УДК 004.896:519.767.6


А.Н. Швецов, А.П. Сергушичева
Вологодский государственный университет

АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ КОМПЬЮТЕРНЫХ ТЕСТОВ КОНТРОЛЯ ЗНАНИЙ

Для автоматизации построения компьютерных тестирующих систем (КТС) авторами разработан метод генерации компьютерных тестов и тестовых заданий, формирующий структуру и содержание конкретной тестовой системы продукционными средствами. Предлагаются направления развития метода для применения в системах электронного обучения.

Компьютерные тестирующие системы, автоматическая генерация тестовых заданий, формальные грамматики, электронные учебно-методические комплексы.

Введение

К концу второго десятилетия 21-го века создано значительное число автоматизированных обучающих систем и средств дистанционного обучения, включающих функции автоматизированного контроля знаний и умений обучаемых [1]. При этом возникает задача обеспечения объективности контроля, требующая вариативности и идентичности контрольно-измерительных материалов по уровню сложности, объему охватываемого материала, размеру предъявляемых заданий.

Во многих КТС отсутствуют средства автоматизации создания тестов – немногочисленные варианты создаются и обновляются автором курса в режиме ручного редактирования. Актуальность проблемы обеспечения вариативности компьютерных тестов подтверждается рядом исследований, направленных на решение этой задачи [2-7].

Методологические подходы к созданию компьютерных тестов

Основные подходы к задачам педагогического, психологического и компьютерного тестирования и их использованию в различных методиках обучения исследованы в работах Аванесова В.С., Башмакова А.И., Башмакова И.А, Брусиловского П.Л., Горбатова Д.С., Галеева И.Х., Клайна П., Майорова А.Н., Третьякова П.И., Капустина Н.П., Талызиной Н.Ф., Sosnovsky S., Ullrich C. и др. [8-11].

За последние десятилетия в среде электронного обучения сформировались два класса программно-технических систем, существенно отличающиеся по экономическим и организационным параметрам, но обладающие очень схожими возможностями и характеристиками с точки зрения методики учебного процесса, состава и функций программных компонентов и используемых форматов данных.

Первый класс составляют коммерческие автоматизированные обучающие системы АОС (LMS – Learning Management System в англоязычной традиции), предназначенные для использования как в организациях и фирмах, так и в высших учебных заведениях (Adobe Captivate Prime (https://captivateprime.adobe.com), Docebo LMS (https://www.docebo.com), Talent LMS (https://www.talentlms.com), Academy LMS (https://www.growthengineering.co.uk/academy-lms/), ExpertusONE (https://www.expertus.com), Administrate (https://www.getadministrate.com), Cisco Learning Network Space (https://learningspace.cisco.com/), Competentum Online (https://competentum.ru), MAESTRO (https://meetmaestro.com/resources), STELLUS (http://www.stel.ru/equipment/distance-learning/), CourseLab (http://www.courselab.com/) и другие.

Второй класс составляют некоммерческие открытые системы электронного обучения (Open-Source LMS): MOODLE (http://moodle.org), ILIAS (http://www.ilias.de), Eliademy (http://eliademy.com), FormaLMS (http://www.ilias.de), Opigno (htttp://www.opigno.org), ГЕКАДЕМ 4.0 (http://www.hecadem.irk.ru/), IDEA (http://www.linkundlink.de/info/ideateam-responsive-design), VeralTest (http://www.veralsoft.com) и др. [12].

Анализ показывает, что в основном системы обоих классов обладают схожими характеристиками, обеспечивая элементы управления курсами, предопределенные сценарии обучения, широкие возможности поиска учебного материала, подключение разнообразных модулей, Интернет-ресурсов – портфолио, блогов, Wiki, FAQ, форумов, чатов, дискуссионных панелей и прочего. Возможности создания тестов остаются достаточно традиционными и не обеспечивают достаточной степени автоматизации этого процесса.

