УДК 004.001;004.5

ПРОБЛЕМНО-НЕЗАВИСИМЫЙ ГЕНЕРАТОР ТЕКСТОВ, УПРАВЛЯЕМЫЙ ОНТОЛОГИЕЙ[1]

В. В. Грибова

Представлен метод автоматической генерации текстов по выходным данным прикладной программы — неупорядоченному множеству кортежей отношений реляционной базы данных. Описана основная идея подхода, управляющая генератором текста онтология, модель генерации текста, а также метод реализации подхода.

ВВЕДЕНИЕ

Одной из основных характеристик современных программных систем различного назначения является возможность представления результатов их работы в соответствии с требованиями, определяемыми не только предпочтениями конкретного пользователя или групп пользователей, но, прежде всего, назначением программной системы и особенностями предметной области.

Существуют различные виды представления выходной информации — таблицы, графики, диаграммы. Особое место занимает представление выходных данных в виде текстов. При генерации таких текстов приходится решать три основные проблемы: каким образом задавать способ изложения текста; как породить текст, соответствующий этому способу изложения на основе выходных данных прикладной программы; как разметить этот текст, чтобы обеспечить возможность его дальнейшего использования.

В настоящее время рынок программных систем не предлагает универсальных средств генерации текстов для произвольных предметных областей. Представление результатов в виде текста возможно с помощью генераторов отчетов [1, 2]. Они, как правило, ориентированы на представление отчетов в виде таблиц, графиков, диаграмм, поэтому с их помощью можно сгенерировать только тексты простой структуры. Генераторы отчетов не имеют специальных средств анализа полученных результатов, а снабжены только функциями их фильтрации и сортировки. Технология формирования отчетов с помощью таких средств основана на создании шаблона отчета, который в большинстве случаев встраивается в приложение, что делает невозможными любые изменения получаемых отчетов без изменения самого приложения; многие генераторы отчетов формируют отчет в собственном формате без возможности его редактирования и изменения.

В данной работе предлагается метод автоматической генерации текстов произвольных структуры и содержания, управляемой онтологией на основе результатов работы прикладной программы, представленных в виде неупорядоченного множества кортежей отношений.

1. ОСНОВНАЯ ИДЕЯ ПОДХОДА

Рассмотрим сначала неформальное описание основной идеи подхода. Стремление к снижению стоимости разработки и сопровождения программных систем приводит к необходимости предоставления разработчикам языков так называемого пятого поколения [3] — высокоуровневых средств описания и автоматической генерации исполнимого кода. Для решения задачи генерации текстов произвольной структуры и содержания в качестве высокоуровневого языка предлагается универсальная онтология, состоящая из конструкций для описания структуры и способа порождения текста, а также конструкций для описания методов анализа результатов (выходных данных прикладной программы). Онтология — явное описание или представление некоторой части концептуализации. Она может иметь различные формы, но обязательно включает в себя словарь терминов и некоторую

[1] Работа выполнена при финансовой поддержке ДВО РАН по программе № 15 ОЭММПУ РАН «Проблемы анализа и синтеза интегрированных технических и социальных систем управления», проект «Синтез интеллектуальных систем управления базами знаний и базами данных».

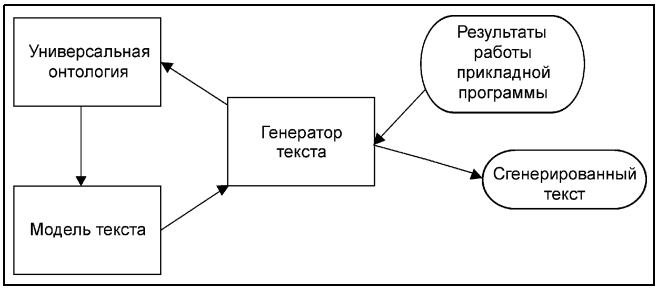

Рис. 1. Графическое представление основной идеи подхода

спецификацию их смысла, определения и указания о связи понятий, что в совокупности налагает структуру на предметную область и ограничивает возможные интерпретации терминов [4]. Используя онтологию, разработчик программного средства формирует модель порождения текста (далее модель текста) для конкретной задачи либо группы задач: конкретизирует значения терминов онтологии. Используя это описание и результаты работы прикладной программы, генератор, управляемый онтологией, автоматически генерирует текст заданной структуры и содержания. Графическое представление основной идеи подхода приведено на рис. 1.

