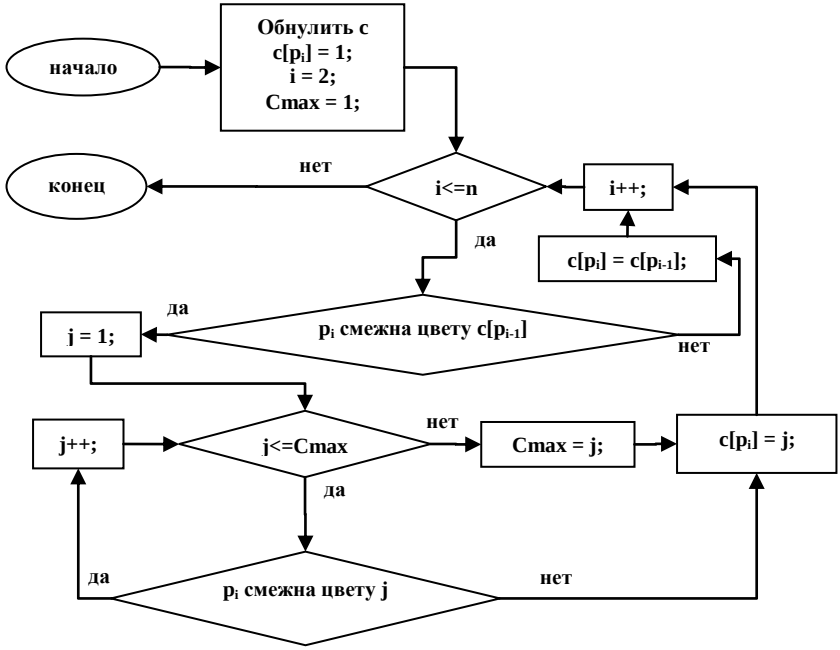


Рис. 2. Блок-схема измененного «жадного алгоритма»

Алгоритм обратного преобразования

Опишем алгоритм обратного преобразования. Пусть дан граф G размерности n и раскраска c. Для получения такого гамильтонова цикла s, что в результате работы «жадного» алгоритма была получена эта же раскраска или раскраска, использующая меньшее количество цветов, необходимо выполнить следующее: для первого цвета выбираем все вершины, которые покрашены им и заносим их последовательно в s. Далее для каждого последующего цвета выбираем все вершины, покрашенные в этот цвет. Среди выбранных вершин ищем такую вершину, что она смежна всем предыдущим цветам. Если такая вершина существует, то вносим в s найденную вершину, а затем последовательно все остальные вершины данного цвета. Если такой вершины нет, то это означает, что множество вершин данного цвета можно разбить на непересекающиеся подмножества, каждое из которых можно приписать к цвету с меньшим номером, чем данный. Сформулируем это утверждение как Теорему 1 (доказательство приведем ниже).

Таким образом, разбиваем множество вершин текущего цвета на подмножества, которые можно перекрасить в цвета с меньшими номерами. Далее для каждого подмножества находим последнюю в *s* вершину цвета, в который это подмножество было перекрашено, обозначим положение этой вершины *index*. Вставляем номера вершин, попавших в подмножество *s*, начиная с позиции *index* + 1, смещая все последующие вершины на необходимое число позиций. После перекраски вершин текущего цвета, все цвета с большим номером переобозначаем. Если цвет имел номер *i*, то после переобозначения он должен иметь номер *i*–1.

После того, как из раскраски будет получен гамильтонов цикл, к нему применяется классический алгоритм кодирования для гамильтоновых циклов, описанный, например, в работе [2].

Блок-схема обратного преобразования для «жадного» алгоритма представлена на рис. 3.

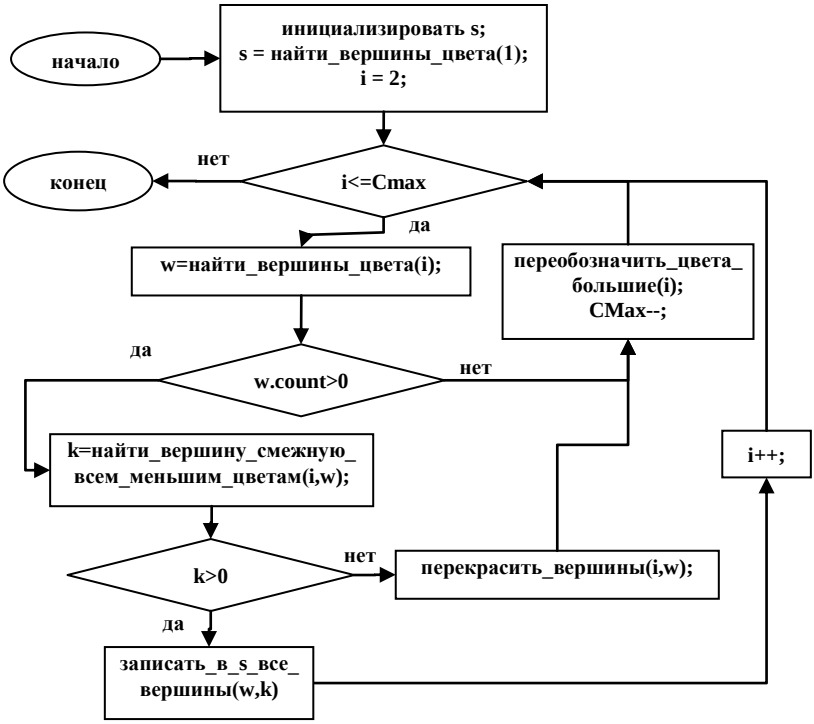


Рис. 3. Блок-схема обратного преобразования

В качестве *s* будем использовать список (list<int>).

Покажем, что полученный таким образом гамильтонов цикл будет давать в результате работы жадного алгоритма ту же раскраску, что была изначально, либо лучшую.

◄Сначала предположим, что в ходе работы алгоритма обратного преобразования для каждого множества вершин, покрашенных определенным цветом, находилась такая вершина, что она была смежна всем меньшим цветам. В этом случае по описанию «жадного» алгоритма будем иметь следующее: не нарушая общности, можно предположить, что если для раскраски использовались *Cmax* цветов, то в полученном гамильтоновом цикле существует *Cmax*–1 точек, в которых один цвет менялся другим при построении этого цикла. Эти точки проходят между двумя номерами вершин. Пусть номера позиций, на которых стоят номера вершин, перед которыми следует смена цвета, обозначены $k_1, k_2, \ldots, k_{Cmax-1}$.

Тогда по «жадному» алгоритму все вершины, номера которых стоят до позиции k_1, будут покрашены в первый цвет. По алгоритму обратного преобразования «жадного» алгоритма на позиции k_1 стоит номер вершины, которая смежна первому цвету. Поэтому по «жадному» алгоритму эта вершина и все вершины, номера которых стоят до позиции k_2, будет покрашена во второй цвет, а т.к. вершина, номер которой стоит на позиции k_1, была покрашена во второй цвет, то цвета до преобразования и после обратного преобразования будут совпадать. Аналогично, на позиции k_i стоит номер вершины, которая смежна первому, второму, … *i*-му цвету и рассуждения для нее повторяются.

Получаем, что все вершины получили те же цвета, в которые они были покрашены до преобразования.

Теперь рассмотрим случай, когда одно или несколько множеств вершин, одинаково покрашенных, были перекрашены в другие цвета. Разберем более подробно случай, когда такое множество одно. Случай, когда таких множеств несколько, разбирается аналогично.

Предположим, что было перекрашено множество вершин, покрашенных в *p*-й цвет. Из доказательства первого случая следует, что все вершины, покрашенные в цвета, номера которых больше *p*, будут покрашены в те же цвета, что и до преобразования, с точностью до переобозначения цветов (чтобы получить цвет, в который вершины были покрашены до преобразования, достаточно прибавить к текущему цвету единицу). Рассмотрим цвета, меньшие *p*. По алгоритмам прямого и обратного преобразований, а также исходя из первого случая, все вершины, которые были раньше покрашены в соответствующие цвета, будут покрашены в них же. Вершины, которые до преобразования бы-

ли покрашены в цвет p, будут покрашены в цвета с меньшими номерами. При этом они перекрашены таким образом, что не будут нарушать порядок смены цветов.

Таким образом, если в преобразовании было перекрашено одно множество цветов, то количество цветов, необходимых для покраски, уменьшается на 1, а все вершины, которые были покрашены в другие цвета, сохранят свою раскраску, т.е. раскраска графа не ухудшится – количество цветов будет меньше или равно изначальному. ►

Теорема 1.

◄Пусть существуют непересекающиеся множества номеров вершин графа G W_1, W_2, …, W_q, где W_i – множество номеров вершин графа, покрашенных в цвет i, K_i – мощность множества W_i. Пусть также выполняется условие, что для $\forall i \in [1; K_q]$, $\exists\ h < q$, $h > 1$, что для $\forall j \in [1; K_h]$, вершина, номер которой, находится на i-ой позиции во множестве W_q, не смежна вершине, номер которой находится на j-ой позиции во множестве W_h.

Разобьем множество W_q на подмножества W_{q1} и $\overline{W_{q1}}$, так, чтобы в W_{q1} вошли номера тех вершин, которые смежны цвету 1, а в $\overline{W_{q1}}$ – несмежных цвету 1. Т.к. вершины, номера которых попали в $\overline{W_{q1}}$, несмежны цвету 1, то их можно перекрасить в первый цвет. Далее разобьем множество W_{q1} на подмножества W_{q2} и $\overline{W_{q2}}$ и перекрасим вершины, номера которых попали в множество $\overline{W_{q2}}$ во второй цвет. Действуя аналогично, разобьем множество W_{qi-1} на подмножества W_{qi} и $\overline{W_{qi}}$ и вершины, номера которых попали в $\overline{W_{qi}}$, перекрасим в цвет i. На последнем шаге останется множество W_{qq-2}, которое будет разбито на подмножества W_{qq-1} и $\overline{W_{qq-1}}$. Вершины, номера которых попали в множество $\overline{W_{qq-1}}$, перекрасим в цвет q–1. Покажем, что множество W_{qq-1} пусто.

Предположим, что это не так. Тогда существует такая вершина, номер которой в результате всех вышеперечисленных действий находится в множестве W_{qq-1}, т.е. сама вершина покрашена в цвет q и смежна цвету q–1. Но т.к. ее номер принадлежит множеству W_{qq-1}, то

он принадлежит и множеству $W_{q_{q-2}}$, т.е. вершина покрашена в цвет q и смежна цвету q–2. Продолжая рассуждения аналогичным образом, получаем, что существует такая вершина, покрашенная в цвет q, что она смежна всем цветам с меньшими номерами, а это противоречит условию теоремы. Следовательно, такой вершины не существует, т.е. множество $W_{q_{q-1}}$ – пусто. А это значит, что все вершины, номера которых принадлежат множеству W_q можно перекрасить в цвета с меньшими номерами. ▶

Заключение

В данной работе приведено описание «жадного» алгоритма, по которому строится обход вершин графа с последующей их покраской. К данному алгоритму приведен алгоритм обратного преобразования, который переводит раскраску графа в обход вершин, который в случае применения к нему прямого алгоритма даст ту же (или лучшую) раскраску. Приведено также доказательство того, что алгоритмы работают вышеописанным образом.

В работе [3] были приведены результаты тестирования комбинированного генетического алгоритма, использующего описанные алгоритмы прямого и обратного преобразования.

Библиографический список

1. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
2. *Данилова Е. Ю., Городилов А. Ю.* Сравнение генетических алгоритмов на примере задачи коммивояжера // Вестник ПГУ. Математика Механика Информатика. Выпуск 3 (29). С. 49-53.
3. *Данилова Е. Ю.* Комбинация генетических алгоритмов для решения NP-полных задач на примере задачи нахождения хроматического числа графа.// Современные проблемы математики и ее прикладные аспекты. Сборник статей (по материалам научно-практической конференции молодых ученых. Пермь, 12 марта 2010 г.). Пермь: Перм. ун-т, 2010. – С. 36-41.

А.П. Остроушко, Н.В. Белоус

Харьковский национальный университет радиоэлектроники

osa@kture.kharkov.ua, belous@kture.kharkov.ua

БЫСТРЫЙ АЛГОРИТМ ВИЗУАЛИЗАЦИИ ОБЛАЧНОГО СЛОЯ ДЛЯ МЕТОДА ОБРАТНОГО ТРАССИРОВАНИЯ

Введение

Одним из требований к современным системам трехмерной компьютерной графики является высокая реалистичность синтезируемых изображений, которая достигается за счет увеличения детализации изображения, формирования статических и динамических теней от источников света, учета прозрачности среды сцены, генерации различных спецэффектов. Кроме того, ключевую роль в качестве формируемых изображений играет метод синтеза. Наиболее перспективным можно назвать метод обратного трассирования лучей, обеспечивающий возможность получения исключительно высокореалистичных изображений. Главным недостатком этого метода является его высокая ресурсоемкость. Именно поэтому разработка эффективных моделей элементов трехмерных сцен и алгоритмов их визуализации для метода обратного трассирования является актуальной задачей.

В статье предложен быстрый алгоритм, позволяющий вести синтез изображения облачного слоя, описанного при помощи модели, предложенной в [1]. Алгоритм использует классификацию проекционных лучей для исключения из обработки лучей, не имеющих пересечения с облаком. Это позволяет значительно сократить объем вычислений и уменьшить время синтеза изображения. Возможность широкого распараллеливания вычислений позволяет использовать предложенный алгоритм в многопроцессорных, распределенных, GRID системах и кластерах.

Особо следует выделить возможность реализации алгоритма для работы на современных архитектурах GPU, благодаря малому объему

требуемой памяти для хранения модели облака и промежуточных результатов вычислений.

Благодаря большой гибкости модели облака и высокой скорости работы алгоритма его визуализации результаты работы могут быть использованы при построении систем визуализации реального времени.

В настоящее время основным подходом при реализации метода обратного трассирования является решение системы уравнений проекционного луча, заданного в параметрической форме, и уравнения поверхности в неявном виде [2]. Классический алгоритм определения точки пересечения проекционных лучей со сферами, составляющими структуру облака, позволяет существенно сократить объем вычислений [3].

Однако, как видно из результатов, приведенных в [1], время синтеза одного кадра по-прежнему достаточно велико и это не дает возможности использовать классический алгоритм для работы в реальном времени. Поэтому требуется разработка быстрого алгоритма для метода обратного трассирования, обеспечивающего поиск параметров пересечения проекционного луча с элементами облака.

Описание алгоритма

Классический алгоритм позволяет вести независимые расчеты для каждого проекционного луча, что дает возможность широкого распараллеливания процесса вычислений [4]. Повысить эффективность и значительно сократить время синтеза можно за счет использования классификации лучей [5].

Предлагается модифицировать алгоритм, приведенный в [5], для нахождения параметров пересечения проекционных лучей со сферами облака.

1. На первом шаге алгоритма осуществляется переход из системы координат (с/к) облака в с/к наблюдателя [3]. Для выполнения первого шага достаточно осуществить пересчет координат центров сфер:

$$p_i = Mp_i^0,$$

где p_i^0 – центр i-й сферы в с/к облака; p_i – центр i-й сферы в с/к наблюдателя; M – матрица перехода из с/к облака в с/к наблюдателя.

2. На следующем шаге определяются параметры пересечения x-го β-среза с базовой сферой облака (сфера 0-го уровня). Угол между соседними β-срезами $\Delta\beta_x$ и α-срезами $\Delta\alpha_y$ не должен превышать мак-

симальную угловую погрешность системы [6]. Исходя из этого вычисляется разрешение формируемого изображения $X \times Y$.

Выполнение первого шага приводит к тому, что β-срез всегда представляет собой плоскость, проходящую через ось y с/к наблюдателя и x-й столбец пикселей формируемого изображения (рис. 1).

Таким образом, уравнение β-среза имеет вид

$$x\cos(\beta_x) - z\sin(\beta_x) = 0, \qquad (1)$$

где $x \in \{1,\ldots,X\}$ – номер столбца в формируемом изображении; β_x – угол между плоскостью x-го β-среза и осью z.

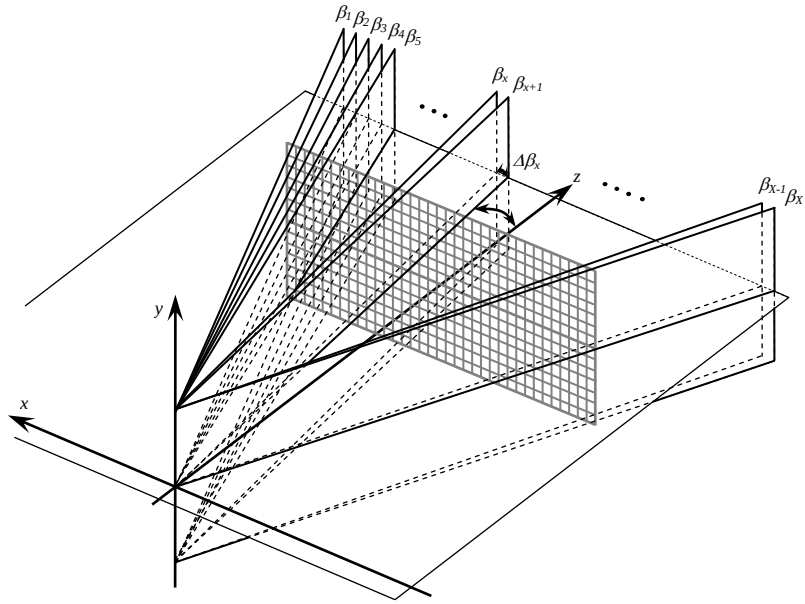


Рис. 1. Формирование β-срезов для классификации лучей

Для наличия пересечения между плоскостью и сферой расстояние d_i от плоскости β-среза до центра i-й сферы не должно превышать радиуса r_i этой сферы:

$$d_i \le r_i. \qquad (2)$$

Расстояние от точки $p(x_0, y_0, z_0)$ до плоскости $Ax + By + Cz + D = 0$ определяется соотношением

$$d = \frac{|Ax_0 + By_0 + Cz_0 + D|}{\sqrt{A^2 + B^2 + C^2}}, \qquad (3)$$

Из уравнения (1) видно, что знаменатель в соотношении (3) для любого β-среза всегда равен 1. Поэтому для проверки неравенства (2) достаточно в уравнение (1) подставить координаты центра сферы, полученные на первом шаге алгоритма. Этот процесс может выполняться независимо для каждого β-среза.

Если пересечения сферы с β-срезом нет, то алгоритм для этого β-среза завершается. Если пересечение есть, то создается два списка [7] для хранения номеров сфер и в первый из них заносится 0 – номер базовой сферы.

3. Для каждого β-среза запускается итерационный процесс, где на каждом уровне итерации формируется список сфер, пересекающих данный β-срез. Из первого списка извлекается номер сферы n^k, с которой было пересечение, и вычисляются номера сфер следующего уровня n_j^{k+1}:

$$n_j^{k+1} = m\, n^k + j; \quad j \in \{1, \ldots, m\},$$

где $m = 5$ – число сфер, на которое разбивается сфера предыдущего уровня детализации; k – номер итерации.

Для каждой n_j^{k+1} сферы осуществляется проверка условия (2) и если оно выполняется, то номер этой сферы заносится во второй список. Так продолжается до тех пор, пока не будет исчерпан первый список. После этого списки меняются местами, номер итерации увеличивается на 1. Достижение уровня $k \le K$ с требуемым уровнем детализации приводит к переходу на следующий шаг алгоритма.

4. Для каждого β-среза, у которого список, полученный на предыдущем шаге, не пустой, осуществляется расчет диапазона α-срезов (рис. 2). Для этого определяются параметры пересечения плоскости и сферы: координаты центра окружности и значение радиуса.

Центр окружности определяется путем поворота центра сферы на угол $-\beta_x$ вокруг оси *y*:

$$x_i^{\beta_x} = z_i\, cos(-\beta_x) - x_i\, sin(-\beta_x) = z_i\, cos(\beta_x) + x_i\, sin(\beta_x);$$
$$y_i^{\beta_x} = y_i,$$

здесь x_i, y_i, z_i – координаты центра *i*-й сферы в с/к наблюдателя; $x_i^{\beta_x}, y_i^{\beta_x}$ – координаты центра окружности на плоскости β-среза.

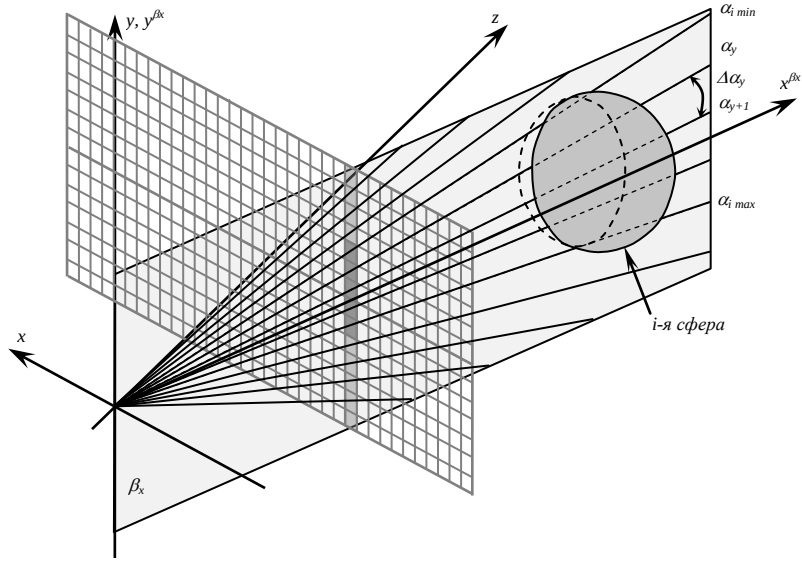


Рис. 2. Классификация по α-срезам

Для вычисления значения радиуса окружности на плоскости β-среза используем параметры, полученные при проверке условия (2):

$$r_i^{\beta_x} = \sqrt{r_i^2 - d_i^2}\ .$$

Если выполняется условие

$$x_i^{\beta_x} - r_i^{\beta_x} < 0,$$

то это означает, что сфера находится позади наблюдателя и должна быть исключена из дальнейшей обработки.

В противном случае определяется диапазон изменения α-срезов, пересекающих окружность:

$$\alpha_{i_{min}} = \frac{x_i^{\beta_x} y_i^{\beta_x} + r_i^{\beta_x}\sqrt{A}}{\left(x_i^{\beta_x}\right)^2 - \left(r_i^{\beta_x}\right)^2};$$

$$\alpha_{i_{max}} = \frac{x_i^{\beta_x} y_i^{\beta_x} - r_i^{\beta_x}\sqrt{A}}{\left(x_i^{\beta_x}\right)^2 - \left(r_i^{\beta_x}\right)^2},$$

где $A = \left(x_i^{\beta_x}\right)^2 + \left(y_i^{\beta_x}\right)^2 - \left(r_i^{\beta_x}\right)^2$.

Если $A < 0$, то это означает, что наблюдатель находится внутри сферы, и диапазон α-срезов в этом случае находится в пределах от α_1 до α_Y, то есть необходимо обрабатывать все лучи для данного β-среза. После этого определяем рабочий диапазон α-срезов как результат пересечения двух диапазонов $\{\alpha_{i_{min}}, \alpha_{i_{max}}\}$ и $\{\alpha_1, \alpha_Y\}$:

$$\{\alpha_{i_{min}}', \alpha_{i_{max}}'\} = \{\alpha_{i_{min}}, \alpha_{i_{max}}\} \cap \{\alpha_1, \alpha_Y\}. \qquad (4)$$

5. Для каждого проекционного луча создается список, куда будут заноситься параметры пересечений луча со сферами облака: значение расстояния от наблюдателя до точек пересечения l_1 и l_2 (координаты *x*, *y* и *z* однозначно восстанавливаются по параметрам проекционного луча и расстоянию *l*), прозрачность и направление нормали. Расстояние и прозрачность необходимы для учета взаимного положения сфер облака и других объектов сцены при вычислении цвета пикселя.

6. Для каждого проекционного луча, попадающего в рабочий диапазон (4), осуществляется расчет точек пересечения с окружностью:

$$x_i^{1,2} = \frac{x_i^{\beta_x} + B y_i^{\beta_x} \mp \sqrt{(x_i^{\beta_x} + B y_i^{\beta_x})^2 - A(1 + B^2)}}{1 + B^2}, \qquad (5)$$

где $B = tan(\alpha_y)$.

Пара координат точек пресечения, являющаяся результатом вычислений соотношения (5), заносятся в отдельный выходной список, формируемый для каждого проекционного луча.

Следует отметить, что значения B и $1 + B^2$ могут быть заранее вычислены и занесены в таблицу, поскольку зависят только от углового разрешения графической системы.

7. Для каждого списка, полученного на предыдущем шаге, выполняется сортировка элементов по первой точке.

8. Обработка выходного списка заключается в определении прозрачности каждой сферы и начинается со сферы, находящейся ближе всех к наблюдателю.

Прозрачность среды внутри облака определяется метеорологической дальностью видимости (МДВ) *Sм*, которая также является входным параметром модели. МДВ определяет дальность видимости абсолютно черного объекта и однозначно характеризует ослабление средой излучения.

Для практических расчетов видимости используют коэффициент пропускания среды *T*, который для оптически однородного слоя среды толщиной, равной единице длины, носит название удельной прозрачности *t*. Зная удельную прозрачность и толщину слоя среды *l*, можно определить коэффициент пропускания [8]:

$$T = t^l . \qquad (6)$$

Удельная прозрачность среды и МДВ связаны соотношением

$$S_м = ln\,0{,}02 / ln\,t .$$

Таким образом, зная МДВ и рассчитав длину пути проекционного луча внутри сферы легко получить коэффициент пропускания.

Расстояния l_1 и l_2 вычисляются на основании данных, полученных в (5), по следующим соотношениям

$$l_i^1 = x_i^1\sqrt{1+B^2};\;\; l_i^2 = x_i^2\sqrt{1+B^2} .$$

Определяется длина участка, на котором проекционный луч проходит внутри сферы

$$l = l_i^2 - l_i^1$$

и осуществляется расчет базового коэффициента пропускания T_i по соотношению (6).

Далее определяется значение функции-модификатора прозрачности облака [1]:

$$f(d_i, r_i) = 1 - \frac{d_i}{r_i} .$$

Осуществляется расчет скорректированного коэффициента пропускания T_i' с учетом функции-модификатора по соотношению:

$$T_i' = T_i f(d_i, r_i) + 1 - f(d_i, r_i).$$

9. Для вычисления цвета сферы облака необходимо определить направление нормали. Для этого потребуется восстановить координаты *x, y* и *z* первой точки пересечения по параметрам проекционного луча:

$$\begin{aligned} x_i^1 &= l_i^1 \cos(\alpha_y)\sin(\beta_x); \\ y_i^1 &= l_i^1 \sin(\alpha_y); \\ z_i^1 &= l_i^1 \cos(\alpha_y)\cos(\beta_x). \end{aligned} \qquad (7)$$

На основании данных, полученных по соотношениям (9) рассчитывается нормаль в первой точке пересечения с i-ой сферой:

$$n_{x_i} = x_i^1 - x_i;\ n_{y_i} = y_i^1 - y_i;\ n_{z_i} = z_i^1 - z_i\,.$$

Затем направление вектора нормали модифицируется в соответствии с соотношением

$$\vec{n}' = f(u,v)\vec{n}\,, \qquad (8)$$

где $f(u,v)$ – параметрическая функция изменения нормали [1].

Дальнейшие действия по вычислению цвета пикселя, учитывающие пространственное положение облака и других объектов сцены относительно источников света и друг друга, могут быть выполнены по стандартным алгоритмам [3, 4].

10. Результирующий цвет определяется соотношением

$$C = C_1\left(1 - T_1'\right) + \sum_{l=2}^{n}\left(C_l(1 - T_l')\prod_{k=1}^{l} T_k'\right),$$

здесь C – цвет сферы, полученный на шаге 9; n – количество сфер на проекционном луче.

Следует отметить, что при выполнении последнего шага алгоритма можно сократить объем вычислений, если при выполнении каждого очередного умножения проверять результирующую прозрачность. При выполнении условия

$$T \le 0.2 \qquad (9)$$

вычисления могут быть остановлены и остальные точка списка исключены из обработки, поскольку в соответствии с законом Вебера [9, 10] глаз перестает различать дальнейшее изменение прозрачности облака.

Практические результаты

Использование предложенного алгоритма позволило ускорить процесс вычислений приблизительно в четыре раза по сравнению с классическим алгоритмом, использующим параметрическое описание проекционных лучей. Результаты расчета на персональном компьютере с процессором Intel CoreDuo E6600 и 4 ГБ ОЗУ изображения облака с

разрешением 800x600 пикселей приведены в табл. 1. Расчет велся с использованием одного процессорного ядра.

Таблица 1. Сравнение производительности алгоритмов

Количество сфер на последнем уровне	**Время счета, мс**	
	Классический алгоритм	**Алгоритм с классификацией лучей**
15625	4750	1171
390625	9203	2422
9765625	22116	8844

Заключение

Предложенный алгоритм визуализации использует разбиение пространства по β-срезам и α-срезам, что позволяет выполнить классификацию лучей по наличию пересечений со сферами и исключить из обработки проекционные лучи не пересекающие облако. Следует отметить, что использование технологий AMD Stream или NVIDIA CUDA позволит существенно улучшить производительность, поскольку существует возможность широкого распараллеливания процесса вычислений как по β-срезам так и по α-срезам при вычислении параметров пересечения проекционного луча со сферами облака и даст возможность осуществлять обработку в реальном времени.