В ряде работ излагаются модели АОС, включающие генерацию тестовых заданий (ТЗ), но отсутствие отработанной методологии построения таких КТС требует разработки оригинальных алгоритмов и уникальной программной реализации порождающих механизмов. Опираясь на результаты ряда работ, перечислим критерии, которым должен удовлетворять метод генерации компьютерных тестов: простота использования; генерация с условиями; дифференциация; множественность; логичность; интегративность; мультигенерация; структурность, интеллектуальность, интерактивность.

Насколько соответствуют известные методы генерации компьютерных тестов установленным выше требованиям?

Методы хранимых и параметризованных шаблонов требуют, чтобы для каждого варианта теста и задания их структуру и информационную наполнен-

ность предварительно установил преподаватель, что существенно ограничивает разнообразие тестов. Метод множественных генераторов обеспечивает разнообразие тестов, но требует контролировать все возможные комбинации множеств, их логические взаимосвязи, отсутствует прозрачность построения логической структуры заданий и тестов. Метод графа знаний и обучающего кластера ориентирован на структуру тестов и обратную связь (используется статистика результатов выполнения тестов). Метод позволяет составить общую картину обучения, но не облегчает наполнения материалами базы данных. Для составления учебного материала самих заданий, установления взаимосвязей и учета обратных связей необходимы большое внимание и значительные трудозатраты.

Метод генерации на основе онтологий основан на использовании отношений между концептами, которые интерпретируются соответствующими учебными материалами. Требуется значительное время на поиск информации и ручное составление заданий, которые необходимо связать между собой, установив отношения между концептами. Метод нейронных сетей используется в основном для создания «идеальных» учебных курсов. По результатам анализа, связь между тестовыми заданиями будет меняться, тем самым достигается самообучаемость теста и способность его подстраиваться под конкретного пользователя. Но изначально учебный курс не является оптимальным, его надо «обучить», что требует участия опытного эксперта предметной области [13].

3. Метод автоматической генерации компьютерных тестов

Для решения проблемы автоматизации построения КТС авторами разработан метод генерации компьютерных обучающих тестов и тестовых заданий, основанный на формализме канонических исчислений Э. Поста, определяющий структуру и содержание конкретного теста с применением формальных грамматик составляющих [14]. Программная реализации метода поддерживает основные типы формальных грамматик по иерархии Н. Хомского [15].

Пусть ST_{PA} – структура предметной области, охватываемой учебной дисциплины, ST_{KN} – модель знаний дисциплины, отражающая структуру предметной области, и содержащая множество учебных единиц – модулей, блоков, разделов. ST_{KN} рассматривается как семантическая сеть, содержащая l уровней иерархии, структура которых определяется как H (рис.).

$$H = (h_0(Sg0), h_1(Sg1, \dots, Sgk), h_2((Sg11, \dots, Sg1k(1)), (Sg21, \dots Sg2k(2)), \dots (Sgk1, \dots, Sgkk(k))), \dots h_l((Sg11 \dots 1_l, \dots Sg11 \dots 1_{l-1}k(11 \dots 1k_{l-1})), (Sg11 \dots 21_l, \dots, Sg11 \dots 2k(11 \dots 2k)), \dots (Sgkk(k) \dots k(l-1), \dots Sgkk(k) \dots k(l-1)k(l)).$$

Множество всех вершин такой семантической сети, включающее все сегменты $Sgi \dots f$ для уровней иерархии от h_0 до h_l, обозначим V. Между сегментами ST_{KN} возможны различные взаимосвязи-отношения, такие как отношения подчинения, структурной вложенности (раздел-подраздел), семантической связи изучаемых в сегментах понятий, логического следования, темпоральные (до – после) и т.п.

Множество возможных отношений обозначим как $\mathcal{R} = (R_i | i = 1, \dots, n)$.

Модель знаний дисциплины представляется структурой $ST_{KN} = (H, V, \mathcal{R}, P)$, где $P = (\Pi_j | j = 1, \dots, n)$, Π_j есть отображение $\underbrace{V \times V \dots \times V}_{m} \to R_j$, значность декартова произведения m может быть различной.

