компьютер в учебном процессе

УДК 004.4'236

Зорин Юрий Алексеевич,
Посов Илья Александрович

ИНСТРУМЕНТАЛЬНЫЕ СИСТЕМЫ ПОСТРОЕНИЯ И ПОЛУЧЕНИЯ МНОГОВАРИАНТНЫХ ТЕСТОВЫХ ЗАДАНИЙ

Аннотация

Рассмотрены проблемы, возникающие при построении алгоритмов генерации тестовых заданий. В статье затронуты вопросы, связанные с инструментальными средствами, позволяющими вести разработку генераторного типа задач. На примерах конкретных задач рассмотрены подходы к разработке многовариантных тестовых заданий на базе двух систем (Платан, Possum), разработчиками которых и являются авторы статьи.

Ключевые слова: генерация заданий, алгоритмы генерации, инструментальные средства, Платан, Possum.

ВВЕДЕНИЕ

Разработка тестовых заданий и обеспечение контроля знаний является неотъемлемой частью процесса создания и поддержки учебно-методических комплексов. Современные тенденции в образовании предъявляют все более жесткие требования к тестовым заданиям, которые затрагивают не только качество поставляемого контента, но и количество поставляемых вариантов в сторону их увеличения.

Применение компьютерных средств при создании многовариантных тестовых заданий существенно сокращает сроки разработки, но требует от разработчика не только хорошего знания предметной области, но и, как правило, понимания алгоритмических конструкций, а в отдельных случаях – знания того или иного языка программирования.

Создание генератора многовариантного тестового задания представляет собой нетривиальную задачу, так как требует индивидуального подхода к каждому разрабатываемому генератору, а в частных случаях требует применения различных программных средств для их разработки.

МНОГООБРАЗИЕ АВТОМАТИЗИРОВАННЫХ СРЕДСТВ СОЗДАНИЯ ГЕНЕРАТОРОВ ТЕСТОВЫХ ЗАДАНИЙ

Задачи, которые приходится решать при разработке многовариантных тестовых заданий с применением компьютерных программ, многообразны по своим предметным областям, и для их решения требуются разные методики и подходы, что нашло свое непосредственное отражение в конкретных автоматизированных средствах. Каждое программное средство основывается на одной или нескольких таких методиках. Разные

подходы при разработке генераторов тестовых заданий закладывали в основу разных программных средств, предназначенных для разработки определённого типа генераторных задач.

Отсутствие общепринятых стандартов разработки генераторов в учреждениях привело к тому, что создание программного обеспечения (ПО), поддерживающего разработку многовариантных заданий, осуществляется в стенах той организации, где эти задания находят свое применение. Выходу подобных приложений в открытый доступ препятствуют ограничения, наложенные спецификой проведения тестирования в том или ином образовательном учреждении. Подтверждением данного факта может служить система генерации задач «Фея» Томского университета систем управления и радиоэлектроники, позволяющая разработчику описывать алгоритмы генерации обширного класса генераторных задач [1]. Выходу данного приложения за стены университета препятствует отсутствие пользовательского интерфейса приложения, а также особенности формата получаемых тестовых заданий, который используется только одной системой тестирования этого же университета.

Помимо разрабатываемого ПО для построения задач генераторного типа, имеются мощные математические приложения (MathCad, Mathematica) [2, 3], обладающие богатыми математическими возможностями, с возможностью описания алгоритмических конструкций, которые, в свою очередь, позволяют описывать генераторные задачи. Распространению генераторов на основе этих приложений мешает высокая стоимость их лицензии. Программные решения с полноценным пользовательским интерфейсом, которые позволяют вести разработку многовариантных тестовых заданий (Moodle, Schoolhouse Test) и обладают возможностями в создании алгоритмов генерации, ограниченны перемешиванием массива ответов и подстановкой численных параметров [4, 5].

Исследования авторов данной статьи вылились в реализацию двух независимых систем разработки генераторных задач. Система разработки многовариантных тестовых заданий на базе языка Possum и система Платан, помимо их внедрения в технологические процессы некоторых образовательных учреждений, претендуют на некую универсальность. В отличие от рассмотренных аналогов, эти системы расширяют возможности построения алгоритмов генерации и не ограничивают разработчика в выборе формы проведения контроля знаний на основе полученных вариантов заданий.

Целью данной статьи является обзор систем разработки многовариантных тестовых заданий Платан и Possum на примерах разработки реальных генераторных задач, демонстрация их функциональных возможностей и процесса построения генераторов.

Подходы к построению алгоритмов генерации описаны в соответствующей литературе: система разработки многовариантных заданий на языке Possum [6], система построения генераторов тестовых заданий на основе деревьев И/ИЛИ [7].

