УДК 004.65

Иерархические виджеты: организация интерфейса пользователя в веб-приложениях на основе ситуационно-ориентированных баз данных

В. В. Канашин[1], **В. В. Миронов**[2]

[1] kanashin@mail.ru, [2] mironov@list.ru

ФГБОУ ВПО «Уфимский государственный авиационный технический университет» (УГАТУ)

Поступило в редакцию 22.12.2012

Аннотация. Рассматривается построение сложно-структурированного интерфейса пользователя в интернет-приложениях, функционирующих на основе ситуационно-ориентированных баз данных (СОБД). Предлагается концепция иерархически организованных объектов интерфейса – виджетов, ассоциированных с состояниями динамической модели СОБД и генерирующих контент в процессе интерпретации динамической модели. Обсуждается состав, структура и функциональность иерархических виджетов. Разбирается содержательный пример задания и функционирования иерархических виджетов.

Ключевые слова. *Веб-приложение; интерфейс пользователя; ситуационно-ориентированная база данных; динамическая модель; XML; XSLT; model-driven development*

Тенденция к абстрагированию при построении прикладных программ находит свое отражение в абстрагировании моделей данных, что позволяет описывать бизнес-процессы приложения на более высоком уровне и делает модели данных основными звеньями разработки, из которых генерируются коды программ и другие звенья (Model-Driven Development [1]).

В русле этого подхода может рассматриваться идея ситуационно-ориентированных баз данных (СОБД) [2], в которых XML-данные ассоциированы с состояниями встроенной динамической модели и могут обрабатываться в контексте ее текущих состояний [3–5]. СОБД могут служить основой веб-приложений с динамическим формированием контента в соответствии со сложившейся ситуацией. Такой подход к построению веб-приложений удобен при обслуживании бизнес-процессов, которые носят ситуационный характер. Например, на одном этапе бизнес-процесса необходимо выполнить одни действия и для этого требуются одни данные, на другом – другие; каждый из этапов включает несколько подэтапов со своей спецификой и т. д. Преимуществом СОБД является возможность обработки данных в контексте ситуации, развитие которой отслеживается с помощью динамической модели.

Работа поддержана грантом РФФИ 13-07-00011.

Важным аспектом обработки данных СОБД является формирование контента, отправляемого пользователю в ответ на его запрос. Обычно это включает: 1) сведения, необходимые пользователю в данной ситуации (в текстовой, табличной, графической и иной форме); 2) элементы управления для ввода пользователем ответных данных и принятых в контексте ситуации решений и отправки их на сервер (поля ввода, кнопки, переключатели и т. п.). Все это образует интерфейс пользователя, который в современных веб-приложениях формируется на сервере в виде HTML / JavaScript / CSS-кода и отправляется для отображения в клиентский браузер.

Данная статья посвящена построению пользовательского интерфейса на основе СОБД.

ВВОДНЫЕ ЗАМЕЧАНИЯ

СОБД мы рассматриваем как набор следующих компонентов [6]: HSML (HSM Library) – библиотека динамических моделей, содержащая набор иерархических ситуационных моделей HSM (Hierarchical Situational Models) в виде иерархии графов переходов с конечным числом состояний (Finite State Model); CSM (Current State Memory) — память текущего состояния, хранящая сведения о текущих состояниях динамических моделей; ADM (Associated Data Memory) — память ассоциированных данных, хранящая XML-данные,

соотнесенные (ассоциированные) с различными состояниями динамической модели; AFL (Associated Functions Library) — библиотека ассоциированных функций, хранящая функции обработки данных, соотнесенные с состояниями динамической модели; HSMI (HSM Interpreter) — интерпретатор динамической модели, который в ответ на внешний запрос *Q* формирует ответ *R* путем обработки некоторой динамической модели HSM из HSML на основе отслеживания ее текущих состояний, сохраняемых в CSM, обработки ассоциированных данных из ADM и выполнения ассоциированных функций из AFL.

Так, если СОБД используется на веб-сервере как основа для интернет-приложения, то в качестве входного запроса *Q* выступает набор параметров, получаемый вместе с URL (например, параметры формы в режиме POST), а в качестве результата *R* — ответный HTML-код, отправляемый клиенту. Параметры запроса задают обрабатываемую динамическую модель и необходимость смены ее текущих состояний [6].

Задание интерфейса пользователя в динамической модели. Рассмотрим имеющиеся возможности формирования интерфейса пользователя на основе СОБД. В настоящее время эта функциональность достигается с помощью размещения в динамической модели:

- *элементов управления*, специфицирующих параметры, на основе которых интерпретатор формирует HTML-код отдельных элементов управления, отправляемый в браузер;
- *элементов-акций*, предусматривающих вызов прикладных функций из AFL, выполнение которых приводит к формированию HTML-кода фрагментов интерфейса;
- *DOM-элементов*, специфицирующих обработку интерпретатором XML-документов из ADM, в ходе которой может быть предусмотрено формирование HTML-кода фрагментов интерфейса.

Указанные возможности являются в той или иной степени альтернативными.

Пример. На рис. 1, *а* в качестве иллюстрации приведен фрагмент динамической модели из работы [6], иллюстрирующий применение элементов управления для формирования интерфейса пользователя, на рис. 1, *б* – фрагменты результирующих экранных форм, а на рис. 1, *в* – соответствующие модели HTML-кода экранных форм.

Фрагмент модели представляет собой субмодель *Студенты* и включает в себя два состояния: *СписокСтудентов* и *ВыбранСтудент*.

Переходы состояний обеспечиваются jmp-элементами, которые становятся активными при нажатии пользователем кнопок «К предметам», «К студентам», «Успеваемость студента» соответственно (факт нажатия кнопок обнаруживается интерпретатором из массива POST, полученного от клиента вместе с URL).

