Вестник УГАТУ
Уфа : УГАТУ, 2012 Т. 16, № 3 (48). С. 159–172

УДК 004.65

В. В. Миронов, А. С. Гусаренко

СИТУАЦИОННО-ОРИЕНТИРОВАННЫЕ БАЗЫ ДАННЫХ: КОНЦЕПЦИЯ УПРАВЛЕНИЯ XML-ДАННЫМИ НА ОСНОВЕ ДИНАМИЧЕСКИХ DOM-ОБЪЕКТОВ

Рассматривается управление XML-данными, ассоциированными с состояниями динамической модели в ситуационно-ориентированных базах данных. Предлагается концепция динамических DOM-объектов, которые автоматически порождаются, загружаются, уничтожаются в процессе смены текущих состояний динамической модели. Предлагаются средства задания динамических DOM-объектов с помощью DOM-элементов, ассоциированных с состояниями динамической модели. Обсуждаются спецификации загружаемого и выгружаемого XML-содержимого DOM-объектов с помощью специальных элементов — источников и приемников данных, задаваемых в динамической модели. Разбирается содержательный пример ситуационно-ориентированной базы данных для веб-приложения. *Динамическая модель*; *интерпретация*; *ситуационно-ориентированные базы данных*; *XML*; *DOM*; *XSLT*

В настоящее время в области информационных технологий активно развивается подход к разработке информационных систем на основе непосредственного использования моделей высокого уровня абстракции (Model-based, model-driven engineering) [1–3]. В русле этого подхода может рассматриваться и идея ситуационно-ориентированных баз данных, в которых данные в формате XML ассоциированы с состояниями встроенной динамической модели и могут обрабатываться в контексте ее текущих состояний [4, 5]. Ситуационно-ориентированные базы данных могут служить основой интернет-приложений, обеспечивающих динамическое формирование контента в соответствии со сложившейся ситуацией [6–8]. В этом плане важными представляются вопросы организация обработки XML-данных, ассоциированных с текущими состояниями динамической модели, обсуждаемые в данной статье.

1. ВВОДНЫЕ ЗАМЕЧАНИЯ

В ситуационно-ориентированнную базу данных входят следующие компоненты (рис. 2.1):

- 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 Memory) – 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.

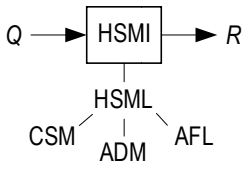


Рис. 2.1. Архитектура ситуационно-ориентированной базы данных

Так, если ситуационно-ориентированная база данных 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, отправляе-

Контактная информация: (347) 272-89-81.

Статья выполнена в рамках исследований, поддержанных грантом РФФИ 10-07-00167-а.

мый клиенту. Параметры запроса задают обрабатываемую динамическую модель и необходимость смены ее текущих состояний.

На рис. 2.2 приведен пример динамической модели HSM. Корневое состояние sta:S0, символизирующее модель в целом, содержит в качестве дочерних три элемента: два определения XML-документов из ADM и одну субмодель из HSML. Первый документ, doc:X1, — это файл «X1.xml», а второй, doc:X2, — файл «X2.xml», оба из папки «XML» ADM. Субмодель sub:M задает для родительского состояния sta:S0 два подсостояния: sta:S1 и sta:S2. Дочерние элементы состояний sta:S1 и sta:S2 содержат, во-первых, элементы-переходы jmp:S2 и jmp:S1, обеспечивающие смену текущего состояния, во-вторых, элементы-акции act:A1 и act:A2, предусмотренные для выполнения определенных действий в соответствующих состояниях.

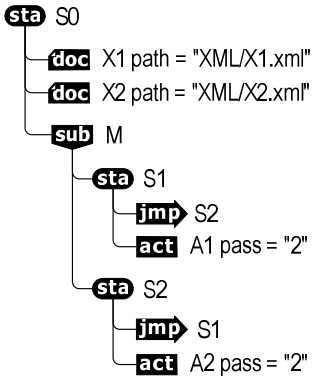


Рис. 2.2. Диаграмма динамической модели с определениями ассоциированных XML-документов

В процессе интерпретации элементов-акций выполняются ассоциированные с ними функции из AFM, которые могут предусматривать обработку XML-документов из ADM. Например, может быть предусмотрено создание DOM-объектов, загрузка в них XML-документов, XSL-трансформация содержимого, вывод результата трансформации в качестве результата запроса. Обработка ассоциированных данных, таким образом, программируется в ассоциированных функциях из AFM, находящихся за пределами HSM, что затрудняет понимание логики обработки данных. Спецификации обработки XML-документов могут быть распределены по различным состояниям HSM. Например, в некотором состоянии задается загрузка XML-документа; его модификация специфицируется в одном из подсостояний указанного состояния, сохранение изменений – в другом, а формирование выходного результата – в третьем. Чтобы понять логику этой обработки, необходимо анализировать совместно несколько функций, ассоциированных с несколькими состояниями. В этом плане было бы полезным, если бы логику обработки XML-документов из ADM можно было бы явно задавать в HSM.

DOM-объекты. DOM (Document Object Model) [9] – это платформо-независимый объектно-ориентированный интерфейс для доступа к документам XML, XHTML, HTML, получивший в настоящее время широкое распространение. Технология обработки XML-данных на основе DOM предусматривает создание DOM-объекта, загрузку в него целиком внешнего XML-документа (или создание нового документа «с нуля» программным путем), манипулирование документом как деревом, сохранение результирующего документа во внешней среде[1].

С учетом возможностей, гибкости, распространенности DOM, представляется целесообразным решать рассматриваемую задачу, в первую очередь, на этой технологии.

Решаемая задача. Таким образом, целесообразно предоставить разработчикам HSM возможность специфицировать DOM-объекты, ассоциированные с текущими состояниями модели. В ходе интерпретации HSM в соответствии с этими спецификациями интерпретатор должен автоматически создавать DOM-объекты, загружать в них XML-данные из ADM, делать их доступными для обработки из других элементов модели, соответствующих текущим состояниям, сохранять измененные XML-данные в ADM, удалять объекты, когда они больше не требуются. При этом следует предоставить возможность специфицировать на уровне HSM преобразование XML-данных, содержащихся в DOM-объектах. Например, разработчик может указать в модели, что в некотором состоянии XML-данные, загруженные в некоторый DOM-объект, необходимо подвергнуть XSL-трансформации в соответствии с определенной XSLT-таблицей стилей. Объекты с подобным поведением будем называть *динамическими DOM-объектами*.

[1] Альтернативой DOM является модель SAX (Simple API for XML) [10] и ее развитие XMLReader / XMLWriter, предполагающие однонаправленную обработку XML-данных «на лету». В ущерб гибкости и функциональности эти технологии позволяют обрабатывать очень большие XML-документы, не загружая их целиком в оперативную память.

2. КОНЦЕПЦИЯ ДИНАМИЧЕСКИХ DOM-ОБЪЕКТОВ

DOM-элементы. Итак, в ходе интерпретации динамической модели могут автоматически создаваться DOM-объекты, содержащие XML-данные, ассоциированные с текущими состояниями модели и доступные для обработки из других элементов модели, ассоциированных с текущими состояниями. Для этого предусмотрим в модели специальные DOM-элементы с заданными для них источниками XML-данных. Соответственно в процессе интерпретации модели предусмотрим обработку указанных DOM-элементов: создание соответствующих им DOM-объектов и загрузку в них XML-данных из различных источников, когда родительские состояния становятся текущими, удаление DOM-объектов, когда родительские состояния перестают быть текущими. Это позволит обрабатывать XML-данные, ассоциированные с текущими состояниями, обращаясь к DOM-объектам из процедур, заданных в элементах-акциях текущего состояния. При этом должна уменьшиться трудоемкость программирования, поскольку рутина подготовки и загрузки DOM-объектов возлагается на интерпретатор.

