В. А. Тарасов,
канд. физ.-мат. наук, профессор кафедры информатики Карельского государственного педагогического университета,

А. С. Кюршунов,
аспирант кафедры информатики Карельского государственного педагогического университета

ОРГАНИЗАЦИЯ ТЕСТИРОВАНИЯ ОБУЧАЕМЫХ В УСЛОВИЯХ ЛОКАЛЬНОЙ СЕТИ

В настоящее время в сфере образования все большую популярность приобретает дистанционное обучение (ДО) на базе компьютерных телекоммуникаций. Одной из актуальных проблем дистанционного обучения является разработка средств сетевого тестирования и контроля [1]. В данной статье рассматриваются вопросы использования компьютеров, объединенных в локальную сеть, для тестирования обучаемых и информационной поддержки электронных сетевых учебных материалов по изучаемым учебным дисциплинам.

Для разработки средств тестирования используется язык программирования Java. Средством выставления обучающих ресурсов и тестов в локальной сети может выступать web-сервер Jakarta Tomcat. Преимуществами этих средств перед средствами организации несетевого тестирования являются их полная интеграция и независимость от платформы.

В статье описана разработка тестов для компьютерного контроля знаний на основе Java-апплетов и сервлетов. При этом рассматриваются вопросы проектирования как тестов для самоконтроля с обработкой данных на стороне пользователя, так и контролирующих тестов с обработкой данных на web-сервере.

Web-сервер Jakarta Tomcat

Данный сервер можно использовать как небольшой автономный сервер. Он позволяет создавать web-документы и выставлять их в локальной сети, использовать клиент-серверные приложения без подключения к Интернету. Целесообразность выбора сервера Tomcat для указанных целей обусловлена тем, что он разрабатывался специально для поддержки сервлетов и JSP (Java Server Pages). На базе сервера Tomcat может быть создан полнофункциональный web-сервер, без изменений во внутренней структуре на пути интеграции с web-сервером Apache, работающим в Интернете, однако эти вопросы не являются предметом нашего рассмотрения.

Среди возможностей рассматриваемого web-сервера можно выделить следующие, позволяющие учебным заведениям, имеющим локальную сеть, организовывать деятельность в поддержку процесса обучения:

- централизованное хранение и управление электронными учебниками, выполненными в виде web-документа;
- эмуляция работы в Интернете;
- отладка web-документов перед выставлением в Интернет;
- организация обучения по методу открытой платформы (в рамках локальной сети учебного заведения);
- тестирование обучаемых.

В данной статье мы рассмотрим тестирование обучаемых.

Установка Jakarta Tomcat и JDK

Как и сервер Apache (который является очень быстрым, высоконадежным, но сложным в установке и конфигурировании), сервер Tomcat требует значительно больших усилий при установке, чем процессоры сервлетов в коммерческих web-серверах. Получить дистрибутив сервера можно на сайте http://jakarta.apache.org/tomcat. Там же можно ознакомиться с информацией по поддержке сервера. Этот сервер разработан под различные операционные системы, однако мы рассмотрим установку сервера только под семейство наиболее распространенных операционных систем Windows. После получения на вышеуказанном сайте файла с дистрибутивом сервера, например jakarta-tomcat-4.0-bl.zip (конкретное имя файла зависит от выбора версии продукта), его необходимо распаковать, полученную папку можно переименовать (например, C:\TOMCAT — короткие имена легче использовать) либо оставить предложенное название.

Поскольку данный web-сервер разработан с помощью языка Java и работает с различными операционными платформами, потребуется установка еще двух пакетов: JDK (Java Development Kit) и J2SDKEE (Java 2 SDK, Enterprise Edition). Первый пакет представляет собой средство разработки Java-приложений, а второй является дополнением, в частности он поддерживает разработку и использование сервлетов и Java-серверных страниц (JSP). Эти пакеты можно найти на сайте http://java.sun.com. Следует учесть, что для корректной работы потребуется JDK версии не ниже 1.2. - именуемый Java2 (или JDK1.2)

Полученные дистрибутивы необходимо установить. Предположим, что JDK установлен в папку C:\JDK2, а J2SDKEE — в папку C:\J2SDKEE. Далее необходимо установить переменные среды, добавив в файл autoexec.bat следующие строки:

```
CATALINA_HOME = C : \TOMCAT
J2EE_HOME  = C:\J2SDKEE
JAVA_HOME  = C:\JDK2
JAXP_HOME  = %CATALINA_HOME%\LIB
JNDI_HOME  = %CATALINA_HOME%\BIN
PATH = %JAVA_HOME%\bin; J2EE_HOME%\bin; CATALINA_HOME%\bin;
CLASSPATH = .;%JAVA_HOME%\lib; %J2EE_HOME%\lib;
%J2EE_HOME%\LIB\j2ee.jar; %CATALINA_HOME%\lib;
%CATALIHA_HOME%\lib\jaxp.jar; CATALINA_HOME%\lib\jasper.jar
```

