УДК 00.004.9

СИСТЕМА ФОРМИРОВАНИЯ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ ДЛЯ СТУДЕНТОВ ПО ТЕХНИЧЕСКИМ ДИСЦИПЛИНАМ

Золотов Д.Ю., студент; Гранков М.В., доц., к.т.н.
(Донской государственный технический университет, г. Ростов-на-Дону, Россия)

Введение. Информационно-коммуникационные технологии, позволяющие студентам быстро и удобно получать учебный материал, проводить самоконтроль знаний и эффективно обмениваться информацией, становятся в современном образовании не просто удобством, а необходимостью. Не менее важным является применение таких технологий в профессиональной деятельности преподавателей. Возможность автоматизировать рутинные процедуры по созданию индивидуальных заданий для студентов, контролю их выполнения, и доставке этого материала студентам, позволит преподавателям освободить время для разработки новых подходов и технологий в образовании. Задания для лабораторных, курсовых работ и проектов по техническим дисциплинам, таким как Детали машин и Электротехника, включают графическую компоненту, отличаются большим количеством параметров. Преподавателям очень сложно варьировать структуру и значение параметров, обеспечивая непротиворечивость и индивидуальность заданий.

Целью данного исследования является разработка и внедрение технологии автоматизации подготовки индивидуальных заданий для студентов технических специальностей.

Основными задачами являются разработка языка описания класса заданий и программная реализация интерпретатора этого языка.

Основная идея исследования. В Донском Государственном Техническом Университете уже около десяти лет проводятся исследования по решению данной проблемы. На первом этапе по заданию кафедры Теория Конструирования Машин была разработана программа, позволяющая генерировать индивидуальные задания по одной курсовой работе. Многочисленные модификации этой программы, которые выполнялись по требованиям кафедры, привели нас к мысли о необходимости использования языка описания заданий. Из анализа наиболее близких к данной проблеме систем управления образовательным контентом[1] (LMS) следовало, что используемые в этих системах средства, например, системы описаний тестовых заданий, совершенно не подходят для достижения поставленной нами цели. Поэтому было принято решения о самостоятельной разработке языка описания заданий и создании интерпретатора этого языка.

Язык описания генерации заданий. Разрабатываемый язык должен позволять описывать сценарии работы генератора заданий как в диалоговом, так и в автоматическом режимах. Операторы языка должны позволять описывать структуру задания, включая графические компоненты, параметры задания, способы выбора их значений и контроля непротиворечивости. Очевидно, что язык должен содержать операторы организации диалога, циклов, ветвления, описания множеств допустимых значений параметров и методы случайного их выбора. Синтаксис и семантика такого языка должны легко поддаваться пониманию и овладению преподавателем высшего учебного заведения. Второй задачей является написания интерпретатора на любом существующем языке программирования, который будет для каждой инструкции языка создания генераторов выводить графическое представление на экран [2].

Язык описания логики генератора основан на стандарте XML. Поэтому он будет иметь древовидную структуру и состоять из тегов и атрибутов [3]. Описываемый в данной работе язык подчиняется системе правил и относится к декларативным. Из этого вытекает, что через его инструкции определяются данные, которые необходимо получить, а не сам процесс их получения [4]. Структура языка представляет собой дерево из элементов, которые

располагаются в произвольном порядке. Сценарий XML требует наличия главного или корневого элемента [3]. Имя сценария непринципиально и следует из содержимого сценария.