Универсальную онтологию можно представить в виде $O = \langle W, G\rangle$, где W — конструкция для формального описания структуры и способа порождения текста, G — конструкция для задания метода анализа текста.

Конструкция W состоит из последовательности элементов описания текста $w_1, ..., w_n$. Каждый элемент описания — конструкция, которая определяет зависимость порождаемого текста от выходных данных прикладной программы. Конструкциями онтологии для формального описания структуры и способа порождения текста служит *альтернатива*, *цикл*, *выводимое множество* и *строка*. Цикл и альтернатива, в свою очередь, также содержат элементы описания текста в качестве компонентов.

Конструкция для задания метода анализа текста G определяет, какие выходные данные прикладной программы анализируются в конструкции w_i. Конструкцию G назовем описанием переменной. Описание переменной G имеет вид: $P(c_1, ..., c_{j*}, ..., c_n)$, где P — имя отношения, представляющего выходные данные прикладной программы, c_{j*} — имя описываемой переменной, $c_1, ..., c_{j-1}, c_{j+1}, ..., c_n$ — имена вспомогательных переменных. Под множеством значений c_{j*} переменной будем понимать множество всех j-х элементов кортежей отношения P. Множествами значений вспомогательных переменных $c_1, ..., c_{j-1}, c_{j+1}, ..., c_n$ являются множества элементов кортежей того же отношения с номерами $1, ..., j-1, j+1, ..., n$, соответственно.

Конструкция онтологии *альтернатива* имеет вид: $\langle G, \{\Omega_1, ..., \Omega_m\}\rangle$. Здесь G — описание переменной, $\{\Omega_1, ..., \Omega_m\}$ — описания альтернатив. Описание каждой альтернативы состоит из двух частей — множества условий выбора δ_k и варианта ψ_k — составного оператора, т. е. $\Omega_k = \langle\delta_k, \psi_k\rangle$, $1 \leqslant k \leqslant m$. Множество условий выбора δ_k задает либо некоторое число, обозначающее возможное число значений переменной, либо значение (или множество значений), которое может получить переменная по результатам работы прикладной программы, либо некоторую метку. Вариант ψ_k — это составной оператор, который будет выполняться, если значения переменной удовлетворяют условию выбора δ_k. Переменная из описания G удовлетворяет условию выбора δ_k, если либо число ее значений совпадает с числом в условии выбора, либо значение переменной (или подмножество ее значений) совпадает со значением (или подмножеством значений) в условии выбора. Выполнение определенной альтернативы состоит в выполнении варианта (составного оператора) первого по порядку описания альтернативы, для которого переменная из описания G удовлетворяет условию выбора. Если переменная из описания G не удовлетворяет ни одному из условий выбора, то выполняется альтернатива, условием выбора которой является метка.

Часто порождаемый текст должен содержать повторяющиеся части, и при каждом повторении эти части несколько отличаются друг от друга, причем эти отличия должны определяться выходными данными прикладной программы. Конструкция онтологии «цикл» служит для представления таких повторяющихся частей и имеет вид: $\langle G$: *усл*, $\alpha\rangle$. Здесь G — описание переменной, *усл* — условие, которое налагается на ее значения (условие может отсутствовать), α — тело цикла. Описание переменной необходимо для задания условия выполнения тела цикла: число повторений тела цикла, а также содержание изменяемых частей в теле цикла. Тело цикла — составной оператор. Содержание изменяемой информации при каждом выполнении тела цикла зависит от значений параметра цикла и вспомогательных переменных. При каждом выполнении тела цикла параметру цикла соответствует некоторое новое значение из множества его значений, определяемое результатами работы прикладной программы и условием «*усл*», которое налагается на значения переменной. Каждому значению переменной соответствуют подмножества значений вспомогательных переменных.

Конструкция онтологии *выводимое множество* указывает, что в тексте необходимо перечислить в определенном порядке (в соответствии с условием) все значения некоторой переменной. Данная конструкция имеет вид $\langle G$: *усл*$\rangle$, где G — описание переменной, «*усл*» — условие, которое налагается на ее значения (условие может отсутствовать). Условие определяет, в каком порядке значения переменной должны быть помещены в текст — в алфавитном порядке либо в порядке, который определен этим условием.

