Алгоритм сравнения следующий.

1. Если объект ПОЗ совпадает с основным объектом описания ПОД (сравнение происходит с учетом иерархии объектов, то есть объект ПОЗ может быть таким же, как объект ПОД, или быть его родителем), документ включается в результат выборки (из подготовленного ПОД выбираются заголовок, адрес, релевантность определяется по степени родства объектов ПОД и ПОЗ (при совпадении максимальная)).

2. Если объект ПОЗ совпадает с одним из часто встречаемых объектов ПОД (без учета родства), документ включается в результат выборки (из ПОД выбираются заголовок и адрес документа, релевантность назначается как 0,5 от максимальной).

По данному алгоритму проверяются все ПОД из БД. Результаты, отсортированные по релевантности, представляются пользователю. Пользователь либо формирует новый запрос к уже отобранным данным, либо просматривает описания найденных ФЭ, загружая их из БД ФЭ по ссылке.

Все программные модули написаны на языке *MS Visual C# 2005* и работают под управлением *Windows 2000/XP/Vista*. Использована бесплатная СУБД – *MS SQL Express 2005*.

Всего проанализировано 1300 эффектов. Также была использована древовидная структура словарей входов/выходов и объектов ФЭ из старой версии системы.

В результате построена онтология, содержащая порядка 1000 понятий предметной области. В онтологии присутствуют связи различных понятий, отражающие соотношения понятий в рамках конкретных ФЭ. Данная онтология является основой для разработанной системы.

Как было отмечено выше, построение онтологии осуществлялось на основе автоматизированного анализа описаний ФЭ, которые имелись в банке данных поиска ФЭ по запросу на естественном языке. В рамках этой работы разработан алгоритм автоматического представления запроса на естественном языке в терминах объектно-ориентированного представления физических данных и осуществления поиска.

К перспективам использования данной системы можно отнести возможность построения на ее основе обучающих систем. Отраженные в ее БЗ зависимости позволяют строить различные наглядные представления физических знаний с разной степенью детализации, а в дополнение к алгоритму поиска по запросу на естественном языке можно получить обучающую систему в виде «вопрос – ответ» с произвольными вопросами из области физических знаний.

Еще одним направлением использования описанной системы является разработка системы автоматизации начальных этапов проектирования, где можно автоматизировать построение физического принципа действия, проверив ограничения, и затем преобразовать его в набор готовых технических решений с определенными параметрами (выбранными элементами конструкции, материалами и т.п.).

Литература

1. Русская морфология для программистов. URL: http://www.aot.ru/download/RusLemmatizer.zip (дата обращения: 27.10.2009).

2. Кобозева И.М. Лингвистическая семантика. М.: Книжный дом «ЛИБРОКОМ», 2009. 352 с.

3. Перспективные технологии: итоги и прогнозы / С.В. Асмаков [и др.] // КомпьютерПресс. 2008. № 1 (217). С. 24–25.

ОПЕРАЦИИ ПРЕОБРАЗОВАНИЯ ГРАММАТИК

В.В. Дрождин, к.т.н.; В.М. Тобольченко
(Пензенский государственный педагогический университет им. В.Г. Белинского, drozhdin@spu-penza.ru)

Для повышения разрешающей способности АИС и выполнения семантических преобразований данных на синтаксическом уровне предложено использовать формат данных, формальной основой которого является грамматика. Рассмотрены операции преобразования грамматик, позволяющие порождать языки с требуемыми свойствами.

Ключевые слова: *информационная система, формальный язык, грамматика, преобразование грамматик.*

Широкое внедрение *автоматизированных информационных систем* (АИС) в различные сферы деятельности человека требует все более интенсивного взаимодействия с ними людей с различным уровнем образования и квалификации. Однако взаимодействие с АИС резко ограничивается низкой способностью распознавания информации системой (только на уровне типов данных и полей форм) и невозможностью ввода и вывода данных в различных альтернативных формах (например, в краткой и полной форме, в виде «фамилия, имя и отчество» или «инициалы и фамилия») без соответствующего изменения программного обеспечения. Поэтому целесообразно ввести некоторый промежуточный уровень – формат данных, который в синтаксическом представлении данных позволил бы хотя бы частично учитывать их семантику и прагматику (ценность и удобство ис-

пользования данных для решения определенных практических задач).