ЗАДАНИЕ ПО АЛГЕБРЕ

Разработка генераторов для решения квадратных уравнений вида

$$ax^2 + bx + c = 0$$

является достаточно тривиальной. Основная задача данного генератора – сформировать численные коэффициенты a, b, c, а также вычислить ответ, используя формулу вычисления корней квадратного уравнения. Усложним задачу построения генератора требованиями формирования численных коэффициентов таким образом, чтобы все арифметические вычисления при вычислении корней квадратного уравнения укладывались в таблицу умножения 10×10, а результатом вычисления дискриминанта являлось целое число. В данном случае на помощь приходит теорема Виета для вычисления корней квадратного уравнения. Согласно теореме Виета, генерацию квадратного уравнения необходимо начать с формирования x_1, x_2, которые являются корнями уравнения, а также один из коэффициентов, например b.

Табл. 1. Значения узлов

Номера	Назначение узлов
1, 2, 3	Узлы определения начальных данных. Узел 1 раскрыт на рис. 2
4, 5	Условные узлы. Истинное выражение – левая ветвь, ложное – правая
6	Узел сообщает генератору, что необходимо пропустить данный вариант и начать генерацию заново
7, 8	Функциональные узла, сообщающие форматы вывода узлов, находящихся ниже. Latex – формула в формате Latex, easy – упрощение математического выражения

Представленное на рис. 1 дерево И/ИЛИ описывает 3200 вариантов данной задачи. Пример одного из полученных вариантов:

Решите квадратное уравнение: $-x^2 - x + 2 = 0$.

Представление данного генератора на языке Possum показано в листинге 1.

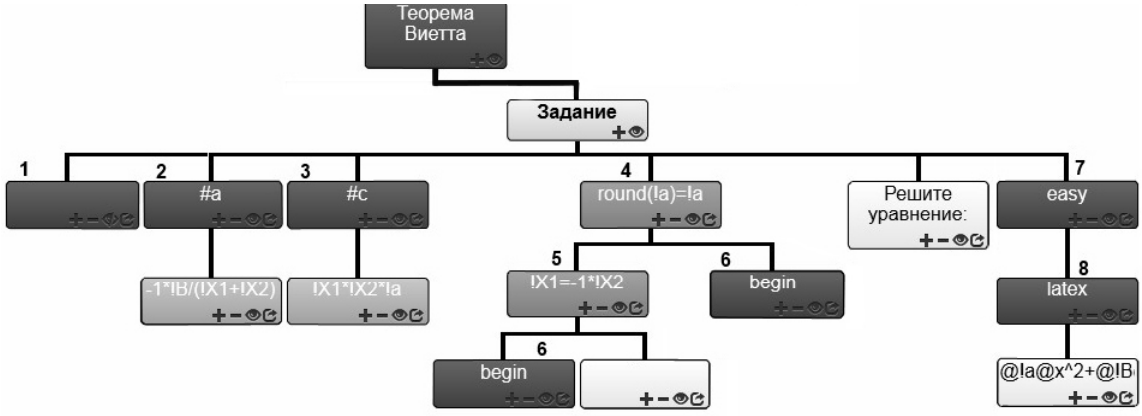

Рис. 1. Генератор задачи квадратного уравнения в системе Платан

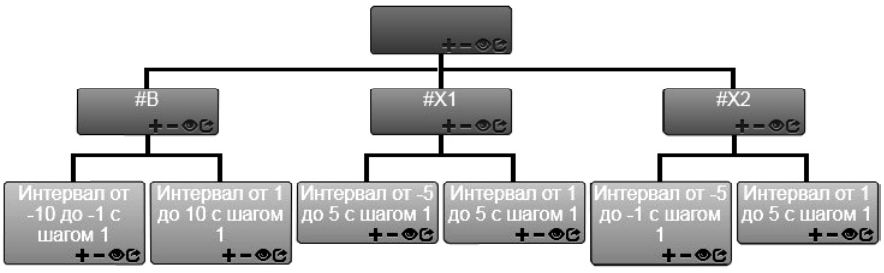

Рис. 2. Узел определения начальных данных

Листинг 1

```
info({'description' : 'Квадратное уравнение с вычислениями в уме'});
statement = 'Решите квадратное уравнение $%p = 0$';
answer = '$x_1 = %x1$, $x_2 = %x2$';
x1 = rnd_not_0(-5, 5);
x2 = rnd_not_0(-5, 5);
assert(x1 != x2 && x1 != -x2);
// Если корни целые, то все коэффициенты делятся на a, поэтому в качестве случайного
коэффициента удобней всего брать a.
a = rnd(1, 3);
p = tex$('expand(%a*(x - %x1)*(x - %x2))');
```

ЗАДАНИЕ ПО ФИЗИКЕ

Часто под генерацией задания понимается не только подстановка численных параметров и вычисление ответа, но и подстановка символьных конструкций или некоторых слов (словосочетаний, предложений), которые также будут влиять как на ответ, так и на алгоритм его вычисления. В представленном ниже тексте задания выделены слова, которые могут принимать участие при генерации вариантов.

Выберите вариант для заполнения пропуска.

*При **уменьшении жесткости** пружинного маятника в **13** раз при неизменной **массе** период*_____.
Точность вычисления – десятые доли.

Синтаксическое описание алгоритма генерации данной задачи на языке Possum выглядит следующим образом (см. листинг 2).