Реализация этой идеи требует концептуальной проработки вопросов о том, как будут создаваться DOM-объекты и загружаются XML-данные, о доступности созданных DOM-объектов для обработки в различных состояниях, об удалении DOM-объектов и сохранении в ADM их XML-содержимого.

2.4. Динамическое порождение DOM-объектов

Источники XML-данных. Может быть предложено несколько вариантов наполнения данными DOM-объектов, создаваемых в ходе интерпретации DOM-элементов: из внешнего XML-файла; из другого DOM-объекта модели; как результат некоторой функции и т. д. На рис. 2.3 показано, как различные варианты источников данных представляются на диаграмме модели. DOM-элемент представляется символом dom, справа от него, как обычно, указывается имя элемента (в данном случае, DOM-элементы dom:D1, dom:D2, dom:D3). Символ источника данных src, прикрепленный к DOM-элементу в качестве дочернего элемента, специфицирует особенности загрузки XML-данных в DOM-объект, порождаемый DOM-элементом. Атрибуты элемента-источника, в свою очередь, специфицируют, откуда берутся XML-данные.

Загрузка из внешнего XML-файла. На рис. 2.3, *а* приведен фрагмент диаграммы модели, на котором для DOM-элемента специфицирован источник данных в виде XML-документа из ADM. Атрибут type = "doc" указывает на то, что это XML-документ из ADM, а имя источника данных «X1» — это ссылка на определение doc:X1. Атрибут method = "copy" говорит о том, что содержимое документа doc:X1 будет целиком копироваться в созданный DOM-объект. При обработке dom:D1 будет создан DOM-объект «D1», а при обработке src:X1 в соответствии с doc:X1 в него будет загружен файл X1.xml.

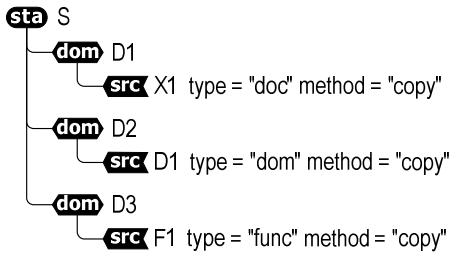


Рис. 2.3. Задание источника XML-данных как XML-документа из ADM (*а*), как другого DOM-элемента (*б*), как результата функции (*в*)

Загрузка из другого DOM-объекта. На рис. 2.3, *б* приведен фрагмент диаграммы модели, на котором для DOM-элемента специфицирован источник данных в виде ссылки на другой DOM-элемент. На это указывает атрибут type = "dom", т. е. имя источника данных «D1» — это ссылка на определение dom:D1. Предполагается, что к моменту обработки элемента src:D1 DOM-элемент dom:D1 уже обработан интерпретатором, т. е. создан и загружен DOM-объект «D1». Интерпретатор обращается к DOM-объекту «D1» и копирует его XML-содержимое в новый DOM-объект «D2».

Загрузка из функции. На рис. 2.3, *в* приведен фрагмент диаграммы модели, на котором для DOM-элемента специфицирован источник данных в виде функции, возвращающей в качестве результата строку XML-данных. На это указывает атрибут type = "func", т. е. имя источника данных «F1» — это ссылка на функцию из AFL. В процессе обработки интерпретатор вызывает указанную функцию и загружает результат ее выполнения —XML-строку — в созданный DOM-объект.

Загрузка, предусматривающая фильтрацию XML-данных. В реальных условиях может потребоваться не просто скопировать XML-данные из источника данных, а выполнить при этом то или иное преобразование данных. Так, часто требуется отфильтровать данные, т. е. загрузить в DOM-объект только ту часть данных, которая удовлетворяет некоторым условиям. На рис. 2.4 иллюстрируется фильтрация для источника данных в виде XML-файла из ADM.

В динамической модели на рис. 2.4, *а* в корневом состоянии sta:S заданы определения XML-документа doc:X1, хранящегося в ADM, и DOM-элемента dom:D3, в котором предусмотрена загрузка документа doc:X1 с фильтрацией. Источник данных src:X1 имеет атрибут method = "cut", указывающий на то, что из исходного XML-документа будет «вырезано» некоторое поддерево. Дополнительные атрибуты «element», «field» и «value», задают условие фильтрации. Атрибут «element» содержит XPath-выражение, которое определяет множество узлов-элементов в XML-дереве, один из которых будет корнем загружаемого поддерева. Атрибут «field» содержит XPath-выражение, которое задает в поддереве проверяемый XML-элемент или атрибут. Атрибут «value» содержит требуемое значение. В данном примере требуется загрузить в DOM-объект поддерево, начинающееся в элементе «e1», атрибута «k1» которого имеет значение «123».На рис. 2.4, *б* представлена модель XML-данных doc:X1 в графической нотации [11]. Корневой XML-элемент «E0» может содержать несколько дочерних XML-элементов «e1», каждый из которых содержит атрибуты «k1» (идентификатор) и «a1». Таким образом, XPath-выражение «/E0/e1», заданное в атрибуте «element» источника src:X1 на рис. 2.4, *а*, адресует множество всех XML-элементов «e1»; атрибут «field» адресует в «e1» XML-атрибут «k1», а атрибут «value» задает для него искомое значение «123». В ходе обработки элемента dom:D3 интерпретатор обращается к источнику src:X1, перебирает XML-элементы «e1», отыскивая единственный тот, у которого XML-атрибут «k1» имеет искомое значение «123», и загружает соответствующее поддерево в DOM-объект. В результате в DOM-объект «D3» будут загружены XML-данные, соответствующие модели, представленной на рис. 2.4, *в*.

Загрузка, предусматривающая трансформацию XML-данных

В более общем случае преобразование XML-данных при загрузке можно рассматривать как XSL-трансформацию XML-документа источника данных и для решения задачи применить технологию XSLT (рис. 2.5). Элемент источника данных в этом случае содержит атрибут method = "xslt", сообщающую интерпретатору о необходимости XSL-трансформации данных, а также атрибут «stylesheet», ссылающийся на используемую для этого стилевую таблицу, и атрибут «params», содержащий список значений глобальных параметров для передачи стилевой таблице. В процессе обработки элемента источника данных интерпретатор определяет необходимость XSL-трансформации, загружает стилевую таблицу, передает ей глобальные параметры, выполняет трансформацию XML-данных источника и загружает результат в созданный DOM-объект.