Формат данных – это форма синтаксического представления данных, отражающая основные закономерности их построения, определяемые семантикой и использованием. Например, представление данных в краткой или полной форме, в форме, требуемой определенным документом или пользователем, и др.

Формальной основой формата данных является грамматика, точно определяющая все синтаксические закономерности их построения. Между форматом данных и грамматикой есть определенные различия:

– грамматика отражает только синтаксическую (конструктивную) сторону данных, а формат учитывает как синтаксическую сторону, так и использование, и семантику данных;

– грамматика задает только единственный способ построения данных, а формат, отражая основные закономерности конструирования данных, может использовать различные грамматики для их построения;

– формат выступает ограничением (играет роль управления) для грамматики представления конкретного набора данных;

– формат может быть сформирован раньше, чем разработаны специализированные грамматики для представления данных, поэтому данные будут представляться и обрабатываться в рамках некоторой универсальной грамматики (например, последовательность символов или текст);

– грамматика может изменяться и совершенствоваться, а формат будет оставаться фиксированным, но изменение формата почти всегда будет приводить к изменению грамматики.

Формат представления набора данных, обладающих высокой степенью подобия, можно задать относительно небольшой совокупностью грамматических правил, что требует минимального объема памяти для хранения данных [1]. Набор данных сложной структуры может быть представлен системой форматов с иерархической или сетевой структурой. Это обеспечивает компактное хранение самих форматов данных и повышает возможность правильного распознавания данных в случае их перестановки или неполноты. Для выявления формата данных целесообразно использовать аналитический подход, разработанный в математической лингвистике для анализа естественных и формальных языков [2].

Пусть значения набора данных $\mathbf{x \in X}$ представляют собой конечные последовательности символов $\mathbf{x=a_1a_2\ldots a_n}$, где $\mathbf{a_i \in A}$. Все множество значений данных **X** составляет язык $\mathbf{L_X=\{x_i \mid i \in I\}}$, где **I** – номерное множество.

Основываясь на анализе языка $\mathbf{L_X}$, АИС должна разработать эффективную грамматику $\mathbf{G_X=(T, N, P, S)}$, удовлетворяющую условию $\mathbf{L_X=L(G_X)}$, где $\mathbf{T=A}$ – множество терминальных символов; $\mathbf{N=\{\nu_j \mid j \in J\}}$ – множество нетерминальных символов; $\mathbf{P=\{p\}}$ – множество правил подстановки, заданных в виде продукций $\mathbf{p: x \to y}$, порождающих по слову **x** слово **y**; **S** – начальный символ грамматики, представляющий слова из $\mathbf{L_X}$ в $\mathbf{G_X}$ [3]. Для обеспечения высокой эффективности алгоритмов порождения грамматики $\mathbf{G_X}$ и обработки данных на ее основе целесообразно строить грамматику $\mathbf{G_X}$ в форме контекстно-свободной грамматики.

Для осуществления преобразования грамматики **G=(T, N, P, S)** необходимо привести множество ее правил **P** к виду

$$\begin{aligned}&\mathbf{P=\{S \to \nu_1\nu_2\ldots\nu_n}\\&\mathbf{p_1:\nu_1 \to \ldots}\\&\mathbf{p_2:\nu_2 \to \ldots}\\&\ldots\\&\mathbf{p_n:\nu_n \to \ldots}\\&\mathbf{\}}\end{aligned} \qquad (1)$$

Принципиальной особенностью грамматик вида (1) является то, что правая часть начального символа **S** состоит только из нетерминальных символов, каждый из которых имеет свою позицию (номер) и обладает семантическими и синтаксическими свойствами. Такая группировка правил отражает структуру порождаемых цепочек символов.

Пример 1. Рассмотрим грамматику $\mathbf{G_A}$, представленную в виде (1), которая порождает последовательности символов типа «адрес»:

$$\begin{aligned}\mathbf{G_A:}\quad &\mathbf{S \to \nu_1\nu_2\nu_3\nu_4\nu_5}\\&\mathbf{p_1:\nu_1 \to}\text{ул.}\\&\mathbf{p_2:\nu_2 \to}\text{Мира|Ленина|Минская...}\\&\mathbf{p_3:\nu_3 \to \nu}\\&\mathbf{p_4:\nu_4 \to}-\\&\mathbf{p_5:\nu_5 \to \nu}\\&\mathbf{p_6:\nu \to 0...9|\nu\nu|\nu\varepsilon}\end{aligned} \qquad (2)$$

где $\boldsymbol{\varepsilon}$ – пустая последовательность.