Для визуального представления алгоритма генерации в системе Платан необходимо понять формулу вычисления периода пружинного маятника, в которой период прямо пропорционален корню из массы груза и обратно пропорционален корню из коэффициента жесткости пружины. Алгоритм генерации также должен исключать неверно сформулированные варианты, связанные с повторением слов («жесткости», «массе») в условии задания и в постановке вопроса. Дерево, описывающее алгоритм генерации условия задания, представлено на рис. 3.

При формировании генератора условия задания также формируются некоторые условно-именованные узлы, которые будут использоваться при вычислении ответа (см. рис. 4). Возможные варианты представлены на рис. 5.

ЗАДАНИЕ ПО МАКРОЭКОНОМИКЕ

При разработке алгоритмов генерации для различного класса дисциплин необходимо учитывать, что под формированием многовариантных заданий понимается не только перебор численных параметров или определенных конструкций слов, но и выдачу

Листинг 2

```
info({'description' : 'Период колебания пружинного маятника'});
constant_value = rnd_choose('weight', 'hardness');
change = rnd_choose('inc', 'dec');
value = rnd(4, 30);
// перечисление ответов
function create_answer(change, value) {
    return (change == 'inc' ? 'увеличится' : 'уменьшится') + ' в ' + value.toFixed(1) +
                                                                                ' раз';
}
// здесь пропущена генерация правильного ответа и всех возможных неправильных;
answers = rnd_subset(3, wrong_answers);
answers.push(right_answer);
answers = rnd_perm(answers);
statement = ('При %change %changing_value пружины пружинного маятника в %value раз при
неизменной %constant_value пружины период колебаний \\underline{\\hskip 2cm}. Ответ
указывается с точностью  до десятых долей.').$({
        'change': change == 'inc' ? 'увеличении' : 'уменьшении',
        'changing_value': constant_value == 'weight' ? 'жесткости' : 'массы',
        'constant_value': constant_value == 'weight' ? 'массе' : 'жесткости',
        'value': value
});
statement += answers.$('\\begin{enumerate}\\item %{}\\item %{}\\item %{}\\item
%{}\\end{enumerate}');
answer = '%right_answer';
```