Библиографический список

1. *Ostroushko A., Bilous N., Bugriy A., Chagovets Ya.* Mathematical Model of the Cloud for Ray Tracing // International Journal "Information Theories and Applications", Vol. 17, Number 1, 2010
2. *Никулин Е.А.* Компьютерная геометрия и алгоритмы машинной графики. – СПб:БХВ-Петербург, 2003. – 560 с.
3. *Fransis S. Hill.* Computer Graphics Using OpenGL (c) Prentice Hall 2001.
4. *Foley J.D., van Dam A., Feiner S.K., Hughes J.F.* Computer Graphics (principles and practice) by Addison-Wesley Publishing Company, Inc. 1996. – 1175 p.
5. *Гусятин В.М.* Алгоритмы сканирования сцены в системах визуализации // Харьков, ХТУРЭ, Радиоэлектроника и информатика. – 2000. – №1.– С. 46-49.

6. *Гусятин В.М., Бугрий А.Н.* Расчет угловой погрешности спецпроцессора в системе визуализации // Автоматизированные системы управления и приборы автоматики: Сб. научн. трудов. Выпуск 112. – Харьков: ХТУРЭ, 2000. – С.4-10.
7. *Bjarne Stroustrup*. The C++ Programming Language— Addison-Wesley Pub Co; 3rd edition, 2000. – 990 p.
8. *McCartney E.J.* Optics of the Atmosphere Scattering by Molecules andParticles (Wiley, New York, 1976), 176–261
9. Атмосфера: Справочник. – Л.: Гидрометеоиздат, 1991. – 512 С.
10. *Дрофа А.С., Кацев И.Л.* Некоторые вопросы видения через облака и туманы. // Метеорология и гидрология, 1981. − № 1. − С. 101-109.

СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

И.А. Белоус, Н.В. Белоус, Т.В. Горбач, И.Ю. Шубин

Украина, Харьковский национальный университет радиоэлектроники

shubin@kture.kharkov.ua

РЕАЛИЗАЦИЯ МЕТОДОВ АДАПТАЦИИ В ГИПЕРМЕДИЙНЫХ СИСТЕМАХ ОБУЧЕНИЯ

Введение

Интернет получил широкое распространение в области электронного дистанционного обучения. В данной области разработчики систем дистанционного обучения хотят направлять пользователей через гиперпространство согласно их предпочтениям и уровню знаний. Достичь этой цели можно с помощью технологий адаптивной гипермедиа, которые и рассматриваются в статье.

На данный момент разработано большое количество дистанционных обучающих систем. Системы управления обучением в своем традиционном виде не в состоянии поддерживать все новые возможности Интернет-технологий и использовать потенциал всех средств, взаимодействия внутри сетей. В этой области создаются все более динамичные платформы, которые приходят на смену традиционным. При активном электронном обучении используется широкий спектр Интернет-технологий, что позволяет внедрять недоступные для традиционных видов обучения методики. Эта проблема частично решается разработкой и внедрением адаптивных обучающих систем. Но в реальности создано не так уж много систем подобного класса, которые действительно отвечают требованиям адаптивности систем [1, 2].

Адаптивность играет большую роль при разработке систем дистанционного обучения, потому что в процессе дистанционного обучения участвует много различных студентов, каждый из которых имеет свои интересы и знания. Однопользовательские обучающие приложения разрабатываются только для определенного класса пользователей с одним складом ума, что может не подойти другим. Под адаптивностью имеется в виду различные пути прохождения контента разными обучаемыми в обучающей системе.

Адаптивная гипермедиа – технология создания гипертекстовых и гипермедийных систем, которая отражает определенные характеристики пользователя в его модели и применяет эту модель для адаптации различных аспектов системы к потребностям пользователя.

Документ гипермедиа - основной компонент WWW, который позволяет пользователям свободно перемещаться (просматривать) информацию в гиперпространстве и состоит из узлов, содержащих информацию и связи, соединяющие их. Гипермедиа – гипертекст, в котором могут быть ссылки на другой тип информации (аудио, видео, анимацию и т.д.) [2].

Системы адаптивной гипермедиа применяют различные виды моделей пользователя для адаптации контента автоматизированных обучающих систем (АОС) и внутренних ссылок под уровень знаний и интересы пользователя. Образование всегда было одной из главных областей применения адаптивной гипермедиа. Большинство адаптивных систем гипермедиа используют методы, которые позволяют разработчикам описывать навигационные правила перемещения обучаемых по контенту АОС [1, 2].

Поэтому целью данной статьи является разработка упрощенного формата навигационных правил для их применения в АОС при дистанционной форме и инструментального средства для проведения экспериментов с ними. Данная задача является актуальной, поскольку существующие АОС, основанные на применении навигационных правил, не предоставляют разработчикам ни средств, ни функций, которые могут уменьшить сроки и трудоемкость составления навигационных правил.

Решение задачи

При проведении данного исследования использован метод сокрытия связей, потому что разрабатываемая АОС должна иметь возможность направить обучаемого по оптимальному пути обучения (рис. 1).

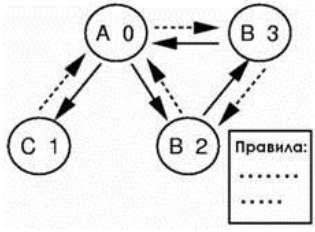


Рис. 1. Гипермедиа модель

Модели обучаемых АОС можно разделить на три вида: оверлейные, стереотипные и модели, которые используют ключевые слова. Оверлейная модель основана на структурной модели предметной области (контента АОС), которая представлена как семантическая сеть понятий. Стереотипная модель назначает пользователю один из нескольких возможных стереотипов для определения соответствующей ему стратегии обучения. Модель обучаемого, использующая ключевые слова, представляет собой вектор или матрицу, элементы которой характеризуют степень интереса в ключевом слове или теме. Системы адаптивной гипермедиа, построенные на основе данной модели позволяют хранить историю взаимодействия обучаемого с системой и относительно просто могут быть преобразованы в строки запросов при реализации модели адаптации [3, 4].

Для разработки АОС предлагаемой архитектуры используем комбинированную модель обучаемого, хранящую как долгосрочную, так и краткосрочную информацию о нем. Для того, чтобы модель обучаемого была более простой, упростим формат навигационного правила. В качестве параметра обучаемого предлагается использовать интересы и уровень знаний пользователя. Это позволит проводить моделирование обучаемого в течение продолжительного времени.

Таким образом, предлагаемая модель обучаемого состоит из последовательности классов узлов. В каждом узле находится содержание, которое показывается обучаемым. Обучаемый может двигаться между узлами по связям между ними. В каждом узле обучаемый имеет возможность выбора связи возврата к предыдущему узлу. Каждый узел идентифицируется номером и классом узла. У всех узлов есть навигационные правила.

Каждому узлу сопоставляем символ, характеризующий его класс. Понятие класса необходимо для формализации навигационных правил и логического вывода на них. Класс также является одной из компо-

нент модели обучаемого. Класс узла определяют разработчики АОС, используя следующие критерии:

- показывает ли АОС фрагмент контента, предназначенный только для определенного вида пользователя?
- предлагает ли узел пользователям объяснение, вопрос, тестовое задание или делает что-то еще в дидактическом плане?
- к каким из категорий контента АОС принадлежит узел (в случаях, когда информация узла может принадлежать более чем к одной категории)?

Предлагаемая структура модели обучаемого содержит два компонента: набор параметров обучаемого и историю взаимодействия с АОС.

Параметры модели обучаемого могут принимать различные значения и характеризовать различные свойства субъекта учебного процесса, такие например как уровень знаний обучаемого. Последний может представлять собой число из некоторого наперед заданного диапазона. Разработчик АОС присваивает значение данному набору параметров, которое используется в навигационных правилах. Необходимо отметить, что указанный набор параметров может быть также задан тьютором или автоматически самой АОС. Например, он может быть установлен по результатам ответа обучаемого на анкетный опрос (тест) или при использовании среднестатистических результатов прохождения тестов несколькими группами обучаемых и т.д.

История взаимодействия обучаемого с АОС представляется как последовательность классов узлов, которые он посетил и информацию с которых он просмотрел.

Предлагаемый навигационный метод основан на адаптивной технологии сокрытия связей (см. выше). АОС, используя навигационное правило, решает, какие узлы скрывать и какие узлы могут быть связаны с текущим.

Представляется целесообразным реализовать следующие четыре вида навигационных правил:

- навигационные правила узла;
- общие навигационные правила;
- локальные пользовательские навигационные правила;
- глобальные пользовательские навигационные правила.

Навигационное правило, которое использует набор параметров обучаемого из его модели, является пользовательским навигационным правилом. Навигационное правило может также быть разделено на два типа: правило узла и общее правило. Правило узла определено и применяется только для определенного узла. Общее правило − для того

чтобы описать наиболее часто встречающиеся навигационные пути в гиперпространстве и часто используемые сегментации диапазона параметров обучаемого.

В навигационном правиле разработчик АОС описывает связи, которые должны быть показаны в соответствии с идентификатором узла или в соответствии с классом узла, который является целью связи. Система скрывает все связи, на которые не ссылаются в навигационном правиле.

Форматы четырех видов навигационного правила следующие:

Навигационное правило узла: $Cn...Cih+...+C_mi...+C_mh=Dj,...,D_n$.

Общее навигационное правило: $Cn...Cih+...+C_mi...+C_mh^{=}Ci,...,C_n$.

Локальное пользовательское навигационное правило:

$$ej\#P;\#e_2\colon D_b...,D_n.$$

Глобальное пользовательское навигационное правило:

$$ei\#\mathrm{Pj}\#e_2\colon C^f{}_b...,C^f\ ,$$

где:

C – класс узла;

D – идентификатор узла, информация которого будет показана обучаемому;

h – количество историй, которые будут задействованы;

m – количество образов пути;

n – количество идентификаторов или классов узлов, которые будут показаны обучаемому;

p – параметр обучаемого;

e – предел параметра обучаемого;

– операция, которая представляется одним из следующих трех логических операторов: '<', '<=' или '='.

С левой стороны от символа '=' в первом и втором правилах специфицирован образец истории пути, который представляет собой результат работы обучаемого с контентом АОС. Навигационное правило означает, что система показывает обучаемому связи, идентификатор узла или класс которых указан в его правой части, в том случае, если история пути обучаемого соответствует одному из образцов историй, которые записаны в его левой части. Отличия третьего и четвертого навигационных правил от первого и второго заключаются в том, что слева от символа ':' указан диапазон значений параметра обучаемого. Пользовательское навигационное правило означает, что система показывает связи, идентификатор узла или класс которых указан в его правой части, если параметр обучаемого, указанный в левой части, находится в пределах заданного диапазона.

Если у узла есть несколько навигационных правил, система пока-

зывает все связи, которые подтверждаются любым навигационным правилом. Таким образом, если, по крайней мере, одно правило из нескольких правил одобряет показ определенной связи, система показывает связь независимо от других навигационных правил.

Пример навигации. На рис. 2 показан пример навигации на основе всех описанных выше навигационных правил. Классы определены следующим образом:

A – узлы с вопросом (тестовым заданием);

B – узлы, содержащие информацию для правильного ответа обучаемого;

C – узлы с информацией для неправильного ответа обучаемого;

D – узлы с объяснением для студентов с высоким уровнем знаний;

E – узлы с объяснением для студентов с низким уровнем знаний.

В качестве параметра обучаемого *p1* выступает его уровень знаний по данной дисциплине.

Навигационное правило узла определено для узла № 5. Данное правило применимо только в этом узле. "ACA = 7" в этом правиле означает, что, когда пользователь заходит в узел № 5 и история пути пользователя – "ACA", система показывает связь к узлу №7 и скрывает связь к узлу № 8. Поскольку класс B означает, что обучаемый ответил правильно, а класс C – неправильно, история взаимодействия обучаемого с АОС показывает, что пользователь отвечал на вопрос в узле № 0 неправильно, а в узле № 4 – правильно. "ABA = 8" означает что, если пользователь ответил правильно на вопросы в узлах № 0 и 4, то система показывает только связь к узлу № 8. То есть система изменяет ход учебного процесса согласно текущим результатам учебной деятельности обучаемого.

В данном примере также задано глобальное пользовательское навигационное правило. Это правило может быть применено в любом узле гиперпространства.

В примере на рисунке 2 оно используется в узле № 6. Так, если параметр обучаемого *pi* больше или равен 0, но не превышает 80, система показывает связи к узлам с классом *D* и игнорирует все другие связи. Если параметр *pi* больше 79, но не превышает 100, система показывает связи только к узлам класса *E*. Таким образом, достигается адаптация учебного процесса в АОС к уровню знаний обучаемого.

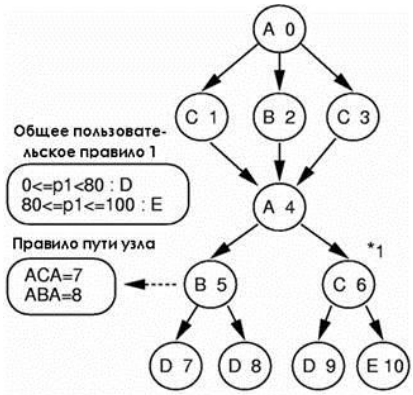


Рис. 2. Примеры навигационных правил

Инструментальное средство исследования навигационных правил предназначено для автоматизации составления навигационных правил разработчиком АОС. Оно является составной частью АОС и помогает разработчику в описании навигационных правил, а также исследует результаты выполнения навигационного правила, прежде чем включить его в узлы. Таким образом, целью создания данного инструментального средства является реализация правильной навигации по контенту АОС с наименьшим количеством ошибок и упрощение составления навигационных правил. Наиболее распространенными навигационными ошибками, которые могут встречаться в контенте любой АОС, являются «тупик» и «петля».

Тупик. Состояние АОС, когда все связи скрыты и обучаемый не может пойти куда-нибудь далее после достижения узла с навигационным правилом. Проблема тупика может быть вызвана плохим навигационным правилом. Появление тупика вынуждает обучаемого прекратить движения по достижению следующих узлов в гиперпространстве. Другими словами, возникает препятствие продвижению обучаемого.

Петля – это состояние АОС, когда обучаемый достигает узла, в котором он уже был. Данная ситуация еще называется «перекручиванием поиска». Использование петли бывает эффективно, например, когда связь необходима для возврата к главной странице. Проблемы с петлями возникают лишь в том случае, когда они являются незапланированными разработчиками АОС. Действительно, поскольку система скрывает динамически связи, которые могли бы вызвать проблемы для навигации обучаемого, это может привести к появлению петель.

Обнаружение тупика. Тупик может быть обусловлен навигационным правилом узла, пользовательским навигационным правилом или другими правилами. Предлагаемый алгоритм обнаружения тупика проверяет возможность появления тупика в узле из-за правил пути или пользовательских правил. Если у узла есть несколько видов навигационных правил, система показывает все связи, которые правило пытается применить. Если инструмент обнаруживает тупик, вызванный одним из видов навигационного правила (например, правило узла) в узле, другой вид правила (например, пользовательское правило) может попытаться показать эти связи в узле. Поэтому, когда алгоритм обнаруживает очевидный тупик в узле, он проверяет, определен ли другой вид правила для него. Если никакое другое правило не определено для узла, значит, обнаружен тупик. Если другой вид правила определен в узле, то существует возможность появления тупика.

Тупик, созданный навигационными правилами, может образоваться в двух случаях: когда образец истории пути, по которому следовал обучаемый, не включен в навигационные правила; когда, ни одна из связей текущего узла не описана в навигационных правилах. Предлагаемый алгоритм не только обнаруживает тупики (возможность появления тупиков), но и определяет образец истории пути, который вызывает тупик.

Алгоритм обнаружения тупиков, основанный на навигационных правилах:

1. Спецификация узла – разработчик АОС определяет узел, который необходимо проверить.

2. Экспертиза визуализируемых связей – система проверяет, существуют ли связи, которые будут показаны согласно образцу истории пути, описанному в навигационном правиле, и определенные в указанном узле гиперпространства. После чего сравниваются узлы, описанные как визуализируемые в навигационных правилах узлов, которые являются непосредственно связанными с текущим.

3. Регистрация так называемого «живого пути» (схемы навигации без тупика) – система распознает образец истории пути, у которого есть связи, которые должны быть показаны. Если обучаемый следует по живому пути к указанному узлу, то гарантировано будут существовать связи для продолжения пути. Каждый живой путь заносится в специальный список, где он ранжируется согласно длине образца истории пути.

4. Система выполняет поиск в глубину, начиная с текущего узла и используя обратные связи, рассматривая таким образом текущий узел как корень перевернутого дерева.

5. Исследование пути – ищем живой путь среди путей, отобранных на предыдущем шаге. Каждый путь, найденный посредством поиска в глубину, проверяется по списку живых путей. Если найден живой путь, возвращаемся на четвертый шаг для продолжения поиска в глубину от верхнего узла. Если найденный путь не является живым, то переходим к шестому шагу.

Обнаружение возможности тупика – если длина текущей глубины поиска достигла максимальной длины путей, зарегистрированных в списке живых путей, то существует возможность появления тупика, когда обучаемый следует по этому пути, и поиск продолжается с седьмого шага. Если максимальная длина не достигнута, возвращаемся на четвертый шаг.

Решение относительно тупика: система проверяет, определено ли правило в любом из узлов на пути. Если никакое навигационное правило не определено ни для одного из этих узлов, то система решает, что тупик появится, когда пользователь будет следовать по этому пути. Если навигационные правила определены, хотя бы для одного узла, то система решает, что есть возможность появления тупика по следованию этого пути пользователем. После этого система возвращается на четвертый шаг.

Тупик, вызванный пользовательским правилом, возникает, когда значение параметров обучаемого выходит за пределы диапазонов, описанных в локальных или глобальных навигационных правилах. Все визуализируемые узлы, описанные в пользовательских правилах, не имеют связи с текущим узлом. Предлагаемый алгоритм не только обнаруживает такие тупики, но также и определяет правило, которое вызывает данный тупик.

Алгоритм обнаружения тупиков, вызванных пользовательскими правилами:

Спецификация узла – разработчик АОС определяет узел, который необходимо проверить.

Исследование визуализируемой связи – система проверяет, существует ли визуализируемая связь для определенного диапазона параметра обучаемого, как описано в пользовательском правиле указанного узла. Данная проверка осуществляется посредством сравнения визуализируемых узлов в пользовательских правилах с узлами, которые непосредственно связаны с текущим. Если такие связи существуют, то данный диапазон считается «живым» (диапазоном, у которого есть визуализируемые связи)

Исследование диапазона параметра – система проверяет, являются ли все диапазоны параметра обучаемого живыми. Если есть диапазон,

который не является живым, следовательно, он и является возможной причиной появления тупика.

Даже если путь определяет петлю, не учитывая технологии сокрытия связей, в действительности она может не являться таковой. Поэтому в определении петлей важную роль играет учет технологии сокрытия связей. Поэтому после определения параметров обучаемого и соответствующих навигационных правил, необходимо проверить, остается ли петля петлей после сокрытия связей. Описанный ниже алгоритм позволяет обнаружить петли, осуществляя поиск в глубину от определенного узла контента АОС. Предлагаемый алгоритм не только обнаруживает петли, но также и получает путь перемещения обучаемого.

Алгоритм обнаружения петли включает следующие шаги:

Спецификация узла и максимальной длины – разработчик АОС определяет узел, от которого начинается поиск в глубину на заданную максимальную длину.

Выполнение навигационных правил – система выполняет навигационные правила текущего узла и скрывает связи. После этого регистрирует показанные связи в списке визуализированных связей, который необходим для выполнения поиска в глубину.

Поиск в глубину - осуществляется один шаг поиска в глубину с начала использованных связей, зарегистрированных в списке, полученном на шаге 2.

Исследование петли – система ищет идентификатор узла, который достигнут при поиске в глубину. Если узел с таким идентификатором уже присутствует в истории пути, то данный путь является петлей.

Определение длины – если текущая глубина поиска меньше значения, указанного в шаге 1, то система возвращается к предыдущему узлу и переходим к шагу 3. Иначе возвращаемся к шагу 2.

Во втором шаге предложенного алгоритма, если навигационное правило определено в узле, где система достигла порогового значения глубины поиска, то она не может выполнить это навигационное правило. В таком случае, система не выполняет навигационное правило, а показывает все связи для обнаружения всевозможных потенциальных петель.

Заключение

В данной статье предложен подход к разработке автоматизированных обучающих систем на основе методов адаптивной гипермедиа. Предложенный подход снижает трудоемкость разработки обучающих

систем за счет автоматического управления ходом учебного процесса посредством навигационных правил. Кроме того, в данной статье предложены алгоритмы, реализация которых приводит к уменьшению количества ошибок в описании навигационных правил, для которых был разработан специальный упрощенный формат, что делает их более простыми для восприятия. Созданная на основе предложенного подхода автоматизированная обучающая система управляет маршрутом движения обучаемого по своему контенту с помощью технологии сокрытия связей для оптимизации учебного процесса. Более того, в данной статье разработано инструментальное средство проверки наличия ошибок в созданных навигационных правилах с целью дальнейшего сокращения их количества. Предложенные подходы и алгоритмы могут быть использованы в любых системах адаптивной гипермедиа, которые управляют перемещением пользователя по гиперпространству.

Библиографический список

1. *Bilous N.; Shubin I.; Vyrodov O*. The conception of interactive training system design // New Solutions in Modern Technologies. Kharkiv, Ukraine, № 79, 2006. С. 68-70.
2. *Bondarenko M.; Bilous N.; Shubin I.* The Ukrainian e-Learning Region // In Proceedings of 10-th International LLinE Conference New Partnerships and Lifelong Learning. Helsinki, Finland, 2008. P. 88-92.
3. *Горбач Т.В., Святкин Я.В., Шубин И.Ю.* Представление и классификация неструктурированных данных в адаптивных обучающих мультимедиа-системах на основе метода компараторной идентификации // Проблемы информационных технологий.-- Херсон: Херсонский национальный технический университет. 2009. № 1 (005). С. 21-25.
4. *Горбач Т.В., Святкин Я.В., Шубин И.Ю.* Методы реализации адаптивной гипермедиа в обучающих системах //Вестник Херсонского национального техничесого университета. Херсон: Херсонский национальный технический университет. 2010. № 2 (38). С. 503-508.

Н.И. Захарова

Пермский государственный университет

Zaharova_N_I@list.ru

ОНТОЛОГИЧЕСКИЙ ИНЖИНИРИНГ В ЗАДАЧАХ РАЗРАБОТКИ БАЗ ЗНАНИЙ УЧЕБНО-МЕТОДИЧЕСКИХ КОМПЛЕКСОВ

Введение

Одной из важных составляющих образовательного процесса являются созданные высококвалифицированными специалистами учебно-методические комплексы (УМК). Однако даже наличие всякого рода документов, содержащих рекомендации по структуре и содержанию УМК, не гарантирует ни соответствие УМК требованиям ГОС по соответствующей учебной дисциплине, ни оценку качества УМК.

Оценка качества УМК – это очень сложный процесс, для которого отсутствуют общепризнанные критерии, что может привести к некоторому несоответствию между предметным содержанием УМК и прописанным в ГОС требованиям к знаниям и умениям по учебной дисциплине.

В то же время правильно и хорошо составленное УМК, без сомнения, очень важно не только при заочной форме обучения, когда студент по большей части должен самостоятельно осваивать материал, а также для дистанционной поддержки традиционной формы обучения.

Так как в настоящее время труд преподавателя автоматизирован в наименьшей степени, и в большей мере это касается процесса составления и оценки УМК. Поэтому основные подходы к оценке качества УМК заключаются ручном труде преподавателя в сопоставлении текста УМК заявленным критериям и ФГОС. Так и ответственность за качество УМК обычно лежит на авторском коллективе, а «сторонняя проверка» зачастую проводится лишь в период аттестации и аккредитации образовательных учреждений. Это, естественно, не гарантирует

качественно составленное УМК. Поэтому разработка программ, даже в незначительной степени автоматизирующих процесс оценивания качества УМК, является актуальной и востребованной на практике, [6].

Учитывая те преимущества, которые дают онтологии при обработке неструктурированной информации, в качестве предлагаемого подхода к процессу составления УМК был выбран онтологический инжиниринг, так как с помощью онтологического подхода мы можем наиболее цельно представить сведения о рассматриваемой проблемной области. Кроме того, материал, представленный в единой форме, вместе с цепочкой взаимосвязей между понятиями, определениями одних терминов посредством других, с учетом типа взаимосвязей, и их синонимами дает не только более полное понимание предметной области, но и лучше воспринимается, тиражируется и воспроизводится. Так, построив онтологию УМК одной дисциплины, мы совершенно логичным образом, без переписывания кода, сможем расширить ее и использовать программное средство для оценки УМК другой дисциплины.

Не менее важно предоставить высококвалифицированному специалисту удобное и доступное средство для создания и визуального представления онтологий, что позволит создавать и редактировать онтологии в простой и наглядной форме. Кроме того, подсистема OntoCtrl предусматривает экспорт описания онтологий представленных в текстовом и графическом виде для ориентации на неподготовленного пользователя, а также имеет дружелюбный Web-интерфейс (Web UI).

В качестве проблемной области была выбрана проблемной области УМК «Информатика» по разделу «Основы защиты информации».

Предлагаемый подход

В самом общем виде наш подход к оценке полноты содержания конкретного УМК заключается в установлении семантической близости анализируемой онтологии текста УМК конкретной дисциплины к онтологической базе знаний общей тематики по этой дисциплине. Для этих целей используются специальные семантические метрики.

Предлагаемый подход базируется на онтологическом инжиниринге, ключевым моментом которого является создание онтологии проблемной области УМК и онтологии общих знаний по дисциплине.

Методы онтологического инжиниринга были выбраны с учетом преимуществ, которые дают онтологии при обработке неструктурированной информации, позволяя наращивать базу знаний (БЗ) проблемной области (ПрО) посредством добавления новых понятии и

связей между ними без необходимости внесения изменений в программный код системы.

Определимся с тем, что такое онтология.

Общепринятым считается определение онтологии, введенное T.Gruber, «онтология – это точная спецификация концептуализации».

Под формальной моделью онтологии *O* будем понимать упорядоченную тройку вида: O = <*T*, *R*, *D*>, где

T – термины, обозначающие объекты и понятия ПрО;

R – отношения между терминами;

D – определения этих понятий и отношений.

Онтология призвана структурировать и упорядочивать знания, а также объединить терминологию данной ПрО, что, несомненно, будет полезно для следующих целей:

- совершенствование организации исследований в данной предметной области;
- усовершенствование методов обучения;
- усовершенствование качества поисковых машин.

База знаний подсистемы OntoCtrl содержит онтологию проблемной области УМК «Информатика» по разделу «Основы защиты информации», которая была построена согласно следующим принципам:

Ясность – онтология должна эффективно передавать смысл введенных терминов, ее определения должны быть объективны, а для их объективизации должен использоваться четко фиксированный формализм, [1].

Согласованность – все определения должны быть логически непротиворечивы, а те утверждения, которые выводимы в онтологии, не должны противоречить ее аксиомам.

Расширяемость – необходимо проектировать онтологию так, чтобы ее словари терминов можно было расширять без ревизии уже существующих понятий.

Минимум влияния кодирования – концептуализация онтологии должна быть специфицирована на уровне представления, а не символьного кодирования.

Минимум онтологических обязательств – онтология должна содержать только наиболее существенные предположения о моделируемой проблемной области, чтобы оставлять свободу расширения и специализации.

Вообще говоря, построение онтологии – это затяжной и итерационный процесс, требующий многократного повторения описанных выше шагов. Это обуславливается, например, наличием скрытых знаний, неявных связей, которые не были очевидными на первом этапе

построения онтологии, выявлением новых, возможно, более абстрактных понятий промежуточного уровня.

Прежде всего, мы определили, что построение онтологии в нашем случае осуществляется по принципу снизу вверх. Этот процесс начинается с определения самых общих понятий предметной области с последующей конкретизацией понятий.

Как уже упоминалось в главе «Принципы построения онтологий» для построения онтологии существует несколько общих шагов:

- Выделение основные понятия и термины проблемной области.
- Построение таксономии понятий.
- Определение взаимосвязей между понятиями.

В дальнейшем предполагается, что онтология будет связана с системой автоматического извлечения знания из текстов на естественном языке, что накопит и модернизирует базовые знания в данной предметной области. Онтология будет играть роль основного справочника, с помощью которого такая система сможет определить, связан ли документ с данной ПрО и касается ли эта категория или концепт документа в целом или его отдельных семантических фрагментов, [2]. Предполагается также, что подсистема будет подключена к большим лингвистическим ресурсам, помогающим накопить дополнительную базу знаний и использовать их для решения проблемы синонимии.

Оценка семантической метрики понятий

Для решения многих задач, связанных с автоматической обработкой текстов на естественном языке, в частности, задачи автоматической проверки соответствия предметного содержания учебно-методических текстов заявленной тематике, возникают серьезные проблемы, связанные с корпусом языка.

Проблема однозначной интерпретации смысла текста на ЕЯ является весьма серьезной проблемой, даже частичное решение которой имеет очень большое значение. Роль ее для текстового поиска заключается в том, что пользователь ищет не тексты, а информацию. И для поиска он задает не кусочки текста (так как этот текст ему в общем случае не известен), а понятия русского языка, характеризующие информацию, которую он хочет найти. К семантическим проблемам относятся проблема семантической близости слов, семантической многозначности и лексико-семантической неоднозначности, которые решаются с помощью семантических метрик. Выбор метрики для нахождения семантической близости слов происходит эмпирически, в зависимости от топологии проблемной области.

В общем случае на выбор метрики влияют:

- наличие перегибов и различные типы перегибов;
- длина пути между понятиями;
- различные типы связей, присутствующие в онтологии;
- максимальная длина пути между понятиями;
- учет локального и глобального контекста.