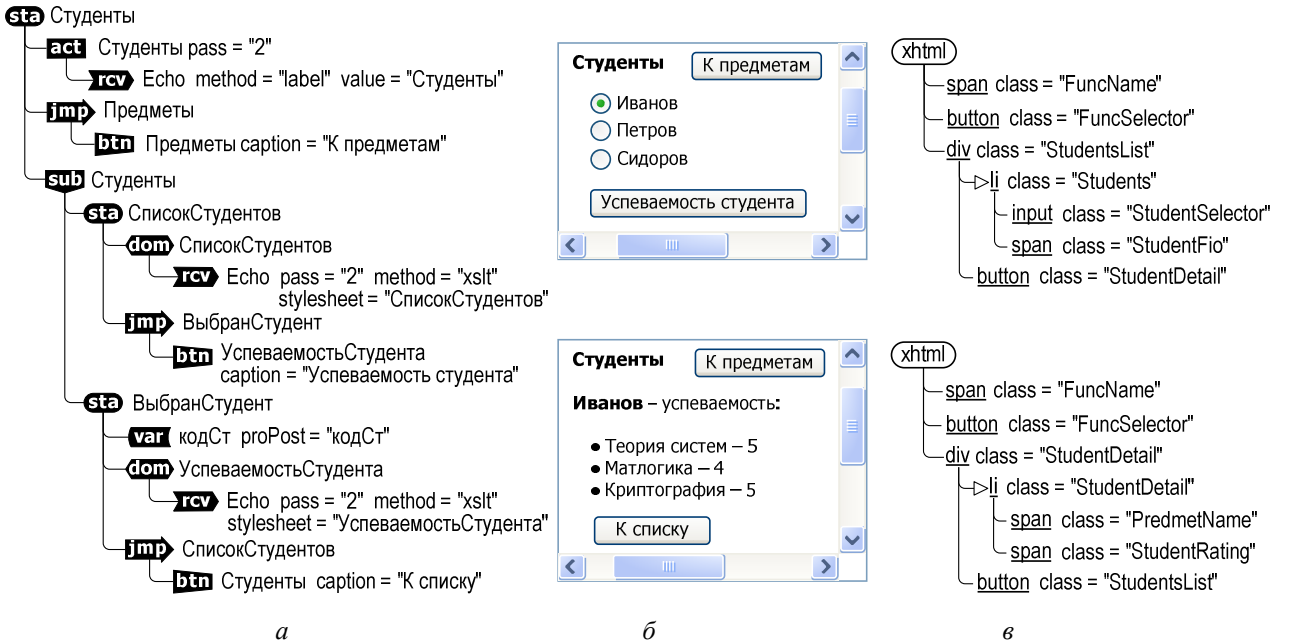


Рис. 1. Формирование клиентской страницы, содержащей интерфейс пользователя, на основе динамической модели (известный подход [6]): *а* – фрагмент динамической модели; *б* – фрагменты изображения для двух состояний; *в* – структура HTML-кода

В состоянии «Студенты» формируются заголовок «Студенты» (HTML-элемент с помощью HSM-акции «Студенты») и кнопка «К предметам» (HTML-элемент <button class = "FuncSelector"> с помощью HSM-элемента btn:Предметы), общие для всех подсостояний.

В подсостоянии «СписокСтудентов» с помощью HSM-элемента dom:СписокСтудентов формируется блок (HTML-элемент), внутри которого размещается набор строк, содержащих радиокнопку (HTML-элемент <input class = "StudentsSelector">) и фамилию студента (HTML-элемент), а с помощью HSM-элемента btn:УспеваемостьСтудента формируется кнопка «Успеваемость студента» (HTML-элемент <button class = "StudentDetail">).

Аналогичным образом в подсостоянии «ВыбранСтудент» с помощью HSM-элемента dom: УспеваемостьСтудента формируется блок (HTML-элемент), внутри которого размещается набор строк, содержащих название сданного предмета (HTML-элемент <input class = "PredmName">) и полученную оценку (HTML-элемент), а с помощью HSM-элемента btn:Студенты формируется кнопка «К списку» (HTML-элемент <button class = "StudentsList">).

Предполагается, что встроенная динамическая модель по своей структуре достаточно близка к модели бизнес-процесса, которая создается разработчиком на этапе концептуального проектирования интернет-приложения, поэтому этапы логического и физического проектирования приложения будут менее трудоемкими, чем программирование кода приложения «с нуля». А это значит, что бизнес-процессам с достаточно сложной логикой, обычно, соответствуют динамические модели иерархического вида, где с каждым состоянием могут быть связаны внутренние подмодели, в свою очередь, включающие набор состояний. Следовательно, отображаемый контент, ассоциированный с состояниями, в этих случаях также имеет иерархическую организацию, что хорошо соотносится с описанной выше концепцией иерархических виджетов.

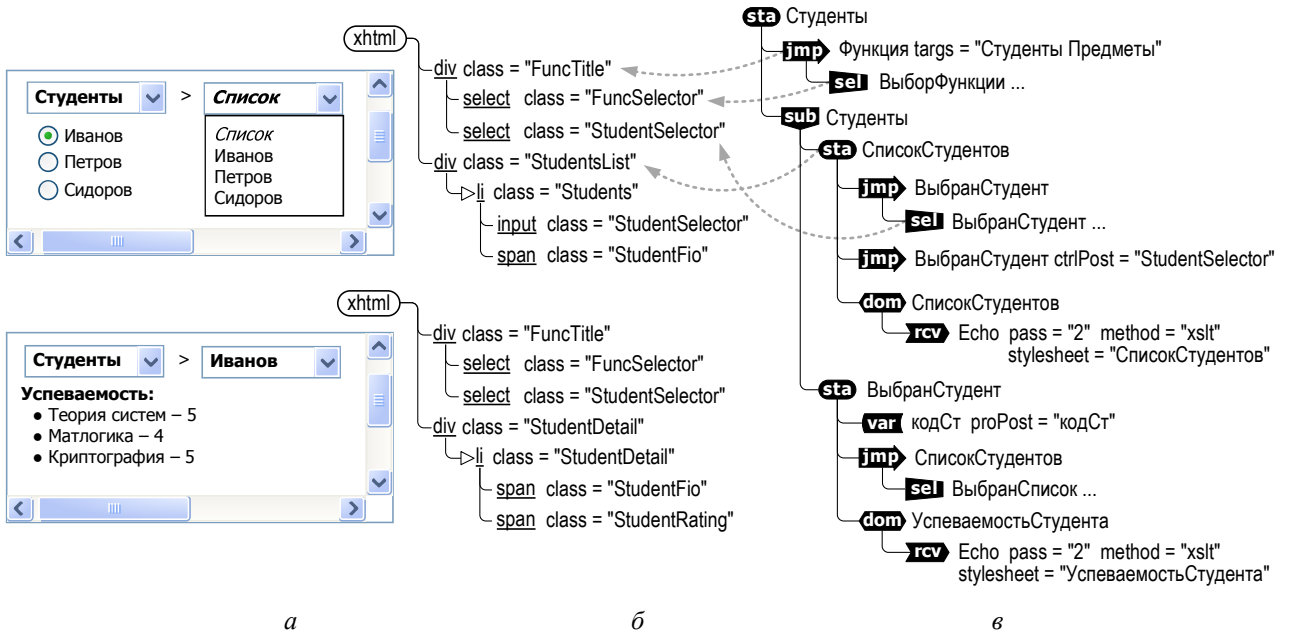
Существующая концепция при реализации встроенных динамических моделей предлагает руководствоваться принципом соответствия структуры изображения структуре динамической модели. В соответствии с этим принципом каждой подмодели, входящей в состав динамической модели, на экране соответствует некоторая область изображения, которая заполняется контентом в соответствии с текущим состоянием подмодели. Область отображения подмодели, в свою очередь, включает постоянную область, содержимое которой не зависит от текущего состояния подмодели, и переменную область, содержимое которой определяется текущим состоянием подмодели. При смене текущего состояния содержимое переменной области изменится. Если текущее состояние, в свою очередь, содержит подмодель, то переменная область будет содержать подобласть, соответствующую этой подмодели. Таким образом, полная структура изображения является оверлейной (перекрывающейся), состоящей из налагаемых друг на друга областей [3].