Назовем «дуплетом» пару $D_i = (Q_i, A_i)$ – «тестовое задание – ответ», компьютерный тест можно считать упорядоченной последовательностью дуплетов $KT = (D_i | i = 1, \dots, n)$, где n – количество заданий в данной реализации теста.

Предметной областью компьютерного теста будем считать подсеть $ST_{KT} \subseteq ST_{KN}$, включающую подмножество вершин V_{KT}, с которыми связаны порождаемые дуплеты D_i. По данной ST_{KT} может быть сгенерировано множество конкретных тестов, имеющих различное количество дуплетов (число тестовых заданий) и различные отображения $ST_{KT} \to (D_i | i = 1, \dots, x)$, где x принимает значения на множестве натуральных чисел.

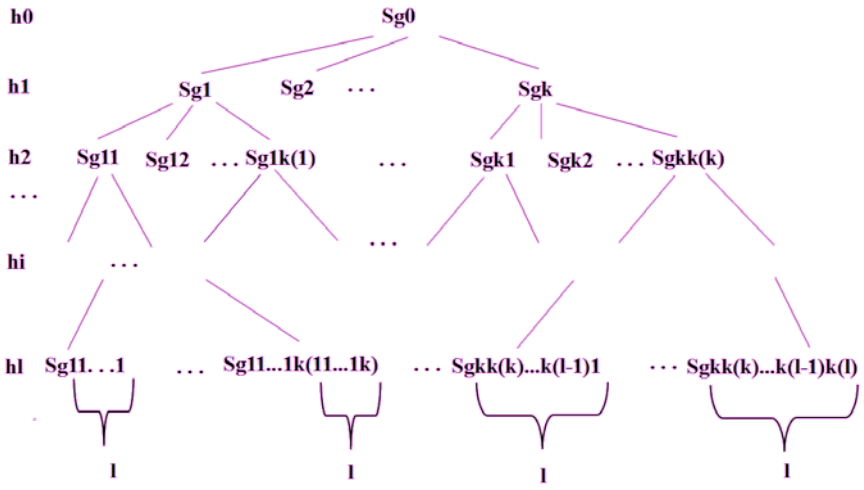

Рис. Граф структуры уровней модели знаний учебной дисциплины

Таким образом, для создания метода генерации компьютерных тестов требуется построить оператор $\mathcal{F}: (ST_{PA}, ST_{KN}, ST_{KT}) \to MT$, позволяющий на основе структуры предметной области дисциплины, модели знаний, предметной области компьютерного теста получить множество тестов $MT = \{KT_\varphi | \varphi = 1, \dots, m\}$. Мощность КТС характеризуется числом различных заданий, которые могут формироваться системой:

$$M_{\text{КТС}} = \sum_{ST_{KN}} |KT|,$$

где суммирование ведется по всей модели знаний и всевозможным структурам предметных областей компьютерных тестов.

На искомый оператор $\mathcal{F}$: накладываются следующие ограничения:

– множество МТ не может быть пустым, его мощность ограничивается некоторым целым числом. Максимальное значение мощности $M_{\text{КТС}}$ определяется программно-технической реализацией;

– должно обеспечиваться соответствие предметной области теста и множества вариантов теста, генерируемых по данной ST_{KT}, внутри теста KT_φ должен быть образ множества вариантов тестового задания $\forall i((Sg_i \in ST_{KT}) \to (\exists D(i) \& D(i) \neq \emptyset))$, где $D(i)$ – возможное множество дуплетов, соответствующих сегменту Sg_i;

– вопросы в порожденных тестах могут повторяться, но должно выполняться ограничение $\forall i \forall j \left(|\{KT_i\} \cap \{KT_j\}| \leq \delta\right), i \neq j$;

– время осуществления преобразования оператором $\mathcal{F}$: ограничивается величиной $t_{dec} \leq t_{max}$.