Некоторые инструкции языка сценариев взяты из существующих языков программирования. Одним из них является оператор ветвления. В данном языке он представлен в виде тега с названием CASE. Эта конструкция служит для создания диалога, в котором пользователю программы предоставляется возможность выбора одного из вариантов. Она включает стандартные атрибуты «id» – уникальный идентификатор и «what» – название блока, которое выводится на экран. Особое место заслуживает атрибут «where», которое указывает на тип источника, откуда интерпретатор будет считывать возможные варианты. Источником может быть файлом XML, файлом INI, текстовым файлом TXT, таблицей реляционной базы данных или сам тег CASE. Если это файл, то используется зарезервированное имя «FILE», если база данных, то «DB», если это простое перечисление вариантов в самом теге, то «HERE». В последнем случае в атрибуте «cont» тега CASE, в котором через разделитель, например двоеточие, перечисляются названия вариантов, во всех остальных «cont» содержит название файла, имя базы данных и таблицы. Тег CASE включает дочерние узлы SEL и ELSE. Конструкция SEL представляет собой ветку, при выборе которой будут произведены определенные действия. Другими словами, это блок связанных по смыслу команд. Конструкция ELSE оператора ветвления отличается от SEL тем, что при старте генератора будет по умолчанию выбираться именно эта ветвь. Конструкция SEL как и ELSE состоит из атрибута «what», который содержит краткое пояснение к содержимому команд этого блока. Работу данной конструкции можно охарактеризовать следующим образом. Между программой и преподавателем происходит диалог через интерфейс программы, где система задает вопросы и в зависимости от ответа преподавателя может последовать следующий вопрос или завершение диалога с выполнением определенных действий. Оператор ветвления CASE в любой своей ветви может включать другой оператор ветвления, то есть быть вложенным.

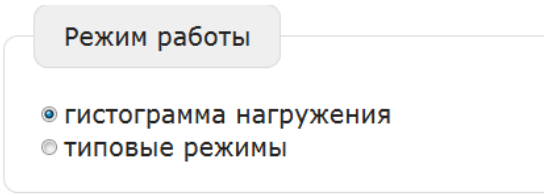

Рисунок 1 – Конструкция CASE

Основным элементом, который позволяет пользователю вручную задавать значение из возможных, является CHOOSE. Значениями конструкции CHOOSE могут быть строки, изображения, числа. Данный тег состоит из стандартных атрибутов «id» и «what». Атрибут «where» указывает тип источника данных, которым может являться текстовый файл, таблица базы данных. В отличие от выше описанного оператора ветвления данный атрибут может иметь в качестве источника данных файловую директорию, для чего служит зарезервированное слово языка «FILE_DIRECTORY». В этом случае возможными значениями будут изображения, и даже целые текстовые файлы. Название файловой директории задается в атрибуте «cont». При создании интерпретатора определение типа файла можно реализовать через анализ расширения имени файла. Для явного задания типа в конструкции CHOOSE имеется атрибут «type», в котором сообщается тип информации источника, например, картинка это или текстовый файл. Если в атрибуте «where» указана файловая директория, а в «type» слово IMAGE, то результатом обработки данной конструкции CHOOSE станет вывод слайдера или названий изображений. На рисунке 2 приведен один из возможных вариантов представления на экране конструкции CHOOSE после обработки конструкции интерпретатором, а на рисунке 3 конструкции CHOOSE, но с атрибутом «type» равным IMAGE.

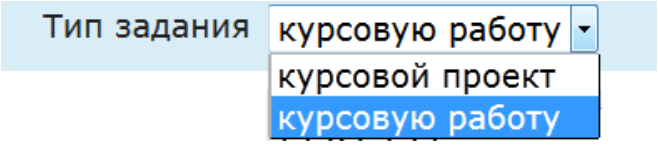

Рисунок 2 – Конструкция CHOOSE

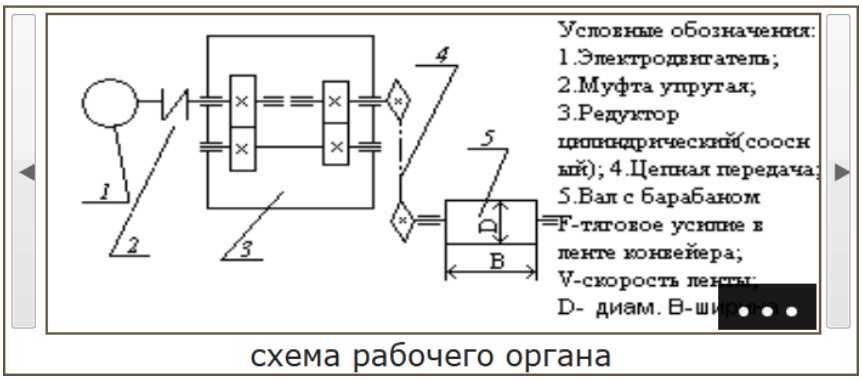

Рисунок 3 – Конструкция CHOOSE с выбором изображения(слайдер)