Таким образом, выходная страница формируется достаточно просто, если ее иерархическая структура в определенном смысле соответствует структуре динамической модели. Вопрос в том, всегда ли можно соблюсти это соответствие?

Сложность реализации множественного доступа как недостаток данного подхода. В современных интерфейсах пользователя широко применяется дублирование управляющих элементов для предоставления пользователю различных способов доступа к функциям и данным. Отображаемые страницы могут содержать несколько панелей управления со сходными функциями (скажем, несколько меню: основное горизонтальное вверху страницы, дополнительное – внизу, боковое древовидное – в левой колонке и т. д.).

В этих случаях иерархическая структура формируемого изображения (определяемая вложенностью элементов HTML-кода, таких как div, span, элементов управления и других) не совпадает с иерархической структурой динамической модели. Возникает затруднение, связанное с определенной последовательностью обработки элементов динамической модели в процессе интерпретации (интерпретатор выполняет обход текущих состояний иерархической модели «сверху вниз, слева направо»). Если, находясь в некотором состоянии, был сгенерирован результирующий HTML-элемент-контейнер (div или span), то HTML-контент, сгенерированный в подсостояниях этого состояния, будет размещен внутри этого контейнера.

Пример. Чтобы пояснить данное затруднение, рассмотрим следующий пример (рис. 3). Пусть нам требуется построить интерфейс, представленный на рис. 3, *а*.



а *б* *в*

Рис. 2. Попытка построения интерфейса с множественным доступом к функциям:
а – фрагменты изображения для двух состояний; *б* – требуемая структура HTML-кода;
в – динамической модель (безуспешная) для реализации интерфейса

В верхней строке мы хотим разместить панель управления, в которой первый раскрывающийся список позволяет выбирать режим представления информации (сведения о студентах, сведения о предметах и т. п.), а второй – режим отображения (общий список или для отдельного студента или предмета). В состоянии «Студент» второй раскрывающийся список имеет текущую опцию «Список», а в состоянии «Выбран студент» – опцию, соответствующую выбранному студенту. Таким образом, в состоянии «Студент» пользователю предоставляется возможность множественного выбора студента: 1) через вертикальный список студентов путем нажатия на соответствующую радиокнопку; 2) через раскрывающийся список в панели управления путем выбора опции, соответствующей тому или иному студенту.

Чтобы реализовать такой интерфейс, необходимо в зависимости от состояния динамической модели генерировать выходной HTML-код, представленный на рис. 3, *б*. Панель управления размещена в div-блоке FuncTitle, а основная информация – в div-блоках StudentSelector и StudentDetail. Раскрывающиеся списки реализуются с помощью HTML-элементов select.

Попытаемся построить динамическую модель, обеспечивающую порождение нужного HTML-кода (рис. 3, *в*).

Контейнер div для панели управления может быть сгенерирован jmp-элементом «Функция» в состоянии sta:Студенты. Первый раскрывающийся список внутри контейнера панели управления генерируем с помощью sel-элемента, вложенного в jmp-элемент.

Заметим, что мы не можем здесь же сгенерировать второй раскрывающийся список, поскольку его вид (текущая опция) зависит от подсостояний состояния sta:Студенты; придется решать эту задачу в субмодели.

Контейнер div для основного фрагмента отображения генерируется HSM-элементом sta:СписокСтудентов или sta:ВыбранСтудент в зависимости от текущего состояния субмодели sub:Студенты.

Далее нужно сгенерировать второй раскрывающийся список в составе панели управления. В состоянии sta:СписокСтудентов это естественно было бы сделать с помощью HSM-элемента sel:ВыбранСтудент, управляющего переходом состояния субмодели. Однако этого не удается выполнить корректно.

Весь контент, генерируемый внутри состояния sta:СписокСтудентов, автоматически будет размещаться внутри div-контейнера StedentsList, а не внутри контейнера FuncTitle, как требуется. Аналогичное затруднение возникает и в состоянии sta:ВыбранСтудент.

Для преодоления подобного рода затруднений приходится либо искусственно строить динамическую модель в соответствии со структурой изображения на экране пользователя, либо прибегать к помощи вспомогательных функций. Первое ведет к нежелательному деформированию структуры динамической модели по отношению к структуре исходных бизнес-процессов (динамическая модель будет изменена ради формы представления данных, а не логики ре-

шаемой задачи). Второе ведет к тому, что структурные моменты логики формирования изображения не отражаются в самой динамической модели, а скрываются в алгоритмах прикладных функций.

Таким образом, возникает необходимость в дополнении динамической модели механизмами формирования отображаемой страницы, которые позволяют более гибко формировать структуру HTML-кода отображаемой страницы в ходе интерпретации динамической модели, размещать интерфейсные элементы, данные, CSS- и JavaScript-код более независимо от структуры динамической модели.

КОНЦЕПЦИЯ ИЕРАРХИЧЕСКИХ ВИДЖЕТОВ

Виджет-элементы. Для преодоления рассмотренных затруднений предлагается в динамической модели предусмотреть особый тип элемента управления – *виджет*, предоставляющий возможность гибко специфицировать фрагменты пользовательского интерфейса, ассоциированные с состояниями динамической модели.

Результат виджета. Каждому виджету соответствует *результат* (*код*) *виджета* – некоторая размеченная контентная область, формируемая на втором проходе интерпретации динамической модели на основе обработки *шаблона виджета* (например, путем XSL-трансформации содержимого определенного DOM-объекта в соответствии с заданной таблицей стилей) и отправляемая в браузер клиента для формирования фрагмента изображения. В результат виджета выводятся специфицированные данные: разметка HTML, код JavaScript и CSS, а также интерфейсные элементы управления (кнопки, селекторы и др.).

Иерархии виджетов. Для построения сложных пользовательских интерфейсов предусмотрена возможность организовывать иерархии виджет-элементов, ассоциированных с различными состояниями динамической модели. Для этого в виджете предусмотрена возможность ссылаться на свой родительский виджет, размещенный в некотором состоянием, предшествующем в динамической модели состоянию исходного (дочернего) виджета. У родительских виджетов может быть несколько дочерних, которые, в свою очередь, могут содержать собственные дочерние виджеты, образуя многоуровневую иерархию.

Результаты дочерних виджетов встраиваются интерпретатором в определенные места результирующей разметки своего родительского виджета. Таким образом, необходимо различать *локальный* результат виджета, формируемый путем обработки данного виджет-элемента, и *глобальный* результат виджета, формируемый с учетом локальных результатов всех текущих виджетов-потомков. *Корневому* виджету (не ссылающемуся на родителя) соответствует результирующий код, включающий результаты виджетов-потомков, соответствующих текущим состояниям. В этом смысле иерархия виджетов является *динамической*: результат иерархии виджетов зависит от текущих подсостояний состояния корневого виджета (в зависимости от текущих состояний динамической модели корневой виджет использует различные виджеты-потомки для формирования результата).