Формальное описание тестовых заданий может содержать следующие виды информации: простые типы данных – целые и рациональные числа, символы и строки символов, рисунки в заданных графических форматах; сложные типы данных – предложения, получаемые комбинированием простых данных; переменные (указывается тип данных и название переменной); функции, в том числе встраиваемые в КТС и подключаемые через библиотеку модулей, а также записываемые посредством переменных; файлы, содержащие грамматики заданий, структуру тестов и информацию, необходимую для генерации тестовых заданий.

Множество генерируемых по данной модели знаний компьютерных тестов представим объединением $MT(ST_{KN}) = \bigcup_{ST_{KN}} B_i$, где $B_i = \{b_{i1}, b_{i2}, \dots b_{ia_j}\}$ – подмножество вопросов i-го типа; a_j – количество вопросов i-го типа в подмножестве B_i. Если n_t – количество типов вопросов, используемых в данной реализации теста, то общее количество вопросов в данном тесте есть $\sum_{j=1}^{n_t} a_j$.

Оператор $\mathcal{F}$: конструируется как исчисление вида:

$$\mathcal{K} = (\mathcal{A}, P, A_0, \mathcal{P}), \quad (1)$$

где $\mathcal{A}$ – алфавит исчисления, P – алфавит переменных, A_0 – аксиома исчисления, $\mathcal{P}$ – множество правил вывода.

Алфавит исчисления $\mathcal{A} = \{N \cup Z\}$, где N – алфавит нетерминальных символов, Z – алфавит вспомогательных символов. В множество N включены нетерминалы V_i – кодирующие структуру вопрос-ответ i-го типа, Π_i – кодирующие множество правил вывода для i-го типа и нетерминалы подтипов $PV_{i,j,\dots,r}$ с индексной последовательностью. Алфавит Z={/,(,), [,], #, $i, j, \dots, r$}. Алфавит переменных $P = \{p, q, \varphi, a_i, \rho_{ij}, \rho_b, \rho_e\}$. Символы алфавитов имеют следующий смысл: | – элементарный конструкт для организации унарных счетчиков; (,) – скобки, обозначающие состояние счетчика; [,] – скобки, обозначающие элементы множества правил вывода; # – элемент разделения, формирующий структуру антецедентов и консеквентов; $i, j, \dots, r$ – индексная последовательность, показывающая уровень и тип вложенности подтипа; p – левая часть последовательности подтипов; q – правая часть последовательности подтипов; φ – переменная, обозначающая текущее состояние счетчика подтипов; a_i – переменная, обозначающая начальное состояние счетчика подтипов; ρ_{ij} – переменная, обозначающая исключаемое правило из множества Π_i; ρ_b, ρ_e – оставшиеся правая и левая (относительно ρ_{ij}) части множества правил Π_i.

Аксиома исчисления A_0 будет иметь вид:

$$A_0 = V_1|^{(a1)}[\Pi_1]\#V_2|^{(a2)}[\Pi_2]\# \dots \#V_k|^{(ak)}[\Pi_k], \quad (2)$$

где $\Pi_i = \{\rho_{i1}, \rho_{i2}, \dots, \rho_{if_i}\}$, где f_i – количество правил вывода для i-го типа. Структура $V_i|^{(ai)}[\Pi_i]$, расположенная между # ... #, означает, что в тест будет включено a_i заданий, генерируемых в соответствии со структурой вопрос-ответ i-го типа, при этом a_i может быть больше, меньше или равно f_i.

Множество правил вывода $\mathcal{P}$ создается с учетом исключения использованных правил из множеств Π_i. Развертывание i-го типа производится правилами вида:

$$\frac{\dots\#V_i|^{(ai)}[\rho_b\rho_{ij}\rho_e]\#\dots}{\dots\#(PV_{i,j})V_i|^{(ai-1)}[\rho_b\rho_e]\#\dots}, \quad (3)$$

которых будет f_i экземпляров для каждого типа i. Далее применяются правила вида:

$$\frac{\dots\#pV_i|^{(\varphi)}[\rho_b\rho_{ij}\rho_e]\#\dots}{\dots\#p(PV_{i,j})V_i|^{(\varphi-1)}[\rho_b\rho_e]\#\dots}, \quad (4)$$

которых также будет f_i экземпляров для каждого типа i, причем переменная p хранит последовательность подтипов $(\dots)(PV_{i,j})(\dots)$.