Конструкция онтологии *строка* предназначена для представления фиксированных фраз. Примером текстовых конструкций могут быть: «Фамилия, имя, отчество», «диагноз при поступлении», «лечащий врач» и др.

Одно и то же формальное задание конструкций W может соответствовать различным выходным данным прикладной программы, но оно должно быть согласовано с этими данными. Текст T, создаваемый на основе конструкций W, получается различным в зависимости от этих данных.

Таким образом, текст T генерируется на основе конструкций для описания структуры и способа порождения текста W, конструкций анализа результатов G, а также выходных данных прикладной программы, представленных в виде конечной совокупности отношений.

2. МОДЕЛЬ ГЕНЕРАЦИИ ТЕКСТА, УПРАВЛЯЕМОЙ УНИВЕРСАЛЬНОЙ ОНТОЛОГИЕЙ

Состояние вычислительного процесса на шаге m определим как четверку: $q_m = (A^m, P^m, T^m, N^m)$, где A^m — вектор значений переменных, P^m — вектор отношений циклов или альтернатив, T^m — текст, N^m — позиционный номер выполняемого элемента описания текста.

Вектор значений переменных имеет вид: $A^m = \langle a_1, a_2, ..., a_{Nm}\rangle$, где Nm — число элементов вектора на шаге m, а каждый элемент вектора a_i для всех $i = 1, ..., Nm$ есть пара $a_i = (nma_i, VAL_i)$. Здесь nma_i — имя переменной, VAL_i — множество значений переменной nma_i такое, что $VAL_i = \{val_1^i, ..., val_p^i\}$, где val_j^i — значение переменной nma_i, $j = 1, ..., p$, p — число значений переменной.

Вектор отношений циклов и альтернатив P^m имеет вид: $P^m = \langle p_1, ..., p_{Km}\rangle$, Km — число элементов вектора в состоянии на шаге m, равное числу вложенных циклов и альтернатив, выполняемых на шаге m. Каждый элемент вектора p_i для $i = 1, ..., Km$ относится к одной из выполняемых на шаге m конструкций *цикл* и *альтернатива*, причем элемент вектора отношений p_1 относится к самой внешней конструкции *цикл* или *альтернатива*, элемент вектора отношений p_2 — к непосредственно вложенной в нее конструкции *цикл* или *альтератива* и т. д., элемент вектора отношений p_{Km} — к самой внутренней конструкции *цикл* или *альтернатива*. Каждый элемент вектора p_i для всех $i = 1, ..., Km$, имеет вид: $p_i = (p_i, nmv_i, Nc_i, Nqn_i, VALVC_i)$, где p_i — имя отношения, входящего в заголовок i-й вложенной конструкции цикл или альтернатива; nmv_i — имя переменной i-й вложенной конструкции *цикл* или *альтернатива*; Nc_i — число повторений i-го цикла или 0 для альтернативы; Nqn_i — номер состояния вычислительного процесса перед выполнением i-й конструкции *цикл* или *альтернатива*; $VALVC_i$ — множество уже пройденных значений переменной цикла nmv_i к шагу m, $VALVC_i = \{valvc_1, ..., valvc_m\}$, где $valvc_j$ — значение переменной цикла $j = 1, ..., m$.

Начальное состояние процесса генерации текста есть $q_1 = (A^1, P^1, T^1, N^1)$, где вектор значений переменных A^1 и вектор отношений цикла и альтернативы P^1 пусты (имеют размерность 0), текст T_1 в начальном состоянии — пустая строка, $N^1 = 1$.

Введем понятие «описание множества», которое имеет вид: $A_i(c_1^i, ..., c_m^{i*}, ..., c_n^i)$: *усл*, где A_i — имя одного из отношений, представляющих выходные данные прикладной программы, $c_1^i, ..., c_n^i$ — имена переменных. Множеством значений каждой переменной c_j^i является множество всех j-х элементов кортежей — $\{a_j\}$ — отношения A_i из результатов ξ работы прикладной программы, c_m^{i*} — помеченная переменная, имеющая свою семантическую трактовку; которая описывается в контексте рассматриваемых далее конструкций; *усл* — условие, налагаемое на значения c_m^{i*}, которое может и отсутствовать. Условие задает порядок, в котором упорядочены значения переменной c_m^{i*}.