Расширением функциональности конструкции CHOOSE является тег с названием AUTOCHOOSE. Данный элемент позволяет выбрать значение вручную как у CHOOSE, так и сгенерировать значение случайным образом из заданного множества. В конструкции CHOOSE имеется возможность задания диапазона значений для автоматического выбора. Если в атрибуте «type» установлено значение number, то диапазон может задаваться через начальное и конечное значение в атрибуте «constr»(ограничение).На рисунке 4 приведен один из возможных вариантов представления конструкции AUTOCHOOSE на экране после обработки конструкции интерпретатором.

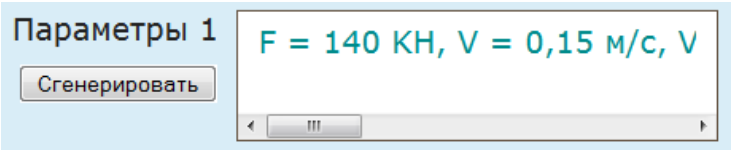

Рисунок 4 – Конструкция AUTOCHOOSE

С помощью конструкции ENTER (ввод) реализуется возможность ввода пользователем произвольного значения параметров задания. С помощью атрибута «type» этого тега имеется возможность описывать типы вводимых данных. На рисунке 5 приведен один из возможных вариантов представления на экране конструкции ENTER после обработки конструкции интерпретатором.

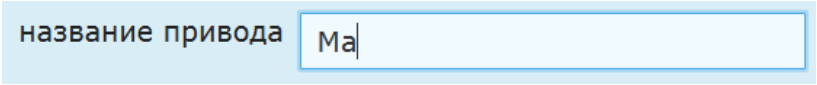

Рисунок 5 – Конструкция ENTER

В языке предусмотрен тег AUTO, являющийся итератором и позволяющий описать автоматический выбор значений нескольких параметров задания.

Описанные конструкции языка располагаются в простом текстовом файле с расширением XML. В файле требуется наличие корневого тега. Для удобства восприятия текста сценария использовать несколько файлов и связать их с помощью конструкций SCRIPT. В атрибуте name тегов SCRIPT указываются имена подключаемых файлов. В результате появляется возможность организовать модульную структуру сценария.

Второй задачей исследования являлась разработка интерпретатора языка сценариев, который будет считывать конструкции из XML файлов и выводить их в удобном графическом представлении. Рассмотрим несколько сценариев, написанных на созданном языке для генератора по предмету «Детали машин». В первом из них, в сценарии под

названием MAIN (Рисунок 6) описываются общие данные, такие как личные данные студентов и преподавателя, тип задания и прочее. Первая конструкция ENTER сообщает интерпретатору, чтобы тот создал поле для ручного ввода ФИО преподавателя, на что указывает атрибут «string».Другая конструкция ENTER этого же скрипта содержит дату выдачи работы на руки студенту, отличие состоит в том, что в атрибуте «type» стоит значение date для которого будет создано специальное поле для удобного ввода даты. Дальше идут две конструкции CHOOSE, в результате обработки которых на экране будет два выпадающих списка. Отличаются эти конструкции источником данных: студенты будут выбираться из таблицы «студенты» подключенной базы данных (атрибут where=”DB”), а тип задания из значений атрибута «cont» (where=”here”) И завершающей конструкцией скрипта является XML-тег SCRIPT, который указывает интерпретатору обработать следующий сценарий с названием details (детали).

```
<MAIN>
    <ENTER id="m2" what="ФИО преподавателя" type="string"/>
    <CHOOSE id="m2"
            what="ФИО студента"
            where="DB"
            cont="студенты" />
    <CHOOSE id="m1"
            what="Тип задания"
            where="here"
            cont="курсовой проект; курсовую работу" />
    <ENTER id="m3" what="Дата выдачи" type="date"/>
    <SCRIPT what="details"/>
</MAIN>
```