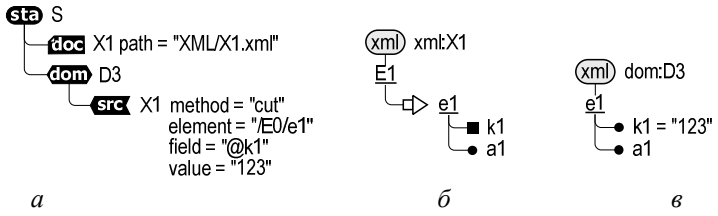


Рис. 2.4. Пример загрузки DOM-объекта с фильтрацией XML-данных: *а* – фрагмент динамической модели; *б* – модель XML-данных источника; *в* – модель XML-данных, загружаемых в DOM-объект

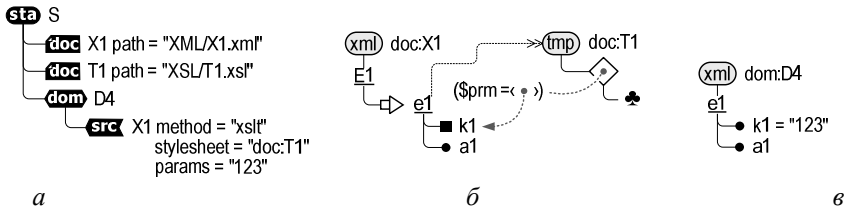


Рис. 2.5. Пример загрузки DOM-объекта с XSL-трансформацией данных: *а* – фрагмент динамической модели данных; *б* – модель XSL-трансформации; *в* – модель XML-данных, загружаемых в DOM-объект

В данном примере с помощью XSLT решается та же самая задача загрузки с фильтрацией, что и в предыдущем примере (см. рис. 2.4). Концептуальная модель XSL-трансформации XML-данных источника (рис. 2.5, *б*) представлена в графической нотации [11]. Трансформация предусматривает поиск в исходном документе элемента «e1», у которого атрибут «k1» равен значению глобального параметра «prm», и выдачу найденного элемента вместе с поддеревом в качестве результата трансформации.

Загрузка, предусматривающая слияние XML-данных

В более сложном случае может потребоваться загрузить в DOM-объект XML-данные, полученные на основе преобразования нескольких исходных XML-документов. Например, общие сведения, содержащиеся в одном документе, требуется дополнить детальными сведениями из другого документа. На рис. 2.6 иллюстрируется источник, предусматривающий слияние XML-документов. Модель первого из сливаемых документов, содержащего общие сведения и выступающего в качестве родительского, приведена на рис. 2.6, *а* (ср. рис. 2.5). Модель второго из сливаемых документов, содержащего детальные сведения и выступающего в качестве дочернего, приведена на рис. 2.6, *б*. Модель результирующего документа, загружаемого в DOM-объект, приведена на рис. 2.6, *в*. Соответствующий фрагмент динамической модели, предусматривающий слияние двух документов при загрузке DOM-объекта «D3», приведен на рис. 2.6, *г*. Элемент-источник имеет атрибут method = "merge", сообщающий интерпретатору, что загружаемые в DOM-объект XML-данные получаются путем слияния двух XML-документов. Эти документы задаются атрибутами perentDoc (родительский документ) и childDoc (дочерний документ). Остальные атрибуты задают XPath-выражения, уточняющие особенности слияния документов:

- parentElement указывает в родительском документе множество XML-элементов, к которым будут прикрепляться поддеревья, взятые из дочернего документа;
- parentField указывает для каждого элемента из предыдущего множества (parentElement) значение, используемое для идентификации прикрепляемого поддерева;
- childElement указывает в дочернем документе множество элементов, которые вместе со своими поддеревьями используются для копирования в родительский документ;
- childField указывает для каждого элемента из предыдущего множества (childElement) значение, используемое для идентификации прикрепляемого поддерева.

В ходе обработки элемента-источника интерпретатор обращается к родительскому документу и для каждого его XML-элемента типа parentElement отыскивает в дочернем документе соответствующий ему XML-элемент типа childElement такой, что значения parentField и childField совпадают. Соответствующий найденный элемент вместе со своим поддеревом копируется в родительский документ в качестве дочернего для обрабатываемого XML-элемента parentElement.

2.4. Динамическое удаление DOM-объектов

Принцип, положенный в основу динамического удаления DOM-объектов, состоит в том, DOM-объект, автоматически созданный при обработке DOM-элемента некоторого текущего состояния динамической модели, должен быть автоматически удален, когда это состояние перестает быть текущим.

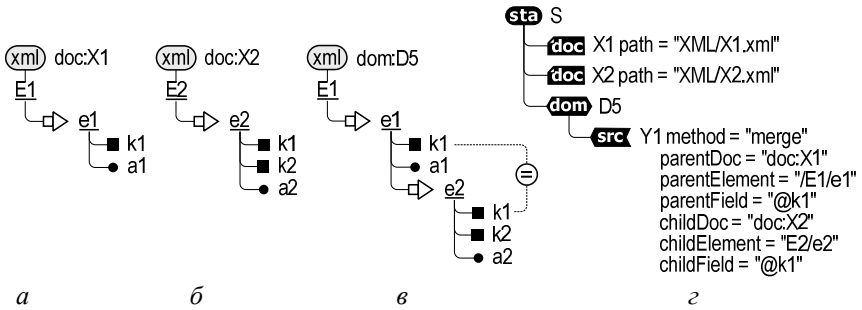


Рис. 2.6. Пример источника со слиянием XML-данных: *а* – модель общих сведений; *б* – модель детальных сведений; *в* – модель результирующих данных, загружаемых в DOM-объект; *г* – фрагмент динамической модели, предусматривающей слияние документов

Данное требование может быть достаточно просто реализовано на основе эпилоговой обработки состояний, предусмотренной общим алгоритмом интерпретации. Эпилоговая обработка выполняется интерпретатором каждый раз перед сменой текущего состояния и заключается в том, что интерпретатор рекурсивно обрабатывает поддерево состояния, переставшего быть текущим, с особым признаком (флагом) эпилогового режима. Используя этот признак, дочерние элементы состояния могут предусматривать в ходе эпилоговой обработки те или иные действия, которые необходимо выполнить перед сменой текущего состояния. Таким образом, обрабатывая DOM-элементы в эпилоговом режиме, интерпретатор должен автоматически удалять соответствующие DOM-объекты.

Сохранение XML-данных DOM-объектов

XML-содержимое DOM-объектов может быть изменено в процессе интерпретации (например, в соответствии с запросом пользователя), и тогда может потребоваться сохранить их в ADM. Следовательно, необходимо предусмотреть возможность задания в HSM процедур сохранения содержимого DOM-объектов.

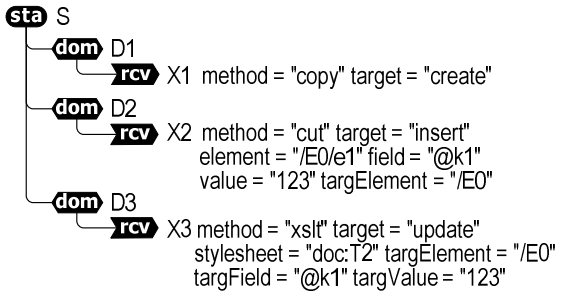


Рис. 2.7. Варианты приемников данных для сохранения содержимого DOM-объектов

Элемент-приемник. Поскольку процедуры сохранения являются обратными по отношению к процедурам загрузки DOM-объекта, введем элемент модели «приемник данных». Для изображения его на диаграмме модели будем использовать символ rcv (как для элемента-источника, но с противоположным направлением и меткой «rcv»). Элемент-приемник иллюстрируется на рис. 2.7, где с его помощью задается сохранение XML-данных DOM-объектов dom:D1, dom:D2 и dom:D3 в целевых XML-документах doc:X1, doc:X2, doc:X3 в ADM. Атрибут «method», как и в случае источника данных, указывает интерпретатору, какие данные DOM-объекта будут сохранены: method = "copy" означает, что необходимо сохранить все XML-содержимое целиком; method = "cut" — что необходимо сохранить XML-поддерево DOM-объекта; method = "xslt" — что необходимо сохранить результат XSL-преобразования содержимого DOM-объекта.