Процесс генерации текста по онтологии текста заключается в последовательном выполнении элементов описания текста, начиная с первого. При этом на каждом шаге формируется очередное состояние вычислительного процесса, которое зависит от выполняемого элемента описания w_i, состояния вычислительного процесса и результатов ξ ра-

боты прикладной программы. Будем считать, что в состоянии $q_s = (A^s, P^s, T^s, N^s)$ имеет место $N^s = i$.

Рассмотрим формирование очередного состояния для каждой из конструкций.

Пусть выполняемым элементом описания текста w_i является конструкция *строка*. Тогда следующее состояние процесса формирования текста есть $q_{s+1} = (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$, причем $A^{s+1} = A^s$, $P^{s+1} = P^s$, $T^{s+1} = T^s + w_i$, где «+» — операция конкатенации, а значением N^{s+1} является позиционный номер, следующий за номером i в лексикографическом порядке. Если такого номера нет, то выполнение универсального рецепта на этом заканчивается.

Пусть выполняемым элементом описания текста w_i является конструкция *цикл* — $\langle\sigma, \alpha\rangle$, где σ — заголовок цикла, $\sigma = A_i(c_1^i, ..., c_m^{i*}, ..., c_n^i)$: *усл* (при этом c_m^{i*} назовем параметром цикла), α — тело цикла (составной оператор).

Выполнение цикла начинается с определения числа повторений цикла. При определении числа повторений цикла r возможны следующие варианты:

— в результатах ξ работы прикладной программы отношения с именем A_i пусто; в этом случае число повторений цикла $r = 0$;

— в результатах ξ работы прикладной программы отношение с именем A_i непусто; в этом случае число повторений цикла r зависит от ξ и состояния процесса формирования текста q_s. Здесь возможны два случая.

- Состояние процесса формирования текста $q_s = (A^s, P^s, T^s, N^s)$ таково, что вектор значений переменных A^s пуст ($Nm = 0$) или в векторе значений переменных A^s нет таких переменных $nma_g, ..., nma_h$, что $nma_g = c_k^i, ..., nma_h = c_p^i$, где $1 \leqslant p, k \leqslant n$. Тогда $r = \mu\{a_m | A_i(a_1, ..., a_m, ..., a_n) \in \xi\}$, где a_m — m-й элемент кортежа A_i, т. е. число повторений цикла равно мощности множества значений параметра цикла c_m^{i*} в результатах ξ работы прикладной программы.
- Состояние $q_s = (A^s, P^s, T^s, N^s)$ таково, что в векторе значений переменных A^s имеются переменные $nma_g, ..., nma_h$ такие, что $nma_g = c_k^i, ..., nma_h = c_p^i$, где $1 \leqslant p, k \leqslant n$. В этом случае число выполнений цикла определяется по результатам ξ работы прикладной программы в соответствии со значениями переменных $nma_g = c_k^i, ..., nma_h = c_p^i$ из вектора A^s следующим образом.

Пусть в результатах ξ существуют кортежи $zn_q \in A_i, ..., zn_r \in A_i$, такие что

$$\begin{gathered} zn_q[k] = val_n^g, ..., zn_q[p] = val_m^g, \\ \cdots \\ zn_r[k] = val_n^h, ..., zn_r[p] = val_m^h, \end{gathered} \tag{1}$$

где $zn[k]$ — элемент кортежа zn с номером k, $val_n^g, ..., val_m^g \in VAL_g$, $val_n^h, ..., val_m^h \in VAL_h$. Тогда $r = \mu\{zn_q[m], ..., zn_r[m]\}$.

Если в результатах ξ нет таких значений, при которых выполняются условия (1), то $r = 0$.

Если $r = 0$, то выполнение цикла на этом заканчивается и формируется следующее состояние процесса порождения текста $q_{s+1} = (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$, где $A^{s+1} = A^s$, $P^{s+1} = P^s$, $T^{s+1} = T^s$, N^{s+1} — позиционный номер конструкции, следующей за выполненным циклом в составном операторе.

Если $r \geqslant 1$, то дальнейшее выполнение цикла зависит от того, какая из перечисленных далее трех ситуаций имеет место:

— данный цикл не является вложенным в другие циклы;

— данный цикл является вложенным в тело некоторых других циклов, но на значения переменных в заголовке цикла никаких ограничений циклами или альтернативами, внешними по отношению к данному циклу, не наложено;

— на некоторые переменные отношения A_i были наложены ограничения внешними циклами либо альтернативами.