Применения правил вида (4) продолжается до наступления одного из следующих событий: исчерпывается $V_i|^{(\varphi)}$, либо исчерпывается множество вил $[\Pi_i]$, что означает, что $f_i < a_i$ и количества правил в $[\Pi_i]$ не хватает для введения подтипов без повторов. В первом случае прекращается процесс раскрытия типа V_i применением правила:

$$\frac{\dots\#pV_i[\rho_b\rho_e]\#\dots}{\dots\#p\#\dots}. \quad (5)$$

Во втором случае подставляется заново множество правил вывода Π_i для продолжения раскрытия типа V_i, что реализуется правилом (символы [] обозначают, что в множестве $[\Pi_i]$ не осталось не примененных правил):

$$\frac{\dots\#pV_i|^{(\varphi)}[\,]\#\dots}{\dots\#pV_i|^{(\varphi)}[\Pi_i]\#\dots}. \quad (6)$$

Вывод предложения целевого языка из подтипа $PV_{i,j,\ldots,r}$ определяется соответствующей формальной грамматикой, содержащей правила вывода, записанные в традиционной нотации Бэкуса-Наура.

Интерпретацию метода генерации тестовых заданий покажем на примере дисциплины «Метрология, стандартизация, сертификация», входящей в учебные планы многих направлений подготовки бакалавров. По теме «Измерения с помощью осциллографа» применяется следующая грамматика:

<постоянная_часть_подтипа>::= Определите частоту исследуемого сигнала по полученной на экране осциллографа

<переменная_часть_подтипа>::= <c>

<варианты_ответа>::=

<описание ответа>::=

<описание грамматики>::=

<c>::= <n1>|<n2>

<n1>::= фигуре Лиссажу <n10><n11><n5> Гц.

<n2>::= круговой развертке <n20><n21><n5> Гц.

<n10>::= < p1>|< p2>|< p3>|< p4>|< p5>| < p6>| < p7>|< p8>|< p11>|< p12>

<n11>::= , если он подается на <n12>

<n12>::= вход X, а частота образцового напряжения fy=| вход Y, а частота образцового напряжения fx =

<n20>::= < p9>|< p10>|< p13>|< p14>|< p15 >

<n21>::= , если он подается на <n22>

<n22>::= модулятор, а частота образцового напряжения fx=fy= | вход Z, а частота образцового напряжения fx=fy= | вход X, а частота образцового напряжения fz=| вход Y, а частота образцового напряжения fz=

<n5>::=100|200|300|400|500|2400|3600|4200|4800|5400

<p1>::= (Metr_p_o1.bmp) < p2>::= (Metr_p_o2.bmp)

< p3>::= (Metr_p_o3.bmp) < p4>::= (Metr_p_o4.bmp)

< p5>::= (Metr_p_o5.bmp) < p6>::= (Metr_p_o6.bmp)

< p7>::= (Metr_p_o7.bmp) < p8>::= (Metr_p_o8.bmp)

<p9>::= (Metr_p_o9.bmp) < p10>::= (Metr_p_o10.bmp)

< p11>::= (Metr_p_o11.bmp) < p12>::= (Metr_p_o12.bmp)

< p13>::= (Metr_p_o13.bmp) < p14>::= (Metr_p_o14.bmp)

< p15>::= (Metr_p_o15.bmp)

По данной грамматике будет сгенерировано до 400 различных по условию задач, число которых может быть легко увеличено добавлением значений нетерминала n5 или определением его как диапазона значений, что делает количество вариантов практически неограниченным.