Атрибут «target» задает способ совмещения сохраняемых данных с уже имеющимися в целевом документе: target = "create" означает, что необходимо заново создать XML-содержимое целевого документа (если такой файл уже существует, он заменяется созданным файлом); target = "insert" — что необходимо вставить сохраняемые XML-данные в дерево целевого документа (задается путь к родительскому элементу целевого документа); target = "update" — что необходимо заменить сохраняемыми XML-данными некоторое поддерево целевого документа (задается путь к этому поддереву).

2.3. Видимость DOM-объектов в модели

Рассмотрим вопрос о том, из каких элементов динамической модели «видны» (т. е. доступны для обработки) DOM-объекты, порожденные DOM-элементами. Ответ зависит, во-первых, от номера прохода интерпретации, во-вторых, от взаимного расположения в модели DOM-элемента и данного элемента.

Влияние прохода интерпретации. В этой работе мы ориентируемся на общий случай многопроходной интерпретации динамической модели, когда в рамках цикла интерпретации интерпретатор несколько раз обрабатывает динамическую модель путем рекурсивного обхода текущего дерева модели. Если на первом (основном) проходе интерпретатор выполняет смену текущих состояний и создает DOM-объекты, ассоциированные с текущими состояниями, то на последующих (дополнительных) проходах — лишь обрабатывает элементы зафиксированных текущих состояний. Следовательно, взаимное расположение элементов влияет на видимость только на основном проходе, а на дополнительных проходах все DOM-объекты, созданные на первом проходе, доступны («видимы») из любого элемента текущего состояния.

Влияние взаимного расположения элементов. На основном проходе интерпретатор создает DOM-объекты при обработке соответствующих DOM-элементов. Эти DOM-объекты видны из тех элементов модели, которые обрабатываются интерпретатором после их создания, и,

соответственно, невидимы из тех, которые обрабатываются до их создания. Следовательно, видимость определяется порядком обработки элементов модели в ходе интерпретации. В динамической модели множество элементов, ассоциированных с некоторым состоянием, упорядочено; при интерпретации дерева текущих состояний модели такие элементы обрабатываются в порядке их следования. Рассмотрим некоторый элемент динамической модели, для которого существуют другие элементы-братья, ассоциированные с тем же состоянием, предшествующие ему («старшие братья») и следующие за ним («младшие братья»). Соответственно, старшим братьям не видны, а младшим — видны объекты, создаваемые рассматриваемым элементом.

На рис. 2.8 приведен фрагмент динамической модели, содержащий иерархию из трех состояний, sta:S0, sta:S1 и sta:S2, с которыми ассоциированы три DOM-элемента, dom:D0, dom:D1 и dom:D2, и шесть акций, act:A0, …, act:A6. Акция act:A0 обрабатывается на втором проходе цикла интерпретации, остальные — на первом. Из акции act:A0, обрабатываемой на втором проходе, видны все DOM-объекты, созданные на первом проходе: dom:D0, а также dom:D1, если текущее состояние sta:S1, или dom:D2, если текущее состояние sta:S2. Из остальных акций, обрабатываемых на первом проходе, видны лишь те DOM-объекты, которые созданы до их обработки. Так, объект dom:D0 виден из акций act:A2, …, act:A6; dom:D1 — из акции act:A4; dom:D2 — из акции act:A6.

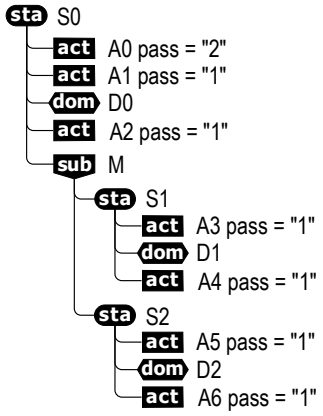


Рис. 2.8. К иллюстрации областей видимости DOM-объектов

2.4. Обработка содержимого DOM-объектов

Итак, с помощью DOM-элементов, ассоциированных с состояниями динамической модели, во время основного (первого) прохода цикла интерпретации создаются DOM-объекты, содержащие XML-данные. Эти объекты, в конечном счете, предназначены для получения выходных данных, отправляемых во внешнюю (по отношению к ситуационно-ориентированной базе данных) среду. Такая результирующая обработка DOM-объектов, как правило, выполняется во время дополнительных проходов интерпретации.

На рис. 2.9 иллюстрируется использование DOM-объекта на втором проходе интерпретации для генерации фрагмента HTML-кода. Предполагается, что интерпретация модели выполняется на веб-сервере, а результат отправляется в браузер клиента по протоколу HTTP. Здесь элемент-приемник с именем rcv:Echo означает стандартный поток результирующих данных, а его атрибут pass = "2" — то, что элемент обрабатывается интерпретатором на втором проходе. Результирующий HTML-код формируется путем XSL-трансформации XML-содержимого DOM-объекта «D3». И хотя детали трансформации скрыты в тексте таблицы стилей doc:T3, общая картина вполне понятна из модели.

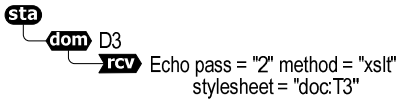


Рис. 2.9. Пример использования DOM-объекта для формирования HTML-кода

3. СОДЕРЖАТЕЛЬНЫЙ ПРИМЕР

Для иллюстрации использования динамических DOM-объектов рассмотрим интернет-приложение, которое предназначено для отображения пользователю сведений о студентах, предметах и сдачах студентами предметов.

Исходные данные

Данные о студентах, предметах и сдачах хранятся в виде XML-файлов stud.xml, predm.xml и sdacha.xml, размещенных в папке XML ассоциированных данных ситуационно-ориентированной базы. На рис. 2.10 представлены концептуальные модели этих документов и тестовые экземпляры их содержимого. Документ stud.xml (рис. 2.10, *а*) в корневом элементе «Студенты» может содержать несколько вложенных элементов «студент» с атрибутами

«код» (код студента) и «фио» (фамилия, имя, отчество студента), причем атрибут «код» является идентификатором. Экземпляр этого документа (рис. 2.10, *г*) содержит сведения о трех студентах: Иванове, Петрове и Сидорове. Аналогичным образом XML-документ predm.xml (рис. 2.10, *б*) в корневом элементе «Предметы» может содержать несколько вложенных элементов «предмет» с атрибутами «код» (код предмета) и «назв» (название предмета), причем атрибут «код» является идентификатором. Модель экземпляра этого документа (рис. 2.10, *д*) содержит сведения о трех предметах: «теории систем», «матлогике» и «криптографии».

Документ sdacha.xml (рис. 2.10, *а*) в корневом элементе «Сдачи» может содержать несколько вложенных элементов «сдача», соответствующих сдачам определенным студентом определенного предмета, с атрибутами «кодСт» (код студента), «кодПр» (код предмета) и «оценка» (оценка студента по предмету), причем пара атрибутов «кодСт, кодПр» является составным идентификатором. Модель экземпляра этого документа (рис. 2.10, *е*) содержит сведения о пяти сдачах: трех сдачах Иванова и двух Петрова (Сидоров не сдал ни одного предмета).