Очень важно при определении семантической близости понятий учитывать особенность пути между понятиями:

- чем длиннее путь между понятиями, тем слабее семантическая близость;
- наличие перегиба на пути ослабляет семантическую близость;
- разные типы перегибов на пути могут по-разному влиять на семантическую близость;
- перегиб пути на высоком уровне иерархии хуже, чем на более низком уровне.

При этом также необходимо при выборе соответствующей метрики учитывать топологию связей между понятиями. Так при обычной классификации, то есть при наличии в основном связи «*is-a*» гораздо выгоднее брать более простую метрику, такую как, например, метрика Leacock и Chodorov, которая учитывает только таксономию понятий. Если же в онтологии присутствует много типов связей, то также необходимо определиться с весом каждого вида связи. Оценки подбираются эвристическим путем в зависимости от топологии сети.

Для получения наиболее семантически верных результатов необходимо также разграничивать такие вещи как локальный и глобальный контекст понятия. В этом случае возникает вопрос в выборе размеров локального контекста. Обычно в виде локального контекста берут линейную окрестность $N \times N$, где N зависит от глубины онтологии.

Так как в построенной онтологии по дисциплине «Основы защиты информации» при анализе ее топологии было установлено, что в онтологии есть семантические перегибы одного вида «перегиб-сверху», то в дальнейшей работе при выборе метрики придется учитывать веса различных типов связей. Необходимо также принять во внимание, что перегиб пути на высоком уровне иерархии хуже, чем на более низком уровне.

Таким образом, при разработке любого алгоритма и выборе метрики для подсчета семантического расстояния, необходимо первым делом провести анализ онтологии по описанным выше критериям.

Применительно к данной топологии проблемной области на данном этапе работы использовалась метрика, которая применялась для концептов Тезауруса Роже, в которой расстояние соответствует числу ребер кратчайшего пути между концептами. Причем связь «*is-a*» бралась за единицу расстояния, а мера связи «*a-part-of*» высчитывалась в зависимости от количества составляющих целое понятий.

Результаты работы

К настоящему времени БЗ OntoCtrl представляет собой онтологию УМК «Информатика» по разделу «Основы защиты информации» и онтологию общих знаний по дисциплине «Основы защиты информации», которые были построены согласно общим принципам построения онтологий.

Для описания онтологий существуют различные языки и системы, однако, наиболее наглядным, удобным, познавательным, но трудозатратным, является графический подход, позволяющий специалистам непосредственно «рисовать» онтологии, что играет важную роль для понимания информации и значительно увеличивает степень ее осмысления. Для графического представления онтологий в системе OntoCtrl был выбран программный продукт StarUML – один из ведущих программных инструментов моделирования.

В табл. 1 приведена сводная характеристика плюсов и минусов основных графических редакторов.

Подсистема OntoCtrl рассчитана на различные категории пользователей, как на пользователей-новичков, так и на пользователей-профессионалов, поэтому необходимо предоставить высококвалифицированному специалисту удобное и доступное средство для создания и редактирования онтологий.

Подсистема OntoCtrl содержит компонент для экспорта и импорта онтологий, позволяя создавать онтологии как в графическом виде, так и в текстовом формате, представляя онтологию в виде взаимосвязанной тройки: «понятие», «связь», «понятие», используя лишь средства Microsoft Word, что позволит редактировать онтологии в простой и наглядной форме для неподготовленного пользователя (рис.1).

Компонент экспорта и импорта онтологий был отлажен на онтологии общих знаний по дисциплине «Искусственных нейронных сетей», созданной в Microsoft Word.

Таблица 1. Сравнение графических редакторов

Критерий	StarUML	Microsoft Visio
Условия распространения	Бесплатный	Платный
Открытость	Открытый	Закрытый
Вес	22 Мб	221 Мб
Работа с русским языком	Поддерживается	Поддерживается
Поддержка многих форматов	UML/XML/RationalRose/JPG	Большое множество стандартных форматов
Удобство визуализации	+	+
Удобство интерфейса	Стиль Microsoft Visual Studio 2008	Стиль Microsoft Office
Расширяемость	Аддины	–
Кодогенерация	+	–
Кроссплатформенность	Windows	Windows

Рис. 1. Экспорт и импорт в системе OntoCtrl

Онтологии, составляющие БЗ подсистемы, были «нарисованы» в графическом редакторе StarUML, но сам по себе графический редактор лишь предоставляет удобный и наглядный графический интерфейс, ни не отражая при этом никакой семантики связи между понятиями в онтологиях, ни не предоставляя возможности оценить степень соответствия УМК заявленной тематике. Импортировав графическое представление онтологии с помощью компонента экспорта и импорта в OntoCtrl, у пользователя появляется гораздо больше возможностей, представляемых подсистемой.

Подсистема OntoCtrl включает в себя компоненту визуального представления онтологии, позволяющую просматривать, пополнять и редактировать иерархию понятий, а также компоненту нестандартного запроса для организации диалога с системой на некотором, соответствующем БЗ, подмножестве естественного языка (ЕЯ) для автоматизации проверки степени полноты соответствия онтологии УМК «Информатика» по разделу «Основы защиты информации» по отношению к онтологии общих знаний по дисциплине «Основы защиты информации».

В рамках системы OntoCtrl компонента для нестандартного запроса по данному полю знаний дает пользователю возможность осуществлять своеобразный диалог с системой, создавая SQL-запросы на некотором подмножестве естественного языка, а также определять взаимосвязи между понятиями и определять, какое из понятий более общее и из каких частей состоит. Пользователь, работая с компонентом нестандартного запроса, составляет запрос на некотором подмножестве ЕЯ, далее этот запрос переводится системой в SQL-запрос и выполняется приложением, обращаясь к БД. В ответ пользователь получает нужную ему информацию.

С помощью компоненты проверки смыслового содержания, входящей в компоненту нестандартного запроса, проводится анализ соответствия между двумя онтологиями: онтологией УМК «Информатика» по разделу «Основы Защиты Информации» и онтологией общих знаний по разделу «Основы защиты информации». Компонент проверки смыслового содержания, работая с заполненной БД, используя эмпирически подобранную семантическую метрику сопоставляет содержание, развернутость и соответствие между терминами предметной области. Что дает нам представление об полноте представления знаний в УМК.

Созданная подсистема OntoCtrl имеет WebUI, который позволит в дальнейшем подключать информационные ресурсы Интернета для повышения качества распознавания смысла текстов на ЕЯ.

В дальнейшем по мере наращивания БЗ предметного содержания учебных дисциплин, поставлена цель расширить онтологию и разработать общий подход к онтологическому анализу УМК учебных дисциплин с целью автоматизации процессов анализа соответствия текстов УМК заявленным целям и ГОС.

Библиографический список

1. *Гаврилова Т.А., Хорошевский В.Ф.* Базы знаний интеллектуальных систем. СПб.: Питер, 2000.
2. *Попов И.Г.* Описание сложных предметных областей на основе интеграции средств представления знаний. М: Москва, 2001.
3. *Козлов С.А.* Генерация образовательных траекторий на основе динамических профилей обучаемых // Вестник пермского университета. Серия «Математика, механика, информатика». Выпуск 3(29), 2009. С. 139-142.
4. *Крижановский А.А.* Математическое и программное обеспечение построения списков семантически близких слов на основе рейтинга вики-текстов. [Электронный ресурс] [Режим доступа: http://www.dialog-21.ru/dialog2006/materials/html/Krizhanovsky.htm] (дата обращения: 20.12.2009).
5. *Лукашевич Н.В., Чуйко Д.С.* Автоматическое разрешение лексической многозначности на базе тезаурусных знаний // Интернет-математика 2007: сб. работ участников конкурса. Екатеринбург: Изд-во Урал. ун-та, 2007. С. 108-117.
6. *Чуприна С.И.* К вопросу о совершенствовании процесса разработки и оценки качества УМК в инновационном вузе // Матер. Всерос. с международным участием науч. практ. конф. "Высшее профессиональное образование, бизнес, власть: опыт и перспективы взаимодействия в подготовке управленческих кадров, ориентированных на инновации" Пермь, 2009. С. 321-324.

Ю.В. Кольцов, Н.Г. Репкин

Кубанский государственный университет, заведующий кафедрой информационных технологий

dean@fpm.kubsu.ru
Кубанский государственный университет, аспирант кафедры информационных технологий

nikolay_r@bk.ru

НЕЙРОСЕТЕВАЯ МОДЕЛЬ ПРОГНОЗИРОВАНИЯ ВРЕМЕНИ ОЖИДАНИЯ ЗАЯВКИ В СМО

Будем рассматривать задачу прогнозирования времени ожидания входящей заявки в очереди для такого класса систем массового обслуживания, как контактные центры (операторские центры, call-центры) [1].

Call-центры – это организации, занимающиеся профессиональной обработкой телефонных звонков. Работой с клиентами занимаются операторы контактного центра (агенты). Заявка, поступающая в контактный центр, направляется одному из свободных операторов, который далее и обслуживает клиента. Выбор этого оператора производится по заранее определенному алгоритму и называется маршрутизацией. Если на момент поступления звонка свободных операторов нет, то он помещается в очередь, где дожидается освобождения агента. Это время и называется временем ожидания входящей заявки.

Первые профессиональные контактные центры появились в США около 35 лет назад [1, 2].

Внедрение контактного центра способно вывести обслуживание клиентов на совершенно новый уровень. Джон Антон из Purdue University (США) приводит следующие данные о том, как зависит вероят-

ность повторного обращения клиента в компанию от качества продукта и эффективности работы контактного центра:

- Хорошее качество продукта – 78%.
- Среднее качество продукта и неэффективный КЦ – 32%.
- Среднее качество продукта и эффективный КЦ – 89% [1, 2].

Сейчас лидером в использовании технологий Call-центров так же является США. По разным оценкам, 70% – 75% всех взаимодействий между потребителями и бизнесом в этой стране осуществляется посредством Call-центров, в которых уже в 1998 году было занято 3% работающего населения.

Однако и в Европе, согласно исследованию Datamonitor [1, 2], в конце 2003 года насчитывалось 29 тыс. центров обслуживания вызовов, а в 2009 году – около 45 тыс.

Для работы операторского центра необходимо вести строгий учет и контроль по ряду ключевых параметров, таких как: уровень обслуживания, средняя скорость ответа, максимальная задержка с ответом, процент обслуженных вызовов и некоторые другие.

Одним из таких ключевых показателей является время ожидания в очереди входящей заявки. Очевидно, что с минимизацией этого времени повышается качество обслуживания, поскольку клиенты меньше времени проводят в ожидании ответа, и большая их часть все-таки дожидается оператора. Кроме того, точное прогнозирование этого параметра может позволить системе выбрать наилучший маршрут для заявки, а клиенту решить самому, оставаться ли ему на линии и ожидать ответа или перезвонить позже. Чем точнее определяется расчетное время ожидания, тем эффективнее методики обслуживания вызовов.

Основными методами прогнозирования времени ожидания, применяемыми на практике, является расчет на основе хронологических данных, расчет на основе оперативной ситуации и расчет на основе одновременного анализа хронологических и оперативных данных [1, 2].

Методы, основанные на анализе хронологических данных за какой-то интервал времени, оперируют такими показателями, как средняя скорость ответа, заданный уровень обслуживания и т.п. Один из наиболее распространенных методов называется Average Speed of Answer (ASA) [1] и основан на определении средней скорости ответа за какой-либо отрезок времени, чаще всего – за последние полчаса. Его суть в следующем. Предположим, что в операторский центр поступил вызов определенного типа. Вычисляется среднее время ожидания $t0$ для данного вызова за некоторый промежуток времени. Далее этот показатель экстраполируется на вновь прибывший вызов и прогнозирует-

ся, что время ожидания для этого вызова также равно $t0$. Значение $t0$ пересчитывается по истечении времени Δt. Величина Δt является параметром алгоритма ASA и может настраиваться статически или динамически. Такая схема работоспособна лишь в случае равномерной нагрузки, что на практике выполняется достаточно редко. Как только происходит скачкообразное нарастание потока вызовов, любой метод, основанный на анализе не текущей, а уже прошедшей ситуации, начинает давать большую погрешность.

Методы, основанные на анализе производительности в данный момент времени, оперируют такими показателями, как число вызовов в очереди, время, которое провел в очереди самый ранний вызов, и т.п. Метод, построенный на анализе времени ожидания самого раннего вызова (Oldest Call Waiting, OCW) [1], является наиболее популярным. Заключается он в следующем.

Предположим, в операторский центр поступил вызов определенного типа. Вычисляется время ожидания $t0$ самого раннего на данный момент вызова этого типа. Далее этот показатель экстраполируется на вновь прибывший вызов и прогнозируется, что время ожидания для этого вызова так же равно $t0$.

Схема работоспособна опять же в случае равномерной нагрузки. При возникновении пиковых нагрузок метод работает неточно.

Причина этого в том, что OCW-методы основаны на следующем предположении: самый последний в очереди вызов будет ждать обслуживания столько же, сколько и самый первый. Но за время, пока этот последний вызов доберется до начала очереди, может произойти множество событий, например, изменится число работающих операторов, количество вызовов в очереди, время обслуживания вызова и т.д. Поэтому чем длиннее очередь, тем хуже работает OCW.

Сочетание же двух предыдущих подходов в едином методе может дать гораздо более точный результат.

В данной работе предлагается нейросетевая модель прогнозирования времени ожидания, где время нахождения заявки в очереди, прогнозируется несколькими нейронными сетями.

Суть метода заключается в прохождении параметров поступившей заявки в сочетании с данными о текущем состоянии контактного центра через несколько нейронных сетей, соединенных определенным образом.

Поступающей заявке вместе с оперативными данными перед подачей в нейронную сеть на основании некоторых показателей ставится в соответствие одна из заранее определенных групп вызовов. После чего вызов может быть либо повторно классифицирован и отнесен к

одной из подгрупп, либо подан непосредственно в одну из прогнозирующих нейронных сетей. Для каждой из таких подгрупп строится и обучается своя нейронная сеть, что дает возможность построить прогноз максимальной точности для каждой из групп вызовов. Процесс кластеризации вызова может продолжаться до тех пор, пока вызов не будет отнесен к наиболее оптимальной для прогнозирования группе.

Для более качественного прогнозирования предлагается классифицировать вызовы по группам так же при помощи специальных нейронных сетей. Это позволяет существенно упростить как процесс определения числа групп вызовов, так и классификацию этих вызовов в процессе работы системы прогнозирования. На вход эти кластеризующие сети могут получать либо все исходные данные целиком, либо их часть, которая в наибольшей степени влияет на группировку. Определяется эта часть заранее путем применения различных методов анализа обучающих данных [3].

В процессе исследования предложенного подхода была построена нейросетевая структура, состоящая из семи различных персептронов в качестве прогнозирующих сетей и одной сети Кохонена в качестве классификатора. Каждый персептрон содержит от двух до трех скрытых слоев с разным числом нейронов, число которых на одном слое достигает 25. При поступлении заявка классифицируется по одному из параметров. Число групп при первой классификации равно четырем. Трем из них соответствуют прогнозирующие персептроны. Если же вызов относится к четвертой группе, то его повторно классифицирует сеть Кохонена, относя к одному из четырех классов, каждому из которых соответствует персептрон (рис. 1).

Полученные сети строят прогноз на основании оперативных и статистических данных за последние несколько минут работы call-центра. Входными параметрами для этих сетей являются: день недели; номер временного интервала – порядковый номер временного интервала в сутках, начиная с 0.00; среднее время в секундах, после которого абонент, находящийся в очереди, вешал трубку; суммарное количество вызовов, принятых всеми операторами; среднее время разговора оператора с абонентом; суммарное число потерянных вызовов; время максимальной задержки при ответе на звонок, в секундах; процент времени, в течение которого операторы были заняты; процент звонков, принятый операторами; среднее число активных операторов.

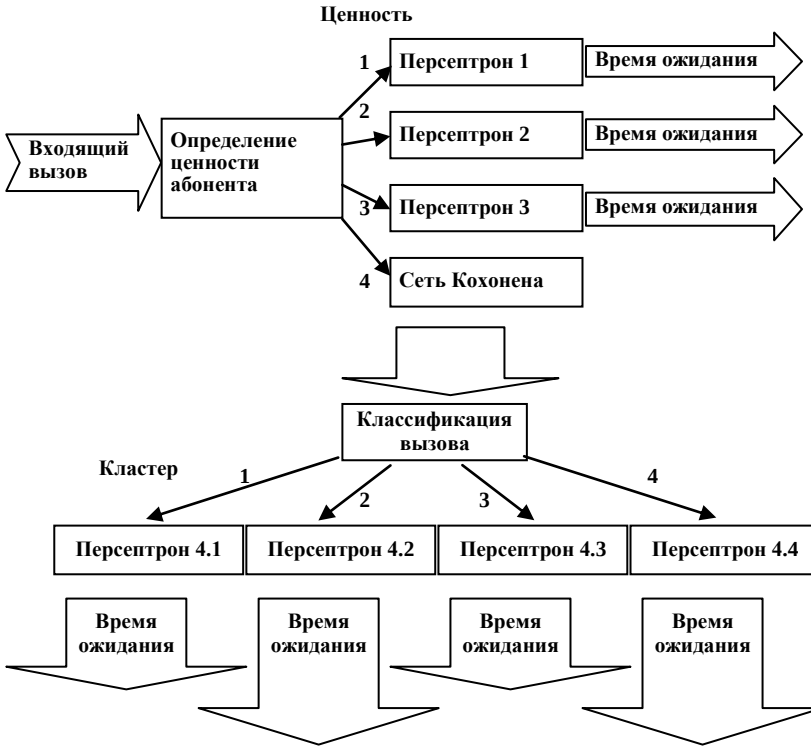


Рис. 1 . Прогнозирующая нейросетевая структура

В результате время ожидания прогнозируется с приемлемой точностью как при равномерной нагрузке на контактный центр, так и в периоды пиковой нагрузки (рис. 2).

Для успешного функционирования описанной схемы должны быть выполнены некоторые условия.

Одно из них состоит в наличии статистических данных для обучения нейронных сетей в течение некоторого периода времени из того контактного центра, время ожидания заявок которого планируется прогнозировать.

Еще одно условие заключается в поддержании прогнозирующих сетей в актуальном состоянии при реорганизации (изменение числа операторов, дополнительные услуги по обслуживанию) операторского центра.

Рис. 2. Графики смоделированного и спрогнозированного времени ожидания

Итак, предложен подход к прогнозированию времени ожидания, основанный на одновременном анализе текущей и статистической информации и использовании нейронных сетей.

Библиографический список

1. *Самолюбова А.Б.* Call-центр на 100%. Москва: Альпина Бизнес Букс, 2004.
2. Call-центр – горячие линии: Электронный ресурс.– Электронные текстовые данные.– Режим доступа: http://www.ihl.ru/, свободный.
3. *Елисеева И.И.* Статистика. М.: Юрайт-Издат, 2010.

Т.В. Осотова

Пермский государственный университет

E-mail: hvostya@gmail.com

ПРИМЕНЕНИЕ СЕМАНТИЧЕСКИХ СЕТЕЙ В СИНТАКСИЧЕСКОМ АНАЛИЗАТОРЕ ПРЕДЛОЖЕНИЙ НА КИТАЙСКОМ ЯЗЫКЕ

Важную роль при обучении китайскому языку играет не только освоение большого числа иероглифов, но и изучение правил синтаксиса этого языка. На кафедре математического обеспечения вычислительных систем Пермского государственного университете разрабатывается интеллектуальная автоматизированная система обучения китайскому языку ОнтоКит 2.0, содержащая подсистему KnowBChin, предназначенную для ускорения процесса освоения синтаксиса китайского языка.

База знаний подсистемы KnowBChin

Знания о синтаксисе китайского языка хранятся в виде расширяемой двухуровневой семантической сети. На первом уровне хранятся знания о принадлежности слов к определённым частям речи, об их представлении в виде иероглифов и т.п.; на втором – знания о том, в роли каких членов предложения могут выступать соответствующие части речи, о структуре различных типов предложений китайского языка и правильном порядке слов [1].

В семантической сети используются как именованные, так и неименованные вершины. Среди именованных выделяются три типа:

1) понятия китайского языка,
2) синтаксические понятия,
3) метапонятия.

Использование метапонятий позволяет определять роль иероглифов или группы иероглифов по «разным уровням» (за счёт объединения нескольких вершин одним метапонятием), а также хранить в структуре самой семантической сети сведения о том, знания какого рода в ней содержатся.

Фрагмент семантической сети представлен на рис. 1.

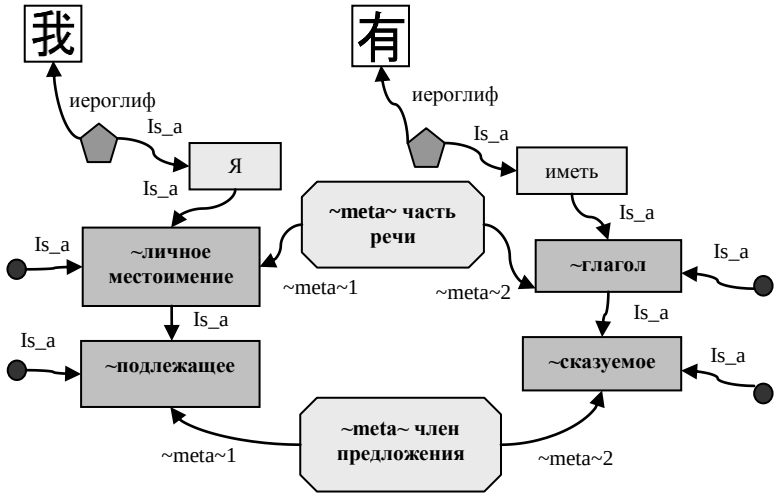


Рис. 1. Фрагмент семантической сети, используемой в подсистеме KnowbChin

Архитектура подсистемы KnowBChin

На внутреннем уровне семантическая сеть организована в виде таблиц. В одной из них хранятся связи между вершинами, представленные в виде троек: (<идентификатор вершины>, <идентификатор дуги>, <идентификатор вершины>).

Для удобства создания и редактирования семантической сети был разработан специальный редактор. Входящая в его состав компонента визуализации позволяет создавать вершины и протягивать между ними дуги определённых типов. Но поскольку пользователю-неспециалисту может показаться сложным представление знаний в виде семантической сети, была добавлена компонента, позволяющая пополнять базу знаний системы на естественном языке: пользователю предлагается

формулировать предложения следующей структуры: <Понятие 1> <Отношение> <Понятие 2>.

В качестве «понятий» могут выступать как уже имеющиеся в базе знаний имена, так и новые (система отслеживает это и в случае необходимости создаёт вершину соответствующего типа). Из «отношений» поддерживаются следующие типы:

1) это есть (*is_a*);

2) имеет экземпляром (обратная к *is_a*);

3) часть от (*a_part_of*);

4) имеет частью (обратная к *a_part_of*).

На настоящий момент подсистема KnowBChin позволяет добавлять в семантическую сеть изображения лишь тех иероглифов, что имеются в базе данных, однако в дальнейшем для создания новых изображений планируется подключение разработанной ранее Шараповым Юрием, совместно с которым ведётся работа над системой Онто-Кит 2.0, «рисовалки» китайских иероглифов [2].

Архитектура подсистемы KnowBChin представлена на рис. 2.

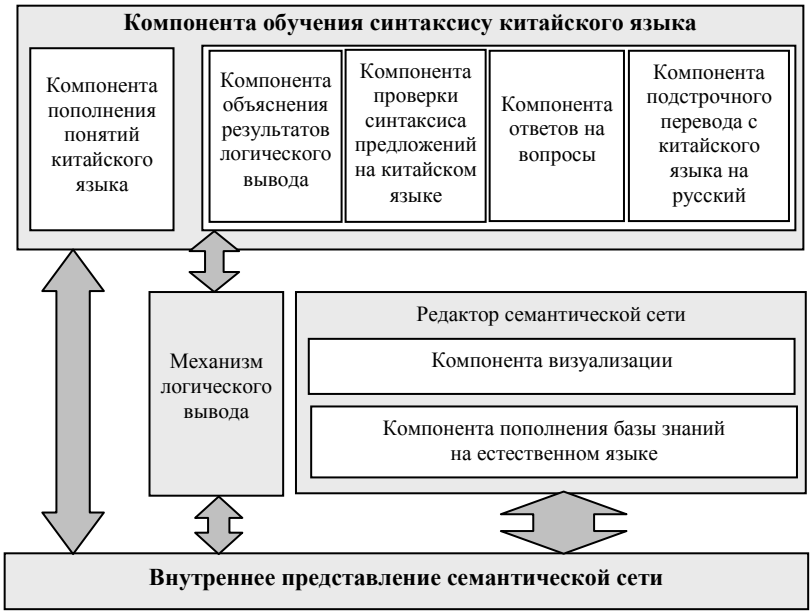


Рис. 2. Архитектура подсистемы KnowBChin

В состав компоненты обучения синтаксису китайского языка входит компонента пополнения понятий китайского языка (рис. 1). Она позволяет пользователю, находясь в режиме обучения, сообщать подсистеме, что какой-то иероглиф или группа иероглифов представляют собой понятие, которого нет в базе знаний. Компонента, основываясь на внутреннем представлении семантической сети, проверяет, действительно ли такие знания отсутствуют, после чего добавляет их, запрашивая при необходимости некоторые дополнительные сведения.

При обучении синтаксису китайского языка также функционируют следующие компоненты: проверки синтаксиса предложений, подстрочного перевода с китайского языка на русский, ответов на вопросы. Их работа основана на результатах, полученных в ходе логического вывода по семантической сети, лог работы которого заносится в компоненту объяснения.

Для ответов на вопросы используется специальная база знаний подсистемы, пополняемая синтаксически правильными утвердительными предложениями на китайском языке.

Развитие подсистемы KnowBChin

На настоящий момент существенным недостатком разработанной подсистемы является то, что она способна находить при проверке синтаксической правильности предложений на китайском языке лишь единственный вариант разбивки предложения на слова. Проблема в том, что в отличие от русского языка в китайском не принято ставить разделители между словами, а синтаксическую роль иероглифа возможно определить лишь исходя из контекста, в котором он употребляется. Таким образом, чтобы проверить синтаксическую правильность предложений на китайском языке, требуется выполнить адекватное разбиение предложения на слова. Кроме того, носителям языков романской группы легче воспринимать китайские тексты, когда между словами поставлены разделители.

Дополнительной трудностью при решении задачи разделения предложений на китайском языке на слова является то, что сами китайцы расходятся во мнении, когда проводят «расстановку пробелов» в тексте на их родном языке.

В настоящее время для решения проблемы выделения слов из текста на китайском языке проводится обзор существующих методов, который позволит в дальнейшем реализовать в подсистеме KnowBChin компоненту с соответствующей функциональностью.

Библиографический список

1. *Чуприна С.И., Осотова Т.В.* Автоматизация синтаксического анализа предложений китайского языка // Сборник тезисов конференции «Актуальные проблемы механики, математики, информатики». – Перм. ун-т. Пермь, 2010. С. 256.
2. *Чуприна С.И., Шарапов Ю.А., Осотова Т.В.* Онтологический подход к созданию автоматизированной адаптивной системы для обучения китайским иероглифам // Труды XXXVI международной конференции «Информационные технологии в науке, социологии, экономике и бизнесе» IT + SE'09. Осенняя сессия. 2009. С. 53-55.

СИСТЕМЫ МОДЕЛИРОВАНИЯ И МАШИННАЯ ИМИТАЦИЯ

Ю.В. Кольцов, Е.В. Бобошко

Кубанский государственный университет

dean@fpm.kubsu.ru, IERC.Evgeniy.Baboshko@gmail.com

ПОДХОД К ИССЛЕДОВАНИЮ ПОТЕРЬ ЭЛЕКТРОЭНЕРГИИ НА ОСНОВЕ ПРОГРАММНО ГЕНЕРИРУЕМЫХ ДАННЫХ

В настоящее время, в связи с рядом реформ энергетической отрасли и переходом к конкурентному оптовому рынку электроэнергии (ЭЭ), все большую роль играют задачи расчета, нормирования и анализа потерь электроэнергии в сетях. Алгоритмы и методы расчета потерь должны обеспечивать максимально точные расчеты в условиях недостаточности исходной информации и ограниченности во времени. Необходимы модели и конкретные их реализации, позволяющие прогнозировать величину потерь ЭЭ, выявлять их очаги, вырабатывать мероприятия по их снижению. В данных направлениях сегодня ведутся обширные исследования [1-3], однако окончательно эти задачи еще не решены. В частности, большое внимание уделяется статистической обработке данных электросетей с целью выявления упрощенных зависимостей величины потерь ЭЭ от ее факторов. Это, прежде всего, регрессионный анализ, однако используются и другие методы, например, методы искусственного интеллекта.

В данной статье рассматривается подход к исследованию потерь ЭЭ, основанный на генерации искусственных выборок данных о зависимости величины потерь от параметров сети и ее режимов. Методы

расчета ориентированы на городские и сельские распределительные сети уровня напряжений СНII (среднее напряжение 2, 1-20 кВ), а также частично НН (низкое напряжение, < 1 кВ), хотя могут быть использованы и для расчета высоковольтных сетей.

В соответствии с [4], технологические потери электроэнергии при ее передаче по электрическим сетям включают в себя потери, не зависящие от величины передаваемой мощности (нагрузки) – условно-постоянные потери, и потери, объем которых зависит от величины передаваемой мощности (нагрузки) – нагрузочные (переменные) потери, а также потери, обусловленные допустимыми погрешностями системы учета. Наибольший интерес представляют первые две категории, объединенные общим понятие «технические потери».