Данный подход предназначен для операционных систем семейства Windows 9X (для семейства Windows NT необходимо в свойствах системы найти средство изменения переменных среды и ввести пары значений — имя переменной и ее значение). Далее надо перезагрузить Windows 9X и в папке C:\TOMCAT\BIN для файла Sturtup.bat создать ярлык и в его свойствах в пункте **Память** установить значение EMS памяти не ниже 2048 (подобные действия для NT-системы не требуются). Теперь ваш web-сервер готов к работе. Для запуска сервера предназначен файл Sturtup.bat (в Windows 9X необходимо использовать созданный ранее ярлык этого файла), а для завершения его работы — файл Shutdown.bat.

Принцип работы с web-сервером

Так как мы не рассматриваем работу сервера в сети Internet, то будем использовать настройки по умолчанию.

Для размещения HTML-документов предназначена папка C:\TOMCAT\webapps\ROOT, в которой изначально размещается стартовая страница. В этой папке вы можете размещать свои папки, содержащие гипертекстовые страницы. Для размещения сервлетов предназначена папка C:\tomcat\webapps\ROOT\WEBINF\classes (если ее нет, то создайте самостоятельно).

Для начала убедимся в работоспособности сервера. Запустим сервер (Sturtup.bat), в случае успеха появятся надписи:

```
Starting service Tomcat-Standalone
Apache Tomcat/4.0-b1
Starting service Tomcat-Apache
Apache Tomcat/4.0-bl
```

Теперь на любом компьютере локальной сети (или на том же, где запущен web-сервер), запустите web-браузер и в строке адреса пропишите http-заголовок, сетевое имя компьютера, на котором работает web-сервер, или его IP-адрес, номер активного порта (по умолчанию он равен 8080). Это может выглядеть примерно так: http:/mainpc:8080 или http://192.168.70.50:8080 (для Windows NT4 используйте второй вариант). Если вы работаете на том же компьютере, где запущен сервер, то можно ввести: http: //localhost:8080

После ввода адреса в окне браузера появится страница с информацией о web-сервере с примерами использования сервлетов и JSP.

Если вы хотите проверить работу с собственным web-документом, то разместите в папке ROOT папку, содержащую файлы документа и файл index.htm (или index.html), который является стартовым по умолчанию. Допустим, создана папка C:\TOMCAT\WEBAPPS\ROOT\Probe и в ней есть файл index.htm . Введите адрес http:/ /mainpc:8080/Probe (или http://mainpc:8080/Probe/index.htm — результат будет аналогичным).

Для корректного завершения работы web-сервера необходимо воспользоваться файлом Shutdown.bat, в частности не следует забывать делать это перед перезагрузкой и выключением компьютера. После запуска этого файла появится еще одно окно и вскоре исчезнет. Далее следует дождаться исчезновения предыдущего окна (в случае длительной задержки вы его можете закрыть самостоятельно, когда в нем будет присутствовать надпись об остановке сервера).

Функциональные возможности тестов, выполняемых на стороне клиента и стороне сервера

С технической точки зрения функционирования компьютерного теста можно выделить два типа тестов:

1. Тесты, выполняемые на стороне клиента. В этом случае по запросу клиента с сервера передается тестовая база данных (как вопросы, так и эталоны ответов). Обработка результатов тестирования производится на стороне клиента, результаты сообщаются только тестируемому, не сохраняются и никуда не передаются. Такие тесты могут выполнять только функции самоконтроля и тренинга, т. е. носить обучающий характер. Заметим, что эти тесты могут быть интегрированы в электронный учебник и передаваться обучаемому вместе с ним.

2. Тесты, выполняемые на стороне сервера. В этом случае клиенту передается только часть тестовой базы данных, содержащая вопросы. Ответы обучаемого передаются серверу, где и обрабатываются, результаты сообщаются тестируемому и сохраняются. Обучаемый не имеет возможности, после того как ответь! отправлены на сервер, вносить в них изменения. Такая организация тестирования наиболее подходит для управления учебной деятельностью обучаемых. В зависимости от результатов тестирования обучаемый либо будет вынужден пройти изучение учебного модуля снова, либо получит доступ к новому учебному модулю.