Интерфейс пользователя

В данном примере предполагается, что одни и те же исходные данные по желанию пользователя требуется представлять в веб-браузере двумя способами: с ориентацией на студентов и их успеваемость по предметам или с ориентацией на предметы и сдачу этих предметов студентами. На рис. 2.11 приведены 4 экранные формы, которые должны генерироваться интернет-приложением для выполнения указанных функций. Форма *а* отображает ситуацию «СписокСтудентов». Здесь кнопка «К предметам» обеспечивает переход к форме *б*, отображающей ситуацию «СписокПредметов». Радиокнопки, размещенные слева от фамилий студентов, предназначены для выбора ситуации (одного из студентов). Кнопка «Успеваемость студента» вызывает переход к форме *в* для отображения ситуации «ВыбранСтудент» успеваемости выбранного студента. Форма *б* аналогичным образом отображает ситуацию «ВыбранПредмет» список предметов. Здесь кнопка «К студентам» обеспечивает переход к форме *а*, отображающей ситуацию «СписокСтудентов». Радиокнопки, размещенные слева от названий предметов, предназначены для выбора одного из предметов. Кнопка «Успеваемость по предмету» вызывает переход к форме *г* для отображения успеваемости по выбранному предмету. Форма *в* отображает сведения об успеваемости выбранного студента в виде ненумерованного списка сданных студентом предметов с указанием полученных оценок. Кнопка «К списку студентов» предназначена для возврата к форме *а*, а кнопка «К предметам» — для перехода к форме *б*. Форма *г* аналогичным образом отображает сведения об успеваемости по выбранному предмету в виде ненумерованного списка студентов, сдавших данный предмет, с указанием полученных оценок.

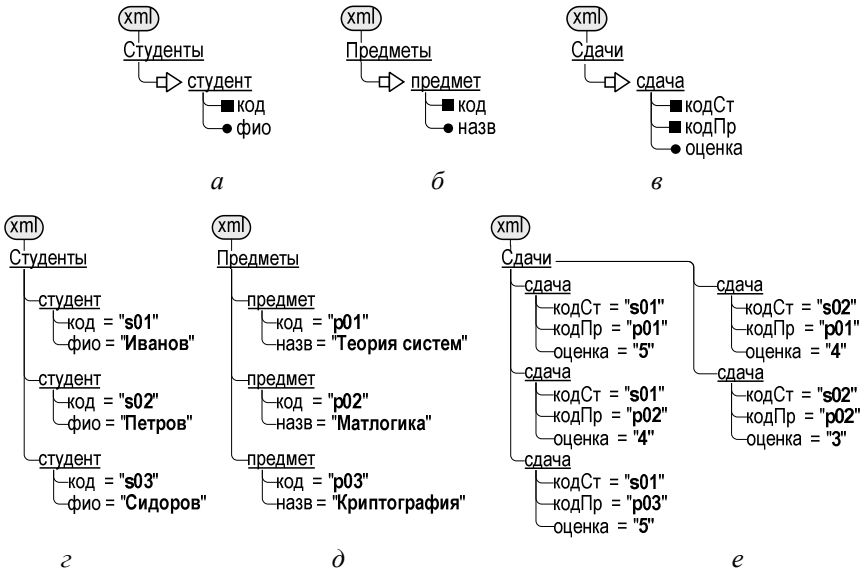


Рис. 2.10. Исходные XML-данные к примеру: *а–в* – концептуальные модели; *г–е* – модели тестовых экземпляров

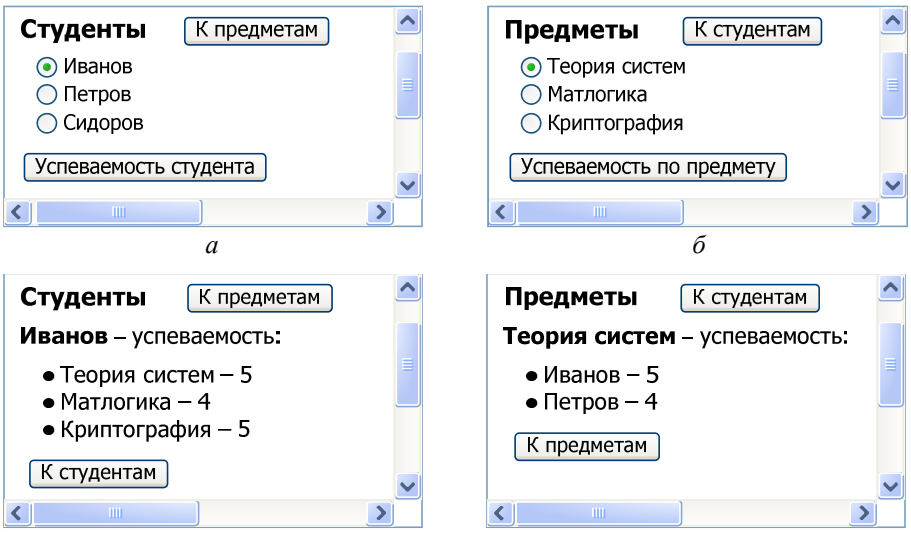


Рис. 2.11. Примеры экранных форм для представления сведений о студентах, предметах и сдачах: *а* – список студентов с возможностью выбора отдельного студента; *б* – список предметов с возможностью выбора отдельного предмета; *в* – список сдач выбранного студента; *г* – список сдач по выбранному предмету

Кнопка «К списку предметов» предназначена для возврата к форме *б*, а кнопка «К студентам» – для перехода к форме *а*. Требуется, чтобы в соответствующих ситуациях интернет-приложение отправляло в браузер клиента соответствующую форму, а также принимало и соответствующим образом обрабатывало воздействия пользователя на элементы управления.

Динамическая модель

Для реализации функций интернет-приложения используется динамическая модель, представленная на рис. 2.12. Модель имеет три уровня иерархии состояний: корневое состояние 1-го уровня sta:Студенты-Предметы; два состояния 2-го уровня, размещенные в субмодели sub:Студенты-Предметы; четыре состояния 3-го уровня, размещенные в двух субмоделях: sub:Студенты и sub:Предметы.

Корневое состояние. Состояние sta:Студенты-Предметы является корневым состоянием модели. В нем декларируются ассоциированные данные: три документа XML, четыре стиля трансформации XSL, библиотека макросов, после чего 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 при нажатии пользователем кнопок «К предметам» и «К студентам» соответственно (факт нажатия кнопок интерпретатор определяет из массива POST, полученного от клиента вместе с URL). 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. Смена текущих состояний (навигация) в пределах субмоделей обеспечивается jmp-элементами, активизируемыми при нажатии пользователем кнопок «Успеваемость студента», «Вернуться к студентам» и «Успеваемость по предмету», «Вернуться к предметам».

Источники данных

DOM-элементы. В элементах-состояниях субмоделей нижнего уровня предусмотрены внутренние dom-элементы, в которые загружаются сведения, необходимые в соответствующих ситуациях: в состоянии sta:СписокСтудентов – обо всех студентах;