4. Программная реализация

Предложенный метод реализован в инструментальной интеллектуальной программной системе (ИИПС) и использован во многих тестирующих программах, созданных на ее базе [16]. При применении метода преподаватель составляет правила и описывает средствами формальных грамматик структуры заданий, ответов, пояснений, определяющих содержание возможного множества тестов. Расширения синтаксиса позволяют использовать в структуре заданий мультимедийные объекты (рисунки, видео- и аудиофайлы), вычисляемые формулы и элементы форматирования текста.

В среде ИИПС реализована структура файлов-хранилищ грамматик, позволяющая хранить несколько грамматик одного или разных типов в едином контейнере. На процесс вывода влияют общие параметры генерации: глубина рекурсии; выборка с памятью; параметры смешивания.

Экспорт полученных тестов реализован в форматах RTF, HTML, SSI, BIN, что позволяет как использовать сгенерированные тесты в текстовом формате, так и осуществлять их передачу в электронные учебно-методические комплексы, созданные на основе АОС Moodle по основным дисциплинам направлений подготовки 09.03.02 «Информационные системы и технологии», 09.03.03 «Прикладная информатика», 09.03.04 «Программная инженерия», 27.03.04 «Управление в технических системах».

Для применения в учебном процессе было разработано более 300 различных грамматик по дисциплинам всех циклов технических направлений подготовки бакалавров и магистров: математический и естественно-научный цикл («Математика», «Физика», «Электротехника»), гуманитарный, социальный и экономический цикл («Русский язык»), профессиональный цикл («Метрология, стандартизация, сертификация», «Оборудование автоматизированных производств», «Интеллектуальные информационные системы», «Представление знаний в информационных системах», «Теория языков программирования и методы трансляции», «Системное программное обеспечение», «Технология разработки программного обеспечения», «Интеллектуальные системы и технологии», «Системы искусственного интеллекта и принятие решений»). Надежность и валидность компьютерных тестов, сгенерированных средствами ИИПС, исследовалась более чем на 30 учебных тестах, содержащих до 50 вариантов конкретного теста по данной дисциплине, при объеме выборки испытуемых от 20 до 300 человек. Надежность полученных тестов лежит в диапазоне от 0,66 до 0,83, что говорит о достаточно высоком качестве компьютерных тестов и возможности их дальнейшего применения в составе АОС.

5. Заключение

Опыт применения метода генерации компьютерных тестов позволяет говорить о широких возможностях использования представленного подхода в современных АОС в условиях массовой подготовки бакалавров и магистров, особенно с использованием дистанционных образовательных технологий. Метод основан на фундаментальных математических формализмах, что открывает широкие возможности для дальнейшего научного поиска и совершенствования механизмов генерации компьютерных тестов. Исследуются возможности комбинирования методов генерации и извлечения тестовых заданий из текстов учебных пособий [17], прорабатываются различные стратегии генерации в рамках алгоритмов реализации продукционных механизмов, рассматривается применение методов формальной семантики для совмещения порождающих правил формальных грамматик с онтологиями предметных областей изучаемых дисциплин [18].

Литература

1. E-learning in European Higher Education Institutions November 2014: Results of a mapping survey conducted in October-December 2013 [Электронный

ресурс] / M. Gaebel, V. Kupriyanova, R. Morais, E. Colucci. – Режим доступа: http://www.eua.be/.

2. Ржеуцкая, С. Ю. Интегрированная информационная среда обучения как средство развития иноязычной коммуникативной компетенции / С. Ю. Ржеуцкая, М. В. Харина // Открытое образование. – 2016. – Т. 20, № 1. – С. 43–48.

3. Зорин, Ю. А. Автоматизация построения многовариантных тестовых заданий на основе деревьев и/или: дис. ... канд. техн. наук: 05.13.06 / Зорин Юрий Алексеевич; [место защиты: С.-Петерб. нац. исслед. ун-т информац. технологий, механики и оптики]. – Томск, 2014. – 139 с.

4. Давыдова, Н. А. Автоматизированный синтез тестовых заданий для систем педагогического контроля знаний / Н. А. Давыдова, И. Д. Рудинский // Информатизация образования и науки. – 2013. – № 1 (17). – С. 77–90.