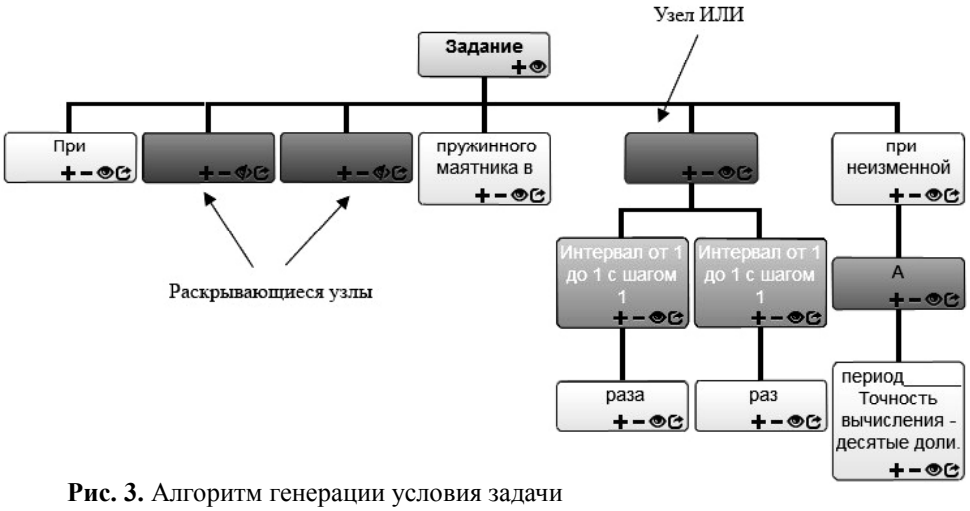

Рис. 3. Алгоритм генерации условия задачи

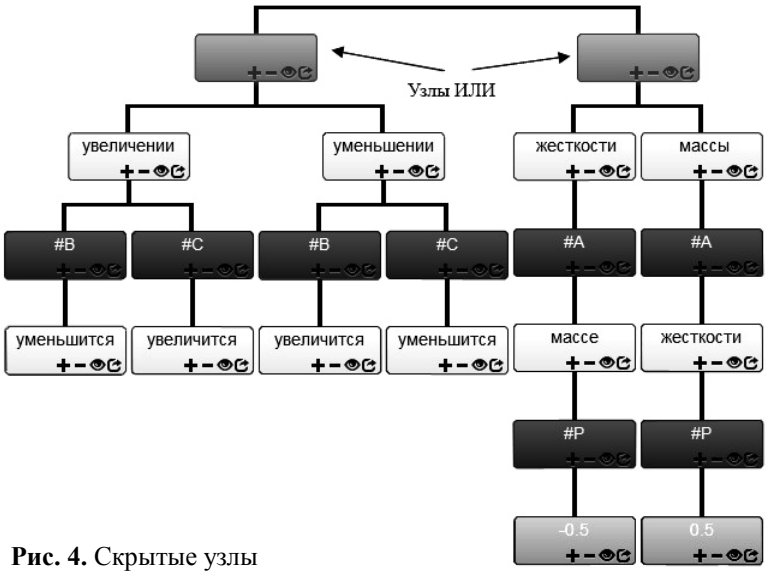

Рис. 4. Скрытые узлы

Вар. 5	**Вар. 6**
При увеличении жесткости пружины пружинного маятника в 24 раз при неизменной массе пружины период колебаний _____. Ответ указывается с точностью до десятых долей. **1.** уменьшится в 24.0 раз **2.** увеличится в 24.0 раз **3.** не изменится **4.** уменьшится в 4.9 раз	При увеличении массы пружины пружинного маятника в 18 раз при неизменной жесткости пружины период колебаний _____. Ответ указывается с точностью до десятых долей. **1.** не изменится **2.** уменьшится в 18.0 раз **3.** увеличится в 4.2 раз **4.** увеличится в 18.0 раз

Рис. 5. Возможные варианты заданий

обучаемому случайного подмножества из заданного в генераторе массива данных. При построении генератора многовариантного задания по дисциплине «Макроэкономика» обучаемому случайным образом выдается две страны, из которых требуется выбрать страны с развитой экономикой. Правильных ответов может быть от 1 до 2. На языке Possum львиная часть кода уходит на описание массива развитых/неразвитых стран. Имеющиеся функции случайной генерации см в листинге 3.

Для системы «Платан» массивы стран представлены в виде двух узлов ИЛИ (рис. 6, 7, 8).

Листинг 3

```
info({description: 'Развивающиеся и развитые страны'});
question = rnd_choose('developed', 'developing');
all_developed = [    'США',    'Канада',    'Италия',    'Франция',    'Великобритания',
'Австралия',    'Германия',    'Дания',    'Ирландия',    'Испания',    'Мальта',
'Япония',    'Словения',    'Бельгия',    'Чехия',    'Португалия',    'Сингапур'];
all_developing = [    'Россия',    'Казахстан',    'Белоруссия',    'Гаити',    'Гвинея',
'Судан',    'Чад',    'Афганистан',    'Лаос',    'Непал',    'ЮАР',    'Бразилия',
'Нигерия'];
num_right = rnd(1, 4);
num_wrong = 4 - num_right;
right_answers = rnd_subset(num_right, question == 'developing' ? all_developing :
all_developed);
wrong_answers = rnd_subset(num_wrong, question == 'developing' ? all_developed :
all_developing);
all_answers = rnd_perm(right_answers.concat(wrong_answers));
question_as_text = question == 'developing' ? 'развивающейся' : 'развитой';
statement = 'Выберите страны с %question_as_text экономикой: ';
statement += all_answers.$('\\begin{enumerate}\\item %{}\\item %{}\\item %{}\\item
%{}\\end{enumerate}');
answer = right_answers.join(', ');
```

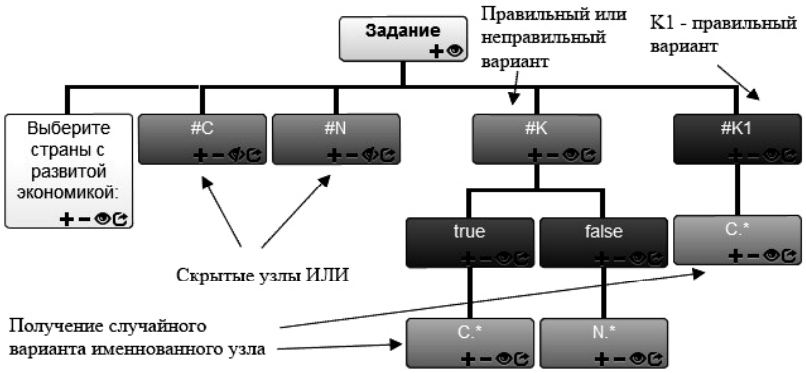

Рис. 6. Формирование массива стран с развитой и развивающейся экономикой

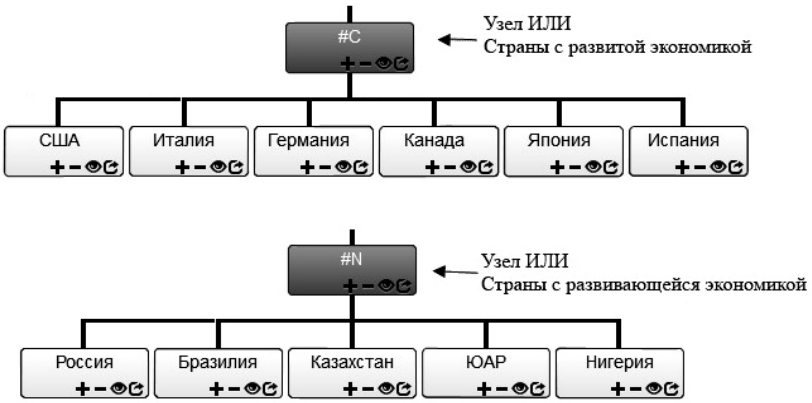

Рис. 7. Скрытые узлы ИЛИ

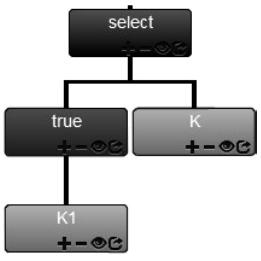

Рис. 8. Получение вариантов ответа

ЗАДАНИЕ ПО МАТЕМАТИЧЕСКОЙ ЛОГИКЕ

Дано случайное логическое выражение, требуется определить его дизъюнктивно-нормальную форму, многочлен Жегалкина и другие свойства.