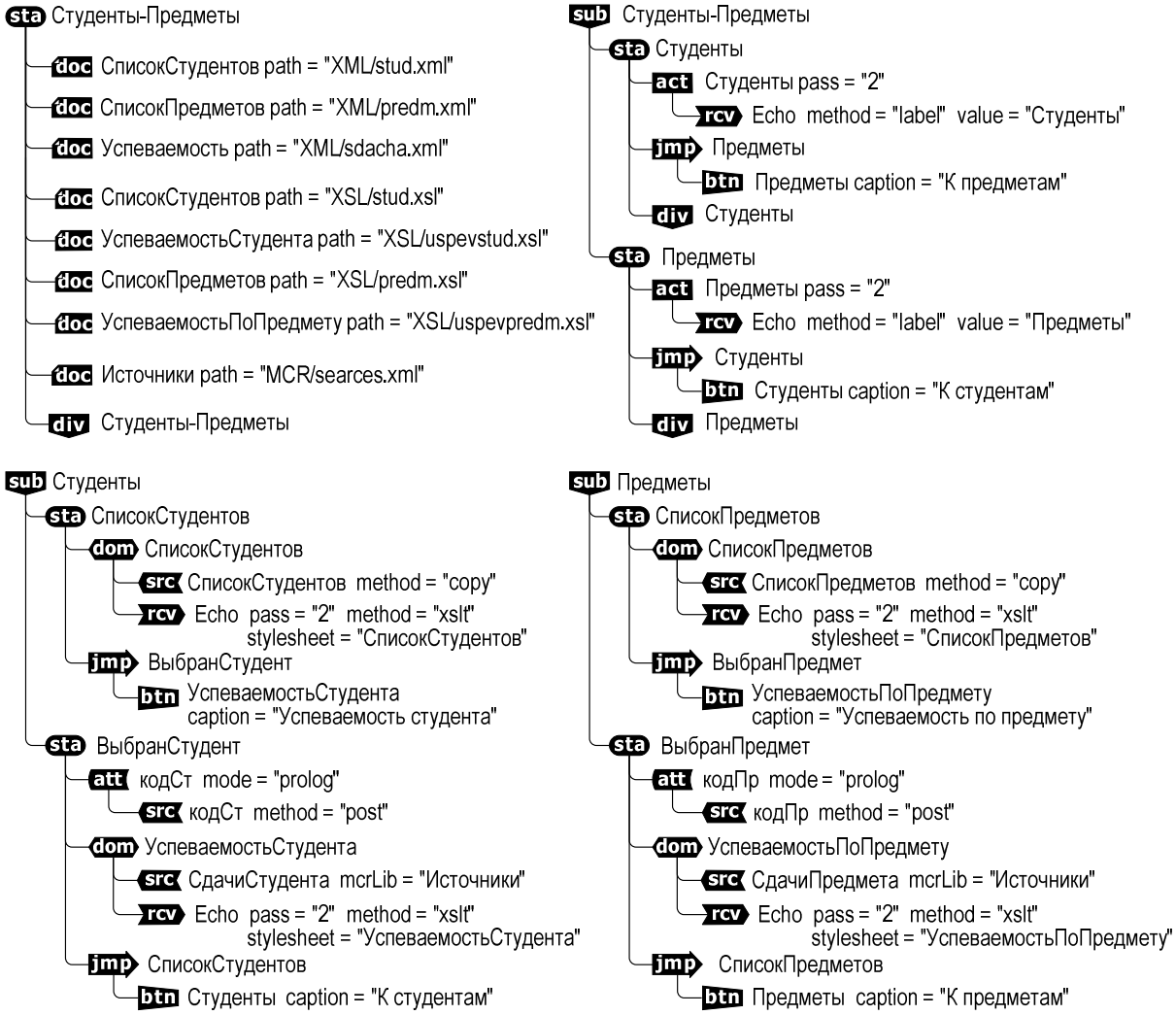


Рис. 2.12. Динамическая модель интернет-приложения

в состоянии sta:СписокПредметов — обо всех предметах, в состоянии sta:ВыбранСтудент — о предметах, сданных выбранным студентом, и полученных при этом оценках; в состоянии sta:ВыбранПредмет — о студентах, сдавших выбранный предмет, и полученных при этом оценках. DOM-элементы dom:СписокСтудентов и dom:СписокПредметов загружаются из простых источников данных — из готовых XML-документов, хранящихся в памяти ассоциированных данных и объявленных в корневом состоянии. Напротив, DOM-элементы dom:УспеваемостьСтудента и dom:УспеваемостьПоПредмету используют сложные источники данных, предусматривающие композицию нескольких XML-документов из памяти ассоциированных данных в зависимости от параметров — выбранного пользователем студента или предмета.

Att-элементы. Для хранения параметров, задающих студента или предмет, выбранных пользователем, в состояниях sta:ВыбранСтудент и sta:ВыбранПредмет предусмотрены att-элементы att:кодСт («код студента») и att:кодПр («код предмета»), создающие атрибуты элемента-состояния в памяти текущего состояния. Создание атрибутов и загрузки их значений происходит на прологовом этапе интерпретации состояний (mode = "prolog"), т. е. на том цикле интерпретации, на котором происходит переход из состояния sta:СписокСтудентов в состояние sta:ВыбранСтудент или из состояния sta:СписокПредметов в состояние sta:ВыбранПредмет. Элементы-источники при этом используют значения идентификаторов выбранного студента или предмета, извлекаемые из массива POST, полученного от пользователя вместе с URL (method = "post").

Сложные источники данных DOM-объектов. Чтобы не загромождать диаграмму модели сведениями, не относящимися непосредственно к навигации, сложные источники

данных представлены в виде макросов (на это указывает атрибут mcrLib), макроопределения которых вынесены в макробиблиотеку (рис. 2.13). В ходе интерпретации макрорасширения подставляются из библиотеки в динамическую модель. Рассмотрим источник данных для dom:УспеваемостьСтудента (источник данных для dom:УспеваемостьПоПредмету имеет аналогичную структуру). Он имеет три уровня вложенности элементов-источников: на верхнем уровне — элемент src:СдачиСтудента, который использует вложенный элемент src:ОценкиСтудента, в котором, в свою очередь, используется элемент src:СписокСтудентов. Начнем с самого внутреннего элемента src:СписокСтудентов.

Элемент-источник src:СписокСтудентов извлекает XML-данные из документа xml:СписокСтудентов в виде поддерева (method = "cut"), начинающегося в XML-элементе «студент». При этом отыскивается тот элемент «студент», у которого атрибут «код» (field = "@код") имеет значение, хранящееся в элементе att:кодСт памяти текущего состояния динамической модели (value = "att:кодСт"). Иными словами, этот источник извлекает XML-данные, соответствующие выбранному студенту. Далее с помощью других источников данные дополняются сведениями о сдачах и предметах этого студента.

Элемент-источник src:ОценкиСтудента выполняет слияние двух XML-деревьев (method = "merge"). Родительское дерево сформировано уже рассмотренным внутренним источником src:СписокСтудентов (parent = "internal"). Дочернее дерево представляет собой документ xml:Успеваемость (childDoc = "Успеваемость"), определенный в корневом состоянии. Условие слияния — равенство кода студента у элемента «студент» родительского дерева (parentElement = "студент" parentField = "@код") и кода студента у элемента «сдача» дочернего дерева (childElement = "Сдачи/сдача" childField = "@кодСт"). В результате сведения о выбранном студенте дополняются сведениями обо всех сдачах этого студента. Каждую сдачу необходимо дополнить сведениями о сданном предмете (названием предмета), что достигается в источнике верхнего уровня.

Элемент-источник src:СдачиСтудента выполняет слияние двух XML-деревьев (method = "merge"). Родительское дерево сформировано уже рассмотренным внутренним источником src:ОценкиСтудента (parent = "internal"). Дочернее дерево представляет собой документ xml:СписокПредметов (childDoc = "СписокПредметов"), определенный в корневом состоянии. Условие слияния — равенство кода предмета у элемента «сдача» родительского дерева (parentElement = "студент/сдача" parentField = "@кодПр") и кода предмета у элемента «предмет» дочернего дерева (childElement = "Предметы/предмет" childField = "@код"). В результате сведения о каждой сдаче выбранного студента дополняются сведениями о сданном при этом предмете.

Загружаемые XML-данные. На рис. 2.14, *а, б* приведены концептуальные модели XML-данных, формируемых сложными источниками для загрузки в DOM-объекты dom: УспеваемостьСтудента и dom:УспеваемостьПоПредмету. На рис. 2.14, *в, г* приведены экземпляры XML-данных для случаев, когда пользователем выбран первый студент (Иванов, код «s01») или первый предмет («Теория систем», код «p01»).