В работе [5] определено формирование очередного состояния в каждом случае. Рассмотрим для примера формирование очередного состояния в первом случае. Очередное состояние процесса порождения текста $q_{s+1} = \langle A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1}\rangle$. Здесь $A^{s+1} = A^s \oplus \langle a_{Ns+1}, ..., a_{Ns+n}\rangle$, где $a_{Ns+1} = \langle nma_{Ns+1}, VAL_{Ns+1}\rangle, ..., a_{Ns+n} = \langle nma_{Ns+n}, VAL_{Ns+n}\rangle$, $nma_{Ns+1} = c_1^i, ..., nma_{Ns+n} = c_n^i$, результатом операции $\oplus$ над векторами A^s и $\langle a_{Ns+1}, ..., a_{Ns+n}\rangle$ является вектор $\langle a_1, ..., a_s, a_{Ns+1}, ..., a_{Ns+n}\rangle$. Каждой переменной nma_{Ns+j}, $j = 1, ..., n$, сопоставляется множество ее значений VAL_{N+j} из результатов ξ. Вектор отношений цикла и альтерна-

тивы примет вид: $P^{s+1} = P^s \oplus \langle p_{Ks+1} \rangle$, причем $p_{Ks+1} = (p_{Ks+1}, nmv_{Ks+1}, Nc_{Ks+1}, Nqn_{Ks+1}, VALVC_{Ks+1})$, где $p_{Ks+1} = A_i$ — имя отношения в заголовке цикла; $nmv_{Ks+1} = c_m^{i*}$, где c_m^{i*} — имя переменной цикла; $Nc_{Ks+1} = r$ — число выполнений цикла; $Nqn_{Ks+1} = s$ — номер состояния до выполнения цикла; $VALVC_{Ks+1} = Y^{s+1}(c_m^{i*})$ — значение переменной цикла c_m^{i*}; $T^{s+1} = T^s$; N^{s+1} — есть позиционный номер, следующий за номером N^s в лексикографическом порядке.

Далее r раз выполняется тело цикла. После завершения каждого выполнения тела цикла формируется новое состояние.

Наконец, рассмотрим последний случай, когда тело цикла уже выполнено r раз и получено состояние q_s. В этом случае состояние $q_s = (A^s, P^s, T^s, N^s)$ таково, что $p_{Ks} = A_i$ и $nmv_{Ks} = c_m^{i*}$. Тогда формируется новое состояние q_{s+1} следующим образом: $q_{s+1} = (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$ такое, что $A^{s+1} = A^f$, $P^{s+1} = P^f$, где $f = Nqn_{Ks}$ — номер состояния вычислительного процесса до выполнения цикла, $T^{s+1} = T^s$, N^{s+1} — позиционный номер, следующий за номером N^s — в лексикографическом порядке.

Пусть выполняемым элементом описания текста w_i является *выводимое множество* — $\langle \gamma = A_i(c_1^i, ..., c_m^{i*}, ..., c_n^i)$: *усл*$\rangle$, т. е. γ — описание множества (при этом c_m^{i*} назовем выводимым элементом). Выполнение данной конструкции зависит от того, какая из перечисленных ниже ситуаций имеет место:

— в результатах ξ работы прикладной программы отношение, имя которого совпадает с именем отношения в описании выводимого множества, пусто;

— на значения переменных в описании выводимого множества никаких условий циклами и альтернативами, внешними по отношению к данной конструкции, не наложено;

— на значения некоторых переменных в описании выводимого множества наложены условия внешними циклами или альтернативами.

Если в выходных данных ξ прикладной программы нет отношения с именем A_i, то выполнение конструкции *выводимое множество* на этом заканчивается. При этом формируется следующее состояние процесса порождения текста $q_{s+1} = (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$ такое, что $A^{s+1} = A^s$, $P^{s+1} = P^s$, $T^{s+1} = T^s$, N^{s+1} — позиционный номер конструкции, следующий за позиционным номером N^s в лексикографическом порядке.