При этом компьютерные тесты могут быть как с выбором вариантов ответов, так и с открытыми ответами. В тестах первого вида в качестве ответа на поставленный вопрос требуется выбрать один или несколько из предлагаемых вариантов.

Мы ограничимся рассмотрением компьютерных тестов с открытыми ответами. Отвечая на вопрос такого теста, обучаемый вводит ответ в компьютер в текстовой форме. Оценка правильности ответа осуществляется с помощью специальной библиотеки ключей. Ключами могут быть словосочетания, слова, части слов (например, корни), числа и др. Особо значима постановка контролирующего вопроса: он должен быть сформулирован настолько ясно и корректно, чтобы не допускать возможности иного толкования, кроме желаемого. Непростой задачей является подбор ключей: основной критерий — подобрать именно те ключи, которые пользователь обязательно должен включить в правильный ответ. Важно программно предусмотреть употребление в ответе слов-синонимов, проверку орфографии.

Компьютерные тесты можно разделить на две основные группы: обучающие и итоговые.

К обучающим тестам относятся тесты самоконтроля и тренажеры. Тесты самоконтроля позволяют учащемуся самому проконтролировать свои знания по изуча-

емой теме. Тренажеры предназначены для выработки у учащихся навыков решения стандартных задач.

Итоговый контроль осуществляется после изучения каждого модуля учебного курса. Успешное выполнение итогового теста является необходимым условием перехода к изучению следующего модуля курса.

Проектирование контролирующего теста

Рассмотрим вопросы проектирования контролирующего теста с открытыми ответами, основанного на Java-апплете и сервлетах, со смешанным типом функционирования.

Структура теста с открытыми ответами

Структурная диаграмма разработанного нами контролирующего теста, отображающая взаимосвязь составляющих его модулей, представлена на рис.1. Эта диаграмма отражает работу с модельной тестовой базой данных Geometry.ot, содержащей четыре вопроса, из которых в окно браузера выводятся только три.

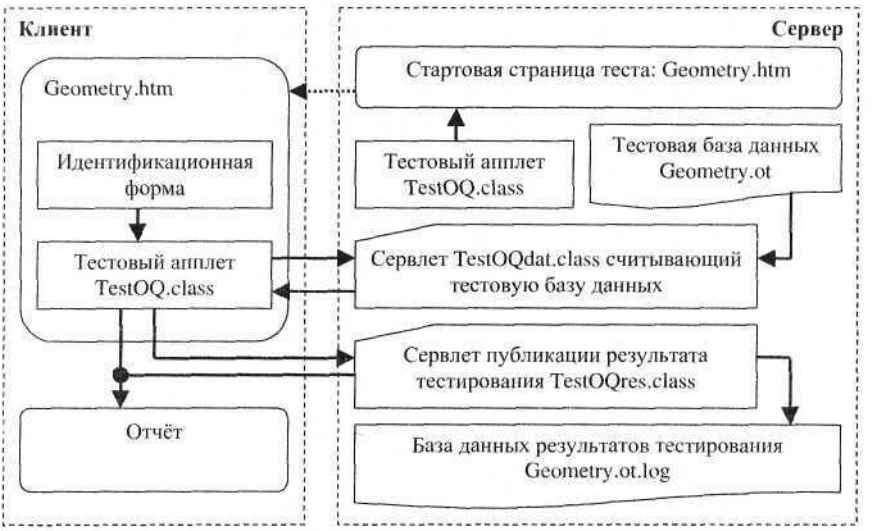

Рис. 1

Использованные в данной диаграмме классы получены компиляцией файлов, содержащих код программы на языке Java, к ним относятся TestOQ.java, TestOQdat.java, TestOQres.java.

Модуль TestOQ.htm выводит в окно браузера шаблон для заполнения обучаемым данных о себе (рис. 2).

После щелчка на кнопке «Начать тест» управление передается апплету TestOQ.class. Структура класса TestOQ.java изображена на рис. 3.

Рис. 2

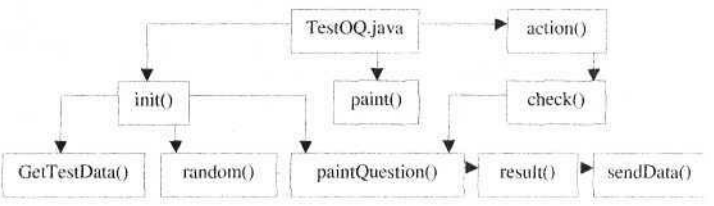

Рис. 3

Апплет TestOQ.java выводит на экран форму для вывода вопросов и ввода ответов учащегося, включающую в себя шесть окон: infwndl, wndl, infwnd2, wnd2, infwnd.3, wnd3 (рис. 4).