Распределительные электрические сети напряжением 6-20 кВ, а также сети 35 кВ, характеризуются большим числом элементов (участков линий, трансформаторов) и меньшей полнотой и достоверностью информации по сравнению с основными замкнутыми сетями энергосистем. Они работают, как правило, в разомкнутом режиме, [5].

Условно-постоянные потери определяются в результате поэлементного расчета сети, в зависимости от типа конкретного элемента и его технических характеристик.

Что касается нагрузок, здесь, в отличие от сетей 35 кВ и выше, где обычно известны значения P и Q в узлах нагрузки и в результате расчета установившегося режима выявляются потоки P и Q в каждом элементе, для сетей 6-20 кВ известен, как правило, лишь отпуск электроэнергии через головной участок фидера, т.е. фактически суммарная нагрузка всех ТП 6-20/0,4 кВ, включая потери в фидере. По отпуску энергии могут быть определены средние значения P и Q на головном участке фидера. Для расчета значений P и Q в каждом элементе необходимо принять какое-либо допущение о распределении суммарной нагрузки между ТП. Обычно принимают единственно возможное в этом случае допущение о распределении нагрузки пропорционально установленным мощностям ТП, [6].

На основании этих данных, можно рассчитать потери мощности в элементах сети в режиме средних нагрузок. Кроме того, иногда используют параметры работы сети в режиме максимальных нагрузок (определяется по контрольным замерам). Таким образом, для расчета потерь электроэнергии используются два схожих метода, основанных

на интегрирующих множителях: *метод средних нагрузок* и *метод наибольших потерь*. Соответственно, формулы:

$$\Delta W = k_k \Delta P_{max} T\tau$$

и

$$\Delta W = k_k \Delta P_{ñð} T k_ô^2$$

где k_k – коэффициент коррекции, 1,01 для сетей 6-20 кВ, ΔP – потери активной мощности в указанном режиме сети (средний либо максимальный по нагрузке режимы), T – количество часов, τ и $k_ô^2$ – интегрирующие множители, которые могут быть рассчитаны на основе T и T_{max} – числа часов использования наибольшей нагрузки сети, [6].

Метод эквивалентного сопротивления также используется для расчета местных сетей 6-20 кВ, однако отличается от предложенных выше. Суть его в том, что реальная распределительная сеть заменяется одним элементом с эквивалентным сопротивлением $R_Э$ и нагрузкой (током, полной мощностью), равной нагрузке головного участка $I_{ГУ}$ в режиме наибольших нагрузок, причем значение эквивалентного сопротивления должно быть таково, что потери электроэнергии в нем равны нагрузочным потерям в реальной сети [5]:

$$\Delta W_í = 3I_{ÃÓ}^2 R_Ý \tau$$

$$R_Ý = \frac{\sum_{i=1}^{n} I_i^2 R_i}{I_{ÃÓ}^2}$$

где $I_{ГУ}$ – ток, проходящий через головной участок сети в режиме максимальных нагрузок, $R_Э$ – эквивалентное сопротивление сети, R_i и I_i – сопротивление и ток в i-м элементе сети.

Расчет нагрузочных потерь ЭЭ в сетях 0,4 кВ допускается производить на основе обобщенной информации о схемах и нагрузках сети. Потери электроэнергии в линии 0,4 кВ со сечением головного участка F_M, отпуском электроэнергии в линию $W_{0,4}$, за период $Д$ дней, рассчитываются в соответствии с методом оценки потерь электроэнергии на основе зависимости потерь от обобщенной информации о схемах и нагрузках сети по формуле:

$$\Delta W = k_{0,4} \times (W_{0,4})^2 \times \frac{(1-d_í)^2 (1+tg^2\phi) L_{ýêâ}}{F_ì \cdot Ä} \times \frac{1+2k_ç}{3k_ç}$$

где $L_{экв}$ – эквивалентная суммарная длина линий; $tg\varphi$ – средний коэффициент реактивной мощности; $k_{0,4}$ – коэффициент, учитывающий характер распределения нагрузок по длине линии и неодинаковость

нагрузок фаз; $d_н$ – доля электроэнергии, потребляемая на расстоянии 1-2 пролета от ТП, по отношению к суммарному отпуску в сеть 0,4 кВ; $k_з$ – коэффициент формы графика нагрузки [4, 6].

Рассмотрим теперь проблему генерации выборок данных, характеризующих зависимость потерь ЭЭ от параметров сети и режима. Как уже было отмечено, важным направлением исследований является статистический анализ ретроспективы величины и факторов потерь ЭЭ в электросетях за некоторые промежутки времени. Подобный анализ позволяет создавать модели упрощенного расчета потерь без существенного уменьшения точности, модели прогнозирования, кластеризации и локализации технологических потерь ЭЭ. Кроме того, это открывает возможность построения расчетных моделей для слабо формализуемых коммерческих потерь ЭЭ.

При исследовании динамики потерь ЭЭ на основе статистических методов одной из серьезных проблем является недостаток или недостаточная репрезентативность выборок данных о работе сетей конкретных электросетевых предприятий. Это может быть обусловлено как организационными проблемами, так и проблемами технического характера. Однако исследования, в частности [1], показывают, что в подобного рода задачах можно с успехом применять искусственные, суррогатные выборки данных на основе тестовых (или реальных) электросетей, лишь затем апробируя полученные результаты на работе реальных энергетических предприятий.

Кроме того, даже имея информацию о схеме сети, ее параметры, данные о некоторых режимах, и ретроспективу значений потерь ЭЭ, можно столкнуться с рядом трудностей. Прежде всего, это недостаточность информации и необходимость обработки вручную крупных массивов данных. Из всего этого вытекает необходимость в некотором программном инструментарии, предоставляющем возможность ввести информацию о сети, а также ее нагрузке, и получить на основе этого совокупность данных, отражающих зависимость потерь ЭЭ от влияющих на них факторов.

Для этих целей была разработана программа «ИссТП». Эта программа позволяет рассчитывать потери электроэнергии в сетях среднего 6-20 кВ и низкого 0,4 кВ напряжения городских и сельских энергосбытовых организаций и формировать на основе этих расчетов выборки данных. Дополнительно, имеется возможность отдельного расчета установившего режима сети произвольного напряжения.

Удобный интерфейс, выполненный на основе платформы .NET, позволяет ввести данные о схеме сети и о некотором ее режиме в табличном виде (рис. 1). Две таблицы – это узлы и ветви сети, отражаю-

щие традиционное представление сети в виде ориентированного графа. Ветвь представляет собой элемент сети, узел – точку сочленения двух и более ветвей. Для получения данных о стандартном электрооборудовании и его электрических характеристиках можно использовать специальные справочники.

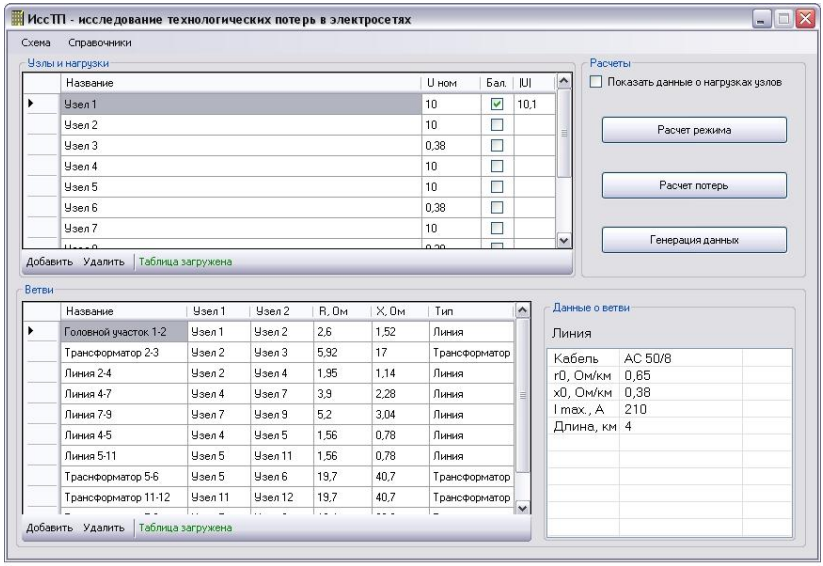


Рис. 1. Главное окно программы «ИссТП»

В качестве локальной базы данных приложения используется распространяемая свободно СУБД PostgreSQL. Подключение осуществляется через программный интерфейс ODBC, что позволяет настроить подключение также к удаленной БД.

Архитектура приложения выбрана таким образом, чтобы максимально упростить добавление новых типов элементов сети, а также связанных с ними алгоритмов расчета условно-постоянных и нагрузочных потерь.

Программа позволяет произвести расчет нагрузочных потерь ЭЭ одним из трёх методов: средних нагрузок, наибольших потерь или эквивалентного сопротивления. Для сетей 0,4 кВ, расчет производится исходя из обобщенных данных о схемах и нагрузках сети. Расчет нагрузочных потерь производится на основе либо введенных мощностей нагрузки и генерации узлов сети, либо отпущенной через головной участок фидера ЭЭ. Также необходимо ввести число часов исполь-

зования наибольшей нагрузки сети T_{max}.

На основании указанных алгоритмов расчета потерь ЭЭ программа может сформировать выборку данных (рис. 2). В первой, более крупной таблице, представлены входные данные для расчета потерь, во второй – выходные. Полужирным шрифтом выделены параметры, значение которых необходимо задать и которые являются общими для всей сети. Остальные представляют собой все возможные исходные данные элементов сети, ее ветвей и узлов. Среди них мы может отметить галочкой те, которые мы желали бы сохранить в результирующей выборке. Далее, мы можем задать новое значение для любого параметра, которое будет присвоено каждому подобному элементу.

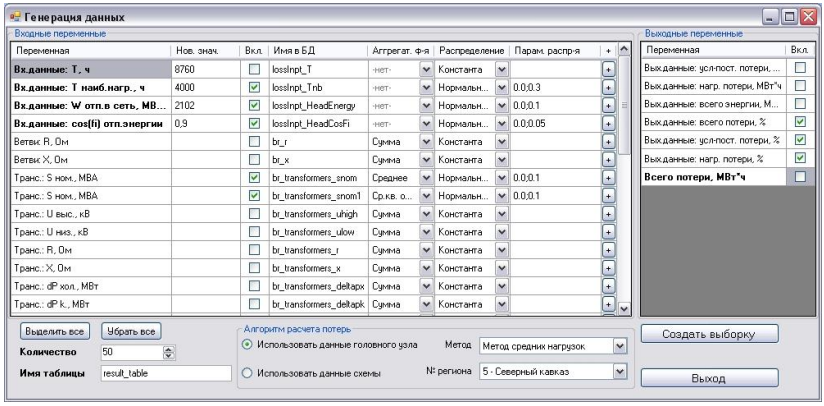


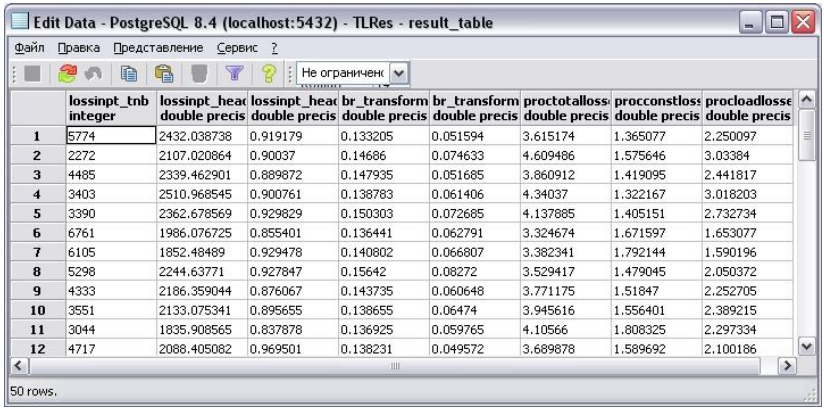
Рис. 2. Окно настройки генератора выборки

Один проход алгоритма расчета потерь ЭЭ формирует одно наблюдение в результирующей выборке, т. е. один ряд в результирующей таблице, задающий сопоставление входных и выходных значений расчета. Количество наблюдений можно задать в специальном поле. Каждое наблюдение может включать в себя:

1. Набор значений входных параметров элементов сети, обобщенных посредством выбранной для каждого параметра агрегатной функции (аналог агрегатных функций языка SQL). Один и тот же параметр может повторяться, будучи обобщаемым при этом разными агрегатными функциями.
2. Набор значений входных параметров, общих для всей сети (например, длительность расчетного периода в часах).
3. Набор значений выходных параметров – условно-постоянных, нагрузочных и суммарных потерь ЭЭ.

Наконец, для любого входного параметра можно указать распределение вероятностей, а также его параметры, что является необходимым для формирования различных образцов выборки. Например, можно считать, что электрическая нагрузка узлов сети как случайная величина распределена по нормальному закону распределения вероятностей, что является следствием центральной предельной теоремы [4]. Таким образом, на каждой итерации основного цикла генерации данных мы получаем набор гипотетических значений параметров сети и режима, а также рассчитанные на их основе значения потерь ЭЭ, что и формирует один образец результирующей выборки.

Полученная выборка может быть сохранена в БД, а затем импортирована оттуда в любую систему анализа данных (рис. 3). Например, была протестирована возможность импорта и анализа полученных данных средствами программного комплекса STATISTICA 6 фирмы StatSoft.


Edit Data - PostgreSQL 8.4 (localhost:5432) - TLRes - result_table

Файл Правка Представление Сервис ?

Не ограничен

	lossinpt_tnb integer	lossinpt_heac double precis	lossinpt_heac double precis	br_transform double precis	br_transform double precis	proctotalloss double precis	procconstloss double precis	procloadlosse double precis
1	5774	2432.038738	0.919179	0.133205	0.051594	3.615174	1.365077	2.250097
2	2272	2107.020864	0.90037	0.14686	0.074633	4.609486	1.575646	3.03384
3	4485	2339.462901	0.889872	0.147935	0.051685	3.860912	1.419095	2.441817
4	3403	2510.968545	0.900761	0.138783	0.061406	4.34037	1.322167	3.018203
5	3390	2362.678569	0.929829	0.150303	0.072685	4.137885	1.405151	2.732734
6	6761	1986.076725	0.855401	0.136441	0.062791	3.324674	1.671597	1.653077
7	6105	1852.48489	0.929478	0.140802	0.066807	3.382341	1.792144	1.590196
8	5298	2244.63771	0.927847	0.15642	0.08272	3.529417	1.479045	2.050372
9	4333	2186.359044	0.876067	0.143735	0.060648	3.771175	1.51847	2.252705
10	3551	2133.075341	0.895655	0.138655	0.06474	3.945616	1.556401	2.389215
11	3044	1835.908565	0.837878	0.136925	0.059765	4.10566	1.808325	2.297334
12	4717	2088.405082	0.969501	0.138231	0.049572	3.689878	1.589692	2.100186

50 rows.

Рис. 3. Готовая выборка данных

Итак, программа «ИссТП» позволяет на основе параметров сети и ее режимов рассчитывать потери в электросетях номинального напряжения 0,4-20 кВ и формировать из них совокупности данных. Эти совокупности могут отражать работу как реальных электросетей энергосбытовых организаций, так и искусственных, тестовых сетей.

Полученные данные могут быть использованы для проведения исследования статистической зависимости потерь ЭЭ от влияющих на них факторов с целью построения эффективных моделей их расчета, прогнозирования и анализа.

Библиографический список

1. *Заиграева Ю.Б.* Нейросетевые модели оценки и планирования потерь электроэнергии в электроэнергетических системах : автореферат дис. ... кандидата технических наук : 05.14.02; [Место защиты: Новосиб. гос. техн. ун-т]. Новосибирск, 2008.
2. *Томин Н.В.* Анализ и прогнозирование режимных параметров и характеристик для субъектов розничного рынка электроэнергии на базе технологий искусственного интеллекта : автореферат дис. ... кандидата технических наук : 05.14.02; [Место защиты: Ин-т систем энергетики им. Л.А. Мелентьева СО РАН]. Иркутск, 2007.
3. *Манусов В.З., Могиленко А.В.* Методы оценивания потерь электроэнергии в условиях неопределенности // Электричество. 2003. № 3. С. 2-9.
4. Инструкция по организации в Министерстве энергетики РФ работы по расчету и обоснованию нормативов технологических потерь электроэнергии при ее передаче по электрическим сетям. Утв. Приказом Минэнерго РФ № 326 от 30.12.2008.
5. *Герасименко А.А., Федин В.Т.* Передача и распределение электрической энергии: Учебное пособие. Ростов-н/Д.: Феникс; Красноярск: Издательские проекты, 2006.
6. *Железко Ю.С., Артемьев А.В., Савченко О.В.* Расчет, анализ и нормирование потерь электроэнергии в электрических сетях: Руководство для практических расчетов. М.: Изд-во НЦ ЭНАС, 2004.

МОДЕЛИРОВАНИЕ И ТЕХНОЛОГИ РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ

А.А. Ахрамейко, И.В.Хмельницкая

ГУВПО «Белорусско-Российский университет»

join@tut.by, inna21@gmail.com

КОНЦЕПТУАЛЬНЫЕ ПРОБЛЕМЫ ПОСТРОЕНИЯ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКИХ СИСТЕМ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЯ

Введение

В последнее десятилетие в различных странах мира появилось значительное число работ, относящихся к системам поддержки принятия решений как новому средству решения многокритериальных проблем. Интерес к СППР непрерывно растет. Также следует отметить значительный рост числа диссертационных исследований по данной тематике как докторских, так и кандидатских.

Создание информационно-аналитической системы поддержки принятия решений (ИА СППР), лежащей в основе современных интегрированных корпоративных информационных систем нового поколения, является современным, перспективным, научно-обоснованным стратегическим, системным подходом к комплексной автоматизации бизнес-процессов.

Повсеместное применение современных информационных технологий привело к пониманию важности решения проблем, связанных с эффективным использованием все возрастающих объемов накопленной информации для оперативного извлечения новых знаний и их применения для повышения степени обоснованности принимаемых решений по управлению предприятиями как в нормальных условиях функционирования, так и в нештатных ситуациях. Это, наряду со стре-

мительным инновационным развитием в области информационных технологий, обуславливает объективную необходимость разработки нового инструментария отдельных предметных областей.

Так основой аналитических систем является сочетание результатов технического и бизнес мониторинга для их последующей обработки средствами многомерной и интеллектуальной обработки данных. Одним из наиболее информативных и технологичных источников данных для таких систем является комплексный дистанционный бизнес-мониторинг объектов с последующим размещением результатов в корпоративном (в перспективе распределенном) хранилище данных. Результаты многоаспектного мониторинга являются основой для системы прогнозирования и информационно-аналитического обоснования принимаемых управленческих решений (информационно-поисковой, оперативно-аналитической и интеллектуальной поддержи принятия решений). В тоже время существует значительное число этапов работ, в которых принимаемые решения основываются на мнении экспертов, в условиях неопределенности, неоднородности данных и т.п. Реализация вышеуказанных методов, моделей и средств в ИА СППР, базирующейся на элементах интеллектуального анализа данных позволит разработать эффективный инструментарий совершенствования тактического и стратегического управления предприятиями.

Для решения выделенных проблем и задач имеется значительный задел у отечественных и зарубежных исследователей. Система мониторинга должна базироваться на результатах аналитической обработки экономической информации. Существенным моментом является наличие разработанных методик, методов и подходов, совершенствование в дальнейшем которых, позволит осуществлять мониторинг не только организации в целом, но и ее отдельных направлений деятельности, структурных подразделений, объектов анализа (ресурсов, затрат, результатов). Предлагаемые концептуальные подходы к проведению комплексного экономического анализа финансово-хозяйственной деятельности организации, а также методология решения многокритериальных задач позволят смоделировать систему мониторинга вне зависимости от количества направлений деятельности организации, ее размеров и т.п.

Несмотря на достигнутые успехи по теме проекта не существует цельной теории построения ИА СППР и ее компонентов в составе КИС нового поколения для обоснования принимаемых решений в нештатных проблемных ситуациях и для совершенствования на их основе процессов предприятиями. Не решены проблемы создания фундаментальных основ построения проблемо-адаптированных знание-ориентированных КИС, а также методов, методик и алгоритмов: полу-

чения и систематизации неполной и неопределенной экономической информации; анализа, классификации, распознавания и формирования признаков, описывающих экономические объекты и процессы; разработки интеллектуальных технологий классификации, мониторинга, диагностики и прогнозирования внештатных проблемных ситуаций.

В связи с этим целесообразна модификация, обобщение и разработка новых теорий и концептуальных подходов построения информационно-аналитической системы поддержки принятия решений управления предприятием в условиях неопределенности, неоднородности и неполноты данных, лежащей в основе современных интегрированных корпоративных информационных систем нового поколения.

Также требуют развития предметно-ориентированные модели и алгоритмы в рамках гибридной методологии для многомерной и интеллектуальной обработки экономической информации с последующей их практической апробацией, что позволит разработать практические рекомендации по проектированию и построению интеллектуальных информационно-аналитических систем поддержки принятия решений в КИС нового типа.

На предприятие воздействует множество внешних факторов, а помимо них динамично меняется его внутренняя среда. Оценка и прогнозирование его состояния в таких условиях становится очень сложным занятием. Для такого рода исследований зарубежные и отечественные ученые разработали множество методов, однако мировой финансовый кризис показал их если не полную неприменимость, то узость областей применимости и используемых алгоритмов. Поэтому необходимо разрабатывать и внедрять в хозяйственную практику новые научно обоснованные методы поддержки принятия решений, ориентированные на адаптивность под задачу, использование знаний. Такого рода системы должны базироваться на эргатическом подходе, поскольку системы искусственного интеллекта далеко не всегда могут эффективно решать управленческие задачи, возникающие ежедневно на современном предприятии.

Любого рода события приводят к изменениям внутри предприятия, его хозяйственной деятельности, что находит свое отражение в изменении показателей, характеризующих его деятельность. Таким образом, накапливая знания вида «действие – результат», можно выработать правила, позволяющие прогнозировать изменения характеризующих предприятие показателей в зависимости от изменения влияющих факторов.

Доказано, что целевые (нормативные) значения показателей, характеризующих деятельность предприятия, целесообразно определять исходя из фактического состояния дел (например, целевое значение

коэффициента текущей ликвидности определяется исходя из фактического бухгалтерского баланса предприятия). Исходя из этого можно утверждать, что вышеописанные изменения будут приводить к соответствующим изменениям целевых значений показателей. В этом случае тоже представляется целесообразным извлекать из хозяйственной практики знания, с тем чтобы построить правила, позволяющие устанавливать целевые значения показателей только по изменениям внешней среды, даже в отсутствие информации об отчетности предприятия.

Важной проблемой в этом случае является правильная организация фиксирования влияющих на состояние предприятия событий (так называемых фактов). Во-первых, необходимо фиксировать не только само изменение и результат, но и начальное (или конечное) состояние предприятия. Это позволит не наивно экстраполировать развитие ситуации, а учесть действие объективных экономических законов и явлений (например, закон убывающей предельной производительности, эффект масштаба и другие). Во-вторых, во избежание ложных отождествлений результата с одним действием, в то время как происходило несколько действий (а также влиял синергетический эффект), необходимо фиксировать факты после каждого значимого действия (например после каждой хозяйственной операции), что не сложно организовать при современном уровне развития корпоративных информационных технологий.

В хозяйственной практике часто необходимо определить пределы варьирования показателей деятельности предприятия или их целевых ориентиров. Для этого существует ряд методов:

- хронологические исследования (горизонтальные исследования);
- пространственные исследования (вертикальные исследования);
- экспертные методы;
- методы балансовых соотношений (для показателей финансового состояния предприятия).

Хронологические исследования предполагают выбор в качестве минимально допустимого значения показателя диагностики его минимальное значение за анализируемый период, а в качестве максимально допустимого – соответственно максимального. У этого метода есть один недостаток – при очередном проведении диагностики могут измениться (и чаще всего бывает именно так) результаты и за предшествующие периоды, то есть получаемая оценка является относительной, она показывает состояние организации по отношению к лучшему и худшему ее результату за анализируемый период. Это необходимо учитывать при интерпретации результатов диагностики и их дальнейшем использовании. Тем не менее, нельзя сказать, что этот недостаток

является критическим. Данный метод является достаточно популярным.

Пространственные исследования предполагают исследование некоторой совокупности организаций. В этом случае в качестве минимально допустимого значения показателя принимается его минимальное значение по исследуемой совокупности предприятий, а в качестве максимально допустимого – соответственно максимальное. Эти исследования могут проводится не только на основании данных за один, но и за несколько периодов. Тогда за пределы варьирования будут приниматься соответствующие экстремальные значения матрицы значений показателя. Исследуемая совокупность организаций может оказаться неоднородной, например в ней могут фигурировать мелкие и крупные, промышленные и торговые, отечественные и иностранные предприятия. Поэтому исследуемую совокупность организаций целесообразно разбить на однородные группы, например при помощи кластерного анализа, а выбор пределов варьирования показателей осуществлять отдельно для каждой группы (кластера).

Экспертные методы предполагают проведение экспертных опросов, в результате которых и определяются пределы варьирования показателей диагностики. Экспертиза может быть проведена любым доступным способом. Нет необходимости подробно останавливаться на процедуре проведения экспертных опросов и обработке их результатов, так как им посвящено много специализированной литературы. Отметим лишь тот факт, что наиболее часто единое мнение экспертной группы определяется как среднее арифметическое отдельных мнений, иногда корректируемое с учетом коэффициентов компетентности экспертов.

В литературе можно встретить модели определения минимально и максимально допустимых значений показателей, основанные на так называемых балансовых соотношениях. Одной из наиболее известных моделей такого рода является модель, приведенная в работах А. Васиной.

Таким образом, разработка проблемо-адаптированных знание-ориентированных КИС является перспективным направлением развития современных информационных технологий. В то же время необходимо совершенствовать методологический аппарат, который требуется для реализации такого рода КИС.

Д.А. Васенина

Пермский государственный университет

natsume_maia@mail.ru

RIGHTUSECHECKER И ПРОЕКТ «ЭЛЕКТРОННЫЙ ПРЕПОДАВАТЕЛЬ ОСНОВ ПРОГРАММИРОВАНИЯ»

Введение

Обучение программированию включает в себя ряд аспектов: постановку задачи, содержательное решение задачи, запись программы на выбранном языке программирования, правильное оформление, тестирование, отладку и др. Причем учить всему этому приходится одновременно, в комплексе. В результате внимание рассеивается сразу по нескольким направлениям. Это требует значительных усилий как от обучаемого, так и от преподавателя.

Для того чтобы уменьшить нагрузку на преподавателя, предлагается автоматизировать его работу. Создание единой программы «автоматического преподавателя информатики» представляется слишком сложным. Более перспективным кажется разделение обучения программированию на несколько аспектов и автоматизация преподавания каждого из них отдельно.

В данной работе будет подробно рассмотрена автоматизация проверки алгоритмической правильности учебных программ.

Описание системы RightUseChecker

Зачем нужна программа

Рассмотрим две простые задачи:

1. Ввести N, A, h. Вывести N чисел от A с шагом h.

2. Ввести A, B, h. Вывести последовательность числе от A до B с шагом h и количество элементов последовательности.

Они довольно похожи, но преподаватель не зря дает их обе, потому что они направлены на два разных типа цикла. Первая – на цикл *for*,

так как там конкретно задано число итераций, а вторая – на цикл *while* или *repeat*, потому что мы заранее не знаем, сколько нам понадобится шагов. Конечно, в случае таких простых задач вполне можно использовать и *for*, и *while*, и *repeat* в обоих случаях и программа будет работать корректно. Но это будет очень плохо с точки зрения обучения, потому что программист должен знать, в каких случаях, какая конструкция будет адекватнее. То же самое касается и структур данных, в некоторых случаях целесообразнее использовать массивы, в некоторых строки, в некоторых записи и т.д. Хороший программист должен уметь выбирать правильные конструкции и структуры.

Именно это и будет контролировать RightUseChecker.

Принцип работы RightUseChecker

Преподаватель описывает эталонную программу, на основе которой будет проводиться оценивание студенческой программы. Данный эталон подается на вход транслятору RightUseChecker'а, который переводит его во внутреннее представление, а именно в семантическую сеть.

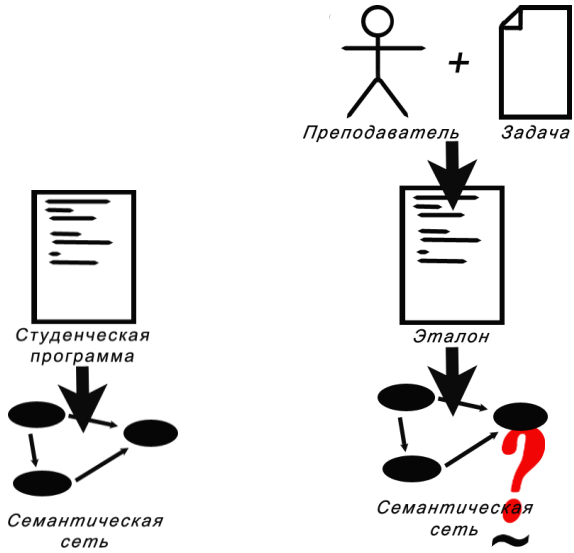


Рис. 1. Схема работы системы

Аналогично обрабатывается студенческая программа. После этого RightUseChecker сопоставляет две получившиеся сети и оценивает студенческую программу. Сеть эталонной программы не обязательно каждый раз генерировать заново, ее можно подгружать из файла. Общий алгоритм представлен на рис. 1.

В ходе трансляции происходит также автоматическая проверка на наличие синтаксических ошибок. В случае, когда ошибки есть, обработка прекращается с соответствующим сообщением.