Обсуждавшееся выше затруднение, связанное с формированием нескольких управляющих элементов на HTML-странице в разных контентных областях, преодолевается за счет того, что при формировании глобального результата локальные результаты автоматически размещаются внутри HTML-контейнера, в котором был размещен локальный результат корневого виджета. Таким образом, разработчик динамической модели должен обеспечить правильное размещение результата корневого виджета, результаты виджетов-потомков «подцепляются» автоматически.

Как нам представляется, предлагаемая концепция иерархических виджетов даст разработчику гибкий инструмент построения и настройки веб-приложения со сложной структурой отображения данных, уменьшая трудоемкость программирования пользовательского интерфейса.

ЗАДАНИЕ ВИДЖЕТОВ В ДИНАМИЧЕСКОЙ МОДЕЛИ

Для дальнейшего развития концепции иерархических виджетов необходимо уточнить синтаксис и семантику предлагаемого виджет-элемента. На рис. 4 представлена синтаксическая диаграмма виджет-элемента.

Имя_виджет-элемента – это название виджета, которое является его идентификатором. Имя должно быть уникальным среди всех встречающихся или видимых в модели виджетов. В модели допускается несколько виджетов с одинаковыми именами, если они находятся в несовместных состояниях и поэтому не видимы друг для друга.

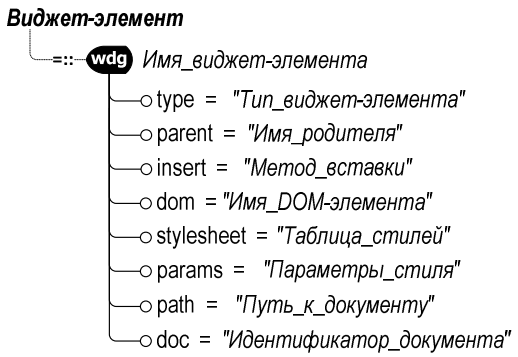


Рис. 4. Синтаксическая диаграмма виджет-элемента

Атрибут type задает тип виджет-элемента:

• "html" означает, что локальный результат виджета представляет собой HTML-код некоторого документа, хранящегося в СОБД; размещение этого документа задается атрибутами path или doc;

• "xslt" означает, что локальный результат формируется путем XSL-трансформации содержимого созданного ранее DOM-объекта; имя этого объекта задается атрибутом dom;

• возможны и другие способы формирования результата виджета.

Атрибут parent хранит имя родительского виджета (аналог внешнего ключа для некорневого виджета). Отсутствие этого атрибута означает, что виджет является корневым – не имеет «родителей» (что не препятствует ему иметь дочерние виджеты).

Атрибут insert определяет способ размещения результата некорневого виджета в составе результата своего родителя:

• значение "append", действующее по умолчанию, означает, что будет производиться прямая конкатенации результата с локальным результатом виджета-родителя;

• значение "prepend" задает обратную конкатенацию – т. е. данные размещаются перед локальным результатом родителя;

• возможны другие способы размещения, например, внутри определенного блока в локальном результате родительского виджета, что требует указания идентификатора или класса этого блока.

ИНТЕРПРЕТАЦИЯ ВИДЖЕТ-ЭЛЕМЕНТОВ

Для обработки динамической иерархической модели (HSM), как описывается в [3–5], используется интерпретатор (HSMI), который «проходит» модель дважды. На первом проходе интерпретатор обрабатывает динамическую модель, контролируя переходы текущего состояния. На втором для зафиксированного текущего состояния выполняется обработка прикладных фрагментов, в результате чего формируется выходной поток данных, направляемый в браузер клиента. Реализация предложенной концепции иерархических виджет-элементов должна учитывать эти особенности общей процедуры интерпретации.

Затруднение, которое необходимо преодолеть при реализации концепции, связано с тем, что как на первом, так и на втором проходах интерпретации выполняется рекурсивный обход модели «сверху-вниз», т. е. от родительских состояний к дочерним, в то время как формирование глобального результата виджета требует обхода «снизу-вверх», т. е. от дочерних виджетов к родительским. Таким образом, локальный результат виджета-родителя, объявляемого выше в иерархии модели, чем его потомки, будет сформирован до того, как результаты дочерних виджетов будут подсоединены к нему.

Может быть предложено два варианта выхода из данного затруднения путем модификации традиционного алгоритма интерпретации динамической модели:

1) Дополнительный восходящий проход при интерпретации дочерних элементов каждого элемента-состояния. Традиционный алгоритм интерпретации обрабатывает дочерние элементы состояния в порядке их размещения в модели. Предлагается после обработки последнего дочернего элемента выполнять повторную их обработку в обратном порядке со специальным признаком, позволяющим интерпретатору на этом этапе выполнять «сборку» глобальных результатов из локальных результатов ранее обработанных виджетов.

2) Сохранение каждого локального результата в буфере и дополнительная (вторичная) обработка виджет-элементов при интерпретации дочерних элементов для сборки глобального результата. Буфер может быть организован на основе DOM-объектов, автоматически создаваемых для каждого виджет-элемента. В ходе вторичной обработки результат из DOM-объекта обрабатываемого виджета переносится в DOM-объект родительского виджета.

Недостатком первого подхода является необходимость адаптировать к нему вывод всех остальных элементов, которые могут быть задействованы в модели. Это означает, что все

элементы должны будут выводиться интерпретатором в информационный поток при «всплытии». Это повлечет за собой изменение принципов построение динамических моделей, и отразиться на наглядности представления иерархии данных.

Недостатки второго подхода проявляются в случае, если виджет-родитель объявлен в модели позже, чем хотя бы один его дочерний виджет. В этом случае предлагается автоматически создавать DOM-объект, который будет буфером для всех виджетов-братьев этого дочернего виджета, пока не найден виджет-родитель. При нахождении родительского виджета, его информация подсоединяется в раннее созданный DOM-объект в соответствии с атрибутом insert. Кроме того, данный подход, как и предыдущий, требует производить вывод данных в информационный поток только после полной обработки всей модели, если в модели встречается хотя бы один элемент-виджет.

Однако в целом второй подход требует меньшей модернизации общей процедуры интерпретации. Он проще в реализации, а также хорошо соотносится с концепцией динамических DOM-объектов [6, 7]. Поэтому именно он взят за основу для реализации иерархических виджетов.

Алгоритм интерпретации виджет-элемента

На рис. 5 представлена структура модуля второго прохода интерпретации, на котором выполняется обработка виджет-элементов и формирования результирующей страницы, отправляемой в клиентский браузер. Серой заливкой выделены блоки 1–4, 11, присутствующие в традиционном алгоритме, белой – новые блоки 5–10, введенные для расширения функциональности.