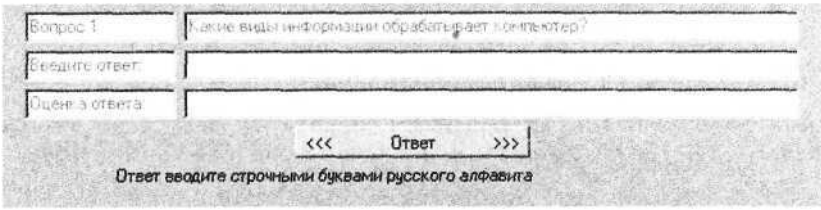

Рис. 4

Дадим краткие пояснения по основным методам класса TestOQ.java. Метод init() осуществляет инициализацию объектов класса, получает информацию о тестируемом и имени тестовой базы данных из HTML-документа. Метод GetTestData() запрашивает на стороне сервера тестовые данные через сервлет TestOQdat.class.

Сервлет TestOQdat содержит метод doGet(). Апплет, обращаясь к сервлету, вызывает метод doGet(), который получает от апплета имя тестового файла и открывает его, распознает, а полученные данные отправляются обратно апплету.

Метод random() осуществляет генерирование вопросов в случайном порядке. Метод paint() используется для отображения информации методом рисования. Метод action() обрабатывает пользовательские события, возникающие в апплете (в частности, щелчок на кнопке «Ответ» после ввода ответа). Метод check() осуществляет обработку ответов учащегося и передает управление методу paintQuestion(), который изображает новый вопрос и определяет момент окончания предъявления вопросов, В случае окончания тестирования управление передается методу resuit(), осуществляющему подсчет четырехбалльной оценки и передачу результатов теста сервлету TestOQres.class посредством метода sendData(). Метод sendData() передает сервлету информацию о тестируемом (идентификационная информация и результаты теста).

Сервлет TestOQres содержит метод doGet(), который отвечает за получение от апплета результирующей информации, которую он формирует в виде HTML-документа и отправляет обратно тестируемому, кроме того, результат тестирования сохраняется в текстовом файле на стороне сервера.

После завершения обучаемым ввода ответов на предъявленные вопросы в окне браузера выводится информация о выполнении теста обучаемым (рис. 5). На рис.6 показано, как эта же информация сохраняется на стороне сервера в текстовом файле.

Рис. 5

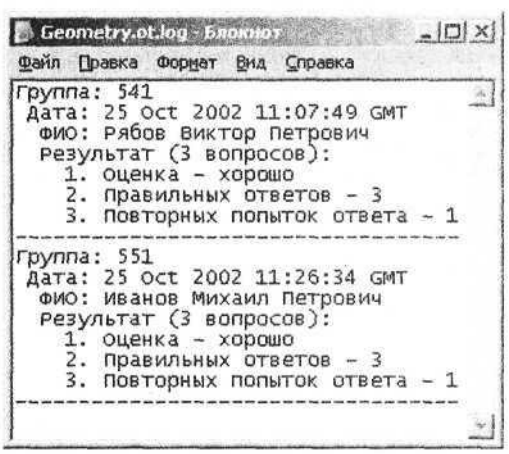

Рис. 6

Дадим краткую характеристику основных из вышеперечисленных методов класса TestOQ.java.

Генерация вопросов

В компьютерных тестах с целью уменьшения вероятности списывания рекомендуется выдавать вопросы на экран из базы данных в случайном порядке. Тогда внешне предъявляемые обучаемым варианты тестов будут казаться разными, этот эффект усиливается, если база данных достаточно большая. Метод random(), генерирующий вопросы из базы данных, можно построить так, как это сделано в [4].

Анализ текстового ответа обучаемого

В контролирующих тестах с открытыми ответами ответ на предъявляемый вопрос вводится обучаемым в текстовой форме. Этот ответ сравнивается с эталонным (правильным) ответом. Для многих предметов школьной программы выбор критерия оценки правильности ответа при такой организации теста очень затруднителен, так как требует решения вопросов степени развернутости ответа, грамотности ученика и т. д. [2].