Что такое эталонная программа?

Сразу подчеркнем, что описываемые здесь механизмы неприменимы к программам произвольной степени сложности, «программам вообще». Речь идет об очень простых программах из наперед заданного множества, а именно, о наборе первых индивидуальных заданий, выполняемых студентами первого курса при изучении основ программирования. Ограниченность и четкое определение набора программ и их простота позволяют заранее определить «все возможные» варианты решения (м.б. точнее было бы говорить обо «всех разумных» вариантах), заранее подобрать нужные структуры управления и структуры данных.

Эталонная программа включает в себя все возможные варианты написания программы для одной конкретной задачи. Каждый такой вариант назовем слоем. Каждый слой имеет имя. Некоторые части этих программ будут одинаковые, т.е. обязательные, а некоторые будут отличаться и образовывать альтернативы. Весь текст эталона разбит на позиции, внутри которых содержится одна или несколько альтернатив, которые в свою очередь состоят из элементов. Элементами являются описание константы, типа или переменной, а также операторы.

В качестве базового языка выбран Паскаль, поскольку именно на Паскале ведется обучение программированию в ПГУ и ПФ ГУ ВШЭ.

Эталонная программа пишется на Паскале, с добавлением некоторых спецсимволов. Внутри каждого блока допускаются альтернативные варианты. Блок состоит из позиций, каждая позиция может иметь несколько альтернатив, внутри альтернативы может быть несколько элементов. Позиция, которая имеет несколько альтернатив (или вариант), должна начинаться со служебного символа !*p* (открывающая позиционная скобка) и заканчиваться символом !!*p* (закрывающая позиционная скобка). Внутри такого блока не может быть обязательных элементов, не принадлежащих какой-либо альтернативе, они все выносятся за ее пределы. Альтернатива начинается символом !*a* (открывающая альтер-скобка) и заканчивается символом !!*a* (закрывающая аль-

тер-скобка). Те элементы, которые присутствуют в любом случае, вне зависимости от варианта, называются обязательными и не заключаются в рамки позиций или альтернатив.

Если альтернатива может использоваться в каком-то слое, то значит она совместима с этим слоем.

Для каждой альтернативы можно в угловых скобках после символа !*a* указать имена слоев, с которыми она совместима. Если же слои не указаны, значит, она совместима с любыми слоями.

В блоке *Begin..End* допускается вложенность «позиция-альтернатива-позиция» любой глубины. В остальных блоках – глубины один.

Также преподаватель сам указывает, какие сообщения следует выдавать в случае несоответствия образцу. Текст ошибки может быть привязан к отдельному оператору, к альтернативе или позиции. Он указывается после соответствующей конструкции и ограничивается символами "#". В общем случае выдается только ошибка наиболее низкого, из имеющих тексты ошибок уровня. Но можно дополнительно указать символ "^" в начале или конце текста ошибки. Это будет означать, что в случае несопоставления, будет выдаваться не только эта ошибка, но и ошибка, принадлежащая конструкции на уровень выше. Например, если не сопоставилась альтернатива, то при наличии символа "^", будет выдаваться сообщение, состоящее из текста ошибки альтернативы и текста ошибки объемлющей ее позиции.

Общая структура эталона приведена ниже.

```
const обязательные константы
  !p
   !a < допустимые слои >
     константы данной альтернативы
   !!a
    ...
  !!p
var обязательные описания
  !p
   !a < допустимые слои >
     описания данной альтернативы
   !!a
    ...
  !!p
type обязательные типы
  !p
   !a < допустимые слои >
     типы данной альтернативы
   !!a
```

```
    ...
   !!p
begin
обязательные операторы
 !p
  !a < допустимые слои >
   операторы данной альтернативы
   . . .
    !p
     !a < допустимые слои >
        . . .
        оператор; #текст ошибки #
        оператор; #текст ошибки #
        . . .
      !!a #текст ошибки #
      . . .
    !!p #текст ошибки #
  !!a #текст ошибки #
   . . .
 !!p #текст ошибки #
. . .
обязательный оператор; #текст ошибки #
обязательный оператор; #текст ошибки #
. . .
end.
```

Структура сети

Эталонная программа, записанная на языке, определенном в предыдущем параграфе, преобразуется в семантическую сеть.

Общая структура сети представлена на рис. 2.

Для каждого элемента альтернативы хранится свой набор атрибутов.

Для типов хранится базовый тип (для массивов, множеств и файлов), размерность (для массивов), границы (для ограниченного типа и для массивов), элементы (для перечислимого типа), имена и типы полей (для записей).

Для переменных хранится описание типа и имя переменной.

Для констант хранится значение и тип.

Для процедур хранятся их имена, типы параметров, вся информация по разделам *var*, *type*, *const* и вся информация о входящих в них операторах. Для функций все то же самое плюс возвращаемый тип.

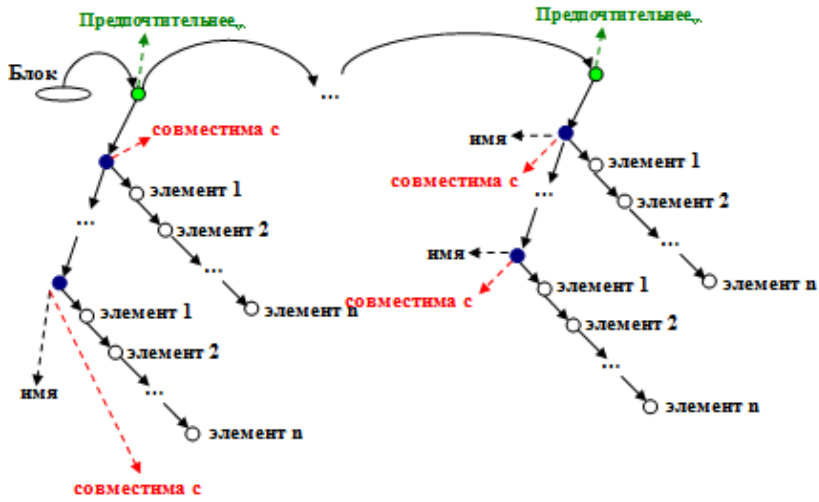


Рис. 2. Структура сети

Для каждого оператора хранится его название и вложенные операторы, если есть. Также для оператора присваивания хранится имя переменной в левой части и выражение в правой. Для операторов *while* и *repeat* – условие, для оператора *for* – начальное и конечное выражения, для оператора *case* – выражение и все метки вариантов с вложенными в них операторами, для вызовов процедур и функций – выражения, переданные в качестве параметров.

Принцип сопоставления

Введём термин «привязка эталона», который будет обозначать процесс сопоставления эталона со студенческой программой.

Программа считается привязавшейся, то есть полностью сопоставимой с эталоном, если все позиции эталонной программы привязались к студенческой программе. Никакие позиции не могут привязаться к одним и тем же элементам студенческой программы. Позиция считается привязавшейся, если привязалась какая-либо альтернатива из этой позиции. А альтернатива привязалась, если привязались все ее элементы. В случае, когда на альтернативу наложены ограничения по совместимости (т.е. четко указаны допустимые для нее слои), она привязывается, только если она совместима с каким-либо слоем из уже привязавшейся части.

Элемент считается привязавшимся, если все атрибуты, кроме имени совпали с тем элементом, к которому привязываем.

Общий алгоритм сопоставления показан ниже:

```
Цикл по всем позициям главной программы эталона
{
    Цикл по всем альтернативам позиции
    {
        Цикл по всем элементам альтернативы
        {
            Если элемент не привязался, то
            {
                Альтернатива не привязалась
                        Выходим из цикла
              }
            Если все привязались, то альтернатива привязалась
          }
        Если альтернатива привязалась, то
        {
            Переходим к следующей позиции
          }
        Если альтернатива не привязалась, то
        {
            Если есть другие альтернативы, то переходим к ним
            Если альтернатив нет, то позиция не привязалась
          }
      }
    Если позиция не привязалась, то программа не сопоставима
  }
```

В общем случае этот алгоритм выливается в рекурсию, из-за вложенности альтернатив и позиций, но суть алгоритма не меняется.

Теперь рассмотрим операцию привязки элементов. Для того чтобы элемент привязался к другому элементу, у них должны совпасть следующие атрибуты:

- Для оператора присваивания – тип переменной и выражение.
- Для оператора *For* – границы и все вложенные операторы.
- Для операторов *While* и *Repeat* – условие и все вложенные операторы.
- Для оператора *Case* – тип переменной, списки меток и соответствующие вложенные операторы.
- Для оператора *If* – условие, вложенные операторы, наличие *Else* и вложенные в него операторы.
- Для вызовов стандартных процедур – имя и параметры.

– Для вызовов своих процедур – операторы процедуры и параметры.
– Для выражений – список присутствующих в выражении переменных и констант, а также знаки операций.
– Для условий – присутствующие в нем выражения и соответствующие знаки сравнения, порядок выражений не важен.

Следует отметить, что проверка сопоставимости выражений и условий выполняется поверхностно, и поэтому эталонные выражения и условия требуют от преподавателя более детального описания, нежели другие конструкции.

Важное замечание: элементы альтернативы в студенческой программе могут идти не в том порядке, в котором идут в эталоне, но порядок позиции должны строго соблюдаться, т.е. поиск сопоставимых элементов для позиции начинается с последнего привязавшегося элемента предыдущей позиции.

Для каждой неудачи в процессе сопоставления сохраняется ошибка, и если программа не привязалась, то выдается весь сохраненный список ошибок.

Реализация

Программа состоит из двух частей – создание эталонов и непосредственно сопоставление эталона и студенческой программы.

Эталонные программы отдаются студентам в зашифрованном виде, для того чтобы у них не было возможности списать. Именно поэтому создание и использование эталонов отделено друг от друга.

Обе части программы написаны на MicrosoftVisualStudio C# и выполняются методом рекурсивного спуска.

Программа создания эталонов – NekoModel – выполняет синтаксический разбор текста программы, попутно проверяя его на наличие ошибок. Чтобы не было путаницы, в разных альтернативах блоков *Var*, *Type* и *Const* имена переменных, типов и констант не должны повторяться, поэтому в ходе обработки переменные, типы и константы записываются в соответствующие списки, При обработке очередной переменной (типа или константы), мы проверяем по этим спискам не встречалась ли переменная (тип или константа) с таким именем раньше. Если нет, то обрабатываем ее и добавляем в соответствующий список. Если имена все же совпали, то это будет ошибка и обработка остановится. Обработка также остановится при наличии любой синтаксической ошибки, как в описании конструкций языка Паскаль, так и в конструкциях поддерживаемого нами расширения. Если ошибок не обнаружено, и трансляция во внутреннее представление прошла

успешно, то на выходе мы имеем семантическую сеть, которую теперь можно сериализовать и сохранить в отдельный файл с расширением .nm.

Семантическая сеть хранится в виде классов. Самый верхний класс имеет следующее описание:

```
public class Onto
{
    public Vars myVar = new Vars();
    public Types myType = new Types();
    public Consts myConst = new Consts();
    public Mains myMain = new Mains();
    public Procs myProcs = new Procs();
    public List<MyVar> varList = new List<MyVar>();
    public List<MyConst> constList = new List<MyConst>();
    public List<MyProc> procList = new List<MyProc>();
}
```

Его поля – это соответственно блок *Var*, состоящий из списка позиций с описаниями переменных, блок *Type*, состоящий из списка позиций с описанием типов, блок *Const*, состоящий из списка позиций с описаниями констант, основной блок, представляющий собой список позиций, состоящих из операторов, блок процедур и три списка с существующими переменными, константами и процедурами (функциями) соответственно.

Вложенные операторы также представляют собой список позиций. Каждая позиция – это список альтернатив, а альтернатива – список элементов. Например, для основного блока, элементом является какой-либо оператор.

```
public class mAlter
{
    public List<Malternative> pos = new List<Malternative>();
    public List<string> forUse = new List<string>();
    public int PosInUse;
    public string erMess = "";
}
public class Malternative
{
    public List<MyMain> alt = new List<MyMain>();
    public string erMess = "";
    public List<string> previous = new List<string>();
}
```

Примерный алгоритм трансляции (на примере блока *Begin.. End*):

1. Считываем *Begin*
2. Смотрим следующий символ. Если он равен !*p*, то пока не встретим !!*p*
   2.1. Считываем символ !*a*
   2.2. Обрабатываем операторы, сохраняя их как элементы в текущей альтернативе, пока не встретим !!*a*
   2.3. Считываем !!*a*
   2.4. Добавляем обработанную альтернативу в текущую позицию
3. Если символ не равен !*p*, то в текущей позиции только одна альтернатива.
   3.1. Обрабатываем операторы. сохраняя их как элементы текущей единственной альтернативы, пока не встретим символ !*p* или финальный *end*.

RightUseChecker начинает свою работу с загрузки эталона и студенческой программы. Эталон загружается в виде сериализованной сети, после чего десериализуется и готов к использованию. Загруженная студенческая программа также переводится во внутреннее представление, т.е. сеть. После чего выполняется сопоставление по рассмотренному выше алгоритму, т.е. совершаем последовательный проход по всем позициям главной программы эталона, и пытаемся найти в студенческой программе сопоставимые с текущей позицией элементы.

Сопоставление ведется по ряду атрибутов, которые хранятся для каждого элемента альтернатив. Если требуемые атрибуты совпали, то элементы сопоставимы.

При выполнении сопоставления, для каждой позиции существует стартовая позиция поиска, т.е. номер элемента студенческой программы в текущей области поиска, с которого можно начинать поиск. Это делается для того, чтобы позиции в студенческой программе следовали в том же порядке, что и в эталоне.

После проверки выдается либо список ошибок, либо сообщение об успешной проверке.

Заключение

К сожалению, при таком подходе тоже существуют проблемы. В частности написание самого эталона – довольно нетривиальный процесс, ведь он должен включать в себя абсолютно все допустимые решения. Эта проблема решается на этапе апробации в ВУЗах, в процессе

которой эталоны дополняются и дорабатываются, т.к. задачи из года в год не меняются, то построенные на этом этапе эталоны можно будет использовать в последующие годы.

Вторая проблема вытекает из первой. Мы можем тестировать лишь небольшие учебные программы, ведь чем сложнее задача, тем больше и запутаннее будет программа, и построить корректный эталон будет практически невозможно. Но, учитывая, ориентацию программы именно на начальные этапы обучения, данная проблема также не является непреодолимой.

В настоящее время RightUseChecker проходит тестирование в двух пермских ВУЗах и планируется его дальнейшее развитие в рамках проекта «Электронный преподаватель».

Е.А. Калашников*

Пермский государственный университет

keatrance@gmail.com

СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА В РЕЖИМЕ WYSIWYG ДЛЯ СИСТЕМ МОНИТОРИНГА ПОКАЗАТЕЛЕЙ ЭНЕРГОПОТРЕБЛЕНИЯ

Введение

Уже не для кого не секрет, что современное производство и потребление начинает все больше и больше удовлетворять условиям меньших энергозатрат. Курс на повышение энергоэффективности уже является официально утвержденным в большинстве современных экономически развитых государств. Это связано как с повышением энергоемкости производственных предприятий и различных учреждений, так и с ростом цен на энергоносители. Современные условия диктуют необходимость радикального изменения отношения к организации энергоучета, оптимизации энергопотребления. Задачи эффективного использования ресурсов (электроэнергия, тепловая энергия, газ, вода и т.д.) актуальны не только для промышленности, но и для других областей деятельности, жизнеобеспечения, и являются одними из ключевых на данный момент.

Для того, чтобы повысить эффективность использования ресурсов и снизить их издержки/утечки, вначале требуется иметь комплексную картину энергопотребления; иметь представление о затратах (в плане ресурсов) происходящих процессов. Для этих целей производится мониторинг потребления различных энергопоказателей с помощью специального оборудования (различных датчиков, счетчиков и измерителей).

Данные, поступающие от этих устройств, обрабатываются программным обеспечением, настроенным для работы с конкретными приборами. Поставляемые разработчиками оборудования программы чаще всего не дают возможности интеграции данных, получаемых из

* Работа выполнена при поддержке РФФИ (проект № 10-01-00794)

различных источников, и их обработки в одной программе, которая бы учитывала показания не одного прибора, а взаимосвязанные параметры, получаемые от различных приборов.

В связи с этим, требуется разработка такого программного обеспечения, которое позволило бы обслуживать множество приборов в единой визуальной среде, интегрировать полученные с них данные. Однако, по мнению разработчиков и пользователей таких систем, главным требованием является возможность адаптации построенной системы под конкретные условия эксплуатации. Система должна динамически настраиваться под новое оборудование и пользовательские нужды, без участия программиста (для изменения исходного кода и последующей перекомпиляции). С этой целью разрабатывается инструментарий, позволяющий собирать виртуальную панель для мониторинга показателей энергопотребления, которые, в дальнейшем, будут рассмотрены экспертами и будут проведены соответствующие политики для повышения энергоэффективности [1].

Разрабатываемая система METAS Control основана на CASE-технологии, позволяющей описывать создаваемую виртуальную панель мониторинга на разных метауровнях. В данной статье основное внимание уделяется средствам создания визуального интерфейса настраиваемой панели в METAS Control.

Презентационная модель

Для создания собственного интерфейса пользователю (или администратору) требуется выполнить несколько действий:

- создать экземпляры компонентов визуализации (элементов управления);
- расположить данные элементы управления в требуемых местах, учитывая их взаимную вложенность и иерархичность;
- настроить основные свойства элементов управления;
- указать источники данных для отображения – те атрибуты, значения которых требуется отобразить, и, при необходимости, задать недостающие значения.

Для обеспечения максимальной гибкости системы, ее адаптируемости (возможности динамической настройки на изменяющиеся условиям эксплуатации, потребностям пользователей) были определены следующие требования [2]:

1. Настройка визуального интерфейса должна быть простой и понятной конечному пользователю-непрограммисту:
   - использование интуитивно понятных действий в стиле

WYSIWYG: технологий Drag & Drop, изменение размеров компонентов мышкой и т.д.;
   - при настройке компонента необходимо отображать те и только те свойства, задание которых требуются непосредственно от пользователя для корректного отображения (например: подпись, выравнивание, шрифт и т.д.), т.е. ничего лишнего (для более быстрого доступа свойства должны быть сгруппированы в определенные категории);
   - при настройке отдельного свойства компонента должно отображаться не только название и значения свойства, но и описание этого свойства (description);
   - при настройке компонента должна быть решена проблема локализации: все отображаемые свойства компонентов должны быть на родном языке пользователя (т.е. на русском языке). Обыденный пользователь вряд ли знает, что означают обозначения Width и Height.
2. Набор поддерживаемых компонент должен быть динамически расширяем:
   - добавление нового компонента должно быть открыто для пользователя, доступно без внесения информации в БД;
   - формат хранения информации о контроле и его свойствах должен быть открытым – например, XML;
   - .NET компоненты визуализации должны иметь четкие интерфейсы для реализации логики отображения поступающих данных и, при необходимости, занесения недостающих значений в базу.
3. Компоненты визуального интерфейса должны поддерживать один из возможных видов привязки данных (binding, «байндинг»):
   - без привязки (none binding): компонент служит лишь для отображения статической информации и/или группировки других компонентов;
   - одиночная привязка (single binding): компонент отображает один параметр системы во времени (например, показания *i*-го термометра);
   - множественная привязка (multi binding): отображение нескольких параметров системы во времени (например, график).
4. Желательно, чтобы компоненты-наследники одного общего родителя (в рамках концепции ООП) имели одни и те же общие свойства, настраивать которые бы не пришлось при до-

бавлении нового элемента-наследника – фактически требуется реализовать наследование описаний свойств.

5. Все изменения в системе должны применяться «на лету», без перекомпиляции исходного кода, по принципу WYSIWYG. Во многих системах модель интерфейса не интерпретируется во время выполнения приложения, а лишь служит для генерации кода пользовательского интерфейса [3].

Для представления настраиваемого пользовательского интерфейса предлагается подход с использованием метаданных [4]. Приложение на уровне пользовательского интерфейса представляется презентационной моделью, показанной на рис. 1.

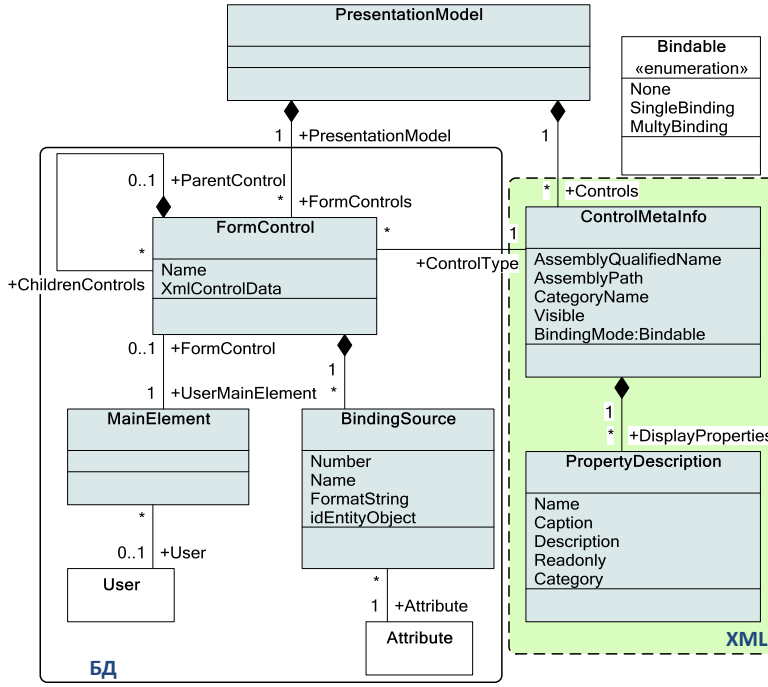


Рис. 1. Презентационная модель

Для обеспечения гибкости и адаптируемости системы в модели были выделены два уровня:

– уровень XML – описаний компонентов визуализации (рис. 1, пунктирная рамка);

– уровень базы данных, на котором хранится информация об экземплярах объектов пользовательских интерфейсов.

Уровень XML-описаний

Набор подключаемых компонентов должен быть легко расширяем. Для представления информации о подключаемых компонентах используется xml-описания компонента, включающие в себя:

– имя и путь к сборке .NET, где располагается скомпилированный компонент;
– название компонента и группы, к которой он принадлежит (компоненты должны быть сгруппированы для более быстрого доступа к ним);
– поддерживаемый режим привязки данных (см. п.3 требований);
– описание свойств, поддерживаемых для отображения и редактирования в редакторе свойств каждого компонента, включает в себя:
  - название свойства (*Name*);
  - заголовок для свойства (*Caption*);
  - описание свойства – для получения пояснений, за что отвечает данное свойство (*Description*);
  - редактируемо оно или используется только для отображения (*ReadOnly*);
  - категория свойства – для группировки и удобного доступа (*Category*).

Описания иерархичны – те свойства, которые задаются для родительских объектов, будут переиспользованы для задания тех же свойств у дочерних объектов. Таким образом, мы получаем возможность единообразного подхода к описанию любого компонента визуализации в понятной для конечного пользователя форме.

Уровень экземпляров объектов

На этом уровне предоставляется описание виртуальной панели мониторинга показателей энергопотребления – описание датчиков, счетчиков и т.д. (рис. 2). Для каждого пользователя задается своя иерархия элементов управления (см. рис. 1 – *FormControl*), содержащая компоненты, вложенные друг в друга. Так как *FormControl* является экземпляром описанного на XML компонента *ControlMetaInfo*, то для хранения измененных значений свойств используется XML-поле в БД, что позволяет нам с большой гибкостью описывать пользовательский интерфейс.

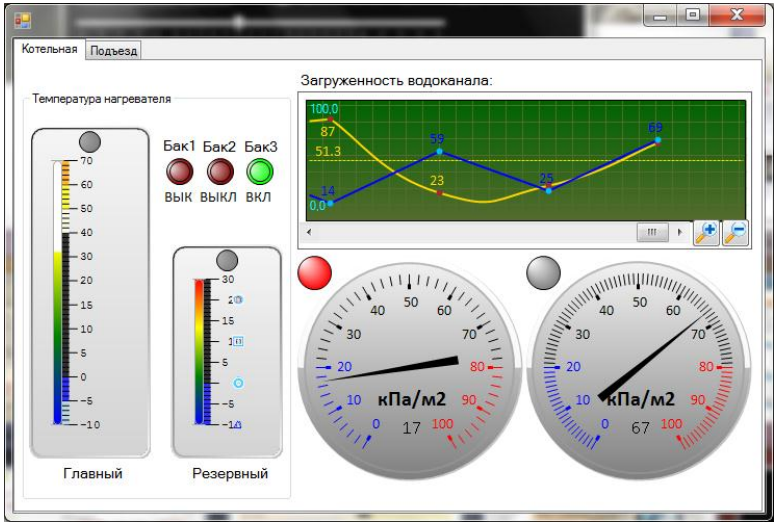


Рис. 2. Внешний вид панели системы мониторинга

Для каждого экземпляра компонента может быть задан *BindingSource*, в котором можно указать, с каким свойством компонента связана тот или иной атрибут определенного экземпляра сущности. В случае режима *MultyBinding* (множественная привязка данных) для обеспечения упорядоченности привязок используется нумерация источников.

Интуитивная настройка панели

Интерфейс для настройки создаваемой виртуальной панели мониторинга внешне напоминает интерфейс Visual Studio 2005. Элементы управления, такие как дерево объектов, редактор свойств, панель инструментов и сама область, на которой располагается наши элемент управления (рис. 3) располагаются на передвигаемых панелях, которые позволяют «пристыковать» их к краю окна и показывать их в нескольких режимах (например, скрывая панель при потере фокуса).

Редактор панелей напоминает упрощенный вариант работы со средой разработки Microsoft Visual Studio и состоит из четырех частей:

- дерева объектов (компоненты могут быть вложены друг в друга, иерархическая структура компонентов панели);
- менеджер свойств, отражающий свойства текущего выбранного элемента (используя возможности .NET рефлексии, заменя-

ем стандартную информацию о свойствах теми значениями, которые были считаны из xml-файла настроек);

– менеджер компонентов, отображающий список доступных компонент для создания панели (данные о доступных компонентах загружаются через XML);
– рабочая область (непосредственно сами наши компоненты, над которыми производится настройка.

Компоненты можно свободно перемещать по панели и изменять размеры (см. ползунки на рис. 3).

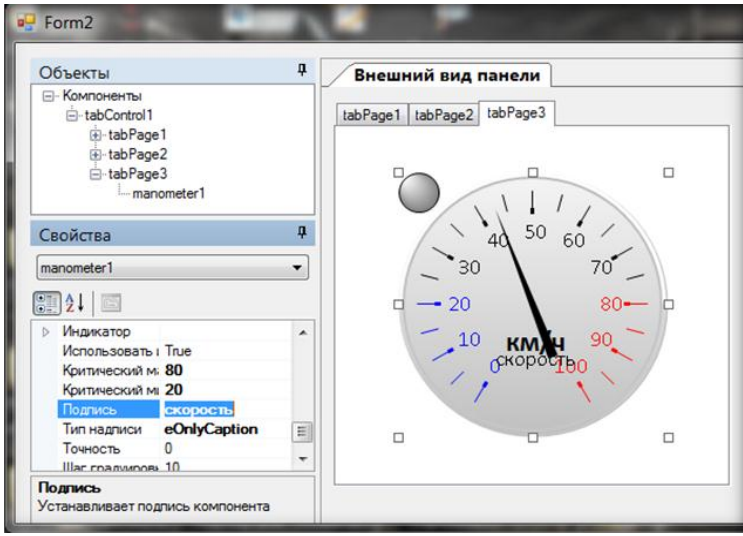


Рис. 3. Форма редактора панели

К редактору свойств предъявляются особые требования. Он не только должен отражать локализованные названия свойств и их описания (которые задаются в XML-маппинге), но и в стандартизованном виде отображать свойства произвольных объектов. При этом *value*-свойства (простые свойства, не имеющие никакой внутренней структуры, например, числовое или строковое значение) задаются в унифицированном виде, а для специфических типов (например, перечислений, структур, коллекций, вложенных объектов и т.д.) предоставляется возможность задавать редакторы свойств. Такими универсальными возможностями обладает компонент *PropertyGrid* – стандартный компонент, включенный в .Net 2.0.

В данной статье был описан подход к созданию гибкого, легко настраиваемого интерфейса приложения мониторинга энергопотребления – виртуальной панели, размещающей в себе подключаемые компоненты визуализации показателей энергопотребления.

Библиографический список

1. *Воронов В.А., Калашников Е.А., Лядова Л.Н.* Технология создания системы мониторинга энергопотребления // Труды Конгресса по интеллектуальным системам и информационным технологиям «AIS'10». Науч. Изд. В 4-х томах. Т. 1. М.: Физматлит, 2010.
2. *Грибова В.В., Клещев А.С.* Использование методов искусственного интеллекта для проектирования пользовательского интерфейса // Информационные технологии. 2005. №8. С. 58-62.
3. *Mohan R., Kulkarni V.* Model Driven Development of Graphical User Interfaces for Enterprise Business Applications // MODELS'09 Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems. Springer-Verlag Berlin, Heidelberg. 2009. P. 307-321.
4. *Fill H.-G.* Visualisation for Semantic Information Systems // PhD Thesis, University of Vienna. 2006.