Описание алгоритма генерации данной задачи проблематично для системы «Платан», так как отсутствует поддержка средств компьютерной алгебры. Язык Possum, наоборот, справляется с задачей путем подключения функций системы MAXIMA. Следует отметить, что реализация указанной задачи требует глубокого знания возможностей MAXIMA, чтобы добавить в нее недостающие логические операции и алгоритмы, в других системах компьютерной алгебры реализация, возможно, была бы проще. Особо отметим существующую в языке возможность генерации случайного слова в КС-грамматике с помощью функции **`generate_term()`** (см. листинг 4). Возможные варианты представлены на рис. 9.

Листинг 4

```
info({description: 'logical functions', latex_packages: ['amsmath']});
statement = 'Определите ДНФ, многочлен Жегалкина для функции $%f$, ' +
    'проверьте функцию на принадлежность классам замкнутости.';
answer = '-';
//====================================================================
// задать значения всем нестандартным логическим функциям
// т.к. в maxima нет xor, =>, <=>
meval('infix(xor, 55, 55, clause, clause, clause)');
meval('(a xor b) := not a and b or not b and a');
//научить новые функции отображаться в TeX
meval('texput('xor', ' \\\\oplus ', infix)');
// аналогично необходимо создать другие операции
//====================================================================
formula = generate_term({
    's': [
        ['(', Nonterm('s'), 'and', Nonterm('s'), ')'],
        ['(', Nonterm('s'), 'or', Nonterm('s'), ')'],
        ['(', Nonterm('s'), 'xor', Nonterm('s'), ')'],
        ['(', Nonterm('s'), 'impl', Nonterm('s'), ')'],
        ['(', Nonterm('s'), 'equiv', Nonterm('s'), ')'],
        ['(', 'not', Nonterm('s'), ')'],
        [Term('X')],
        [Term('Y')],
        [Term('Z')]
    ]
}, 's', 5).join(' ');
//построить Многочлен Жегалкина
f = tex_no_eval(formula);
//далее идут дополнительные проверки качества построенной формулы и вызовы вычислений
ДНФ и требующихся выражений.
```

Вар. 35 Определите ДПФ, многочлен Жегалкина для функции $(Z \oplus X]) \wedge X \wedge (Z \Rightarrow Y)$, проверьте функцию на принадлежность классам замкнутости.	**Вар. 36** Определите ДПФ, многочлен Жегалкина для функции $Y \wedge (Y \Rightarrow (X \equiv Z \vee Y))$, проверьте функцию на принадлежность классам замкнутости.
Вар. 37 Определите ДНФ, многочлен Жегалкина для функции $X \wedge (Y \Rightarrow Z) \wedge X \equiv Y$, проверьте фупкцию па припадлежпость классам замкпутости.	**Вар. 38** Определите ДНФ, многочлен Жегалкина для функции $Z \wedge (X \oplus (Z \equiv X) \Rightarrow Y)$, проверьте фупкцию па припадлежпость классам замкпутости.

Рис. 9. Возможные варианты заданий

ЗАДАНИЕ НА ВЫЧИСЛЕНИЕ НОД

Усложним задачу генератора при формировании чисел таким образом, чтобы ответ не был равен единице, а количество шагов алгоритма Евклида было от 4 до 6.

Сложность реализации данного алгоритма в системе «Платан» вызывает отсутствие реализации циклических операций на деревьях И/ИЛИ. Данный недостаток устраняется за счет возможности вставки JavaScript функций в узлы дерева (рис. 10).

В скрытом узле «znacheniya» определяются начальные данные для функции «nod» (рис. 11).

Проверка вычисленного значения НОД на неравенство единице, и количества шагов алгоритма Евклида определены в узле «usloviya» (рис. 12). Узел «begin» оповещает систему, что вариант задания не подходит и необходимо продолжить генерацию.

На языке Possum данный алгоритм реализуется на одной функции (см. листинг 5).