Если на значения переменных в описании выводимого множества не наложено условий внешними циклами и альтернативами, то $q_s = (A^s, P^s, T^s, N^s)$ таково, что в векторе значений переменных A^s имеет место $nma_j \neq c_k^i$ для всех $1 \leqslant j \leqslant Nm$ и $1 \leqslant k \leqslant n$. В этом случае формируется очередное состояние процесса порождения текста $q_{s+1} = (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$ такое, что $A^{s+1} = A^s$, $P^{s+1} = P^s$, $T^{s+1} = T^s + t$, где t — все значения переменной c_m^{i*}, разделенные запятой, из упорядоченного множества $M(c_m^{i*})$. Упорядоченное множество $M(c_m^{i*})$ формируется из множества значений переменной c_m^{i*} — всех m-х элементов кортежей с именем A_i из результатов ξ работы прикладной программы.

Если на значения некоторых переменных в описании выводимого множества наложены условия внешними циклами или альтернативами, то состояние процесса формирования текста $q_s = (A^s, P^s, T^s, N^s)$ таково, что в векторе значений переменных A^s имеются переменные nma_g, ..., nma_h такие, что $nma_g = c_k^i$, ..., $nma_h = c_p^i$, где $1 \leqslant p, k \leqslant n$ (некоторые переменные отношения A_i в описании выводимого множества совпадают с некоторыми переменными вектора значений переменных A^s). В этом случае формируется очередное состояние процесса порождения текста $q_{s+1} = (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$ такое, что $A^{s+1} = A^s$, $P^{s+1} = P^s$, $T^{s+1} = T^s + t$, где t — все значения переменной c_m^{i*}, разделенные запятой, из упорядоченного множества $M(c_m^{i*})$. Упорядоченное множество $M(c_m^{i*})$ формируется из множества значений переменной c_m^{i*}, которое определяется по результатам ξ работы прикладной программы в соответствии со значениями переменных $nma_g = c_k^i$, ..., $nma_h = c_p^i$, где $1 \leqslant p, k \leqslant n$ из вектора A^s.

Пусть выполняемым элементом описания текста w_i является конструкция *альтернатива* — $\langle \beta, \{\Omega_1, ..., \Omega_m\} \rangle$, где β — описание переменной, т. е. $\beta = A_i(c_1^i, ..., c_m^{i*}, ..., c_n^i)$ (при этом c_m^{i*} назовем альтернативной переменной), $\{\Omega_1, ..., \Omega_m\}$ — описания

альтернатив. Описание каждой альтернативы есть $\Omega_k = \langle \delta_k, \psi_k \rangle$, $1 \leqslant k \leqslant m$. Множество условий выбора есть $\delta_k = N$, где N — либо некоторое число, обозначающее возможное число значений альтернативной переменной, либо $\{значение_i\}$, где $i \geqslant 1$, т. е. заданное множество значений альтернативной переменной по выходным данным прикладной программы. Множеством условий выбора δ_k может быть также метка #, на которую произойдет переход, если не будет выполняться ни одно из заданных условий выбора, т. е. δ_m — либо число, либо $\{значение_i\}$, где $i \geqslant 1$ либо метка. Вариант ψ_k — это составной оператор, который будет выполняться, если значения переменной удовлетворяют условию выбора δ_k.

Выполнение альтернативы начинается с определения множества значений альтернативной переменной c_m^{i*} по описанию переменной, состоянию процесса порождения текста q_s и выходным данным ξ прикладной программы. Способ определения множества значений переменной c_m^{i*} совпадает со способом определения множества значений для конструкции *выводимое множество*.

Дальнейшее выполнение конструкции *альтернатива* зависит от условий выбора.

Пусть мощность множества значений альтернативной переменной c_m^{i*} либо совпадает с числом в одном из условий выбора, либо заданное подмножество значений в одном из условий выбора есть подмножество множества значений альтернативной переменной, либо если эти условия не выполняются, но последним условием выбора является метка. Тогда формирование очередного состояния процесса порождения зависит от способа определения значений альтернативной переменной c_m^{i*}. При этом в каждом из возможных случаев формируется очередное состояние.

Далее выполняется выбранный составной оператор, т. е. оператор ψ_k, условие выбора δ_k которого является первым по порядку условием выбора среди множества условий выбора, удовлетворяющих значению альтернативной переменной, иначе выполняется оператор ψ_m, если $\delta_m = \#$.