В.В. Ланин, Л.Н. Лядова*

Пермский государственный университет,
Высшая школа экономики (Пермский филиал)

Lanin@perm.ru, LyadovaLN@hse.perm.ru

ИСПОЛЬЗОВАНИЕ ОНТОЛОГИЧЕСКОГО ПОДХОДА ДЛЯ РАЗРАБОТКИ И ПОДДЕРЖАНИЯ ЖИЗНЕННОГО ЦИКЛА ЭЛЕКТРОННЫХ АДМИНИСТРАТИВНЫХ РЕГЛАМЕНТОВ

Введение

В настоящее время в России, в различных регионах реализуются многочисленные проекты по созданию «электронных правительств». Основное направление в этой области – использование возможностей Internet, современных информационных и телекоммуникационных технологий с целью оптимизации предоставляемых услуг, совершенствования внутренних процессов и повышения уровня участия общества в вопросах государственного управления. Основные предпосылки выполнения поставленной задачи – необходимость массовой разработки административных регламентов (АР) и начало внедрения электронных АР (ЭАР), необходимость снижения трудоёмкости разработки и ведения ЭАР.

Целью исследований является создание технологии, программного инструментария, предназначенного для описания административно-управленческих процессов в органах исполнительной власти, их анализа и оптимизации, позволяющего перенести значительную часть работ по созданию и ведению ЭАР (сбор и первичная обработка информации, построение формальных описаний АР, подготовка обоснованных управленческих решений, связанных с необходимостью разработки новых ЭАР или внесения изменений в существующие и пр.) на госу-

* Работа выполнена при поддержке РФФИ (проект № 10-01-00794)

дарственных служащих, специалистов в соответствующих предметных областях.

Проектируемые средства должны удовлетворять следующим требованиям:

- максимально упрощенный процесс описания АР с использованием предметно-ориентированных языков, основанных на привычной для специалистов терминологии и классификаторах, а также средств визуализации описаний;
- возможность использования ранее созданных описаний административных процедур, что позволит начать работу по созданию ЭАР не «с нуля»;
- наличие средств анализа разработанных АР с целью их оптимизации;
- отсутствие привязки к конкретному средству моделирования бизнес-процессов, технологии реализации ЭАР.

С одной стороны, в последние годы выполняется множество научно-исследовательских работ, посвященных решению поставленных задач [15, 13, 2, 12, 11], разрабатываются нормативные документы, ведущими ИТ-компаниями предлагаются решения в данной области [9, 3, 14, 5], но, с другой стороны, отмечается, что ни одна из существующих методик не удовлетворяет приведенным требованиям.

Электронные административные регламенты: определение, разработка, оптимизация

Административный регламент (АР) – это правовой документ, который может, как и любой другой документ в «бумажной» форме, иметь «электронный аналог» – электронный документ. При этом *электронный административный регламент* (ЭАР) не является электронным документом – *ЭАР является реализацией административного регламента*, представляет собой «электронный формат публичной деятельности органов государственной власти по реализации своих полномочий, основанный на внедрении информационных технологий в области взаимодействия государственных структур, граждан и юридических лиц» [1]. При этом ЭАР не обладает самостоятельной юридической силой, а основывается на положениях соответствующего АР. Таким образом, средства создания и ведения ЭАР должны поддерживать правовую связь между административным регламентом и электронным административным регламентом при его разработке, при выполнении процедур его изменения и отмены, а также процедуры разрешения коллизий между АР и ЭАР.

Важная задача, решаемая в настоящее время, – *разработка понятийного аппарата*, используемого в соответствующих нормативных правовых актах. Единообразие терминологии, четкость и ясность применяемых определений являются залогом эффективной реализации административных процедур и гарантией качества исполняемых функций и предоставляемых услуг. Еще одна задача – *создание типовой модели административных регламентов*: административные процедуры в органах исполнительной власти должны выстраиваться в единообразном порядке (с учетом специфики применительно к конкретному виду деятельности), причем необходимо обеспечить сочетание административных и должностных регламентов. Кроме того, до сих пор *отсутствует единая полная нормативно-правовая база для разработки, принятия и функционирования АР*, нормативно не определены либо сформулированы недостаточно исчерпывающе признаки, виды и требования к административным регламентам, их место в системе актов управления.

Термин «*электронный административный регламент*» встречается во многих нормативно-правовых актах (Программа социально-экономического развития Российской Федерации на среднесрочную перспективу (2003-2005 годы), утвержденная распоряжением Правительства РФ от 15.08.2003 № 1163-р; Концепция использования информационных технологий в деятельности федеральных органов государственной власти до 2010 года, утвержденная распоряжением Правительства РФ от 27.09.2004 № 1244-р; Концепция административной реформы в Российской Федерации в 2006-2008 годах и соответствующий план мероприятий, утвержденные распоряжением Правительства РФ от 25.10.2005 № 1789-р и др.). В различных работах, посвященных этой тематике, даются различные определения ЭАР. В данной работе мы будем использовать следующее определение: «*электронным административный регламент (ЭАР) – реализация административного регламента с использованием информационно-коммуникационных технологий* при условии обязательного обеспечения юридической значимости автоматических и автоматизированных административных процедур, в том числе в случае отсутствия непосредственного взаимодействия участников административного регламента». Аналогичные определения даются во многих документах. Таким образом, при широком толковании этого термина, *под электронным регламентом можно понимать совокупность информационных технологий,* позволяющих реализовать полномочия и обязанности органов власти, поддержку административных процессов и автоматизацию контроля исполнения требований, предъявляемых

законодательством к административной деятельности. Следовательно, для создания ЭАР можно применить подходы, используемые для создания информационных систем.

При разработке административных регламентов для органов исполнительной власти должна быть сформирована *единая функционально-процессная модель организации их деятельности с учетом возможностей современных информационно-коммуникационных технологий* [4]. Административную деятельность целесообразно рассматривать как «процессуальную» деятельность. Во многих случаях эффективность деятельности государственного аппарата, взаимодействия граждан с органами власти снижается из-за того, что административные процессы урегулированы лишь фрагментарно и, как правило, отражают интересы исполнителей, а не граждан (заявителей). Если рассматривать административную деятельность как процессуальную, то государственный аппарат нужно рассматривать как своеобразную корпорацию, в деятельность которой внедряется «процессный подход», целью которого является оптимизация бизнес-процессов и повышения качества продукции или услуг. Таким образом, к деятельности государственного аппарата можно применять методологию и инструментарий «процессного подхода», получившего широкое распространение в современных предприятиях. Но при этом существует специфика внедрения «процессного подхода» в деятельность органов исполнительной власти: цели государственных органов не являются коммерческими, следовательно, должна применяться социальная, а не финансовая система показателей эффективности деятельности; «продукция» государственной деятельности является нематериальным продуктом, поэтому должны применяться правовые показатели качества продукции, а не технические; государственные услуги предоставляются гражданам на монопольной основе, что требует реализации «не денежной» формы влияния граждан на качество предоставляемых государственных услуг. Все это должно быть учтено при разработке модели.

Анализ типового, обобщенного административного процесса позволяет выявить общие для многих регламентов и процессов характеристики [13], которые должны быть отражены при создании формальных моделей.

Любой административный регламент и процесс может быть *разбит* (*декомпозирован*) на отдельные подпроцессы, операции и микрооперации:

– *Процесс* (услуга, функция) разбивается на подпроцессы. Подпроцесс – это процесс, для которого могут быть четко обозна-

чены границы, а также элементы (информация), которые имеют место на входе и выходе.

- *Подпроцессы* разбиваются до уровня отдельных операций, последовательность выполнения которых составляет процесс. С каждой операцией связаны определенные действующие лица, такие как государственные служащие, граждане или информационные системы.
- *Операции* могут быть разбиты на последовательность микроопераций.

Выделяются следующие *этапы создания ЭАР* [13]:

- определение приоритетных для регламентации процессов в органах власти;
- описание процессов в режиме «как есть»;
- проведение оптимизации процессов и создание моделей процессов «как должно быть»;
- формирование текстовых регламентирующих документов (административных и должностных регламентов);
- создание технических заданий на разработку ЭАР.

Отмечается [13], что особое внимание следует уделить изучению и описанию процессов в режиме «как есть»: сформированные рутинные процессы, зачастую оптимизированные временем, являются основой устойчивого управления. Даже в случае их технологической неэффективности только на основе модели «как есть» можно начать итеративный процесс оптимизации системы целей и показателей, организационной структуры, распределения полномочий, персонифицированной ответственности и т.д. на основе сложившихся (часто нерегламентированных) процессов; оптимизации самих процессов на основе перечисленных системных характеристик.

С точки зрения поставленной цели важной является *задача первичного семантического анализа нормативной документации*, регламентирующей порядок выполнения административного процесса [12].

При построении и оптимизации АР *анализ нормативной документации необходим*, чтобы:

- выделить элементарные операции, инцидентные процессу, и отсечь операции, не относящиеся непосредственно к выполнению процесса;
- определить исполнителей операций, состав и объем используемых при выполнении операций ресурсов;
- определить источники информации, используемой в ходе исполнения процесса;

- составить список документов, используемых и порождаемых в процессе исполнения операций;
- выявить порядок выполнения инцидентных процессу операций и определить операции, которые могут выполняться одновременно;
- выработать рекомендации по использованию имеющихся и созданию необходимых информационных ресурсов, баз (хранилищ) данных, используемых в ходе выполнения административного процесса, и выявить ресурсы, которые могут быть исключены из числа используемых благодаря переходу к новой (электронной) технологии выполнения процесса или расход которых может быть уменьшен;
- построить и оптимизировать сетевой график исполнения процесса;
- произвести расчет показателей эффективности регламента (объема сэкономленных ресурсов, уменьшения времени выполнения, повышения надежности результатов и качества выполнения процесса, уменьшения риска утечки конфиденциальной информации и возникновения ошибок);
- при необходимости, пересмотреть регламент с целью приближения значения показателей эффективности к желаемому уровню.

Для *оценки административных регламентов* можно сформулировать следующие требования [1, 13]:

- соответствие административного регламента действующему законодательству;
- достаточность регламентации участия и разграничения полномочий государственных органов исполнительной власти при реализации административных процедур;
- соответствие формальным требованиям, предъявляемым к содержанию АР исполнения государственных функций и АР предоставления государственных услуг нормативными правовыми актами;
- достаточность регламентации внутриведомственного контроля качества реализации административных процедур;
- надлежащее описание процедур обжалования;
- оптимизация административных процедур в регламенте.

Контроль соблюдения этих требований можно обеспечить только при регулярном проведении экспертизы соответствия электронного административного регламента соответствующим положениям норма-

тивных правовых актов [11]. Проведение этой экспертизы также требует поддержки.

Для оптимизации порядка осуществления услуги (функции) до реализации ЭАР должны быть предприняты определенные меры («оптимизирующие преобразования» регламентов), основные из них можно перечислить:

- упрощение процессов и процедур;
- уменьшение длительности цепочек рабочих шагов процессов;
- уменьшение точек контактов между различными пользователями процессов;
- максимальное исключение итераций;
- уменьшение цикла реализации и «времени ожидания» выполнения очередных шагов.

Важными *приемами оптимизации* административных процедур являются:

- принцип «одного окна» (высвобождение граждан от роли «соисполнителей»);
- принцип «обезличивания» (недопустимость непосредственного взаимодействия заявителя с исполнителем, что позволяет исключить неформальное влияние заявителя на результат, а также позволяет исключить неформальные требования исполнителя в адрес заявителя);
- принцип «реформ за кулисами» (оптимизация административного процесса не должна усложнять процедуру общения граждан с органами власти).

Нарушение этих принципов может быть выявлено с использованием формальной модели регламента, которая может быть построена на основе данных, полученных в результате семантического анализа документов.

Оптимизация документооборота и использования информационных ресурсов при реализации ЭАР предполагает, что:

- информация, используемая и порождаемая в ходе выполнения процесса, не дублируется, но при этом обеспечивается доступ к ней для всех нуждающихся в этой информации и имеющих соответствующие права лиц;
- обеспечивается приемлемый уровень безопасности хранения и использования информации, скорости доступа к ней и разделения прав доступа при использовании источника информации несколькими уполномоченными лицами одновременно.

Все эти оптимизирующие преобразования возможны только при осуществлении комплексной информатизации исполнения функций государственной власти, с учетом особенностей всех административных процессов (как минимум – информационно связанных или близких по сфере деятельности). Выявление всех связей, установление и оптимизация внутри- и межведомственного взаимодействия в многоуровневой системе – сложная задача, требующая использования специальных средств анализа административных регламентов. Анализ может быть выполнен с использованием формальных моделей АР. Однако задача их создания является достаточно сложной и трудоемкой, требующей участия специалистов, работающих в различных областях. При этом необходимость анализа изменений предметной области, внесения соответствующих изменений в модель, существует постоянно, в течение всего жизненного цикла ЭАР.

Целесообразным представляется создание интеллектуальной системы, обеспечивающей поддержку всех этапов создания и ведения АР, перечисленных выше задач.

Использование онтологических моделей для создания и ведения административных регламентов

Результатом анализа нормативно-справочных документов должна стать *концептуальная модель регламента*, на базе которой могут быть сформированы различные представления регламента. Таким образом, при анализе текстов необходимо применение методов концептуального индексирования, при которых для текста автоматически строится не пословный, а понятийный индекс, в котором синонимичные выражения представлены одним и тем же понятиям, многозначность текстовых выражений отражена различными понятиями, и присутствуют отношения между понятиями [10].

Основным подходом к разработке программного инструментария для поддержки ведения ЭАР является *онтологическое моделирование*. Используемые в процессе работы онтологии размещаются в *многоуровневом репозитарии* [6], в котором находятся онтологии предметной области и онтологии нормативно-справочных документов.

Онтология предметной области описывает понятия, используемые в документах, именно в ней описаны такие понятия как «процесс», «операция», «исполнитель» и т.д., кроме того, туда включены различные классификаторы.

Онтологии нормативно-справочных документов, в частности, онтология самого регламента (рис. 1), описывают структуру и характерные элементы документов.

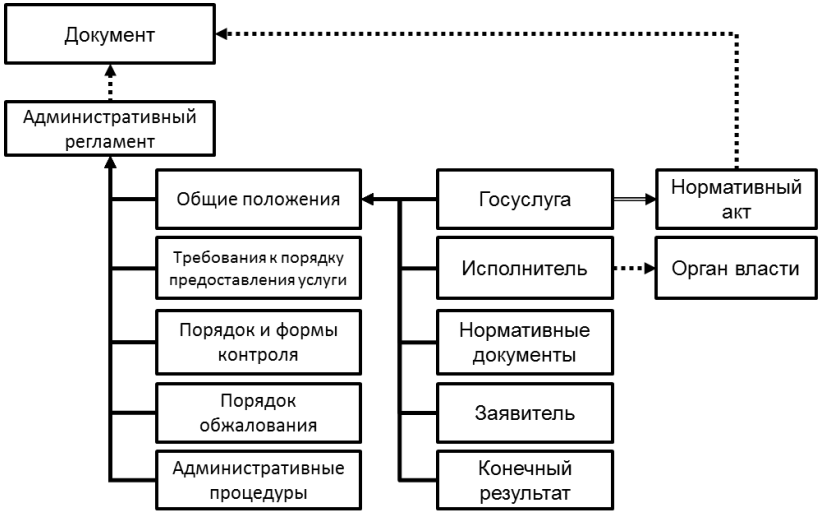


Рис. 1. Фрагмент онтологии регламента

В результате анализа текстового описания (декомпозиции) будет построена концептуальная модель регламента, которая, во-первых, позволит верифицировать его (проверить структуру, выявить дублирование информации и т.д.), во-вторых, позволит связать фрагменты текстового документа с соответствующими понятиями онтологии. Кроме того, концептуальные модели документов могут быть использованы для установки «смысловых» связей между различными документами и визуализации этих связей [8].

Сценарии применения подхода

Можно выделить три сценария использования разрабатываемого программного инструментария поддержки ведения ЭАР:

1. Анализ существующих административных регламентов с целью формализации их описания для последующей обработки.
2. Создание новых регламентов в соответствии с нормативными документами.
3. Поддержка жизненного цикла регламента (внесение возможных изменений).

Первый сценарий предполагает анализ существующего АР и его преобразование в концептуальную модель, на основе которой могут быть построены структурированное описание регламента, блок-схема

описываемого процесса, описание бизнес-процесса на языке XPDL (XML Process Definition Language) и различные аналитические отчеты. Структурированное описание регламента подразумевает представление регламента в форме таблиц, описывающих основные информационные объекты.

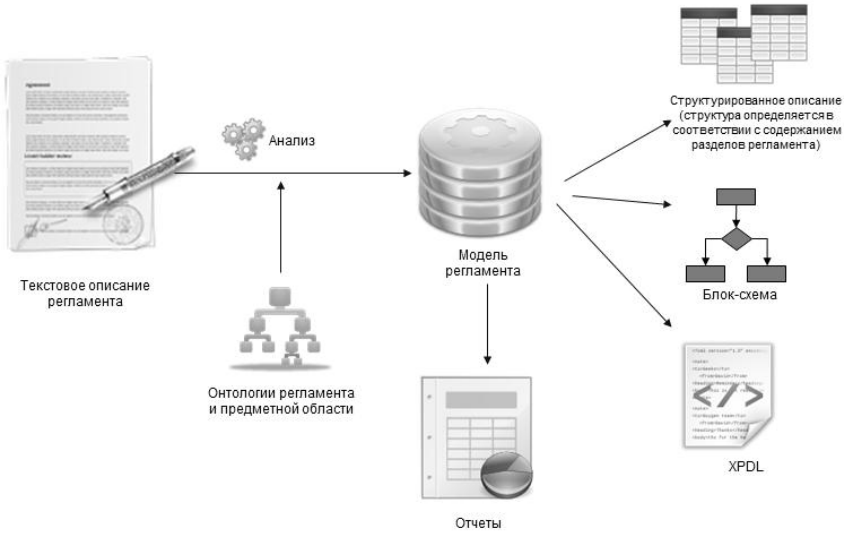


Рис. 2. Схема анализа существующих регламентов

Данное представление пригодно для анализа регламентов с использованием различных методов (теории графов и пр.) с целью верификации и оптимизации как отдельных регламентов, так и их совокупности. Описание на языке XPDL позволит передать описания бизнес-процессов, соответствующих регламентам, в workflow-системы для реализации ЭАР. Схема процесса анализа регламента представлена на рис. 2.

Второй сценарий позволяет аналитику получить поддержку при составлении нового регламента. При разработке регламента аналитик работает не с текстовым представлением регламента, а с его концептуальной моделью.

Схема процесса создания нового регламента представлена на рис. 3.

Пользователю доступны следующие функции системы:

– интеллектуальный поиск необходимых документов на основе разработанных онтологий, привязка их к соответствующим

разделам АР;

- автоматическая проверка корректности структуры создаваемого регламента;
- генерация структурированного описания регламента, текстового представления регламента, блок-схемы соответствующего процесса, описания этого бизнес-процесса на языке XPDL;
- получение аналитических отчетов о возможности оптимизации процессов.

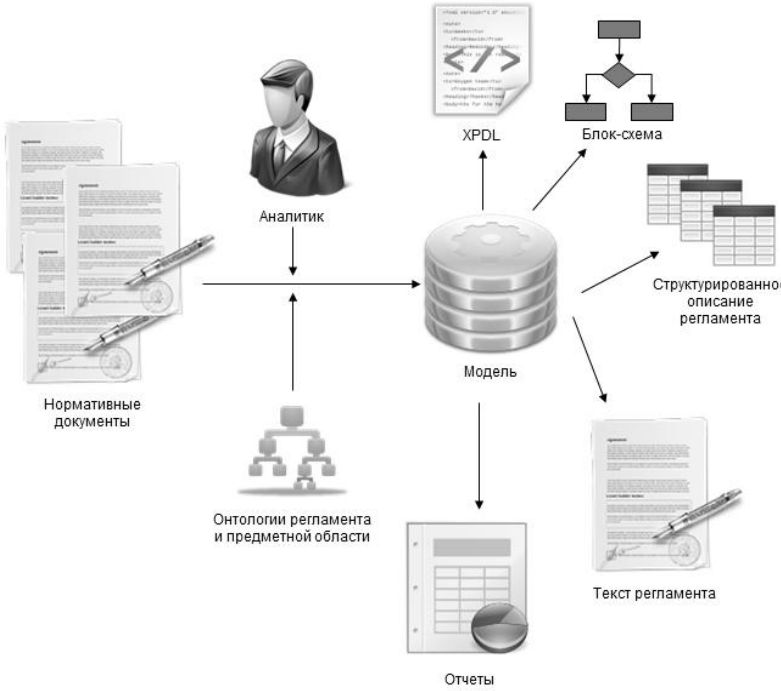


Рис. 3. Схема создания нового регламента

Третий сценарий позволяет поддерживать регламенты в актуальном состоянии. Системой периодически производится анализ состояния предметной области, а именно, изменений нормативных документов. Задача может быть решена на основе мультиагентного подхода [7].

Схема процесса поддержки жизненного цикла регламента представлена на рис. 4.

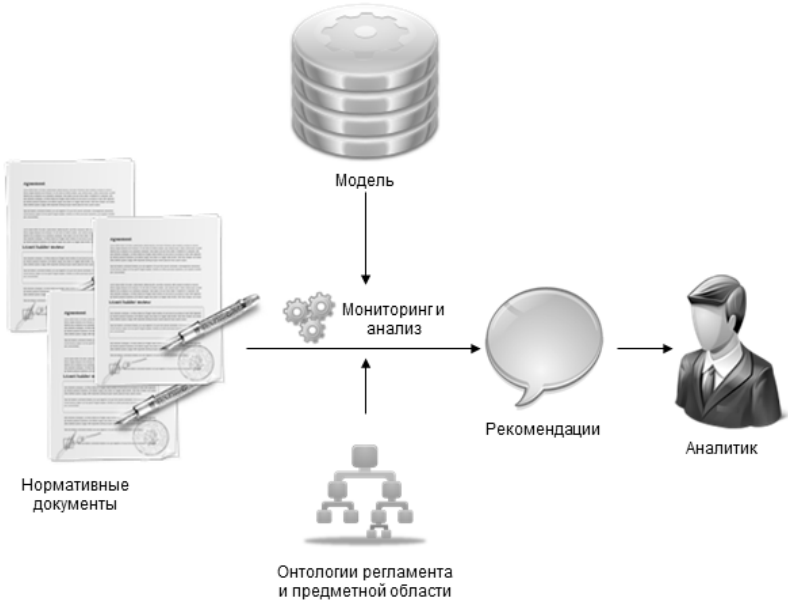


Рис. 4. Схема поддержки жизненного цикла регламента.

Данные, полученные в результате анализа и мониторнга предметной области, могут стать источником оптимизации бизнес-процессов. В частности, полезным может быть:

- выявление связей различных услуг по документам, используемым при предоставлении этих услуг;
- выявление связей исполнительных органов государственной власти и др. при предоставлении услуг.

На основе полученной информации может быть произведена выработка предложений по изменению порядка предоставления отдельных услуг, формированию пакетов услуг, соответствующих так называемым «жизненным ситуациям» (рождение ребенка, выход на пенсию и т.п.) и т.д.

Описание документа с помощью онтологий

Согласно общепринятому определению под онтологией (в широком смысле) понимается база знаний специального типа, которая может «читаться» и пониматься, отчуждаться от разработчика и/или физически разделяться ее пользователями. Учитывая специфику решаемых в данной работе задач, можно конкретизировать понятие

онтологии: *онтология* – это спецификация некоторой предметной области, которая включает в себя словарь терминов (понятий) предметной области и множество связей между ними, которые описывают, как эти термины соотносятся между собой.

Для построения иерархии понятий онтологии используются следующие базовые типы отношений:

- “is_a” («экземпляр – класс», гипонимия);
- “part_of” («часть – целое», меронимия);
- “synonym_of” (синонимия).

Следует учесть, что данные типы отношений являются базовыми и не зависят от онтологии, но необходимо предоставить пользователю возможность добавления новых отношений, которые бы учитывали специфику описываемой предметной области.

В рассматриваемом подходе предполагается наличие трех типов онтологий, используемых для описания документов:

- онтология предметной области конкретной информационной системы (ИС) − ЭАР;
- онтология как база знаний (БЗ) интеллектуального агента;
- онтология как описание документа.

Рассмотрим назначение каждого из перечисленных типов онтологий для решения поставленной задачи.

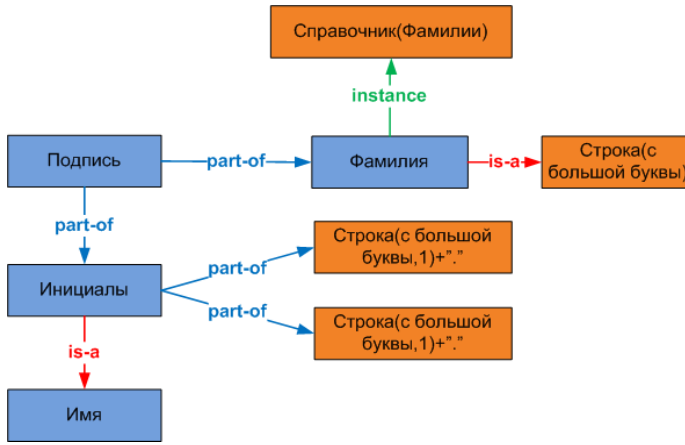


Рис. 5. Представление базы знаний агента с помощью онтологии

Онтологии предметной имеют наиболее типичное применение, они используются для описания понятий предметной области ИС. Например, образование, социальная помощь гражданам или оплата коммунальных услуг. В этой онтологии описывается связь понятий,

языковые единицы для их выражения, аксиомы предметной области. Онтология предметной области используется для семантического индексирования и анализа всех документов системы.

Для анализа документов в проекте используется мультиагентный подход. Интеллектуальные агенты, руководствуясь *онтологией как базой знаний* (второй тип онтологий), производят поиск и анализ конкретных понятий документа (рис. 5). Каждая из вершин такой онтологии имеет определенный прототип, интерпретация которого известна агенту. Таким образом, агент использует онтологию как определенную программу своих действий. Вершинами онтологии данного типа могут являться понятия из онтологии предметной области.

Председателю комитета
социальной защиты
населения Иванову И.И.

заявление.

Прошу предоставить бесплатную путевку для оздоровления сына, Сидорова Ильи.

10.02.2008 **Сидорова А.А.**

а)

б)

Рис. 6. Пример простого документа «Заявление» (а) и онтологии, описывающей класс документов «Заявление» (б)

Третий тип онтологий используется для *описания структуры и содержания документов* (рис. 6). Этот тип онтологий включает в себя два класса (две плоскости) вершин (рис. 7). К первому классу относятся вершины, описывающие *структуру документа*. Например: таблица,

дата, должность и т.д. (они представляют собой общие понятия, не зависящие от конкретной предметной области). Другим типом будут являться вершины, содержащие *понятия документа*. Первый тип вершин будем называть *структурными вершинами*, второй тип – *семантические вершины*.

ПРИКАЗ

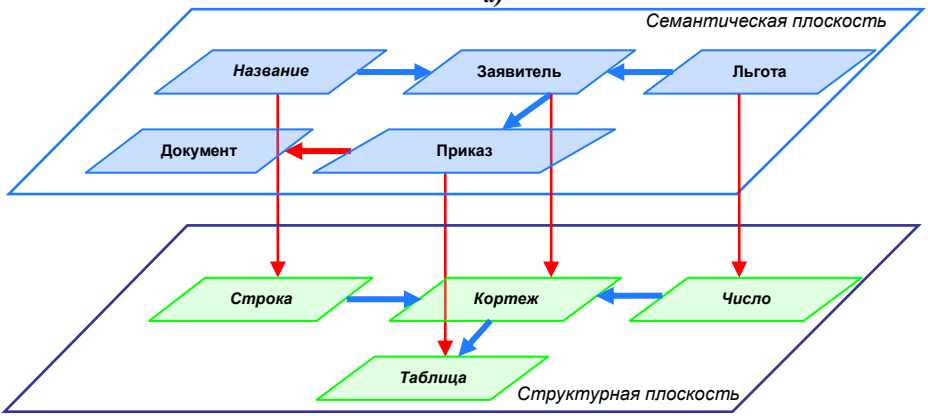
№ 17 **от 01.11.2009**

Установить на 2010 г. следующие льготы по оплате путевок (в процентах от стоимости):

Сидорова А.А.	**100**
Мухин В.В.	**80**
Кашкин П.О.	**50**
Сирык В.Л.	**25**

Председатель комитета **И.И. Иванов**

а)



б)

Рис. 7. Пример документа «Приказ» (а) и разбиение вершин онтологии для документа на две плоскости (б)

С использованием предлагаемого подхода из документа можно получить необходимые для решения задач, связанных с ведением административных регламентов, данные: известно, где искать данные и как они могут быть интерпретированы.

Онтологии располагаются на трех уровнях репозитария (рис. 8). На первом уровне расположены онтологии описывающие объекты, используемые в конкретной системе и учитывающие ее особенности. На втором уровне описываются объекты, инвариантные к предметной

области. Объекты третьего уровня описывают наиболее общие понятия и аксиомы, с помощью которых описываются объекты нижележащих уровней.

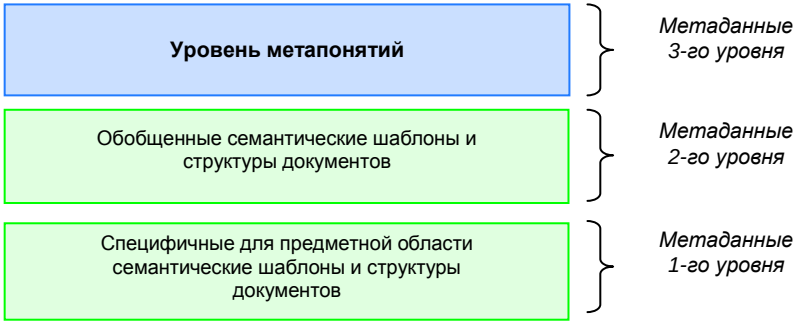


Рис. 8. Уровни метаданных

Для решения проблемы выделения общих понятий на основе формальных описаний предлагается агентный подход. Описанные средства являются основой для решения описанных ниже задач, автоматизация которых позволит значительно снизить трудоемкость ведения ЭАР.