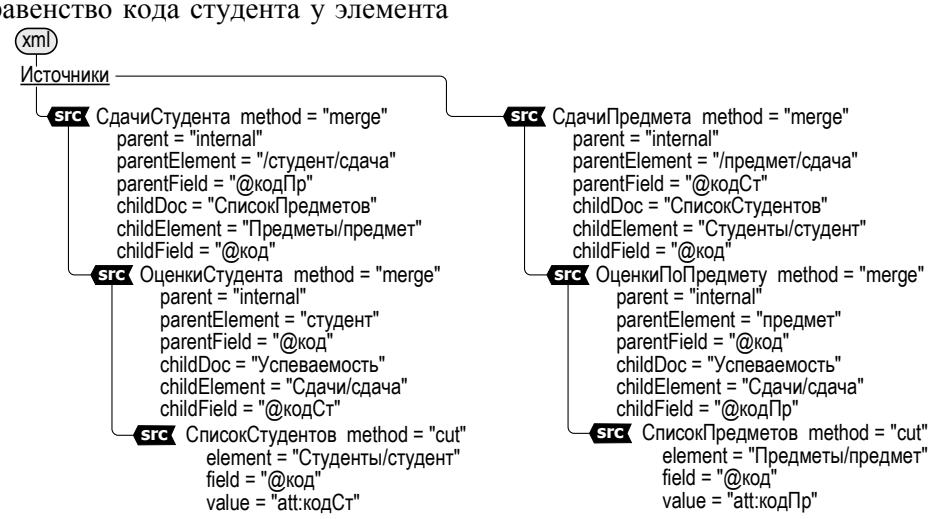


Рис. 2.13. Макроопределения сложных источников данных

Формирование результата для вывода в браузер клиента

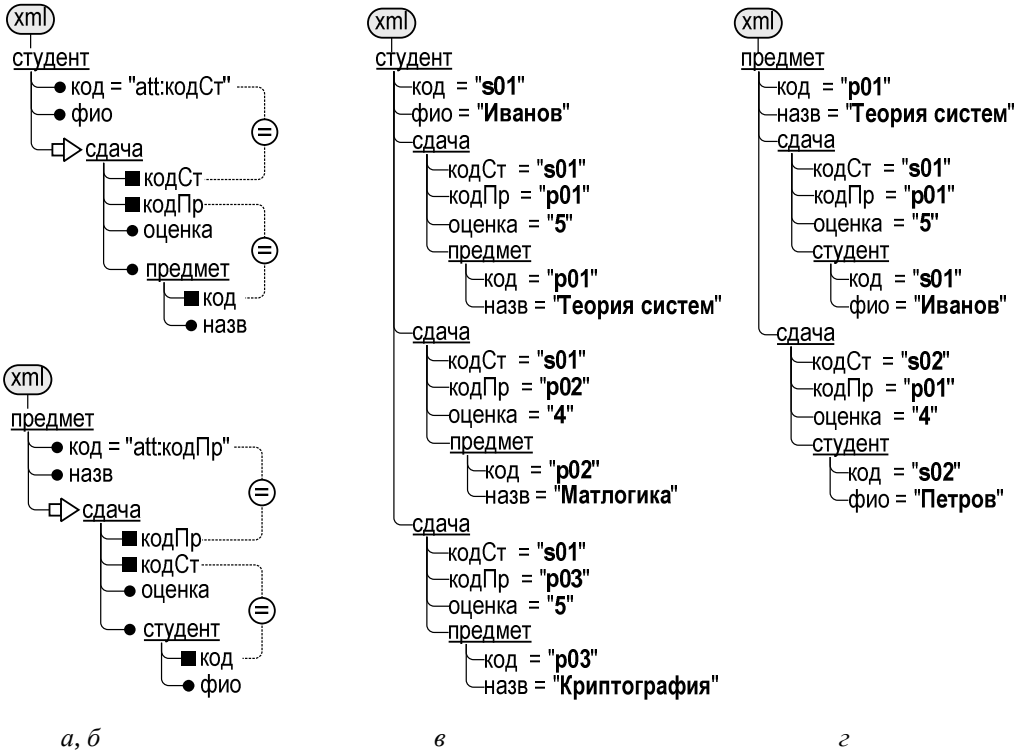
Результирующий HTML-код, отправляемый клиенту в ответ на входной запрос, должен обеспечить формирование в окне веб-браузера изображений, соответствующих текущему состоянию динамической модели (см. рис. 2.12). HTML-код формируется на втором проходе интерпретации динамической модели и состоит из фрагментов, порождаемых в результате интерпретации отдельных элементов текущих состояний на различных уровнях иерархии. Так, на втором уровне иерархии, в субмодели sub:Студенты-Предметы элементы-акции act:Студенты и act:Предметы обеспечивают вывод заголовков «Студенты» и «Предметы». В той же субмодели элементы btn:Предметы и btn:Студенты формируют HTML-кнопки «К предметам» и «К студентам» в верхней части изображения. Далее, на третьем уровне иерархии, в субмоделях sub:Студенты и sub:Предметы с помощью элементов-приемников rcv:Echo путем XSL-трансформации содержимого DOM-объектов в средней части изображения формируются списки: студентов, сдач выбранного студента, предметов, сдач по выбранному предмету. В этих же субмоделях элементы btn:УспеваемостьСтудента, btn:Студенты, btn:УспеваемостьПоПредмету, btn:Предметы формируют HTML-кнопки «Успеваемость студента», «К студентам», «Успеваемость по предмету», «К предметам» в нижней части изображения.

XSL-трансформация. Особую важность для нас представляет формирование фрагментов HTML-кода путем XSL-трансформации XML-содержимого динамических DOM-объектов. Подобная трансформация в динамической модели (см. рис. 2.12) предусматривается приемниками данных rsv:Echo DOM-объектов dom:СписокСтудентов, dom:СписокПредметов, dom:УспеваемостьСтудента и dom:УспеваемостьПоПредмету. Трансформация выполняется на втором проходе интерпретации динамической модели в соответствии со стилевыми таблицами, заданными в атрибутах «xsl» соответствующих rsv-элементов. На рис. 2.15 приведены концептуальные модели такого преобразования, следующие нотации работы [11]. В каждой такой модели справа показана концептуальная модель XML-данных, подвергающихся трансформации, а слева — модели XSL-шаблонов (Template), используемых для получения результирующих HTML-данных. Применяется так называемое форсирующее преобразование, в ходе которого процессор трансформации выполняет обход исходного XML-дерева и применяет соответствующие шаблоны, ассоциированные с его узлами.

Список студентов (рис. 2.15, *а*) формируется на основании документа xml:СписокСтудентов (см. рис. 2.10, *а*, *г*), загруженного в dom:СписокСтудентов. Предусмотрен единственный шаблон, ассоциированный с XML-элементом «студент»;этот шаблон будет выполняться для каждого студента (Иванова, Петрова, Сидорова согласно рис. 2.10, *г*), представленного в трансформируемом дереве. При каждом выполнении шаблона будет сформирован HTML-элемент <INPUT> с атрибутом type = "radio", который в окне клиентского браузера отобразится в виде радиокнопки. Атрибут name = "кодСт" задает имя кнопки, по которому ее состояние можно проверить в массиве POST, а атрибут value, которому присваивается значение кода студента, позволяет определить при этом выбранного студента. Условная конструкция создает для самого первого элемента <INPUT> атрибут checked = "checked", задающий выбор первого студента по умолчанию. Вслед за элементом <INPUT> выводится имя студента и HTML-элемент
, обеспечивающий переход в начало новой строки.