Преобразование грамматики **G** вида (1) в грамматику **G'** такого же вида зададим формулой:

$$\mathbf{G \xrightarrow{\theta} G} \text{ или } \mathbf{G'=\theta(G)}, \qquad (3)$$

где $\boldsymbol{\theta}$ – допустимая операция над грамматикой **G**.

Операции преобразования грамматик (3) предоставляют возможность задавать грамматику **G'** через грамматику **G** и наоборот. Таким образом, необязательно задавать обе грамматики с помощью четверок **(T, N, P, S)**, достаточно иметь одну грамматику и множество допустимых операций $\boldsymbol{\theta}$, чтобы сформировать требуемое разнообразие грамматик с определенными характеристиками.

Рассмотрим подробнее операции преобразования грамматик.

Замена цепочки символов – это операция получения грамматики **G'** из грамматики **G** вида (1) заменой правила $\mathbf{p_k}$ грамматики **G**, порождающего цепочку символов $\boldsymbol{\alpha}$, на правило $\mathbf{p_k'}$ грамматики **G'**, порождающего цепочку символов $\boldsymbol{\beta}$:

$$\mathbf{G'=\theta_{зам.}(G, p_k, p_k')}, \qquad (4)$$

где $\mathbf{p_k : \nu_k \to \alpha}$; $\mathbf{p_k' : \nu_k \to \beta}$.

Пусть имеется грамматика **G=(T, N, P, S)** вида (1), множество правил **P** которой содержит некоторое правило $p_k: \nu_k \to \alpha$, $p_k \in P$, $\nu_k \in N$, где α – некоторая цепочка, состоящая из символов, принадлежащих словарю **T**.

Для получения грамматики **G'** операция замены цепочки символов (4) выполняет следующие действия.

1. Изменяет множество правил **P**: $P'=P \setminus p_k \cup p_k'$.

Примечание. Множество правил **P'** может быть дополнено и другими правилами, необходимыми для вывода цепочки β;

2. Изменяет множество терминальных символов **T**:

– дополняет символами множества H_1, которые входят в цепочку β, но не входят во множество **T** ($H_1 \cap T=\varnothing$): $T'=T \cup H_1$;

– удаляет символы множества H_2, которые входят в цепочку символов α, но не порождаются другими правилами грамматики **G'**: $T'=T \setminus H_2$.

Таким образом, после применения операции (4) к грамматике **G** получим новую грамматику **G'=(T', N, P', S)**, которая будет порождать такие же цепочки символов, как грамматика **G**, но вместо цепочки α будет порождать цепочку β.

Пример 2. Операцию замены цепочки символов продемонстрируем на грамматике (2). Грамматика G_A порождает адреса в формате «ул. *улица №_дома-№_квартиры*» (например, «ул. Мира 12-34»). Курсивом в формате выделены поля, значения которых изменяются в различных порождаемых цепочках. Преобразуем грамматику G_A таким образом, чтобы новая грамматика G'_A порождала адреса в формате «улица *улица №_дома-№_квартиры*» (например, «улица Мира 12-34»): $G'_A = \theta_{зам.}(G_A, p_1, p_1')$, где $p_1 : \nu_1 \to$ **ул.**; $p_1' : \nu_1 \to$ **улица**.

Удаление цепочки символов, $\theta_{удал}$, является частным случаем операции замены цепочки символов, так как в качестве β выступает пустая цепочка символов ε:

$$G' = \theta_{удал.}(G, p_k) = \theta_{зам.}(G, p_k, p_k'), \quad (5)$$

где $p_k : \nu_k \to \alpha$; $p_k' : \nu_k \to \varepsilon$.

Для получения грамматики **G'** операция удаления цепочки символов (5) выполняет следующие действия:

1) изменяет множество правил **P**: $P'=P \setminus p_k \cup p_k'$;

2) изменяет множество терминальных символов путем удаления символов множества H_2: $T'=T \setminus H_2$.

Множество H_2 состоит из символов цепочки α, которые больше не порождаются другими правилами грамматики **G'**.

Таким образом, после применения операции (5) к грамматике **G** получим новую грамматику **G'=(T', N, P', S)**, которая будет порождать такие же цепочки символов, как грамматика **G**, за исключением цепочки α.