Автоматическое реферирование, классификация и каталогизация документов

Функция *автоматического реферирования* является необходимой для разрабатываемой системы. При работе с документами пользователю полезно бывает предоставить аннотацию документа, по которой он сможет принять решение о значимости данного документа для решения стоящих перед ним задач. На данный момент для автоматического реферирования применяются в основном два подхода. Традиционный подход (*квазиреферирование*) основан на выделении и выборе фрагментов текста из исходного документа и соединении их в короткий текст. Подход, *основанный на знаниях*, предполагает подготовку краткого изложения и передачу основной мысли текста (пересказ).

Основное преимущество квазиреферирования заключается в простоте реализации. Однако выделение фрагментов текста, не учитывающее взаимоотношений между ними, часто приводит к формированию «бессвязных» текстов, в реферате могут встретится «висящие» слова или словосочетания. Второй метод базируется на онтологических справочниках, отражающих понятия «здравого смысла» и термины

предметной области документа. Очевидно, что в системе возможна реализация второго подхода, т.к. в системе уже присутствуют онтологические знания. В результате процесса анализа документа его содержание структурируется на основе онтологии. Таким образом, необходимо только реализовать лингвистический компонент, формирующий связный реферат.

Задача *автоматической классификации и каталогизации документов* является задачей разбиения поступающего потока текстов на тематические подпотоки в соответствии заранее заданными рубриками. Как и для автоматического реферирования, существует два противоположных подхода к каталогизации. Наиболее эффективными, но сложными в реализации, являются методы, основанные на знаниях. При каталогизации текстов на основе знаний используются заранее сформированные базы знаний, в которых описываются языковые выражения, соответствующие той или иной рубрике, и правила выбора рубрик. Другим классом методов для автоматической рубрикации текстов являются методы машинного обучения, которые в качестве обучающих примеров могут использовать заранее отрубрицированные вручную тексты.

Процесс построения системы документов

Одно из применений разрабатываемой системы − поиск зависимостей и *установление связей между документами*, регламентирующими деятельность различных учреждений, их сотрудников, оказания услуг.

В результате выполнения анализа должна быть построена система взаимосвязанных документов:

- относящихся к определенным направлениям деятельности (к определенным понятиям, объектам предметной области);
- отражающих связи между этими понятиями (с каждым понятием может быть связан документ или совокупность документов, связи между документами отражают связи между понятиями);
- содержащих нормативную информацию, которая также может быть выделена на основе анализа содержания документов.

На основе построенной системы взаимосвязанных документов можно частично автоматизировать процесс анализа изменений предметной области и внесения изменений в модель предметной области ЭАР (т.е. реализовать поддержку процесса разработки и адаптации ЭАР). Таким образом, система управления документами становится не только частью ЭАР, позволяющей получать результаты обработки

данных, хранящихся в БД, в удобной для пользователей форме, но и становится основой средств поддержки разработки ЭАР.

Заключение

Центральным понятием разрабатываемой системы является концептуальная модель регламента, которая строится на основе анализа различных типов документов с использованием онтологий. Она позволяет получить различные представления административных регламентов (текстовое, графическое, на языке XPDL). Наличие репозитария онтологий делает возможным верификацию регламентов на всех этапах работы с ними. Создание структурированных представлений АР позволяет осуществлять анализ регламентов с целью их оптимизации. Описание АР на XPDL может быть передано для реализации ЭАР во внешние системы, позволяет интегрировать данную систему с различными средствами разработки ЭАР.

Разрабатываемый инструментарий поддерживает все этапы жизненного цикла административных регламентов. Его применение должно быть особенно эффективным при массовой работе над электронными административными регламентами, при создании средств их ведения специалистами органов власти, отвечающими за реализацию услуг.

Библиографический список

1. *Буряга В.О.* Административный регламент в сфере реализации исполнительной власти в российской федерации [Текст] : Автореферат диссертации на соискание ученой степени кандидата юридических наук : 25.11.2009 / Буряга Владимир Олегович. – М., 2009. 24 с.
2. Выполнение работ по подготовке методической и нормативной базы для оказания услуг в формате МФЦ: Отчет о НИР (итоговый) / Некоммерческое партнерство «Центр развития малого бизнеса, образования и международных связей «Сократ» (НП «Центр «Сократ») : рук. Н.Ю. Свиридов; Госконтракт 84-ЭА-09. Липецк, 2010. 71 с.
3. *Дрожжинов В*. Исследование: «Новые бизнес модели для электронного правительства в Российской Федерации» [Электронный ресурс]. – М.: EuropeAid/126204/SER/RU Проект G2C по поддержке э-правительства Российской Федерации. Компонент 2: Политические, законодательные и нормативно-правовые рамки. Активность 2.2: Подготовить аналитические заметки по

ключевым стратегическим вопросам о развитии э-правительства в России, 2010. – [Режим доступа: свободный, http://open-gov.ru/wp-content/files/Drozhzhinov.pdf]. Проверено: 29.04.2011.

4. *Загоруйко А.Е.* Электронные административные регламенты. Принципы и аспекты реализации в документационном обеспечении управления [Электронный ресурс]. М.: ГИС-Ассоциация, 2007. – [Режим доступа: свободный, http://www.gisa.ru/40399.html]. Проверено: 29.04.2011.
5. *Ключко Н.В.* Электронные административные регламенты в условиях электронного правительства [Электронный ресурс]. – [Режим доступа: свободный, www.ifap.ru/pr/2009/090326aa.rtf]. Проверено: 29.04.2011.
6. *Ланин В.В.* Использование многоуровневого репозирария онтологий для анализа электронных документов // Труды международных научно-технических конференций «Интеллектуальные системы» (AIS'08) и «Интеллектуальные САПР» (CAD-2008). Научное издание в 4-х томах. Т. 1. – М.: Физматлит, 2008. С. 202-206.
7. *Ланин В.В.* Мультиагентая система для интеллектуального анализа документов // International Book Series Journal "Information Science & Computing" / Sofia (Bulgaria) – Number 4, 2008. С. 166-172.
8. *Ланин В.В.* Построение системы взаимосвязанных документов на основе средств интеллектуального анализа // Труды Конгресса по интеллектуальным системам и информационным технологиям (AIS-IT'09) / М.: Физматлит, 2009, Т.1. С. 426-431.
9. *Лозинов М.* Автоматизированная межведомственная информационная система для оказания государственных и муниципальных услуг в электронном виде [Электронный ресурс]. – Тюмень: Систематика, 2010. – [Режим доступа: свободный, http://admtyumen.ru/files/upload/2010-09-06.pdf]. Проверено: 29.04.2011.
10. *Лукашевич Н.В., Добров Б.В.* Двуязычный информационный поиск на основе автоматического концептуального индексирования // Компьютерная лингвистика и интеллектуальные технологии. Труды Международной конференции Диалог-2003 / М.: Наука, 2003. С. 425-432.
11. Моделирование деятельности органов власти, государственных и муниципальных организаций: Отчет о НИР (итоговый) / АНО КМЦ «Бизнес-инжиниринг» : рук. Л.Ю. Григорьев; Москва, 2008.

12. Разработка концепции и методики построения электронных регламентов взаимодействия федерального министерства с агентствами и службами, а также центрального аппарата министерства с территориальными органами: Отчет о НИР (заключительный). Часть 1 / ООО «ФОРС-Центр разработки» : рук. Г.Ф. Свердлов; № темы 5. Москва, 2004. 497 с.
13. Разработка методических рекомендаций по описанию и оптимизации процессов в органах исполнительной власти в рамках подготовки внедрения ЭАР: Отчет о НИР (промежуточный). Часть 1 / Институт проблем государственного и муниципального управления Государственного университета – Высшей школы экономики : рук. А.В. Клименко; № темы 4. М., 2004. 193 с.
14. *Сенченко П.В., Павинич Е.С.* Web-ориентированные информационные технологии поддержки ведения электронного регламента деятельности органов местного самоуправления // Доклады ТУСУР. – 2008. – № 2 (18), часть 2. – С. 91-94.
15. *Шульгин А.О.* Моделирование административных регламентов государственных и региональных информационных систем // Вестник Ставропольского государственного университета. – Ставрополь: Изд-во СГУ, 2009. № 63(4). С. 164-169.

А.О. Сухов*

Пермский государственный университет

Sukhov_PSU@mail.ru

ОПИСАНИЕ МНОГОУРОВНЕВОЙ МОДЕЛИ ПРЕДМЕТНОЙ ОБЛАСТИ С ПОМОЩЬЮ ПСЕВДО-МЕТАГРАФОВ

При разработке информационных систем (ИС) широко используются технологии, основанные на применении метамоделирования и предметно-ориентированных языков [2]. Система MetaLanguage представляет собой инструментарий для создания визуальных динамически настраиваемых предметно-ориентированных языков моделирования, используемых при разработке ИС. Для описания метамоделей – моделей предметно-ориентированных языков – MetaLanguage использует метаязык, базовыми конструкциями которого являются сущность, отношение, ограничение [3].

Используя конструкции сущность и отношение можно построить любую модель, в том числе и невалидную в рассматриваемой предметной области. Для задания формальных правил построения моделей воспользуемся графовыми грамматиками. В [4] приведен обзор различных средств формального описания синтаксиса визуальных языков моделирования. В результате обзора было определено, что наиболее подходящим формализмом, учитывающим назначение системы MetaLanguage, являются формальные грамматики, представленные ориентированными псевдо-метаграфами. Дадим определение метамодели и модели предметной области, применяя выбранный инструментарий, а также построим прямое и обратное отображение графа метамодели на граф модели.

Граф метамодели

Обозначим множество сущностей метамодели через $Set = \{set_i\}, i \in \mathrm{N}, i < \infty$ (N – множество натуральных чисел), причем в каждый фиксированный момент времени их число конечно, но при

* Работа выполнена при поддержке РФФИ (проект № 10-01-00794)

создании/удалении сущности множество расширяется/сокращается.

Каждая сущность метамодели

$set_i = \{SName_i, SICount_i, Attr_i, Opp_i, SRest_i, SUnique_i\}$

характеризуется

- именем $SName_i$, однозначно идентифицирующим сущность в рамках метамодели;
- количеством экземпляров сущности $SICount_i$, которое может быть создано при построении модели;
- набором атрибутов сущности $Attr_i = \{attr_{j_i}\}, j_i \in N, j_i < \infty$;
- набором операций над сущностью $Opp_i = \{opp_{j_i}\}, j_i \in N, j_i < \infty$;
- набором ограничений, налагаемых на сущность, $SRest_i = \{srest_{j_i}\}, j_i \in N, j_i < \infty$;
- флагом уникальности $SUnique_i$, определяющим пределы уникальности имен экземпляров сущности.

Множества $Attr_i$, Opp_i, $SRest_i$ являются конечными в каждый фиксированный момент времени.

Множество всех характеристик i-ой сущности разделим на две группы SG_i^1 и SG_i^2. К первой группе отнесем те характеристики, которые в графовой модели будут представлены отдельными вершинами: набор атрибутов, операций и ограничений, налагаемых на сущность, т.е.

$$SG_i^1 = \{Attr_i, Opp_i, SRest_i\}.$$

Характеристики второй группы $SG_i^2 = \{SName_i, SICount_i, SUnique_i\}$ (имя сущности, количество экземпляров, флаг уникальности) будут приписываться непосредственно вершине, соответствующей сущности.

Обозначим множество отношений метамодели $Rel = \{rel_i\}, i \in N, i < \infty$, причем в каждый фиксированный момент времени их число конечно, но при создании/удалении отношения число элементов множества увеличивается/уменьшается.

Любое отношение метамодели

$$rel_i = \{RName_i, RType_i, RMult_i, RRest_i, RUnique_i\}$$

характеризуется

- именем $RName_i$, которое однозначно задает отношение в рамках данной метамодели;
- типом $RType_i$, определяющим семантику отношения;

– кратностью $RMult_i$, которая указывает, сколько экземпляров сущности может участвовать в отношении;
– множеством ограничений, налагаемых на отношение, $RRest_i = \{rrest_{j_i}\}, j_i \in N, j_i < \infty$, причем множество $RRest_i$ является конечным в каждый фиксированный момент времени;
– флагом уникальности $RUnique_i$, определяющим пределы уникальности имен экземпляров отношений.

Множество характеристик i-ого отношения разобьем на две группы RG_i^1 и RG_i^2.

К первой группе отнесем множество ограничений, налагаемых на отношение.

Во вторую группу включим такие характеристики отношения как: «имя», «тип», «кратность», «флаг уникальности», т.е.

$$RG_i^1 = \{RRest_i\},$$
$$RG_i^2 = \{RName_i, RType_i, RMult_i, RUnique_i\}.$$

Рассмотрим ориентированный псевдо-метаграф $GMM = (V,E)$. Представим множество вершин графа в виде объединения шести попарно непересекающихся подмножеств

$$V = Set \bigcup_{i=1}^{|Set|} Attr_i \bigcup_{i=1}^{|Set|} Opp_i \bigcup_{i=1}^{|Set|} SRest_i \bigcup Rel \bigcup_{i=1}^{|Rel|} RRest_i . \qquad (1)$$

Множество всех дуг псевдо-метаграфа E разделим на пять попарно непересекающихся подмножеств:

– $ESA = \{esa_i\}, \overline{i = 1, |Set|}$ – множество дуг, соединяющих каждую сущность метамодели с множеством атрибутов, ей соответствующих;
– $ESO = \{eso_i\}, \overline{i = 1, |Set|}$ – множество дуг, соединяющих каждую сущность метамодели с множеством операций над ней;
– $ESR = \{esr_i\}, \overline{i = 1, |Set|}$ – множество дуг, соединяющих каждую сущность метамодели с множеством ограничений, налагаемых на нее;
– $ERR = \{err_i\}, \overline{i = 1, |Rel|}$ – множество дуг, соединяющих каждое отношение метамодели с множеством ограничений, налагаемых на него;
– $ESRR = \{esrr_i\}, i \in N, i < \infty$ – множество дуг, соответствующих связям между сущностями и отношениями, причем в каждый

фиксированный момент времени их число конечно, но при создании/удалении сущности (отношения) множество расширяется/сокращается.

Таким образом, получили, что

$$E = ESA \cup ESO \cup ESR \cup ERR \cup ESRR. \qquad (2)$$

Определение 1. Граф метамодели MM – это ориентированный псевдо-метаграф $GMM = (V,E)$, для которого выполняется (1) и (2), где V – это непустое множество вершин графа, E – это множество дуг графа.

Рассмотрим пример. Построим граф метамодели для сущности «Прецедент» диаграмм вариантов использования языка UML [1]. Метамодель для данного вида диаграмм представлена на рис. 12. Атрибутами сущности «Прецедент» являются «Имя», «Описание», «Дата создания». Операции, которые можно выполнять над сущностью – «Изменить имя», «Задать описание», «Изменить дату создания», т.е. для рассматриваемой сущности

$$Attr_i = \{\text{«}Имя\text{»}, \text{«}Описание\text{»}, \text{«}Дата\ создания\text{»}\},$$

$$Opp_i = \{\text{«}Изменить\ имя\text{»}, \text{«}Задать\ описание\text{»},$$
$$\text{«}Изменить\ дату\ создания\text{»}\},$$

$$SRest_i = \varnothing.$$

Граф, соответствующий фрагменту метамодели диаграмм прецедентов, представлен на рис. 1.

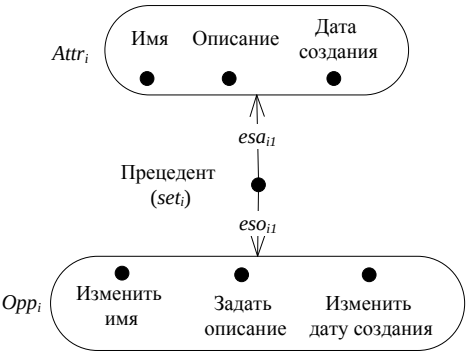


Рис. 1. Фрагмент графа метамодели для сущности «Прецедент»

Как видно из рисунка, $ESA = \{esa_{i_1}\}$, $ESO = \{eso_{i_1}\}$, $ESR = \varnothing$, $ERR = \varnothing$, $ESRR = \varnothing$.

Граф модели

Модель – это фактически «*экземпляр*» *метамодели*, в котором:

- атрибуты сущности – это конкретные значения;
- отсутствуют операции над экземплярами сущностей и ограничения, налагаемые на экземпляры сущностей и отношений;
- экземпляры отношения наследования создать нельзя.

Обозначим множество всех моделей, которые были созданы на основе текущей метамодели, через $M = \{m_k\}, k \in N, k < \infty$, причем в каждый фиксированный момент времени их число конечно, но при создании/удалении модели число элементов множества увеличивается/уменьшается.

Введем следующие обозначения:

- $SetI_i$ – множество экземпляров *i*-ой сущности;
- $AttrI_{j_i}$ – множество значений атрибутов *j*-ого экземпляра *i*-ой сущности;
- $RelI_k$ – множество экземпляров *k*-ого отношения.

Множества $SetI_i$, $AttrI_{j_i}$, $RelI_k$ являются конечными в каждый фиксированный момент времени, но при создании/удалении экземпляра сущности (отношения) множества расширяются/сокращаются.

Рассмотрим ориентированный псевдо-метаграф $GM = (VI, EI)$. Множество вершин графа представим в виде трех попарно непересекающихся подмножеств:

$$VI = \bigcup_{i=1}^{|Set|} \left(SetI_i \bigcup_{j=1}^{|Attr_i|} AttrI_{j_i} \right) \bigcup_{k=1}^{|Rel|} Rel_k \ . \qquad (3)$$

Рассмотрим пример. Создадим экземпляр описанной ранее сущности «Прецедент» (см. рис. 2).

Как видно из рисунка, $AttrI_i$ = {«*Сдать экзамен*», «*Прецедент описывает процесс сдачи экзамена*», «21/06/09»}.

Множество EI разделим на два попарно непересекающихся подмножества:

- $ESAI = \{esai_i\}, \overline{i = 1, |SetI|}$ – множество дуг, соединяющих каждый экземпляр сущности с множеством атрибутов, ему соответствующих;
- $ESRRI = \{esrri_i\}, i \in N, i < \infty$ – множество дуг, соответствующих связям между экземплярами сущностей и экземплярами отношений, причем в каждый фиксированный момент времени

число элементов множества конечно, но при создании/удалении экземпляра сущности (отношения) число элементов увеличивается/уменьшается.

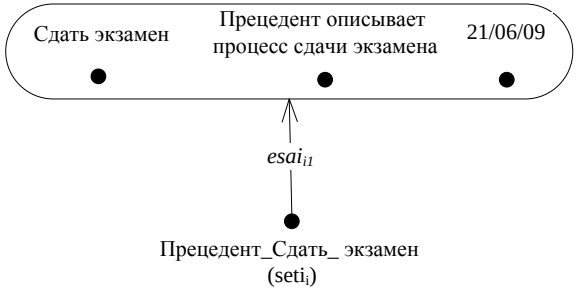


Рис. 2. Фрагмент графа модели, соответствующий экземпляру сущности «Прецедент»

Таким образом, получили, что

$$EI = ESAI \bigcup ESRRI \,. \qquad (4)$$

Из рис. 2 видно, что для представленного экземпляра сущности «Прецедент»

$$ESAI = \{esai_{i_1}\},$$

$$ESRRI = \varnothing.$$

Определение 2. Граф модели – это ориентированный псевдометаграф $GM = (VI, EI)$, для которого выполняется (3) и (4), где VI – это непустое множество вершин графа, EI – это множество дуг графа.

Операция создания графа модели

Построим отображение графа метамодели на граф модели, ему соответствует операция создания графа модели. Такое отображение позволит поддерживать модели в актуальном состоянии, поскольку при модификации метамодели во все модели, созданные на ее основе, будут внесены необходимые изменения.

Введем следующие обозначения:

– $SetI = \bigcup_{i=1}^{|Set|} SetI_i$ – множество вершин графа модели, соответствующих экземплярам всех сущностей метамодели;

– $RelI = \bigcup_{i=1}^{|Rel|} RelI_i$ – множество вершин графа модели, соответствующих экземплярам всех отношений метамодели;

– $AttrI = \bigcup_{i=1}^{|Set|} \bigcup_{j_i=1}^{|SetI|} AttrI_{j_i}$ – множество вершин графа модели, соответствующих значениям атрибутов всех экземпляров всех сущностей модели.

Множества $SetI$, $RelI$, $AttrI$ являются конечными в каждый фиксированный момент времени, но при создании/удалении экземпляра сущности, его атрибута или экземпляра отношения множества расширяются/сокращаются.

Построим отображение $fs:Set \to SetI$, которое для каждой вершины-сущности графа метамодели, определяет множество вершин графа модели, соответствующих экземплярам этой сущности, т.е.

– $(\exists set_i \in Set)(\exists seti_{j_i} \in SetI):fs(set_i) = seti_{j_i}$, если сущность не является абстрактной и имеет экземпляры;

– $(\exists set_i \in Set):fs(set_i) = \varnothing$, если сущность является абстрактной или не имеет экземпляров.

Отображение fs задает *операцию создания вершины, соответствующей экземпляру сущности*.

Построим отображение множества вершин $Attr$ графа метамодели, соответствующих атрибутам сущностей, на множество вершин графа модели $AttrI$:

$$fa:Attr \to AttrI .$$

Тогда

$$(\forall a_{j_i} \in Attr)(\exists ai_{k_{j_i}} \in AttrI):fa(a_{j_i}) = ai_{k_{j_i}}, i = \overline{1,|Set|},$$

$$j_i = \overline{1,|SetI_i|}, k_{j_i} = \overline{1,|AttrI_{j_i}|} .$$

Отображению fa соответствует *операция задания атрибутов экземпляра сущности*.

Рассмотрим множество вершин графа метамодели, которые соответствуют отношениям. Каждой такой вершине поставим в соответствие множество вершин графа модели, которое соответствует экземплярам определенного отношения, в результате получим отображение

$fr{:}Rel \to RelI$, такое, что выполняются условия

- $(\exists rel_i \in Rel)(\exists reli_{j_i} \in RelI){:}fr(rel_i) = reli_{j_i}$, если отношение имеет экземпляры;
- $(\exists rel_i \in Rel){:}fr(rel_i) = \varnothing$, если отношение экземпляров не имеет.

Данное отображение определяет *операцию создания вершины, соответствующей экземпляру отношения*.

Таким образом, три отображения fs, fa, fr задают соответствие между множеством вершин графа метамодели и множеством вершин графа модели (см. рис. 3).

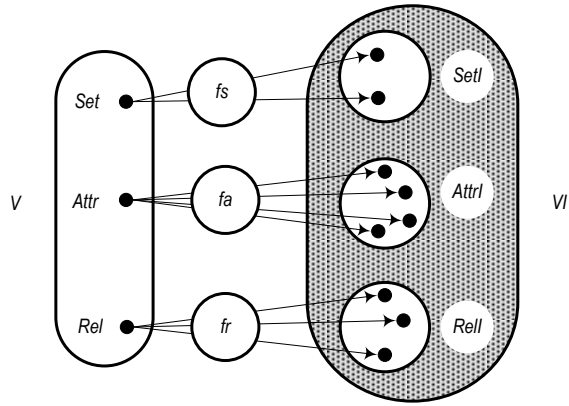


Рис. 3. Отображение множества вершин графа метамодели на множество вершин графа модели

Теперь определим правила, согласно которым дугам графа GMM ставятся в соответствие дуги графа GM .

Построим отображение $ga{:}ESA \to ESAI$, ставящее каждой дуге множества ESA графа метамодели в соответствие определенные дуги из множества $ESAI$ графа модели, т.е.

$$(\forall esa_{j_i} \in ESA)(\exists esai_{k_{j_i}} \in ESAI){:}ga(esa_{j_i}) = esai_{k_{j_i}}, i = \overline{1,|Set|},$$

$$j_i = \overline{1,|SetI_i|}, k_{j_i} = \overline{1,|AttrI_{j_i}|}.$$

Построим отображение $gsr:ESRR \to ESRRI$, ставящее каждой дуге множества $ESRR$ графа метамодели в соответствие определенные дуги из множества $ESRRI$ графа модели, т.е.

$$(\forall esrr_i \in ESRR)(\exists\, esrri_{k_{j_i}} \in ESRRI): gsr(esrri_{k_{j_i}}) = esrr_i,$$

$$i = \overline{1,|Set|},\ j_i = \overline{1,|SetI_i|}, k_{j_i} = \overline{1,|AttrI_{j_i}|}.$$

Таким образом, отображения ga,gsr задают соответствие между множеством дуг графа метамодели и множеством дуг графа модели.

Определение 3. Создание графа модели – это отображение графа метамодели на граф модели, при котором выполняются преобразования fs,fa,fr,ga,gsr.

Операция интерпретации модели

Построим отображение графа модели на граф метамодели. Оно определяет операцию интерпретации модели, что позволяет выполнять операции над экземплярами сущностей и проверять ограничения, наложенные на сущности и отношения.

Поскольку вершины графа модели являются экземплярами вершин графа метамодели, то можно задать отображение множества вершин графа GM на множество вершин графа GMM.

Построим сюръекцию $fs^{-1}:SetI \to Set$, которая каждому экземпляру сущности модели ставит в соответствие сущность метамодели

$$(\forall seti_{j_i} \in SetI)(\exists !\, set_i \in Set): fs^{-1}(seti_{j_i}) = set_i,\ i = \overline{1,|Set|},\ j_i = \overline{1,|SetI_i|},$$

причем несколько элементов множества $SetI$ могут соответствовать одной сущности, т.е. выполняется

$$(\forall set_i \in Set)(\exists\, seti_{j_i}, seti_{k_i} \in SetI, seti_{j_i} \neq seti_{k_i}):$$

$$fs^{-1}(seti_{j_i}) = fs^{-1}(seti_{k_i}) = set_i.$$

Построим отображение обратное отображению fa:

$$fa^{-1}:AttrI \to Attr.$$

Такая сюръекция каждой вершине множества $AttrI$ ставит в соответствие единственную вершину множества $Attr$, т.е.

$$(\forall ai_{k_{j_i}} \in AttrI)(\exists! a_{j_i} \in Attr): fa^{-1}(ai_{k_{j_i}}) = a_{j_i}, i = \overline{1,|Set|},$$

$$j_i = \overline{1,|SetI_i|}, k_{j_i} = \overline{1,|AttrI_{j_i}|},$$

причем несколько элементов множества $AttrI$ могут соответствовать одному элементу множества $Attr$, т.е. выполняется

$$(\forall a_{j_i} \in Attr)(\exists\, ai_{k_{j_i}}, ai_{l_{j_i}} \in AttrI, ai_{k_{j_i}} \neq ai_{l_{j_i}}):$$

$$fa^{-1}(ai_{k_{j_i}}) = fa^{-1}(ai_{l_{j_i}}) = a_{j_i}.$$

Рассмотрим множество вершин графа модели, которые соответствуют экземплярам отношений. Каждой такой вершине поставим в соответствие единственную вершину графа метамодели, которая соответствует заданному отношению, в результате получим сюръективное отображение $fr^{-1}: RelI \to Rel$, такое, что выполняется

$$(\forall reli_{j_i} \in RelI)(\exists\, !rel_i \in Rel): fr^{-1}(reli_{j_i}) = rel_i,$$

$$i = \overline{1, |Rel|},\ j_i = \overline{1, |RelI_i|},$$

причем несколько экземпляров отношения могут быть созданы на основе одного отношения, т.е. выполняется

$$(\forall rel_i \in Rel)(\exists\, reli_{j_i}, reli_{k_i} \in RelI, reli_{j_i} \neq reli_{k_i}):$$

$$fr^{-1}(reli_{j_i}) = fr^{-1}(reli_{k_i}) = rel_i.$$

Таким образом, три отображения fs^{-1}, fa^{-1}, fr^{-1} задают соответствие между множеством вершин графа модели и множеством вершин графа метамодели (см. рис. 4).

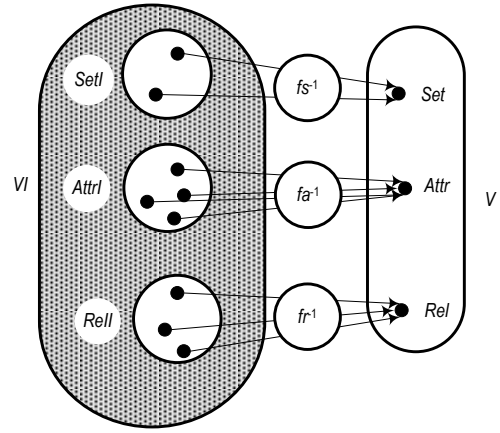


Рис. 4. Отображение множества вершин графа модели на множество вершин графа метамодели

Поскольку операции над экземплярами сущностей и отношений не определены, то для перемещения между сущностями, отношениями и их экземплярами расширим множество дуг графа модели дугами-ссылками, соединяющими экземпляры сущностей и отношений с теми сущностями и отношениями метамодели, на основе которых они созданы. Обозначим множество таких дуг через T:

$$T = \bigcup_{i=1}^{|Set|+|Rel|} T_i, T_i = \{t_{j_i}\}, j = \overline{1, |SetI_i| + |RelI_i|}.$$

Теперь определим правила, согласно которым дугам графа модели GM ставятся в соответствие дуги графа метамодели GMM.