ЗАДАНИЕ ПО АУДИОТЕХНИКЕ

На практике часто используются задачи, в которых необходимо правильно выбрать

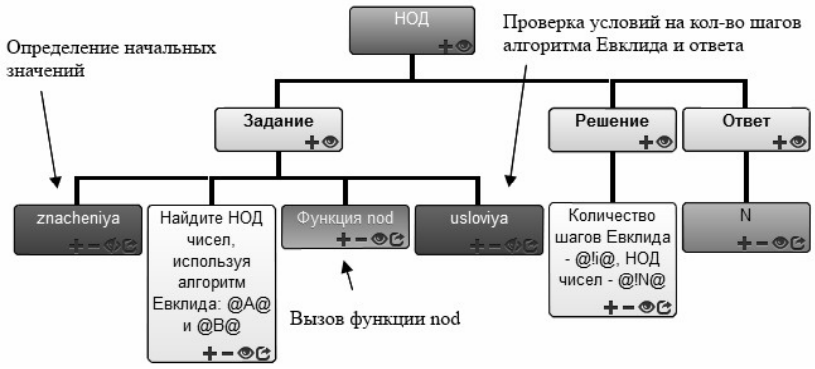

Рис. 10. Алгоритм генерации вычисления НОД на дереве И/ИЛИ

```
1  //Инициализируем переменные. A и B определеются случайным образом в узле znacheniya.
2  //i - кол-во шагов алгоритма Евклида определется в нуль в узле znacheniya.
3  var max = 0, min = 0, o = 0;
4  //Определяем максимальное число
5  if(A > B) {max = A; min = B;}else {max = B; min = A;}
6
7  //Остаток от деления максимального числа от минимального
8  o = max % min;
9
10 //Находим НОД, считаем кол-во шагов Евклида
11 while(o > 0)
12 {
13     max = min;
14     min = o;
15     o = max % min;
16     i++;
17 }
18 N = min;
19 //Переменные N - НОД, A, B, i - кол-во шагов алгоритма Евклида
20 //доступны в дереве  И/ИЛИ как именованные узлы
```

Рис. 11. Функция nod на языке JavaScript

термин из списка определений или определение из списка терминов (табл. 2).

При разработке алгоритма генерации данной задачи количество вариантов будет зависеть от количества имеющихся терминов, в данном случае их 8. Сам алгоритм в системе «Платан» достаточно громоздкий, но при понимании принципа позволяет в дальнейшем быстро добавлять новые определения и термины, получая, тем самым, новые варианты и увеличивая мощность генератора. Для начала необходимо определить варианты терминов и их определений в древовидной структуре И/ИЛИ (рис. 13). Условно-именованный узел ИЛИ «Т» позволяет инициализировать узлы «о» и «v» в зависимости от выбранного варианта.

Алгоритм генерации данной задачи состоит в произвольном выводе вариантов и выборе правильного. На рис. 14 представ-

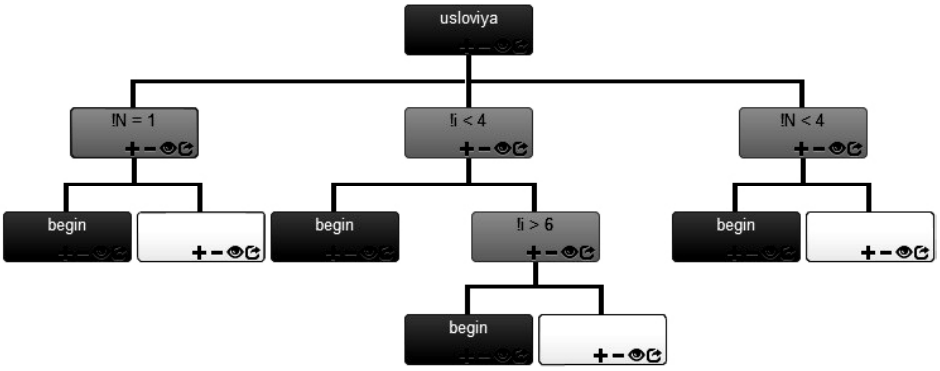

Рис. 12. Валидация варианта

Листинг 5

```
info({'description' : 'Вычисление наибольшего общего делителя'});
statement = 'Чему равен НОД чисел $%a$ и $%b$?';
answer = '$НОД(%a, %b)=%gcd$, шагов: $%steps$';
a = rnd(100, 999);
b = rnd(100, 999);
res = gcd_and_steps(a, b);
gcd = res['gcd'];
steps = res['steps'];
assert(gcd != 1 && 4 <= steps && steps <= 6);
function gcd_and_steps(a, b) {
    var steps = 0;
    while (a > 0 && b > 0) {
        if (a > b)
            a = a % b;
        else
            b = b % a;
        steps ++;
    }

    return {
        'gcd': a + b,
        'steps': steps
    };
}
```