Пример 3. Преобразуем грамматику (2) таким образом, чтобы новая грамматика G'_A порождала адреса без цепочки «ул.»: $G'_A = \theta_{удал.}(G_A, p_1)$.

Множество правил **P'** грамматики G'_A примет вид:

$P'=\{S \to \nu_1\nu_2\nu_3\nu_4\nu_5$
$p_1' : \nu_1 \to \varepsilon$
$p_2 : \nu_2 \to$ Мира\|Ленина\|Минская...
$p_3 : \nu_3 \to \nu$
$p_4 : \nu_4 \to -$
$p_5 : \nu_5 \to \nu$
$p_6 : \nu \to 0...9 \| \nu\nu \| \nu\varepsilon$
$\}$

Вставка цепочки символов, $\theta_{встав}$, позволяет получить грамматику **G'** путем добавления цепочки символов α в позицию **k** начального символа **S** грамматики **G**:

$$G' = \theta_{встав}(G, \nu', k). \quad (6)$$

Таким образом, получим грамматику **G'=(T', N', P', S')**, в которой:

1) в позиции **k** начального символа **S** установлен нетерминальный символ ν';

2) множество терминальных символов **T** дополнено множеством символов **H**, входящих в цепочку α ($H \cap T=\varnothing$): $T'=T \cup H$;

3) множество нетерминальных символов дополнено символом ν', порождающим цепочку α:

$N'=N \cup \{\nu'\}$;

4) множество правил **P** изменено следующим образом:

$P'=\{S \to \nu_1\nu_2 ... \nu' ... \nu_n$
$p_1 : \nu_1 \to ...$
$p_2 : \nu_2 \to ...$
$...$
$p_n : \nu_n \to ...$
$p' : \nu' \to \alpha$
$\}$

Примечание. Множество правил **P'** может быть дополнено и другими правилами, необходимыми для вывода цепочки α.

Пример 4. Изменим грамматику (2) таким образом, чтобы она порождала адреса, в которых вначале указывается почтовый индекс:

$G'_A = \theta_{встав}(G_A, \nu_{индекс}, 1)$.

В новой грамматике G'_A в начальном символе **S** первым нетерминальным символом будет задан $\nu_{индекс}$. Множество терминальных символов **T** останется неизменным, так как почтовые индексы состоят только из цифр, которые уже содержатся во множестве **T**. Изменится множество нетерминальных символов **N**: $N'=N \cup \{\nu_{индекс}, \nu'\}$, а множество правил **P** примет вид:

$P'=\{S'\rightarrow\nu_{индекс}\nu_1\nu_2\nu_3\nu_4\nu_5$
$p_1:\nu_1\rightarrow$**ул.**
$p_2:\nu_2\rightarrow$Мира|Ленина|Минская...
$p_3:\nu_3\rightarrow\nu$
$p_4:\nu_4\rightarrow-$
$p_5:\nu_5\rightarrow\nu$
$p_6:\nu\rightarrow$**0...9**$|\nu\nu|\nu\varepsilon$
$p':\nu_{индекс}\rightarrow\nu'\nu'\nu'\nu'\nu'\nu'$
$p_1':\nu'\rightarrow$**0...9**
$\}$

Модификация числовых цепочек позволяет получить грамматику **G'** из грамматики **G** вида (1) путем изменения цепочки символов **x**, являющейся числовой переменной и порождаемой нетерминальным символом ν_k правила p_k, на цепочку символов **y**, порождаемую из **x** по закону **f**:

$$G'=\theta_{ч\_мод}(G,p_k,p_k',f), \quad (7)$$

где $p_k:\nu_k\rightarrow\alpha$, ν_k**.val=x;** p_k' **:** $\nu_k\rightarrow\beta$, ν_k**.val=y**; **y=f(x)**.

Обозначение ν_k**.val**, приписывающее значение параметра **val** нетерминальному символу ν_k, будем рассматривать как расширение контекстно-свободной грамматики Хомского, называемое атрибутной грамматикой. Атрибутная грамматика позволяет каждому символу (терминальному и нетерминальному) порождающей контекстно-свободной грамматики приписать конечное число семантических параметров – атрибутов, которые принимают некоторые значения (они могут быть целыми, символьными цепочками, признаками, матрицами значений и т.п.) [4].