Пусть после выполнения выбранного составного оператора было получено состояние $q_s = (A^s, P^s, T^s, N^s)$. В этом случае формируется очередное состояние $q_{s+1} \leqslant (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$ такое, что $A^{s+1} = A^f$, $P^{s+1} = P^f$, где $f = Nqn_{Ks}$ — номер состояния вычислительного процесса до выполнения альтернативы, $T^{s+1} = T^s$, N^{s+1} — позиционный номер конструкции, следующей за альтернативой в составном операторе.

Если ни одно из условий выбора не удовлетворяется и среди условий выбора нет условия выбора, совпадающего с меткой, то очередное состояние процесса формирования примет вид: $q_{s+1} = (A^{s+1}, P^{s+1}, T^{s+1}, N^{s+1})$, где $A^{s+1} = A^s$, $P^{s+1} = P^s$, $T^{s+1} = T^s$, N^{s+1} — позиционный номер конструкции, следующей за альтернативой в составном операторе.

3. МЕТОД РЕАЛИЗАЦИИ ПОДХОДА

Сформулируем сначала основные требования, которым должен удовлетворить генератор текстов:

- генератор текста должен иметь средства форматирования, которые не уступают по своим возможностям существующим средствам форматирования документов;
- выходные данные прикладной программы не должны ограничиваться форматом какой-либо реляционной базы данных; все широко известные и используемые форматы реляционных баз данных могут использоваться как средство хранения выходных данных прикладной программы;
- модель текста по универсальной онтологии должна формироваться с помощью редактора, помогающего разработчику без заучивания конструкций онтологии формировать конкретную модель текста.

Для реализации этих требований онтология расширена конструкциями оформления. Конструкции оформления — это конструкции (теги) одного из языков разметки, например, HTML, RTF. Выбор языка разметки зависит от требований к оформлению текста.

Прикладная программа записывает свои выходные данные в базу данных в соответствии с форматом, определенным в модели текста. База данных может находиться как в локальном компьютере, так и быть сетевой. Генератор текстов формирует текст на том языке разметки, вставки конструкций которого присутствуют в описании шаблона текста. На следующем этапе оно приводится к виду, понятному пользователю с помощью интерпретатора языка разметки (например, с помощью браузера, компонента HTMLView или интерпретатора RTF). На рис. 2 приведена обобщенная схема процесса формирования текста.

Метод создания модели текста заключается в следующем: с помощью редактора создается модель переменных прикладной программы (или повторно используется уже созданная), далее, с

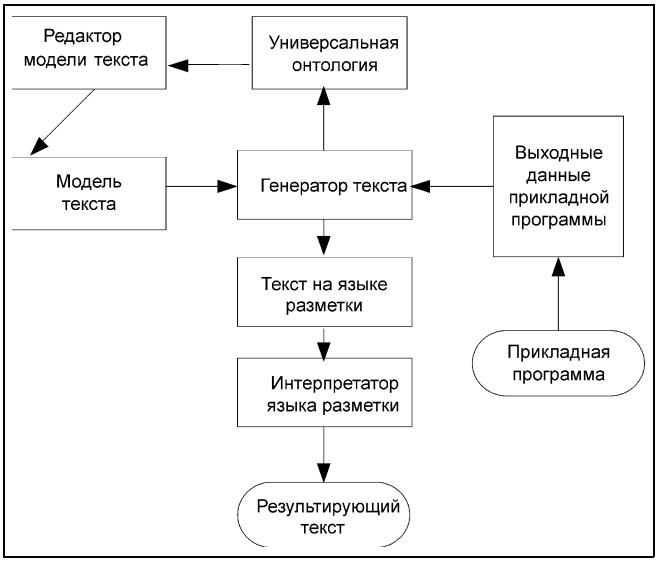

Рис. 2. Обобщенная схема формирования текста

помощью этого же редактора, создается модель структуры и содержания текста. С помощью редактора модели текста вставляются конструкции языка разметки для описания оформления текста либо эти конструкции вставляются в описание с помощью редактора языка разметки. Заметим, что редактор модели текста не имеет средств автоматизированного создания конструкций языка разметки, поскольку он не «привязан» к какому-либо конкретному языку разметки.