Табл. 2. Термин-определение

Термин	Определение
ЭМИ	Любой музыкальный инструмент, в котором звук создается в результате генерирования, усиления и преобразования электрических сигналов с помощью электроники
Терменвокс	Электромузыкальный инструмент, в котором высота звука изменяется в зависимости от расстояния правой руки исполнителя до одной из антенн, а громкость – от расстояния левой руки до другой антенны
Эмиритон	Электромузыкальный инструмент, снабженный клавиатурой фортепианного типа
Музыкальный синтезатор	Электронный блок, который создает сложный сигнал путем комбинации цифровых импульсов, представляющих образы звука
Одноголосый ЭМИ	Электромузыкальный инструмент, имеющий только один генератор, который поет только одним голосом без возможности аккордов
Многоголосый ЭМИ	Электромузыкальный инструмент, в котором одновременно может работать большое число источников сигнала, возможны различные звукосочетания, аккорды
Электронный барабан	Электронное устройство, типа контура ударного возбуждения, генерирующее импульсный сигнал с крутым фронтом, пологим затухающим спадом, заполненный колебаниями почти синусоидальной формы частотой 100...400 Гц
Приставка к электрогитаре	Электронное устройство, выполняющее различные эффекты, типа «вау», «бустер», «дистошн», «тремоло» и другие, при исполнении мелодии, за счет изменения фронтов, спадов импульсов, модулирования звуковых сигналов и т.п.