Используемый здесь и далее «иерархический» способ представления блок-схемы имеет отличия от традиционного. Вход в очередной блок того же уровня вложенности происходит по соединительной линии сверху или справа; выход всегда снизу; вход в первый дочерний (внутренний) блок из данного блока – справа; если внутренний блок последний в цепочке (не имеет соединительной линии снизу), то по его завершении происходит возврат в родительский блок. Такая нотация в большей степени соответствует конструкциям современных процедурных языков высокого уровня.

Модуль выполнен в виде рекурсивной функции Pass2, список входных параметров $s которой представляет состояние обрабатываемой субмодели (исходной динамической модели HSM и соответствующей ей памяти текущего состояния CSM). Интерпретация начинается путем вызова функции Pass2 (блок 1) для корневого состояния динамической модели, которое всегда является текущим, и продолжается для текущих состояний внутренних субмоделей путем рекурсивного погружения (блок 4).

Для обрабатываемого состояния $s выполняется первичная циклическая обработка всех дочерних элементов (блок 2) в зависимости от их типов (блок 3). Если дочерний элемент является субмоделью (sub) или ссылкой на нее (div), функция рекурсивно вызывается (блок 4) для обработки этой субмодели с ее текущего состояния (функция Submodel); это обеспечивает обход всей текущей иерархии модели. Веткой «Другие элементы» обозначена обработка других традиционных дочерних элементов состояния, таких как акция (act), dom-элемент (dom) и т. д.

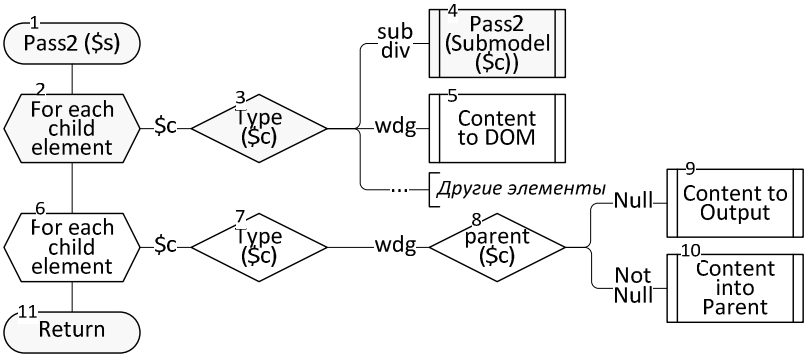


Рис. 5. Алгоритм второго прохода интерпретации, модифицированный с учетом иерархических виджетов

Если дочерний элемент является виджет-элементом (wdg), вызывается функция (блок 5) формирования локального результата виджета. Результат сохраняется в DOM-объекте.

Таким образом, по завершении первичной обработки дочерних элементов состояния (блок 2) для всех виджет-элементов (как дочерних, так и их потомков более глубоких уровней, соответствующих текущим внутренним состояниям) будут сформированы DOM-объекты, содержащие локальные результаты виджетов.

Вторичная обработка (блок 6). Для формирования глобальных результатов и вывода их в выходной поток выполняется повторная обработка дочерних виджет-элементов (блок 6, 7). Если обрабатываемый виджет-элемент является дочерним для другого виджет-элемента (непустой атрибут parent, блок 8), то содержимое DOM-объекта вставляется в содержимое родительского DOM-объекта (блок 9). Если виджет является корневым (пустой атрибут parent, блок 8), то сформированное содержимое DOM-объекта (глобальный результат) выводится в выходной поток (блок 10).

Алгоритм функции формирования локального результата виджета (блок 5) включает:

1) создание нового глобального DOM-объекта с именем, соответствующим имени виджета;

2) генерацию контента и загрузку его в созданный DOM-объект. Этот этап управляется атрибутом type виджет-элемента:

• "xml" или "html" – контент, представляющий локальный результат, загружается из документа ADM;

• "xslt" – контент формируется путем XSL-трансформации содержимого DOM-объекта.

Алгоритм вставки в родительский DOM-объект (блок 9) предусматривает следующие действия:

1) по значению атрибута parent отыскивается родительский DOM-объект, соответствующий родительскому виджет-элементу;

2) по значению атрибута corner отыскивается в родительском DOM-объекте родительский элемент для вставки контента;

3) содержимое DOM-объекта обрабатываемого виджет-элемента вставляется в качестве дочернего в найденный родительский элемент после уже имеющихся детей ("append") или перед ними ("prepend") в соответствии со значением атрибута insert.

Пример. На рис. 6 иллюстрируется последовательность формирование результирующего контента иерархического виджета. Динамическая модель содержит состояние sta:A, непосредственно в котором размещены виджет-элементы wdg:A_1 и wdg:A_2. Виджет A_1 формирует код div-блока, содержащего элемент управления в виде кнопки (Кнопка А), а виджет A_2 – код div-блока, содержащего текст (Текст А). Потомками состояния A являются состояния sta:B и sta:C, расположенные в субмоделях где-то на более низких уровнях иерархии и содержащие виджет-элементы wdg:B_1 и wdg:C_1 соответственно. Эти виджеты формируют div-блоки, содержащие кнопку и текстовый параграф (Кнопка В и Текст С). Рассмотрим второй проход интерпретации, предполагая, что указанные состояния являются текущими.

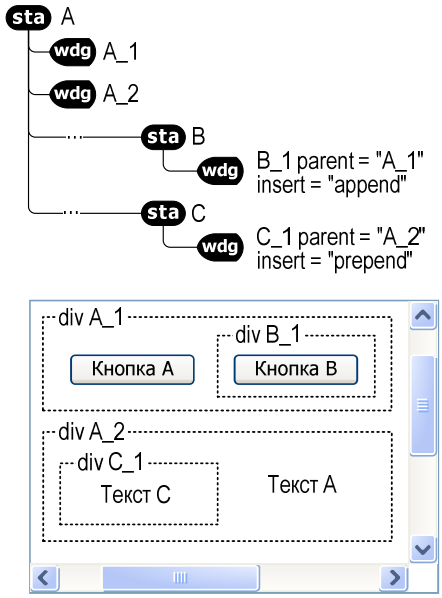


Рис. 6. Этапы формирования результирующего контента иерархического виджета.

В ходе первичной обработки вложенных элементов состояния sta:A интерпретатор сначала обработает виджет wdg:A_1 и создаст соответствующий DOM-объект, куда поместит div-контейнер A_1 с элементом управления Кнопка А. После этого аналогичным образом будет обработан виджет wdg:A_2 и создан DOM-объект с div-контейнером A_2, содержащим параграф Текст А. Далее, в ходе первичной обработки субмоделей состояния sta:A будет обработано состояние sta:B с вложенным виджетом wdg:B_1 и создан DOM-объект, содержащий div-контейнер B_1 с элементом управления Кнопка В. Аналогичным образом будет обработано состояние sta:C с вложенным виджетом wdg:C_1 и создан DOM-объект, содержащий div- контейнер C_1 с параграфом Текст С.