Допустимы следующие операции числовых преобразований, используемые в качестве закона преобразования **f**:

- сложение: **y=x+k**;
- вычитание: **y=x-k**;
- умножение: **y=x*k**;
- деление: **y=x/k**;
- тригонометрические функции: например, **y=sin(x)**;
- другие математические операции над числами.

Здесь **k** – произвольный коэффициент, учитываемый в законе **f**.

После применения операции (7) к грамматике **G** будет получена грамматика **G'=(T, N, P', S)** со множеством правил **P'**: $P'=P\backslash p_k\cup p_k'$.

Пример 5. Рассмотрим грамматику $G_Д$**=(T, N, P, S)** вида (1), порождающую даты 21-го столетия, в которой запись года представлена в сокращенном формате (01, 09, 14 и т.д.):

$$\begin{aligned} G_Д: \quad & S\rightarrow\nu_1\nu_2\nu_3\nu_4\nu_5 \\ & p_1:\nu_1\rightarrow 1|2|...|31 \\ & p_2:\nu_2\rightarrow . \\ & p_3:\nu_3\rightarrow 1|2|...|12 \\ & p_4:\nu_4\rightarrow . \\ & p_5:\nu_5\rightarrow 00|01|...|99 \end{aligned} \quad (8)$$

Преобразуем грамматику $G_Д$ при помощи операции (7) таким образом, чтобы запись года была представлена в полном формате (например, 2001, 2009 и т.д.): $G_Д'=\theta_{ч\_мод}(G_Д,p_5,p_5',2000+\nu_5.val)$.

Изменение порядка следования цепочек позволяет получить грамматику **G'** из грамматики **G** вида (1) путем взаимного изменения позиций двух нетерминальных символов в начальном символе **S** грамматики **G**:

$$G'=\theta_{порядк.}(G,\nu_k,\nu_\ell). \quad (9)$$

Пусть задана грамматика **G=(T, N, P, S)** вида (1) со множеством правил **P**:

$P=\{S\rightarrow\nu_1\nu_2...\nu_k...\nu_\ell...\nu_n$
$p_1:\nu_1\rightarrow...$
$p_2:\nu_2\rightarrow...$
$...$
$p_k:\nu_k\rightarrow\alpha$
$...$
$p_\ell:\nu_\ell\rightarrow\beta$
$...$
$p_n:\nu_n\rightarrow...$
$\}$

Для получения грамматики **G'** операция изменения порядка следования цепочек (7) модифицирует только начальный символ **S**, переставляя местами нетерминальные символы ν_k и ν_ℓ.

Рассмотрим комплексное преобразование грамматики **G** с помощью множества допустимых операций преобразования **Θ**.

Пример 6. Модифицируем грамматику (8), порождающую даты в формате «дд.мм.гг» (европейский формат), таким образом, чтобы новая грамматика порождала даты в формате «мм/дд/гг» (американский формат). Для этого используем множество операций **Θ**: $\Theta=\{\theta_{порядк}, \theta_{зам}\}$.

Применим к грамматике $G_Д$ следующую последовательность операций:

$$[\theta_{порядк}(G_Д,\nu_1,\nu_3)\circ\theta_{зам}(G_Д,p_2,p_2')\circ\theta_{зам}(G_Д,p_4,p_4')],$$

где p_2' **:** ν_2 **= /** , p_4' **:** ν_4 **= /** и получим грамматику $G_Д'$ вида: $G_Д'$**=(T', N, P', S')**, в которой множество терминальных символов $T'=T\backslash\{.\}\cup\{/\}$, а множество правил P':

$P'=\{S\rightarrow\nu_3\nu_2\nu_1\nu_4\nu_5$
$P_1:\nu_1\rightarrow 1|2|...|12$
$P_2:\nu_2\rightarrow /$
$P_3:\nu_3\rightarrow 1|2|...|31$
$P_4:\nu_4\rightarrow /$
$P_5:\nu_5\rightarrow 00|01|...|99$
$\}$

Таким образом, в работе дано развернутое определение формата данных и перечислены его свойства. Рассмотрена формальная составляющая формата – грамматика, и определено множество основных операций преобразования грамматик.

Литература

1. Дрождин В.В., Баканов А.Б. Грамматика описания домена фамилий // Вопросы радиоэлектроники. Сер.: Электрон-

ная вычислительная техника. 2007. Вып. 1. С. 77–82.