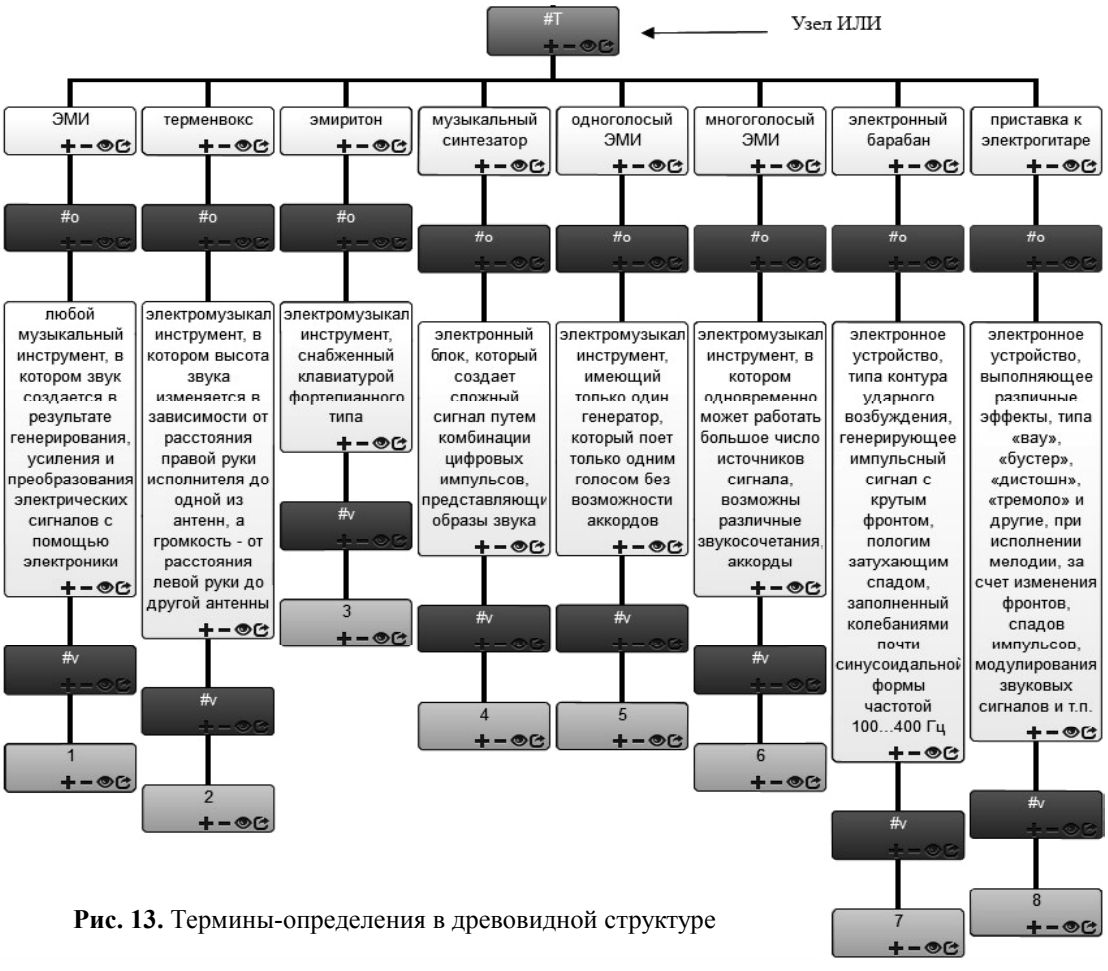

Рис. 13. Термины-определения в древовидной структуре

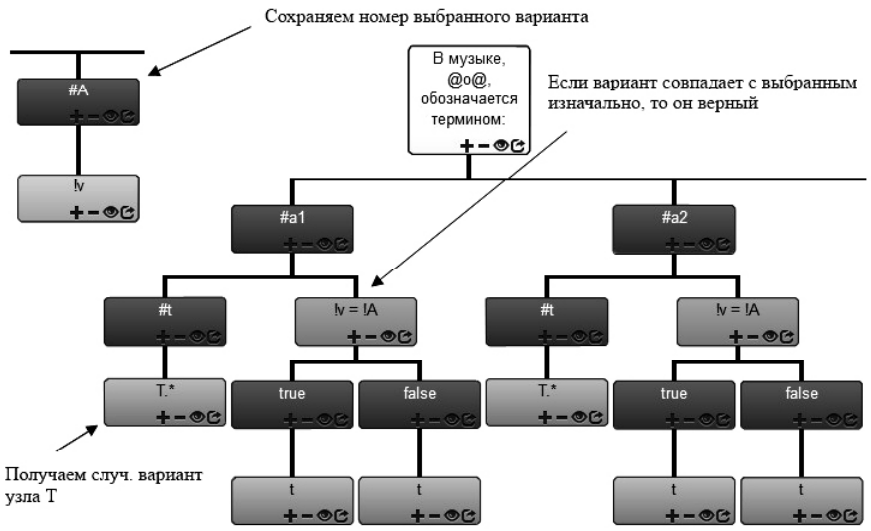

Рис. 14. Выбор вариантов и определение правильного варианта

Листинг 6

```
info({description: 'Аудиотехника'});

objects = [
    'ЭМИ',
    'Терменвокс',
    'Эмиритон',
    'Музыкальный синтезатор',
    'Одноголосый ЭМИ',
    'Многоголосый ЭМИ',
    'Электронный барабан',
    'Приставка к электрогитаре'
];
descriptions = [ {Массив определений}];
//выбираем вопрос из questions, выбираем варианты из descriptions
if (coin()) {
    questions = objects;
    variants = descriptions;

   statement = 'Для музыкальных электронных инструментов термином '%text' обозначается:';
} else {
    questions = descriptions;
    variants = objects;

    statement = 'В музыке, %text, обозначается термином:';
}
ind = rnd(0, objects.length - 1);
text = questions[ind];

right_variant = variants[ind];
variants.splice(ind, 1);
ask_variants = rnd_subset(3, variants);
ask_variants.push(right_variant);
ask_variants = rnd_perm(ask_variants);
statement += ask_variants.$('\\begin{enumerate}\\item %{}\\item %{}\\item %{}\\item
%{}\\end{enumerate}');

answer = 'Вариант: ' + (1 + ask_variants.indexOf(right_variant));
```

лены 2 варианта, аналогичным образом описываются варианты для всех 8 терминов.

За счет использования синтаксиса на языке Possum алгоритм генерации данной задачи выглядит несколько компактнее, где большую часть кода занимает описание определений (см. листинг 6).

ЗАКЛЮЧЕНИЕ

Рассмотренные в статье задачи показывают подходы, которые можно использовать при разработке генераторов. Система «Платан» и язык Possum позволяют существенно упростить процесс построения многовариантных тестовых заданий для различного диапазона дисциплин. Несмотря на многообразие решений, методов и подходов к организации процесса подготовки тестовых заданий, разработку генераторов тестовых заданий сложно свести к решению задачи построения алгоритма одного типа. Зачастую необходимо для каждого конкретного генератора создавать отдельный алгоритм и свои методы разработки генераторов. Разработка инструментальных средств, связанных с разработкой многовариантных тестовых заданий, не получит своего завершения. Тенденции развития подобных систем склоняются не только к применению новых алгоритмов и методов построения генерации, но и в сторону визуализации процесса построения многовариантных тестовых заданий.

Литература

1. *Кручинин В.В.* Генераторы в компьютерных учебных программах. Томск: изд-во Томск. ун-та, 2003.
2. MathCad. Режим доступа: http://mathcad.com.ua/ (дата обращения 13.02.2014).
3. Mathematica. Режим доступа: http://www.wolfram.com/mathematica/ (дата обращения 13.02.2014).
4. LMS Moodle. Режим доступа: http://moodle.org/ (дата обращения 13.02.2014).
5. Schoolhouse Test. Режим доступа: http://schoolhouse-test.software.informer.com/ (дата обращения 13.02.2014).
6. *Посов И.А.* Автоматизация процесса разработки и использования / Дисс. на соискание степени канд. техн. наук. СПб., 2012.
7. *Зорин Ю.А.* Использование алгоритмов комбинаторной генерации при построении генераторов тестовых заданий // Дистанционное и виртуальное обучение, 2013. № 6. С. 54–59.

INSTRUMENTED SYSTEMS FOR CONSTRUCTION AND OBTAINING MULTIVARIATE TEST TASKS

Abstract

We consider the problems arising in the construction of algorithms for test tasks generation. The article touches upon the tools that enable the development of tasks of the generating type. On the examples of specific tasks, the approaches to develop multiple choice test items are considered. They are based on the two systems (Sycamore, Possum) developed by the authors.

Keywords: generating of items, algorithms of generation, tools, Platan, Possum.

Зорин Юрий Алексеевич,
аспирант кафедры Промышленной электроники ТУСУР, программист систем дистанционного обучения,
yura@freebrains.ru,

Посов Илья Александрович,
кандидат технических наук,
старший преподаватель кафедры Информационных систем в искусстве и гуманитарных науках СПбГУ, iposov@gmail.com.