ЗАКЛЮЧЕНИЕ

В работе представлены основная идея и метод реализации подхода к генерации текстов произвольной структуры и содержания. Тексты могут служить объяснениями результатов работы прикладной программы, отчетами и другими документами. Описание текста не встраивается в архитектурные компоненты программной системы — интерфейс и прикладную программу, а автоматически генерируется на основе модели текстов и результатов работы прикладной программы.

ЛИТЕРАТУРА

1. *Cohen R., Spencer B., Sanmugasunderam A.* Providing responses specific to a user's goals and background // Int. J. Expert Systems. — 1989. — N 2. — P. 135—162.
2. *Маклаков С. В.* Анализ данных. Генератор отчетов Crystal Reports. — СПб.: БХВ-Петербург, 2003. — 489 с.
3. *Бобровский С.* Технология Пентагона на службе российских программистов. Программная инженерия. — СПб.: Питер, 2003. — 222 с.
4. *Uschold M.* Knowledge Level Modeling: Concepts and Terminology // The Knowledge Engineering Review. — 1998. — Vol. 13:1, — P. 5—29.
5. *Грибова В. В., Клещев А. С.* Модель вербального объяснения результатов работы экспертной системы для индивидуального пользователя // Известия РАН. Теория и системы управления. — 2000. — № 3. — С. 155—160.

☎ *(4232) 31-40-01*
e-mail: gribova@iacp.dvo.ru ❑

Новые книги

***Алгоритмы** и S-модели гибридных систем адаптивного управления. — Благовещенск: Амурск. гос. ун-т, 2005. — 204 с.*
***Амбарцумян А.А.** Событийное логическое управление производственными процессами поточного типа. — М.: Ин-т проблем управления, 2006. — 99 с.*
***Боголюбов Н.Н.** Собрание научных трудов. Т. 3. — М.: Наука, 2005. — 605 с.*
***Нормальные** формы и бифуркации векторных полей на плоскости. — М.: МЦНМО, 2005. — 415 с.*
***Винниченко И.В.** Автоматизация процессов тестирования. — СПб.: Питер, 2005. — 202 с.*
***Воронин В.В.** Теоретические проблемы диагностических экспертных систем.— Владивосток: Дальнаука, 2005 с.*
***Информационные** технологии регионального управления. — М.: УРСС, 2004. — 398 с.*
***Малугин В.А.** Математика для экономистов. Линейная алгебра. — М.: Эксмо, 2006. — 216 с.*
***Мао В.** Современная криптография. — М.: Вильямс, 2005. — 763 с.*
***Мартюшев Л.М.** Развитие экосистем и современная термодинамика. — М.; Ижевск: Ин-т компьютер. исслед., 2004. — 78 с.*
***Морозов А.Д.** Визуализация и анализ инвариантных множеств динамических систем. — М.; Ижевск: Ин-т компьютер. исслед., 2003. — 303 с.*
***Нарышкин А.К.** Цифровые устройства и микропроцессоры. — М.: Academia, 2006. — 317 с.*
***Нечепуренко М.И.** Итерации вещественных функций и функциональные уравнения. — Новосибирск, 2005. — 231 с.*
***Новое** в искусственном интеллекте. — М.: Интелл, 2005. — 280 с.*
***Петров М.К.** Философские проблемы "науки о науке". — М.: РОССПЭН, 2006. — 623 с.*
***Пузанов В.П**. Критерии устойчивости линейных систем автоматического управления и регулирования. — М.: МГТУ им. Н.Э. Баумана, 2005. — 92 с.*
***Реструктуризация** экономики дотационного региона. — М.: Экономика, 2005. — 663 с.*
***Суздалев И.П.** Нанотехнология. — М.: URSS, 2006. — 589 с.*
***Тархов Д.А.** Нейронные сети. — М.: Радиотехника, 2005. — 256 с.*
***Турчин В.И.** Введение в современную теорию оценки параметров сигналов. — Н. Новгород, 2005. — 114 с.*
***Фёдоров Ю.Н.** Основы построения АСУТП взрывоопасных производств. — М.: СИНТЕГ, 2006. — Т. 1. 710 с.; Т. 2. 620 с.*
***Шаронов А.В.** Методы функционального анализа в теории систем автоматического управления. — М.: Горный ун-т, 2005. — 246 с.*
***Экстремальная** робототехника. — СПб.: Астерион, 2004. — 436 с.*