Вторичная обработка сначала выполняется для самых низких уровней иерархии. В ходе вторичной

обработки вложенных элементов состояния sta:B интерпретатор обработает виджет wdg:B_1 и в соответствии с атрибутом parent поместит div-контейнер B_1 из DOM-объекта B_1 в родительский DOM-объект A_1. При этом в соответствии с атрибутом insert размещаемый div-элемент будет вставлен после уже имеющегося контента (Кнопка B будет следовать за Кнопкой A). Аналогичным образом в ходе вторичной обработки вложенных элементов состояния sta:C интерпретатор обработает виджет wdg:C_1 и в соответствии с атрибутом parent поместит div-контейнер C_1 из DOM-объекта C_1 в родительский DOM-объект A_2. При этом в соответствии с атрибутом insert размещаемый div-элемент будет вставлен перед уже имеющимся контентом (Текст C будет першествовать Тексту A). Таким образом, к моменту вторичной обработки дочерних элементов состояния sta:A в DOM-объекты A_1 и A_2 будут вставлены результаты виджетов-потомков B_1 и C_1. В ходе вторичной обработки, не обнаружив атрибутов parent, интерпретатор выведет в выходной поток заполненный div-контейнер A_1, а в след за ним заполненный div-контейнер A_2.

СОДЕРЖАТЕЛЬНЫЙ ПРИМЕР

Для иллюстрации использования виджетов и тестирования алгоритма интерпретации виджет-элементов было рассмотрено интернет-приложение, предназначенное для отображения пользователю сведений о студентах, о предметах и о сдачах студентами предметов. Исходные данные примера базируются на работе [6] с целью продемонстрировать различия традиционного подхода и предлагаемого, основанного на использовании иерархических виджетов. В данном примере предполагается, что одни и те же исходные данные по желанию пользователя требуется предоставлять различными способами: 1) с ориентацией на студентов и на их успеваемость по предметам; 2) с ориентацией на предметы и сдачу этих предметов различными студентами. Для этого наряду с отображаемыми данными пользователю предоставляется совокупность элементов управления, позволяющих выбирать тот или иной режим отображения для того или иного объекта или группы объектов. Стиль такого отображения частично обсуждался выше (см. рис. 2, *а*).

Динамическая модель

Для реализации функциональности интернет-приложения используется динамическая модель, представленная на рис. 7. Модель имеет набор состояний, размещенных на трех уровнях:

• корневое состояние sta:Студенты-Предметы;

• два состояния 2-го уровня, размещенные в субмодели sub:Студенты-Предметы (sta:Студенты и sta: Предметы);

• четыре состояния 3-го уровня, размещенные в двух субмоделях: sub:Студенты и sub:Предметы (sta:СписокСтудентов, sta:ВыбранСтудент, sta:СписокПредметов, sta:ВыбранПредмет).

Данные о студентах, предметах и сдачах хранятся в виде XML-файлов stud.xml, predm.xml и sdacha.xml, размещенных в ADM СОБД. Методы трансформации данных указаны в соответствующих XSL файлах: stud.xsl, uspevstud.xsl, predm.xsl, uspevpredm.xsl. Объявлены 3 виджета: Главный и его дочерние виджеты – Панель и Детали. Они видны остальным виджетам из подсостояний модели. Виджет Панель отвечает за управление, а виджет Детали – за информационное наполнение.

Для задания html-разметки содержимого виджетов подключаются отдельные файлы main.html, panel.html и detail.html.

Корневое состояние. Состояние sta:Студенты-Предметы является корневым состоянием модели. В нем декларируются 3 виджета и выполняется погружение в субмодель следующего уровня иерархии.

Субмодель второго уровня. Субмодель sub:Студенты-Предметы задает два состояния, sta:Студенты и sta:Предметы, которые определяют текущий режим представления данных. В состоянии sta:Студенты данные ориентированы на студентов, а в состоянии sta:Предметы — на предметы. Переходы состояний обеспечиваются jmp-элементами, которые становятся активными в зависимости от выбранного варианта из выпадающего меню МенюФункций (факт нажатия кнопок интерпретатор определяет из массива POST, полученного от клиента вместе с URL). При этом происходит заполнение виджета Панель данными соответствующего типа: либо списком предметов, либо списком студентов. В зависимости от текущего состояния этой субмодели div-элементы обеспечивают погружение в субмодель *sub:Студенты* или *sub:Предметы*.

Субмодели третьего уровня. Эти субмодели задают состояния нижнего уровня иерархии: sta:СписокСтудентов и sta:ВыбранСтудент в субмодели sub:Студенты и sta:СписокПредметов и sta:ВыбранПредмет в субмодели sub:Предметы. Происходит наполнение виджета Детали соответствующими данными. Смена текущих состояний (навигация) в пределах субмоделей обеспечивается jmp-элементами, активизируемыми при выборе соответствующих опций меню.

Иерархия виджетов

Таким образом, динамическая модель веб-приложения содержит 13 виджет-элементов, образующих 3-уровневую иерархию. Изображение на экране строится путем комбинации локальных результатов, генерируемых виджетами, в зависимости от текущих состояний динамической модели.