2. Маркус С. Теоретико-множественные модели языков. М.: Наука, 1970. 332 с.

3. Линьков В.М., Дрождин В.В., Лушникова Е.В. Порождение грамматики описания данных в информационных доменно-ориентированных средах // Проблемы информатики в образовании, управлении, экономике и технике: сб. стат. III Всеросс. науч.-технич. конф. Пенза, 2003. С. 8–11.

4. Карпов Ю.Г. Теория и технология программирования. Основы построения трансляторов. СПб: БХВ-Петербург, 2005.

КЛАСТЕРЫ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ТРЕНАЖЕРНО-МОДЕЛИРУЮЩЕГО КОМПЛЕКСА В ЗАДАЧЕ АГРЕГАЦИИ ФРАКТАЛОВ

В.В. Янюшкин

(Донской филиал Центра тренажеростроения, г. Новочеркасск, vadim21185@rambler.ru)

Рассматривается концепция построения и работы современных распределенных информационных систем тренажерно-моделирующих комплексов, при этом особое внимание уделено перспективным методам представления структур системы в приложении к задаче оптимизации размещения данных на основе эволюционного моделирования и фрактальных кластерных агрегатов.

Ключевые слова: *кластер, фрактал, самоподобие, агрегация, модели представления данных, распределенное информационное пространство, сервис данных.*

Многие задачи подготовки специалистов успешно решаются с помощью компьютерных тренажерных систем. Создание таких сложных программно-аппаратных комплексов требует поддержания необходимой функциональности распределенных по узлам вычислительной сети моделей и взаимосвязанных данных. Для решения этих задач создается распределенная информационная система, которая обеспечивает своевременное поступление исходных данных и команд моделям, передачу результатов моделирования средствам имитации, а также размещение данных модельного мира в информационной системе.

При этом существуют задачи проектирования компонент, обеспечивающих такое распределение информации в системе, которое позволяло бы минимизировать временные характеристики при получении необходимых входных данных для потребителей тренажера, сокращать совокупный объем для хранения информационных массивов, соблюдая при этом высокие требования, накладываемые функционированием системы реального времени.

Формализация и постановка задачи. Описание структуры тренажера можно представить в виде иерархии нескольких уровней (рис. 1):

– уровень аппаратной конфигурации, который представлен компьютерами и другим оборудованием тренажера, а также набором характеристик, таких как объем оперативной памяти, скорость передачи данных, частота процессора, объем видеопамяти и т.д.;

– уровень функциональной конфигурации, характеризующий распределение программного обеспечения и моделей тренажера по узлам вычислительной сети, то есть по элементам аппаратной конфигурации; при этом каждая модель имеет определенный состав данных, необходимых для ее функционирования и правильной работы приложений;

– уровень информационной модели, на котором представлен состав данных, необходимых для функционирования системы в целом и отдельных приложений и моделей в частности.

Предложенная иерархическая организация архитектуры может являться практическим примером и попыткой декомпозиции сложной системы, на основе которой удобно рассматривать функциональные составляющие системы и их взаимодействие. Наибольшее внимание в данной работе уделяется верхнему уровню организации системы. Например, дальнейшая декомпозиция данного уровня при использовании подхода *Service-Oriented Architecture* (*SOA*) и современной технологии *Windows Communication Foundation* (*WCF*) может быть такой, как представлено на рисунке 2, из которого видно, что именно массивы данных, конкретные объекты и значения их атрибутов являются основой формирования распределенного общетренажерного ресурса, находясь на вершине

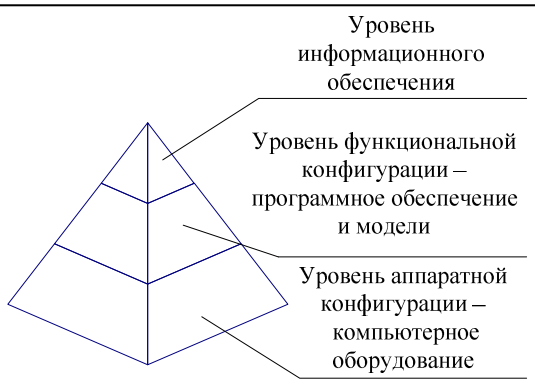

Рис. 1. Структура тренажера в виде иерархии уровней