С целью уменьшения трудностей анализа ответа обучаемого эталонный ответ обычно конструируется из ключевых слов, существенных для передачи смысла ответа [3]. Например, ответом на вопрос «Чему равна площадь ромба, выраженная через его диагонали?» может служить предложение «Площадь ромба равна половине произведения его диагоналей». В качестве ключевых слов выберем следующие три: «половине», «произведения», «диагоналей». Эти слова сократим до четырех символов с целью уменьшения ошибок, связанных с набором слов с помощью клавиатуры, а также ошибок склонения существительных и прилагательных и спряжения глаголов. Таким образом, эталонные слова образуют массив

```
etalon=new Array("поло","прои","диаг")
```

В нижеприведенном фрагменте программы, являющимся частью метода check(), для анализа текстового ответа учащегося, вводимого с клавиатуры и находящегося в окне wnd2, выделяются лексемы, размещаемые в классе StringTokenizer(), которые урезаются до четырех символов:

```
StringTokenizer st=new StringTokenizer(wnd2.getText());
  int mm = (int) st.countTokens ();
  Vector s = new Vector(mm);
  while(st.hasMoreTokens()) {
    String str=st.nextToken();
    if(str.length()>4) str=str.substring(0,4);
      s.addElement(str);
  }
```

Обработка ответа обучаемого

Предполагая, что выделенные из ответа учащегося лексемы (урезанные до четырех символов) находятся в векторе Vector s. Приведем фрагмент программы, являющимся частью метода check(), осуществляющего сравнение компонент вектора s с компонентами массива *etalon,* содержащего ключевые слова предъявленного вопроса с номером *numbers[indexl]:*

```
int q = 0;
int nn = etalon[numbers[index]].length;
for(int j=0;j<nn;j++)
  for(int i=0;i<mm;i++)
   if((s.elementAt(i) ).equals(etalon[numbers[index]][j]))
     {q++;}
wnd2.setText("");
if(q==nn) {
  wnd3.setText("Правильно! Отвечайте на следующий вопрос.");
  r.ight++;
  paintQuestion();}
else{
```

```
    if(second_attempt){wnd3.setText("Неправильно!
    Отвечайте на следующий вопрос.");
      paintQuestion();}
    else{ wnd3.setText("Неправильно!  Повторите попытку");
      second_attempt=true;repeated++;}}
}
```

Оценка ответа учащегося

Число верных ответов учащегося уже является показателем уровня выполнения теста. Можно также на основе этого показателя вывести традиционную четырехбалльную оценку с помощью метода result(), который можно составить по аналогии с методом give_mark() из работы [4].

Структура базы данных

Создание тестовой базы данных заключается в следующем. С помощью любого текстового редактора, например Блокнота, формируете текстовый файл следующего вида:

```
[T<]Геометрия
[N=]3
[Q=]Чему равна площадь ромба, выраженная через его диагонали?
[K=]поло прои диаг
[Q=]В каком прямоугольнике диагонали взаимно перпендикулярны?
[K=]квад
[Q=]В каком четырехугольнике диагонали равны?
[K=]прям
[Q=]В каком четырехугольнике диагонали в точке пересечения делятся
    пополам?
[K=]пара
[T>]
```

Сервлет TestOQdat воспринимает текст между тегом [T<] и [T>] как содержимое теста. Тег [N=] определяет количество предъявляемых тестируемым вопросов. Тег [Q=] указывает сервлету на то, что за этим тэгом стоит текст вопроса, а тег [K=] — на набор ключевых слов для оценки ответа. Весь текст, размещаемый вне строк с тегами, игнорируется. Подобная структура была выбрана с целью упрощения написания программного кода сервлета, наиболее удобным с точки зрения формирования структуры данных является использование XML-разметки, что предполагается реализовать в ближайшее время.

Разработка подобной системы тестирования требует хорошего знания языка программирования Java. Однако созданная система может быть легко использована людьми, не знакомыми с языками программирования, поскольку конструирование нового теста потребует создания только текстового файла, содержащего вопросы и ключевые слова по приведенному образцу, и стартовой страницы, в которой повторяется весь HTML-текст, кроме имени тестового файла. Полный набор исходных файлов и откомпилированных классов рассмотренного теста с открытыми ответами можно получить по адресу http://aks.dem.ru/resources

Литература

1. http://www.ed.gov.ru/ntp/fp/pfzp

2. *Ермаков М. Г. Андреева Л. Е.* Вопросы разработки тестирующих программ//Информатика и образование. 1997. № 3.

3. *Матушанский Г. У.* Проектирование педагогических тестов для контроля знаний //Информатика и образование. 2000. № 6.

4. *Тарасов В. А., Тарасов В. В.* Разработка контролирующих HTML-документов//Информатика и образование. 2001. № 3.