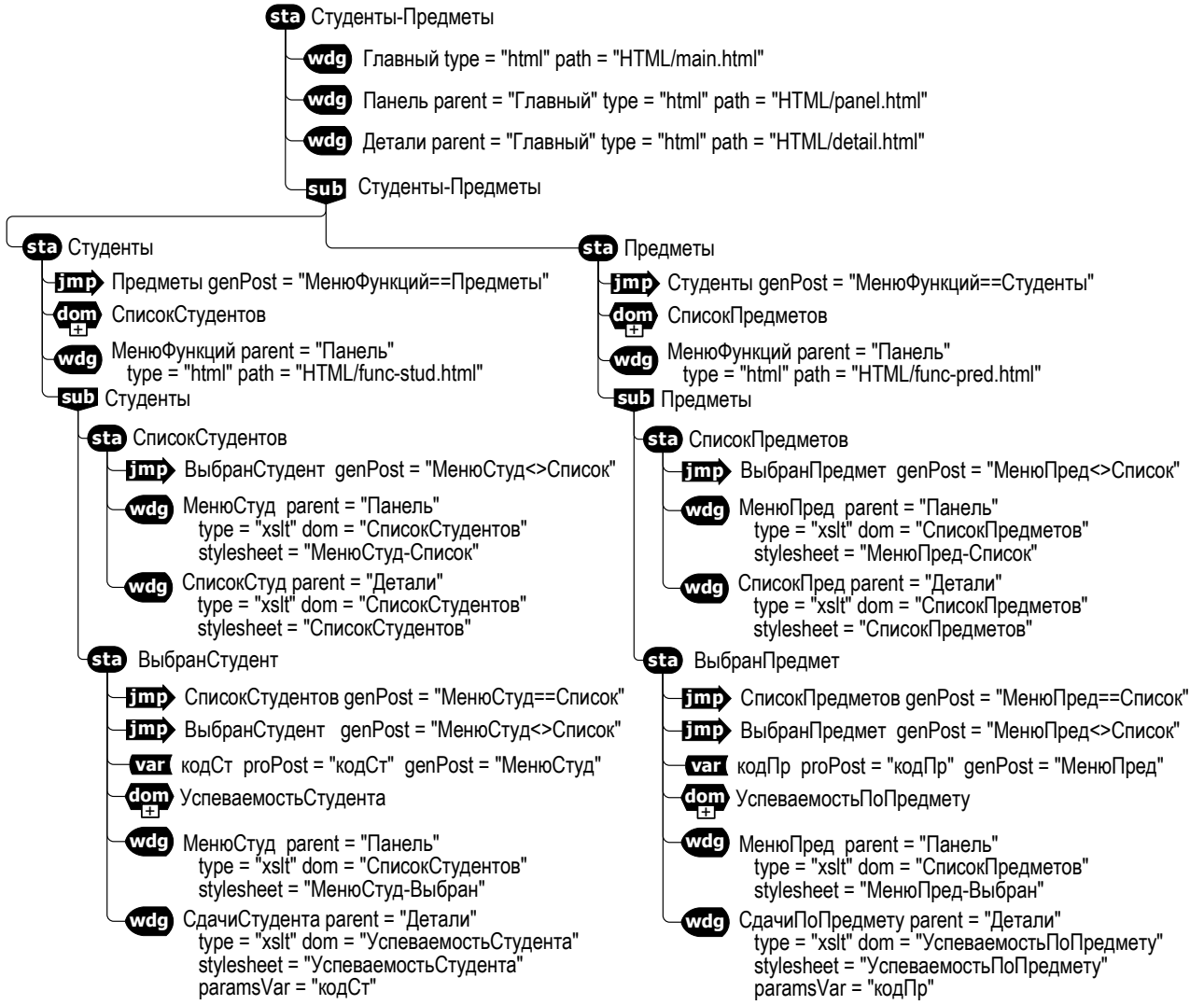


Рис. 7. Динамическая модель тестового интернет-приложения

На рис. 8 вложенность виджет-элементов, задаваемая атрибутами parent, иллюстрируется с помощью И-ИЛИ-дерева, в котором И-узлы указывают на совместно используемые дочерние виджеты, а ИЛИ-узлы – на альтернативно используемые (в зависимости от текущего состояния).

Виджет Главный является единственным корневым виджетом приложения, он генерирует локальный результат в виде div-контейнера для изображения в целом. Внутри контейнера могут присутствовать HTML-элементы, содержащие сведения общего характера (заглавия). Эти сведения не зависят от состояния модели, соответствующий код хранится в виде HTML-документа и используется «как есть».

Виджеты Панель и Детали встраивают свой код-результат в код-результат виджета Главный. Виджет Панель генерирует div-контейнер для панели управления, а виджет Детали – div-контейнер для области детальных сведений. Как и в случае корневого виджета, генерируемый код не зависит от состояния модели и хранится в ADM виде HTML-документов.

Заметим, что, вообще говоря, можно обойтись без виджетов Панель и Детали, если порождаемый ими неизменный код предусмотреть в корневом виджете, а для виджетов следующего уровня в атрибутах insert явно указать точки вставки. Такой подход, скорее всего, будет более эффективным в плане производительности, в то время как используемый подход преследует иллюстративные цели.

В локальный результат виджета Панель в зависимости от текущего состояния модели встраиваются результаты одного из виджетов группы *1*, формирующих меню функций, и одного из виджетов группы *2*, формирующих меню студентов или предметов. Выбор виджетов группы *1* и *2* зависит от текущего состояния субмодели Студенты-Предметы. При этом виджеты группы *2*, в свою очередь, подразделяются на группы *3* и *4*, управляемые состояниями субмоделей Студенты и Предметы соответственно.

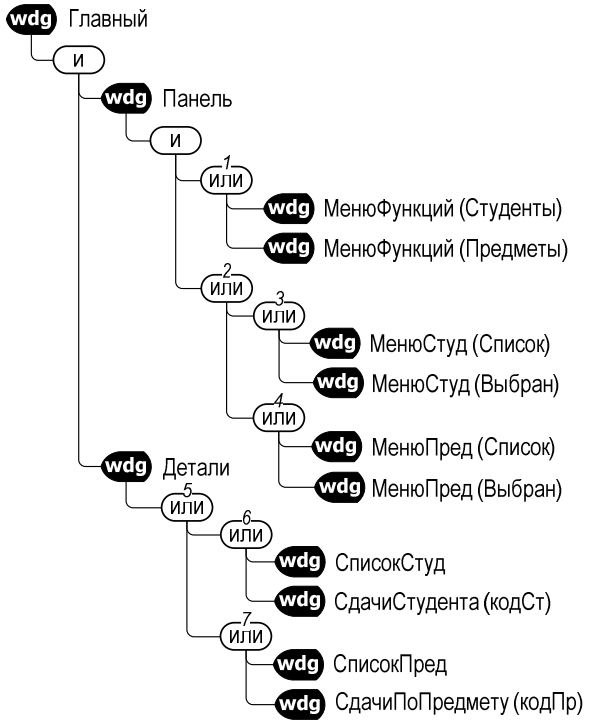


Рис. 8. И-ИЛИ-дерево, иллюстрирующее вложенность виджет-элементов

Меню, генерируемые виджетами МенюФункций, имеют одинаковый, фиксированный состав опций («студенты» и «предметы»), поэтому они заданы с помощью HTML-документов, хранящихся в ADM. Хотя состав опций одинаковый, необходимо два документа, поскольку меню имеют разные текущие опции.

Меню, генерируемые виджетами МенюСтуд и МенюПред, имеют состав опций, определяемый списком студентов и списком предметов. Они формируются путем xsl-трансформации соответствующих XML-документов, предварительно загруженных из ADM в DOM-объекты (dom-элементы СписокСтудентов и СписокПредметов, детали загрузки скрыты, чтобы не загромождать рисунок). Таблицы стилей трансформации также размещены в ADM.

В результат виджета Детали в зависимости от текущего состояния субмодели Студенты-Предметы встраиваются результаты одного из виджетов группы *6*, формирующих список студентов или сдачи выбранного студента в зависимости от текущего состояния субмодели Студенты, или одного из виджетов группы *7*, формирующих список предметов или сдач по выбранному предмету в зависимости от текущего состояния субмодели Предметы.

Списки студентов и предметов, генерируемые виджетами СписокСтуд и СписокПред, формируются путем xsl-трансформации соответствующих XML-документов, предварительно загруженных из ADM в DOM-объекты (dom-элементы СписокСтудентов и СписокПредметов). Таблицы стилей трансформации размещены в ADM.