5. Монахов, М. Ю. Методы и модели обработки и представления информации в распределенных образовательных системах: дис. ... д-ра техн. наук: 05.13.01, 05.13.10 / Монахов Михаил Юрьевич; [место защиты: Владимирский гос. ун-т]. – Владимир, 2005. – 418 с.

6. Башмаков, А. И. Разработка компьютерных учебников и обучающих систем / А. И. Башмаков, И. А. Башмаков. – Москва: Филин, 2003. – 430 с.

7. Горбатов, Д. С. Практикум по психологическому исследованию: учеб. пособие / Д. С. Горбатов. – Самара: БАХРАХ-М, 2000. – 248 с.

8. Клайн, П. Справочное руководство по конструированию тестов / П. Клайн. – Киев: Ника-Центр Лтд, 1994. – 283 с.

9. Brusilovsky, P. Addictive links: the motivational value of adaptive link annotation / P. Brusilovsky, S. Sosnovsky, M. Yudelson // New Review of Hypermedia and Multimedia. – 2009. – Vol. 15, № 1. – P. 97–118.

10. Галеев, И. Х. Проблемы и опыт проектирования ИОС [Электронный ресурс] / И. Х. Галеев // Образовательные технологии и общество. – 2014. – Т. 17, № 4. – С. 526–542. – Режим доступа: https://elibrary.ru/item.asp?id=22673873.

11. Ullrich, C. Pedagogically Founded Courseware Generation for Web-Based Learning: An HTN-Planning-Based Approach Implemented in PAIGOS: Lecture notes in computer science. Lecture notes in artificial intelligence. Vol. 5260 / C. Ullrich. – Berlin; New York: Springer, 2008. – 257 p.

12. Learning Management Systems [Электронный ресурс]. – Режим доступа: https://elearningindustry.com/subjects/elearning-software/learning-management-systems-lms.

13. Посов, И. А. Обзор генераторов и методов генерации учебных заданий [Электронный ресурс] / И. А. Посов // Образовательные технологии и общество. – 2014. – Т. 17, № 4. – С. 593–609. – Режим доступа: https://elibrary.ru/item.asp?id=22673878.

14. Сергушичева, А. П. Гибридный подход к синтезу тестовых заданий в тестирующих системах / А. П. Сергушичева, А. Н. Швецов // Математика, Компьютер, Образование: сборник науч. трудов. Вып. 13. Т. 1 / под ред. Г. Ю. Ризниченко. – Москва; Ижевск, 2006. – С. 215–228.

15. Chomsky, N. Syntactic Structures / N. Chomsky, W. David Lightfoot. – Walter de Gruyter, 2002. – 117 p.

16. Швецов, А. Н. Система синтеза учебных тестов на основе формальных грамматик / А. Н. Швецов, Ю. О. Мамадкулов, С. И. Сорокин // Программные продукты и системы. – 2013. – № 2 (102). – С. 181–185.

17. Швецов, А. Н. Метод автоматизированной генерации контрольно-тестовых заданий из текстов учебных материалов / А. Н. Швецов, А. М. Куртасов // Вестник Череповецкого государственного университета. – 2014. – № 7 (60). – С. 7–11.

18. Швецов, А. Н. Проблема интеллектуализации систем дистанционного обучения / А. Н. Швецов / Системные стратегии: наука, образование, информационные технологии: сборник науч. статей. Вып. 1 / под ред. О. Б. Голубева и Н. А. Ястреб. – Вологда, 2013. – С. 132–139.

A.N. Shvetsov, A.P. Sergushicheva
Vologda State University

AUTOMATION DESIGN OF COMPUTER TESTS FOR KNOWLEDGE CONTROL

To automate the computer testing systems (CTS), the authors developed a method of generating of computer tests and test tasks that form the structure and content of a particular test system using production tools. Ways for the development of the method for use in e-learning systems are proposed.

Computer testing systems, automatic computer test generation, formal grammars, e-learning complexes.