Построим отображение $ga^{-1}: ESAI \to ESA$, ставящее каждой дуге множества $ESAI$ графа модели в соответствие единственную дугу из множества ESA графа метамодели, т.е.

$$(\forall esai_{k_{j_i}} \in ESAI)(\exists !esa_{j_i} \in ESA): ga^{-1}(esai_{k_{j_i}}) = esa_{j_i}, i = \overline{1, |Set|},$$

$$j_i = \overline{1, |SetI_i|}, k_{j_i} = \overline{1, |AttrI_{j_i}|}.$$

Как видно из определения, это отображение – сюръекция.

Построим сюръективное отображение $gsr^{-1}: ESRRI \to ESRR$, ставящее каждой дуге множества $ESRRI$ графа модели в соответствие единственную дугу из множества $ESRR$ графа метамодели, т.е.

$$(\forall esrri_{j_i} \in ESRRI)(\exists !esrr_i \in ESRR): gsr^{-1}(esrri_{j_i}) = esrr_i,$$

$$i = \overline{1, |ESRR|}, j_i = \overline{1, |ESRRI_i|}.$$

Таким образом, отображения ga^{-1}, gsr^{-1} определяют однозначное соответствие между множеством ребер графа модели и множеством ребер графа метамодели.

Определение 4. Интерпретация модели – это отображение графа модели на граф метамодели, при котором выполняются преобразования $fs^{-1}, fa^{-1}, fr^{-1}, ga^{-1}, gsr^{-1}$.

Заключение

Таким образом, была построена формальная метамодель и модель предметной области с помощью графовых грамматик, представленных ориентированными псевдо-метаграфами.

Использование формальной математической модели метаязыка системы MetaLanguage позволяет описать его свойства, разработать алгоритмы горизонтальной и вертикальной трансформации метамоделей и созданных на их основе моделей ИС, их предметных областей.

Библиографический список

1. *Буч Г., Рамбо Д., Джекобсон А.* Язык UML. Руководство пользователя, 2-е издание.: пер. с англ. М.: Издательство «ДМК Пресс», 2007. – 496 с.
2. *Лядова Л.Н.* Многоуровневые модели и языки DSL как основа создания интеллектуальных CASE-систем // Сборник трудов Третьей международной научно-технической конференции «Инфокоммуникационные технологии в науке, производстве и образовании» (Инфоком-3): Часть 2 / Ставрополь, 2008. С. 65-71.
3. *Лядова Л.Н., Сухов А.О.* Языковой инструментарий системы MetaLanguage // Математика программных систем: межвуз. сб. научн. тр. / Перм. ун-т. Пермь, 2008. С. 40-51.
4. *Сухов А.О., Лядова Л.Н.* Использование графов для описания синтаксиса визуальных языков моделирования // Технологии Microsoft в теории и практике программирования / Материалы межвуз. конкурса-конференции. Политехнический ун-т. СПб, 2010. С. 172.

И.В. Хмельницкая, В.В. Мякинькая

Белорусский государственный экономический университет

Inna21@gmail.com, violettasuper@gmail.com

ИННОВАЦИОННЫЕ ПОДХОДЫ К ОЦЕНКЕ КАЧЕСТВА СИСТЕМЫ СБОРА И ОБРАБОТКИ ИНФОРМАЦИИ: АУДИТОРСКИЙ ПОДХОД

Введение

В условиях инновационной экономики повышается роль и значение управленческой составляющей функционирования предприятий, основанной на анализе поступающей на предприятие информации из внутренней и внешней среды и осуществлении мониторинга развития предприятия. Определяющее значение в общей совокупности экономической информации имеют данные учета (бухгалтерского, управленческого, налогового). Для превращения учетной информации в решение используются знания аналитика, который с помощью моделей и алгоритмом проводит ее аналитическую обработку. Гарантированное качество работы достигается путем стандартизации деятельности организации с целью производства продукции (предоставления услуги, работ), гарантированно отвечающей заданным требованиям, а также совершенствование этой продукции (услуги, работ).

Инструментом стандартизации является *компьютерная система менеджмента качества* (СМК), одним из основных принципов которой является *принятие решений на основании качественной информации*. Первая предпосылка для использования информационных технологий при разработке и внедрении такой СМК – сокращение временных затрат, трудозатрат на внедрение СМК. Вторая предпосылка для использования информационных технологий – в необходимости оптимально управлять большим объемом документации, обеспечивать эффективное ознакомление сотрудников с ней, организовывать удобный доступ и поиск необходимых документов. Такое решение представляет собой *универсальный инструмент по автоматизации боль-*

шинства процессов СМК предприятия, как на основе уже разработанных шаблонов процессов, так и с использованием подсистемы моделирования процессов, в том числе и процесса сбора и аналитической обработки информации. Однако для самостоятельного моделирования таких процессов необходима методика оценки достоверности данных.

Контроль достоверности информации в системе менеджмента качества

Достичь высокого уровня достоверности обрабатываемой информации в системе возможно как технологическими (при разработке программного обеспечения), так и организационными (последующий контроль вводимой информации) методами.

При разработке программных средств, считается, что информация, вносимая в систему пользователями либо получаемая от внешних источников, верна априори. В условиях сжатых сроков разработки данное предположение часто находит одобрение как у рядовых исполнителей, так и архитекторов систем. Программистов в первую очередь волнует корректность типов данных и соответствие программного обеспечения требованиям, указанным в техническом задании. Они озабочены тем, чтобы система удовлетворяла требования пользователей и успешно проходила тестирование. Исходя из этих соображений, и делаются проверки в коде. Зачастую контроль достоверности информации техническими средствами даже не предусматривается. Надежды возлагаются на организационные методы, т.е. на дополнительную проверку информации пользователями. Однако контроль вводимой информации независимым подразделением требует привлечения значительных человеческих ресурсов, что при больших объемах обрабатываемой информации затруднительно. Еще одна особенность такого подхода – значительное увеличение временных затрат на производственные процессы. Поэтому данный вид контроля часто сводится к формальным процедурам и ощутимой пользы не приносит [1, 2].

Таким образом, вопросам обеспечения должного уровня качества обрабатываемой информации в автоматизированных системах не всегда уделяется должное внимание. Анализ современных подходов к оценке качества информации позволил авторам выделить три подхода (табл. 1). При разработке методики оценки качества аналитической информации в условиях компьютерной обработки данных в организациях рекомендуется использовать все три подхода, что позволит оценить информацию наиболее полно, выявить скрытые закономерности, оценить количественно и качественно показатели качества автоматизированной обработки информации. Тем самым полученные знания

помогут выявить источники недоброкачественной информации, вырабатывать предложения по модернизации документооборота организации, а аналитикам получить достоверную информацию для анализа, что повысить ценность аналитических решений.

Таблица 1. Основные подходы к созданию методик оценки качества информации

Название	Сущность подхода
Математический подход	суждение об оценке, основанное на результатах статистического моделирования массовых данных или применение строго формализованных математических процедур, позволяющих свести к минимуму субъективные допущения математика
Аналитический подход	заключается в формировании строгих математических зависимостей между показателями, осуществляется по законам диалектики, формальной логики, с применением общенаучных методов исследования
Аудиторский подход	оценка информации и особенности ее искажения определяются аудитором на основании ее существенности

Рассмотрим сущность аудиторского подхода.

Аудиторский подход к созданию методик оценки качества информации

Независимое подтверждение достоверности информации о результатах деятельности предприятия и соблюдения им действующего законодательства необходимо как внешним так и внутренним пользователям информации, необходимой для принятия решений в области учета, анализа, с целью последующего управления предприятием. Действенным механизмом реализации данного подхода является аудит. Изучение научной литературы отечественных и зарубежных авторов показало, что имеется значительный накопленный опыт в трудах ученых экономистов, раскрывающих широкий спектр задач, стоящих перед аудиторами. В то же время методология и методика отбора и оценки качества данных аудитором до сих пор не разработаны в достаточной степени. Аудитор отбирает необходимые данные из различных сфер специальных знаний и источников аудиторских доказательств, принимает решение о достоверности данных для анализа с целью последующего управления зачастую интуитивно, опираясь на традиционные методы аудита, исходя из своих знаний и опыта, в условиях ограниченного времени, что придает процедуре ярко выраженный субъективный характер.

Функция аудитора сводится к оценке риска получения аналитиками искаженной информации с целью последующего выявления недоброкачественной информации и внесением в нее соответствующих корректировок.

Основное отличительное свойство предлагаемого решения – акцент на автоматизации выполнения процесса сбора и обработки информации аудитором, наличие универсального конструктора процесса и методических материалов, позволяющих самостоятельно разрабатывать собственные, авторские подходы к технологии работ в рамках СМК аудита, основные принципы построения которой предложены авторами [3].

В аудите под *достоверностью данных* следует понимать способ уменьшения или снятия неопределенности информации в отношении объектов учета и возможности принятия на основе такой информации адекватных управленческих решений. Получение достоверной информации о деятельности хозяйствующего субъекта невозможно без создания *системы менеджмента качества и количественной оценки влияния всех факторов*, влияющих на качество обрабатываемой информации. Формализация оценки качества информации позволит перевести категорию качество из разряда теоретических понятий в разряд практически достижимых качественных характеристик данных.

Аудиторская оценка информации должна включать оценку следующих групп факторов:

1. *Существенность информации*. Существенность характеризует уровень ценности информации для решения аналитической задачи. Существенность зависит от многих факторов, главным из которых является способность подробно проанализировать объект анализа.

2. *Надежность информации*. Автоматизированная система обработки информации состоит из аппаратной и программной частей. Аудитор при анализе группы факторов в части аппаратного обеспечения исследует тип и возраст применяемой вычислительной техники, количество рабочих мест, наличие сети, способ передачи данных, наличие в штате организации штатных работников, занимающихся техническим обслуживанием техники, уровень их квалификации и т.д. В части ПО аудитор должен исследовать тип применяемого ПО, наличие на нее лицензии, наличие договоров на техническое обслуживание фирмой-производителем и т.д.

3. *Безопасность данных*. Аудитор оценивает обеспечение в организации безопасность данных в системе обработки информации, разработанные основы разграничения прав доступа в системе, защиты от несанкционированного доступа.

4. *Достоверность*. Проверку достоверности данных аудитор должен начинать с оценки системы бухгалтерского учета и внутреннего контроля в части контроля подготовки данных, контроля предотвращения ошибок во время работы, возможности изменения ПО и методов первичной регистрации данных, наличие квалифицированного персонала для работы с электронными документами и т.д.

В соответствии с Правилом аудиторской деятельности «Аудиторские доказательства» [4] аудитор может получать аудиторские доказательства, в том числе и доказательства качественности информации путем выполнения ряда процедур. В табл. 2 приведены возможные способы получения аудиторских доказательств при оценке качества информации.

Таблица 2. Выбор возможных способов получения аудиторских доказательств в разрезе показателей качества аудируемой информации

Показатели качества информации	Инспектирование	Наблюдение	Запрос	Подтверждение	Пересчет	Аналитические процедуры
Существенность	+	+	+	+	+	+
Надежность	+	+	+	+	+	+
Безопасность данных	+	+	+	+	–	–
Достоверность	+	+	+	+	+	+

Оценку предложенных показателей предлагается производить методом количественной комплексной оценки [5], основные этапы которой представлены на рис. 1.

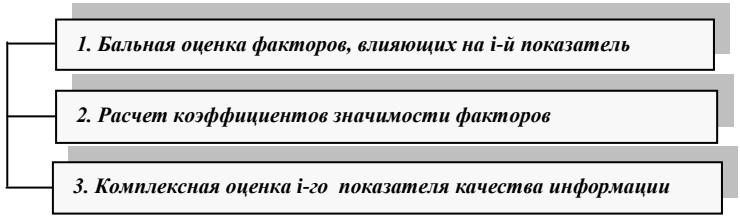


Рис. 1. Комплексная оценка качества информации (аудиторский подход)

Заключение

Задача обеспечения высокого качества информации в автоматизированной информационной системе представляется серьезной исследовательской и практической работой. Полученные разнообразными путями данные станут полезными лишь после того, как все они подвергнутся необходимому анализу и предельно точному истолкованию с использованием различных подходов. Предлагаемое решение направлено на автоматизацию сбора и обработки информации аудитором через использование универсального конструктора процесса и методических материалов, который позволяют аудитору самостоятельно разрабатывать подходы к технологии работ в рамках СМК аудита.

Библиографический список

1. *Кульба В.В.* Достоверность и сохранность информации в АСУ. Москва.: Синтег, 2003.
2. *Березкин Е.Ф.* Аппаратные средства функционального и тестового диагностирования. М.: МИФИ, 1986.
3. *Панков Д.А, Мякинькая В.В*. Система менеждмента качества аудита как базовая основа контроля качества аудита //Новые концепции развития бухгалтерского учета, анализа и контроля в условиях экономических изменений. Житомир: ЖДТУ. 2010.№ 9. С. 171-182.
4. http://www.minfin.gov.by/data/audit_pril/postmf26_10_00_114_2.pdf
5. *Хмельницкая И.В.* Модели формирования комплексных оценок финансово-хозяйственной деятельности предприятия // Бухгалтерский учет и анализ. Минск: БГЭУ. 2008.№ 6 (138). С. 23-30.

П.А. Мальцев*

Национальный исследовательский университет
«Высшая школа экономики» (Пермский филиал)

pavel_maltsev@mail.ru

ПРИМЕНЕНИЕ МОДЕЛИ «ФАКТ-СВЯЗЬ» К РЕШЕНИЮ ЗАДАЧИ СЕКВЕНЦИАЛЬНОГО АНАЛИЗА

Введение

Знания о причинно-следственных связях в некоторой предметной области оказываются неоценимыми при принятии решений в данной области. Основная проблема заключается в том, что подобные знания не всегда доступны в явном выражении – в виде правил и законов. Зачастую аналитику доступен лишь набор данных, так или иначе описывающих некоторые свершившиеся факты. Но сами по себе данные не позволяют судить о закономерностях без применения специализированных методов анализа данных, позволяющих выявить их. Здесь исследователю могут помочь методы Data Mining, а именно – методы поиска последовательных шаблонов (*секвенциальный анализ*).

На практике имеющиеся данные также не всегда полны, т.е. зачастую исследователю доступны данные не обо всех свершившихся или возможных событиях. Значит, выявленные закономерности не могут быть применены для любых событий. Другими словами мы не можем принять решение в связи с наступление события "A", пока у нас в базе данных не будет записи о прецеденте наступления события "A". Предлагается подход, позволяющий частично решить данную проблему путём обобщения извлечённых из данных знаний. Обобщение позволит построить абстрактную модель, которая описывала бы взаимосвязи между классами событий, а не между отдельными событиями.

В данной работе описана *модель «факт-связь»* лежащая в основе предлагаемого подхода.

* Работа выполнена при поддержке РФФИ (проект № 10-01-00794)

Понятие факта

Пусть нами наблюдается некоторая система S. Под способностью наблюдать систему S будет пониматься способность фиксировать значения некоторых её переменных: $f_1,f_2,\ldots,f_k$. Набор данных переменных – это всё, что имеет в своём распоряжении исследователь, наблюдающий за системой. Полагаем, что система S изменяется с течением времени, а вместе с ней меняются и значения её переменных. Таким образом, мы должны иметь возможность фиксировать отдельные состояния наблюдаемой системы. *Состояние системы* S в момент времени t будем обозначать $S(t)$. Ввиду того, что при наблюдении за системой в распоряжении исследователя имеется лишь ограниченный набор переменных, состояние системы мы будем описывать *вектором* их значений:

$$S(t) = (f_1^t, f_2^t, \ldots, f_k^t)$$

Это значит, что одно состояние отличается от другого значением хотя бы одной переменной. Таким образом, будем рассматривать систему S в динамике, т.е. будем рассматривать последовательность состояний системы S:

$$S(t_1) \to S(t_2) \to \cdots \to S(t_n)$$

На рис.1 представлен пример динамики состояния двухмерной системы.

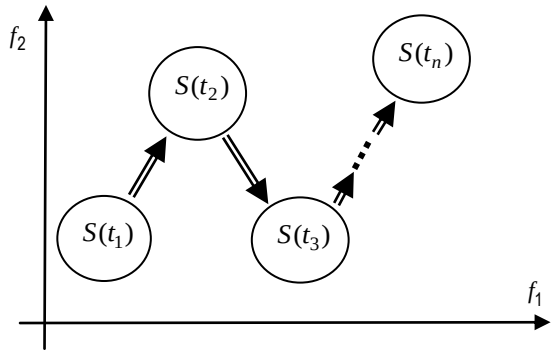


Рис. 1. Динамика двухмерной системы

Состояние двухмерной системы описывается значениями двух переменных; положим, что это f_1 и f_2. Будем считать, что система меняет своё состояние вследствие наступления некоторых событий

$\alpha,\beta,\gamma,\ldots$. В каждый момент времени в любой системе может происходить огромное количество событий, но не все события «полезны» для анализа. Для выделения значимых для анализа событий введём понятие факта. *Фактом* будем называть *значимое для анализа событие*. Другими словами, *фактами будем называть такие события, которые переводят наблюдаемую систему в новое состояние и данное изменение нами может быть обнаружено*.

Но что является причиной возникновения событий? Рассматривая любую систему, мы можем выделить несколько сил $F_1, F_2, \ldots, F_m$, которые действуют на систему изнутри или снаружи (рис. 2).

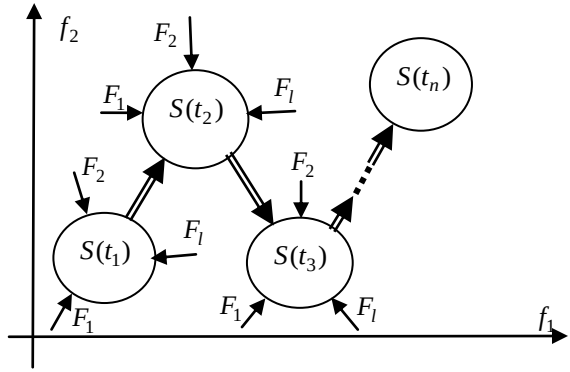


Рис. 2. Действие сил на систему

Каждая из сил, действуя на систему, стремиться перевести её в «желательное» для себя состояние (см. рис. 3).

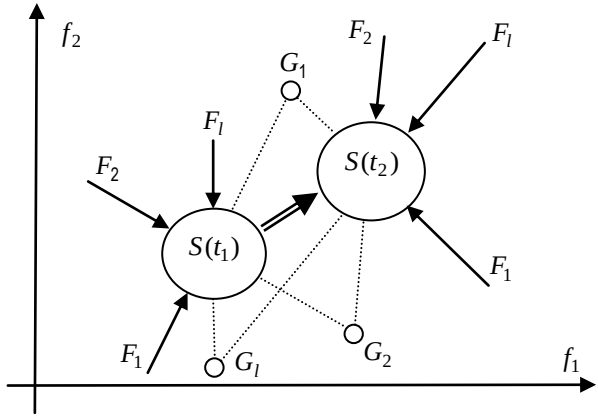


Рис. 3. Цели сил, действующих на систему

Таким образом, факт есть результат совместного действия на систему «заинтересованных» сил. Основная сложность при моделировании таких систем заключается в наличии большого числа неопределённостей.

Во-первых, нам не всегда известны все силы, действующие на систему. Например, моделируя учёбу студента, нам вполне ясно, что данный процесс определяется совместными усилиями самого студента и его преподавателей. Но мы можем не знать, что студент где-то подрабатывает или занимается спортом, что также оказывает влияние на его учёбу.

Во-вторых, силы не постоянны. То же можно сказать и о целях, иначе бы система нашла равновесное состояние. На примере студента мы можем сказать, что нам хорошо известно, что не все студенты демонстрируют одинаковое рвение к учёбе на протяжении всего периода обучения.

В-третьих, нам не всегда известны цели даже известных нам сил. Например, работодатель может декларировать своей целью получение студентом высшего образования и при этом отказывать ему в сокращённой рабочей неделе, что не приближает работодателя и студента к достижению данной цели.

Мы ничего не знаем о стратегиях, реализуемых каждой силой, поэтому будем полагать, что действие каждой силы определяется текущим состоянием системы. Это значит, что и очередной факт в нашем представлении определяется только текущим состоянием системы. В результате свершения любого факта система переходит в конкретное состояние. Таким образом, можно уверенно сказать, что наблюдаемая нами система развивается не случайным образом – развитие системы определяется всей совокупностью действующих на неё сил, о которых, как мы выяснили, известно очень мало. Раз не случайна последовательность состояний системы, то не случайна и последовательность фактов. Это значит, что в последовательности свершившихся фактов имеет смысл искать закономерность.

Структура факта

Каждый факт характеризуется *набором атрибутов*, комбинация значений которых однозначно идентифицирует факт среди остальных. Для разных фактов наборы их атрибутов могут различаться, но каждый факт обязательно обладает атрибутами пространства и времени.

Таким образом, будем считать, что *факт представляется совокупностью атрибутов* с зафиксированными значениями:

$$\alpha = (t, g, x_1, x_2, \ldots, x_k)$$

Подобное описание факта будем отождествлять с самим фактом.

Будем говорить, что *два набора атрибутов* $(x_1, x_2, \ldots, x_m)$ и $(y_1, y_2, \ldots, y_n)$ *подобны*, если выполняются следующие условия:

$$1)\ m = n;$$
$$2)\ \left(\forall x_i \mid i \in \overline{1,m}\right)\left(\exists j \in \overline{1,n}\right): D(x_i) = D(y_j)$$

Факты, наборы атрибутов которых подобны, также будем называть *подобными*. Множество фактов A будем называть *классом фактов*, если для него выполняются следующие условия:

$$1)\ \left(\forall \alpha, \beta \in A\right): \alpha\ like\ \beta$$
$$2)\left(\forall \gamma \in G \setminus A\right)\left(\forall \alpha \in A\right): \overline{\gamma\ like\ \alpha}$$

где G – множество всех фактов, $\alpha\ like\ \beta$ – отношение подобия фактов α и β.

В современных системах Business Intelligence данные для анализа берутся из хранилищ данных, построенных на основе многомерной модели данных [2]. При использовании многомерной модели в хранилище данные хранятся в гиперкубах. Ячейка гиперкуба и представляет собой факт, а отметки измерений, соответствующие данной ячейке с показателями, представляют собой атрибуты фактов. Таким образом, *гиперкубы хранилищ данных своей структурой определяют классы фактов.*

Причинно-следственные связи

Нами было установлено, что в последовательности свершившихся фактов существуют закономерности. В основе предлагаемой модели лежит аксиома о том, что *все факты связаны причинно-следственными связями*, другими словами, у любого факта есть причина и следствие.

Рассмотрим два состояния системы $S(t')$ и $S(t'')$, причём $t' < t''$. Возьмём момент времени τ такой, что $t' < \tau < t''$. Пусть существует также состояние $S(\tau)$, такое, что:

$$S(t') \xrightarrow{\alpha} S(\tau) \xrightarrow{\beta} S(t'') \quad \alpha, \beta \in G.$$

Видно, что факты α и β связаны.

Введём на множестве G *отношение следствия*:

$$\alpha \to \beta,$$

здесь α – *причина*, а β – *следствие.*

Изучив, как тот или иной конкретный факт меняет состояние системы, можно делать точный прогноз о том, в какое состояние перейдёт система, если будет иметь место один из известных фактов. Но проблема заключается в том, что не всегда известно, какое событие произойдёт в будущем.

Поставим задачу поиска условной вероятности

$$P_{\alpha}(\lambda), \text{ где } \alpha,\lambda \in G, \qquad (*)$$

которая показывает вероятность свершения факта λ при условии свершения факта α.

Имея инструментарии оценки данной условной вероятности, можно легко разрешать неопределённости при принятии решений, но получение подобной оценки требует глубоких знаний законов предметной области. В качестве примера подхода к решению данной задачи, можно привести *теорию Байесовских сетей* [3]. Однако применение данного подхода требует знаний об условной зависимости переменных, их априорных и условных вероятностей. На практике же, как правило, исследователю доступен лишь огромный набор данных о свершившихся фактах и ему необходимо сделать прогноз либо выбрать наиболее удачное решение, не имея чёткого представления обо всех законах предметной области. Таким образом, построить Байесовскую сеть, описывающую все факты, не обладая достаточными знаниями о законах предметной области, – это крайне сложная задача.

С другой стороны, среди задач интеллектуального анализа данных хорошо изучена задача *секвенциального анализа* [4]. Методы решения данной задачи можно с успехом применить при выявлении причинно-следственных связей между фактами, но только между теми, записи о которых имеются в базе данных. Таким образом, данный подход также не позволяет решить задачу на всем множестве G, а лишь на некотором ограниченном наборе фактов $\overline{G}$, записи о которых имеются в хранилище данных, такой что:

$$\overline{G} \subset G.$$

Решение задачи в общем виде на всем множестве G, без сомнения, представляет большой теоретический интерес, но на практике может быть достаточно решить задачу на более узком множестве Φ, таком что:

$$\overline{G} \subset \Phi \subset G.$$

Методы решения задачи поиска закономерностей свершения фактов на конечном множестве зафиксированных фактов $\overline{G}$ лежат в области секвенциального анализа и хорошо изучены. Предлагается свести

решение этой задачи на множестве Φ к решению задачи на множестве $\overline{G}$.

В основе предлагаемого подхода лежит комплекс методов построения модели «факт-связь» [5]. В отличие от частных наборов (или последовательных шаблонов), получаемых алгоритмами, решающими задачу секвенциального анализа (например Apriori), модель «факт-связь» позволяет реализовать дополнительный уровень абстракции. Это делает возможным оперировать не конкретными фактами, а целыми группами, классами, строить обобщения и т.д. [1]. Применение данной формальной модели позволяет обобщить решение задачи секвенциального анализа на множестве $\overline{G}$ и получить условную вероятность (*) для фактов из множества Φ.

В предлагаемой модели «факт-связь» предусмотрено *отношение обобщения фактов*:

$$\alpha \; is \; \beta, \text{где } \alpha, \beta \in G,$$

здесь α *есть частный случай* β. Механизм обобщения позволяет строить новые абстрактные факты на основе реальных фактов, о которых имеются записи.

Для обобщения справедливы следующие *основные свойства обобщения*:

$1^{\circ}\ (x_1 = c_1, x_2 = c_2, \ldots, x_m = c_m, x_{m+1}, \ldots, x_n)\ is\ (x_1 = c_1, x_2 = c_2, \ldots, x_m = c_m)$

$2^{\circ}\ (\forall \alpha \mid \alpha \to \beta; \gamma\ is\ \beta) : \alpha \to \gamma$

$3^{\circ}\ (\forall \beta \mid \alpha \to \beta; \gamma\ is\ \alpha) : \gamma \to \beta$

Допустим, что в нашем распоряжении имеется некоторое множество зафиксированных фактов $\overline{G}$, пусть также методами секвенциального анализа (например алгоритмом Apriori) были выявлены следующие закономерности:

$$\alpha_i \to \beta_i, \alpha_i, \beta_i \in \overline{G}, i = \overline{1,q}, q \in N,$$

такие, что:

$$\alpha_i \; is \; \alpha \;\text{ и }\; \beta_i \; is \; \beta, \; i = \overline{1,q} \;\; \alpha, \beta \in G.$$

Тогда при достаточной поддержке найденных секвенций [4] можно с определённой уверенностью утверждать, что

$$\alpha \to \beta.$$

А это означает, что

$$\left(\forall \delta, \lambda \in G \mid \delta \; is \; \alpha, \lambda \; is \; \beta\right) : \delta \to \lambda.$$

И в этом случае факты δ и λ могут лежать вне множества $\overline{G}$.

Таким образом, отношение обобщения позволяет свести задачу поиска условной вероятности (*) на множестве Φ к задаче на

множестве $\overline{G}$.

Проблема в том, что не ясно, откуда взять знания о наличии отношения обобщения между фактами. Видится два решения этой задачи: во-первых, знания об обобщении могут быть внесены в автоматически построенную модель аналитиком [5]; во-вторых, эти знания могут быть извлечены из структуры гиперкуба. Измерения гиперкубов имеют иерархическую структуру [2] и это можно использовать при автоматическом построении модели «факт-связь», т.е. знания об обобщении фактов можно получить на основе иерархии измерений.

Заключение

В данной работе предложен подход к решению задач моделирования и анализа фактов и причинно-следственных связей между ними. В основе предлагаемого подхода лежит идея полуавтоматического построения модели факт-связь. Построение абстрактной модели позволит применить знания о связях между фактами, построенных на основе ограниченного набора данных, для более широкого множества фактов. Данный подход может быть использован при решении задач прогнозирования и оптимизации.

Библиографический список

1. *Мальцев П.А.* Моделирование и анализ фактов и связей между ними // Natural and Artificial Intelligence: International Book Series / Sofia, 2010. С. 194-199.
2. *Мальцев П.А., Лядова Л.Н.* Формализация многомерной модели данных // Математика программных систем: межвуз. сб. научн. тр. / Перм. ун-т. Пермь, 2006. С. 74-87.
3. *Рассел С., Норвиг П.* Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006.
4. *Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И.* Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP, 2-е изд. – СПб.: БХВ-Петербург, 2008.
5. *Мальцев П.А.* Моделирование и анализ фактов и связей между ними // Труды Конгресса по интеллектуальным системам и информационным технологиям «AIS IT'11»: Научное издание в 4-х томах. Т. 1. – М.: Физматлит, 2011. С. 239-245.

СОДЕРЖАНИЕ

Научное издание

Математика программных систем

Межвузовский сборник научных статей

Выпуск 7 (2010 г.)

Издается в авторской редакции

Подписано в печать 28.12.2010. Формат 60×84/16.
Усл. печ. л. 8,14. Уч.-изд. л. 7,5. Тираж 100 экз.

Редакционно-издательский отдел
Пермского государственного университета.
614990. Пермь, ул. Букирева, 15