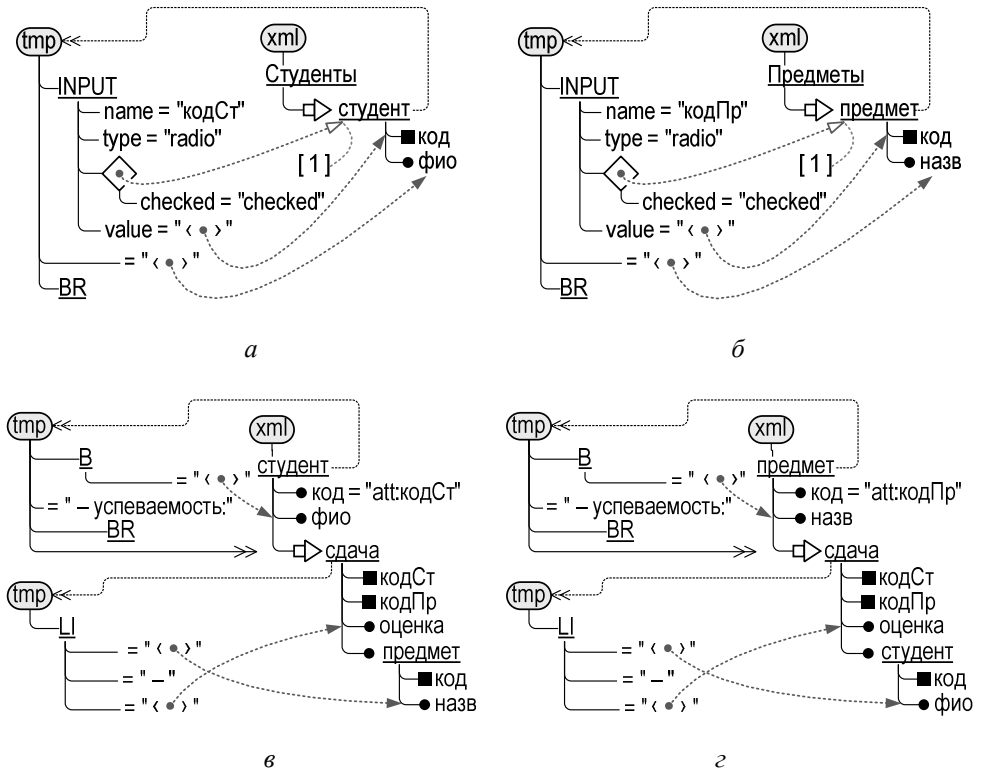
Список предметов (рис. 2.15, *б*) формируется аналогичным образом на основании документа xml:СписокПредметов (см. рис. 2.10, *б*, *д*), загруженного в dom:СписокПредметов.

Список сдач выбранного студента (рис. 2.15, *в*) формируется на основании XML-данных (см. рис. 2.14, *а*, *в*), загруженных в dom:УспеваемостьСтудента. Трансформация выполняется с помощью двух шаблонов, первый из которых ассоциирован с XML-элементом «студент», а второй — с XML-элементом «сдача». Первый шаблон срабатывает единственный раз, при обработке корневого элемента; он формирует строку заголовка, содержащую, в том числе, выделенное жирным шрифтом имя выбранного студента, и запускает обход внутренних элементов корневого элемента. Второй шаблон срабатывает при обработке каждой сдачи выбранного студента; он выводит HTML-элемент <LI>, формирующий в браузере строку маркированного списка. В каждой строке выводятся название сданного предмета и полученная оценка, разделенные символом тире.



а, б *в* *г*

Рис. 2.14. Модели содержимого DOM-объектов «УспеваемостьСтудента» и «УспеваемостьПоПредмету», сформированного сложными источниками данных: *а*, *б* — концептуальные модели; *в*, *г* — модели экземпляров для случая выбора первого студента и первого предмета



а *б*

в *г*

Рис. 2.15. Концептуальные модели трансформации XML-содержимого DOM-объектов при формировании HTML-кода, отправляемого в браузер клиента: *а* — формирование списка студентов; *б* — формирование списка предметов; *в* — формирование списка сдач выбранного студента; *г* — формирование списка сдач по выбранному предмету

Список сдач по выбранному предмету (рис. 2.15, *г*) формируется аналогичным образом на основании XML-данных (см. рис. 2.14, *б, г*), загруженных в dom:УспеваемостьПоПредмету.

ЗАКЛЮЧЕНИЕ

В данной работе на концептуальном уровне (без учета особенностей и возможностей среды реализации) предложен и исследован возможный подход к манипулированию XML-данными в ситуационно-ориентированных базах данных на основе динамически создаваемых DOM-объектов.

Концепция динамических DOM-объектов основана на привязке DOM-объектов к состояниям динамической модели: DOM-объекты создаются, загружаются, используются для обработки XML-содержимого, когда соответствующие состояния динамической модели становятся текущими, и удаляются после смены текущего состояния. В известном подходе это достигается за счет программирования функций в элементах-акциях, ассоциированных с состояниями динамической модели.

Отличие концепции состоит в том, что 1) у элементов-состояний динамической модели в качестве дочерних предусматриваются DOM-элементы, у которых, в свою очередь, дочерние элементы-источники задают загружаемые XML-данные, а дочерние элементы-приемники – сохраняемое XML-содержимое; 2) в ходе интерпретации динамической модели интерпретатором выполняется автоматическое создание DOM-объектов для текущих состояний модели и загрузка XML-данных с возможным преобразованием, а также автоматическое удаление DOM-объектов при смене текущих состояний.

Реализация концепции позволит разработчику динамической модели в декларативной форме гибко специфицировать XML-данные, требуемые в тех или иных ситуациях, и способы их получения из различных источников, избавляя его при этом от необходимости программирования соответствующей функциональности.

Дальнейшие исследования в этом направлении связаны с программной реализацией предложенных концептуальных решений в виде соответствующих дополнений в структуре динамической модели HSM и в алгоритмах интерпретатора динамических моделей HSMI.

СПИСОК ЛИТЕРАТУРЫ

1. **Model Based Systems Engineering** [Электронный ресурс]. URL: http://mbse.gfse.de (дата обращения 29.02.2012).

2. **Model Based System Development** [Электронный ресурс]. URL: http://www.ru.nl/mbsd (дата обращения 29.02.2012).

3. **Model-Driven Engineering** [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Model-driven_engineering (дата обращения 29.02.2012).

4. **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.

5. **Миронов В. В., Юсупова Н. И., Шакирова Г. Р.** Ситуационно-ориентированные базы данных: внешние представления на основе XSL // Там же. 2011 Т. 14, № 4 (39). С. 200–209.

6. **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.

7. **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.

8. **Миронов В. В., Маликова К. Э.** Интернет-приложения на основе встроенных динамических моделей: элементы управления пользовательского интерфейса // Там же. 2011 Т. 14, № 5 (40). С. 170–175.

9. **Document Object Model** (DOM) [Электронный ресурс]. URL: http://www.w3.org/DOM (дата обращения 29.02.2012).

10. **Simple API for XML** [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Simple_API_for_XML (дата обращения 29.02.2012).

11. **Миронов В. В., Юсупова Н. И., Шакирова Г. Р.** Иерархические модели данных: концепции и реализация на основе XML. М.: Машиностроение, 2011. 453 с.

ОБ АВТОРАХ

Миронов Валерий Викторович, проф. каф. автоматизир. систем упр-я. 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.

Гусаренко Артем Сергеевич, асп. той же каф. Дипл. информатик-экономист (УГАТУ, 2010). Готовит дис. о динамических DOM-объектах в ситуационно-ориентированных базах данных.