Список сдач выбранного студента и список сдач по выбранному предмету, генерируемые виджетами СдачиСтудента и СдачиПоПредмету, также формируются путем xsl-трансформации соответствующих XML-документов, предварительно загруженных из ADM в DOM-объекты (dom-элементы УспеваемостьСтудента и УспеваемостьПоПредмету). Таблицы стилей трансформации размещены в ADM. В таблицы стилей передаются параметры, обеспечивающие селекцию сдач, относящихся к выбранному студенту или к выбранному предмету.

ЗАКЛЮЧЕНИЕ

В данной работе на концептуальном уровне (без учета особенностей и возможностей среды реализации) предложен и исследован подход к созданию сложно структурированных пользовательских интерфейсов для веб-приложений на основе СОБД.

Предложенная концепция иерархических виджетов состоит в том, что в состояниях динамической модели предусматриваются виджет-элементы, соответствующие фрагментам изображения на экране пользователя и задающие способ и параметры формирования результирующего HTML-кода, а также ссылки на родительские виджеты, объединяющие виджет-элементы в иерархию. В процессе интерпретации в зависимости от текущего состояния динамической модели автоматически формируется в буфере результирующий контент иерархии виджет-элементов, который по завершении выводится в выходной информационный поток.

Реализация концепции предусматривает, что алгоритм интерпретации при обработке текущего состояния на втором рекурсивном обходе динамической модели дважды обрабатывает виджет-элементы: 1) первичная обработка, в ходе которой формируется локальный контент виджетов, 2) вторичная обработка, в ходе которой из локальных контентов собирается глобальный контент для вывода в выходной поток.

Как ожидается, реализация концепции даст разработчикам СОБД гибкий инструмент, облегчающий за счет декларативного описания проектирование веб-приложений со структурой пользовательского интерфейса, зависящей от текущих состояний динамической модели.

Дальнейшие усилия в данном направлении предполагается направить на алгоритмическую и программную реализацию предложенных концептуальных решений в виде соответствующих дополнений к XML-структуре динамической модели и к программному обеспечению интерпретатора.

СПИСОК ЛИТЕРАТУРЫ

1. **Model-driven_engineering** [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Model-driven_engineering (дата обращения 01.10.2012).

2. **Миронов В. В., Юсупова Н. И., Шакирова Г. Р.** Ситуационно-ориентированные базы данных: концепция, архитектура, XML-реализация // Вестник УГАТУ: науч. журн. Уфимск. гос. авиац. техн. ун-та. 2011. Т. 14, № 2 (37). С. 233–244.

3. **Миронов В. В., Маликова К. Э.** Интернет-приложения на основе встроенных динамических моделей: идея, концепция, безопасность // Там же. 2009. Т. 13, № 2 (35). С. 167–179.

4. **Миронов В. В., Маликова К. Э.** Интернет-приложения на основе встроенных динамических моделей: архитектура, структура данных, интерпретация // Там же. 2010. Т. 14, № 1 (36). С. 154–163.

5. **Миронов В. В., Маликова К. Э.** Интернет-приложения на основе встроенных динамических моделей: элементы управления пользовательского интерфейса // Там же. 2010. Т. 14, № 5 (40). С. 170–175.

6. **Миронов В. В., Гусаренко А. С.** Ситуационно-ориентированные базы данных: концепция управления xml-данными на основе динамических dom-объектов // Там же. 2012. Т. 16, № 3 (48). С. 159–172.

7. **Миронов В. В., Гусаренко А. С.** Динамические dom-объекты в ситуационно-ориентированных базах данных: лингвистическое и алгоритмическое обеспечение источников данных // Там же. 2012. Т. 16, № 6 (51). С. 167–176.

ОБ АВТОРАХ

Миронов Валерий Викторович, проф. каф. АСУ. Дипл. радиофизик (Воронежск. гос. ун-т, 1975). Д-р техн. наук по упр. в техн. системах (УГАТУ, 1995). Иссл. в обл. иерархических моделей и ситуационного управления.

Канашин Виталий Владленович, асп. той же каф. Дипл. инж. по АСУ (УГАТУ, 2011). Готовит дис. об иерархических виджетах в ситуационно-ориентированных базах данных.

METADATA

Title: Hierarchical widgets: user interface organization in web applications on the basis of situation-oriented databases.

Authors: V. V. Kanashin and V. V. Mironov

Affiliation: Ufa State Aviation Technical University (UGATU), Russia.

Email: mironov@list.ru.

Language: Russian.

Source: Vestnik UGATU (Scientific journal of Ufa State Aviation Technical University), vol. 17, no. 3 (55), pp. 138-149, 2013. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract: Complex structured user interface in the Internet applications functioning on the basis of the situation-oriented databases (SOBD) is considered. The concept of hierarchically organized interface objects – the widgets – associated with states of the SOBD dynamic model and generating content in the course of dynamic model interpretation is offered. The structure, structure and functionality of hierarchical widgets are discussed. The substantial example of hierarchical widgets organization and functioning is presented.

Key words: Web application; user interface; situation-oriented database; dynamic model; XML; XSLT; model-driven development.

References (English Transliteration):

1. (2012, October 01) *Model-driven_engineering* [Online]. Avaliable: http://en.wikipedia.org/wiki/Model-driven_engineering
2. V. V. Mironov, N. I. Yusupova, G. R. Shakirova, “Situation-oriented databases: concept, architecture, XML realization”, (in Russian), *Vestnik UGATU* (scientific journal of Ufa State Aviation Technical University), vol. 14, no. 2 (37), pp. 233-244, 2011.
3. V. V. Mironov, K. E. Malikova, “Internet applications based on embedded dynamic models: idea, concept”, (in Russian), *Vestnik UGATU*, vol. 13, no. 2 (35), pp. 167-179, 2009.
4. V. V. Mironov, K. E. Malikova, “Internet applications based on embedded dynamic models: architecture, data structure, interpretation”, (in Russian), *Vestnik UGATU*, vol. 14, no. 1 (36), pp. 154-163, 2010.
5. V. V. Mironov, K. E. Malikova, “Internet applications based on embedded dynamic models: user interface controls”, (in Russian), *Vestnik UGATU*, vol. 14, no. 5 (40), pp. 170-175, 2010.
6. V. V. Mironov, A. S. Gusarenko, “Situation-oriented databases: concept of XML data management based of dynamic DOM objects”, (in Russian), *Vestnik UGATU*, vol. 16, no. 3 (48), pp. 159-172, 2012.
7. V. V. Mironov, A. S. Gusarenko, “Dynamic DOM objects in situation-oriented databases: lingware and knoware of data sources”, (in Russian), *Vestnik UGATU*, vol. 16, no. 6 (51), pp. 159-172, 2012.

About authors:

1. Kanashin, Vitaliy Vladlenovich, Postgrad. (PhD) Student, Automated Systems Dept.. Dipl. Ing. (UGATU, 2011).
2. Mironov, Valeriy Viktorovich, Prof., Automated Systems Dept. Dipl. Radiophysicist (Voronezh State Univ., 1975). Dr. (Habil.) Tech. Sci. (UGATU, 1995).