Рисунок 6 – Сценарий main.xml

В следующем сценарии details (Рисунок 7) рассматриваются параметры деталей машин. Сценарий примечателен конструкцией CHOOSE с атрибутом «where» равным «FILE_DIRECTORY». Результатом обработки интерпретатором станет вывод содержимого трех файлов с названиями из атрибута «cont»

```
<DETAILS>
    <CHOOSE id="d1"
            what="коэффициент использования в течении года"
            where="here"
            cont="0.1;0.2;0.3;0.4;0.5;0.6;0.7;0.8;0.9;1.0" />
    <CHOOSE id="d2"
            what="коэффициент использования в течении суток"
            where="here"
            cont="0.1;0.2;0.3;0.4;0.5;0.6;0.7;0.8;0.9;1.0" />
    <CHOOSE id="d3"
            what="вариант задания для графической части"
            where="FILE_DIRECTORY"
            type="text"
            cont="A.txt;B.txt;C.txt" />
    <CHOOSE id="d4"
            what="срок службы"
            where="here"
            cont="1;2;3;4;5;6;7;8;9;10" />
    <SCRIPT what="work_mode"/>
</DETAILS>
```

Рисунок 7 – Сценарий details.xml

Теперь рассмотрим более сложный сценарий с такими конструкциями как CASE и AUTO (Рисунок 8). Пользователю предстоит задать рабочий режим детали машины. Для этого он должен выбрать либо типовые режимы или гистограмму нагружения. Эти значения будут взяты согласно атрибуту «where» прямо из самого тега CASE, а именно из атрибута «cont». Конструкция ELSE выводится на экран, уже выбранной по умолчанию. После обработки ELSE на экране будет один выпадающий список с меткой “типовые режимы” и

двумя значениями. С конструкцией SEL дело обстоит иначе, дочерним узлом которой является автогенерируемый блок AUTO. При выборе конструкции SEL пользователь увидит два поля со случайными данными.

```
<WORK_MODE>
<CASE id="wm1"
    what="Режим работы"
    where="here"
    cont="гистограмма нагружения; типовые режимы">
    <SEL what="Гистограмма нагружения">
        <AUTO what="Генерировать гистограмму">
            <CHOOSE
                name="wm2"
                what="Гистограмма"
                where="here" type="image" cont="гистограмма.bmp"/>
            <CHOOSE id="wm3"
                    what="a1"
                    where="here"
                    cont="0.1;0.2;0.3;0.4;0.5;0.6;0.7;0.8;0.9;1.0"/>
            <CHOOSE id="wm4"
                    what="a2"
                    where="here"
                    cont="0.1;0.2;0.3;0.4;0.5;0.6;0.7;0.8;0.9;1.0"/>
        </AUTO>
    </SEL>
    <ELSE>
        <CHOOSE id="wm11"
            what="типовые режимы"
            where="here"
            cont="Типовой постоянный; типовой тяжелый" />
    </ELSE>
</CASE>
</WORK MODE>
```

Рисунок 8 – Сценарий work_mode.xml

Выводы. Проведенные исследования показали актуальность и возможность создания специальных языков описания образовательного контента. В качестве базовой платформы описания таких языков эффективно использовать язык XML. Разработанный язык сценариев генерации классов заданий по техническим дисциплинам на основе XML и его интерпретатор внедрены в программное обеспечение портала «Образовательный процесс ДГТУ» Донского Государственного Технического Университета. (http://ec.donstu.ru).

Перечень ссылок

1. Learning_management_system // Википедия URL: 3. https://en.wikipedia.org/wiki/Learning_management_system (дата обращения: 14.04.2015).
2. Создание языка на основе XML для декларативного описания пользовательского интерфейса // www.ibm.com URL: http://www.ibm.com/developerworks/ru/library/x-decxmlui/ (дата обращения: 19.03.2015).
3. Хантер Д., Рафтер Д. XML. Базовый курс. М.: Вильямс, 2009. – 1344 с.
4. Императивные и декларативные языки программирования // lisiynos.googlecode.com URL: http://lisiynos.googlecode.com/git-history/2f44be2be1b5cc92e92705d51393a3ffcd0a74a9/dp/imperative_declarative.html (дата обращения: 15.04.2015).