Jet Info

ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ № 1 (80)/2000

Современное состояние языков и средств разметки документов

ИНТРАНЕТ И JAVA

Современное состояние языков и средств разметки документов

Илья Дуров

СОДЕРЖАНИЕ

1. Введение

Так уж сложилось, что большую часть информации человек предпочитает хранить в виде документов. Но хранение документов не является самоцелью — это лишь промежуточный этап работы с информацией. Документ представляет собой объект, предназначенный для дальнейшего использования читателем, в роли которого может выступать и сам человек, и, что становится все более актуальным, компьютерные программы.

Каждый документ имеет три составляющие — содержание (смысловое наполнение), структуру и внешнее представление. В рамках данной статьи мы не будем останавливаться на анализе первой составляющей — содержании, а обратим свое внимание, в первую очередь, на структуру, а также на внешнее представление. Структура документа позволяет правильно определить составляющие его части и взаимоотношения между ними. Внешнее представление направлено на повышение эффективности восприятия информации читателем, что достигается за счет выделения смысловых частей документа теми или иными средствами, доступными для данной формы представления.

1.1. О разметке

В документе, помимо смыслового наполнения, должна содержаться некоторая метаинформация, позволяющая определить его структуру и внешнее представление. Такая метаинформация называется разметкой документа.

Разметка документа преследует следующие две основные цели:

- выделение смысловых частей (логических элементов) документа и связей между ними;
- указание действий, которые должны быть осуществлены с этими элементами.

Для достижения первой цели предназначена структурная разметка. Действия, направленные на получение внешнего представления, задаются разметкой представления.

В качестве примеров на листингах 1 и 2 приведены два возможных способа разметки начала нашей статьи.

В первом случае мы описываем раздел, который имеет заголовок и текст в виде абзаца, то есть определяем структуру документа. Структурная разметка говорит о том, как текст устроен, то есть из каких он частей состоит, и как эти части друг с другом соотносятся.

Во втором случае мы показываем, каким образом данный текст должен быть отображен на бумаге или на мониторе — выделить шрифтом Arial Bold размера 16, отступить по вертикали 20, сделать табуляцию 5, выделить шрифтом Times New Roman размера 12. Здесь мы имеем дело с разметкой представления документа, которая говорит о том, что делать с текстом, как его отображать.

Исторически разметка представления появилась раньше, и в течение длительного времени разметка документа была ориентирована исключительно на внешний (бумажный) вид документа. Но в последнее время ситуация существенно меняется — быстрый рост числа документов, их создание, хранение и использование в электронном виде, автоматизированная обработка и обмен документами предъявляют новые требования к разметке. В числе этих требований — независимость от среды представления, возможность осуществления эффективного поиска, возможность повторного использования как документа целиком, так и отдельных его элементов.

Сейчас существует большое число устройств, с помощью которых можно отображать документы. Среди таких устройств — и дисплеи от компьютерных до мобильных, и принтеры от формата A1 до встроенных в кассовые аппараты, и различные синтезаторы речи, и многое другое. Для воспроизведения некоторого документа на всех этих устройствах требуется либо наличие огромного количества ва-

```
<div1 type="Section">
 <head>Введение</head>
  <p>Так уж сложилось, что большую часть информации человек
  предпочитает хранить в виде документов...</p>
</div1>
```

Листинг 1. Пример структурной разметки

```
<font face="Arial Bold" size=16>1. Введение<hspace size=20>
<tab size=5><font face="Times New Roman" size=12>
Так уж сложилось, что большую часть информации человек
предпочитает хранить в виде документов...
```

Листинг 2. Пример разметки представления.

риантов одного и того же документа, только размеченного разными способами, либо существование единой универсальной разметки и программных средств для корректного преобразования в соответствующее внешнее представление «на лету».

Быстрый рост количества документов привел к тому, что поиск нужной информации стал занимать все больше и больше времени. Например, если нам необходимо найти в Интернет информацию об авторе статей по фамилии Дуров, то простой контекстный поиск даст нам огромное количество ссылок на те места, где встречается данная фамилия. После чего нам придется либо просмотреть все полученные ссылки, либо задавать дополнительную информацию для сужения области поиска. Если бы мы могли сразу указать, что фамилию следует искать только среди авторов журнальных статей технического плана, это во много раз упростило бы поиск. Но для этого необходимо, чтобы документы, среди которых ведется поиск, были размечены должным образом с явным выделением элементов «автор», «тематика» и т.п.

Возможность повторного использования документов или отдельных его частей приводит к тому, что мы не составляем каждый раз заново отчет или деловое письмо, используем в своей работе шаблоны контрактов, изменяя лишь некоторую существенную для данного случая информацию. Но делаем мы это преимущественно вручную. Если говорить об автоматизированном формировании, связывании, повторном использовании документов, то это становится возможным только тогда, когда документы как информационные объекты являются структурированными, а используемая метаинформация полно и ясно описывает характеристики каждого элемента документа.

Все перечисленные задачи можно решить, используя исключительно структурный подход при разметке документов. Именно структурная разметка позволяет выделять смысловые элементы, определять их связи с другими элементами как в рамках одного документа, так и вне этих рамок.

1.2. О языках и средствах разметки

Далеко не всякая разметка настолько формализована, что можно говорить о языке разметки. Язык разметки должен определять ряд специальных инструкций, правил и соглашений для описания структуры элементов документа и отношений между элементами этой структуры. Специальные инструкции, их еще называют маркерами или тэгами, в структурированных документах должны определенным образом кодироваться, то есть выделяться среди основного текста. Их главное назначение — служить управляющими инструкциями для программных средств обработки структурированных текстов.

В данной статье мы остановимся на истории возникновения таких языков разметки как SGML (Standard Generalized Markup Language, стандартный обобщенный язык разметки) и HTML (HyperText Markup Language, язык разметки гипертекстов), а также подробно расскажем о том, что собой представляет XML (eXtensible Markup Language, расширяемый язык разметки).

Для эффективной работы с языками разметки необходимо наличие специализированных средств для создания и редактирования размеченных текстов, их просмотра и разного рода обработки. Естественно, что развитие языков разметки, принятие стандартов и начало их широкомасштабного использования не могло не сказаться на росте количества предлагаемых программных продуктов. О некоторых из них будет рассказано в последующих разделах.

2. SGML

Стандартный обобщенный язык разметки (Standard Generalized Markup Language, SGML) был утвержден международной организацией по стандартизации (International Standards Organisation, ISO) в качестве стандарта ISO 8879:1986 в 1986 году [1].

SGML — это метаязык, то есть средство формального описания прикладных языков разметки, предназначенных для кодирования структурированных документов.

2.1. Разметка в SGML

Разметка, определяемая в рамках SGML, основывается на двух постулатах:

- разметка должна описывать структуру документа, а не указывать, что с документом или его частями должно происходить;
- разметка должна быть строгой, чтобы программы и базы данных могли быть использованы для хранения и обработки размеченных документов. Структура документа с точки зрения SGML представляет собой граф компонентов, вершины которого являются компонентами, а ребра — связями между ними. Основным компонентом структурированного текста является элемент. Таким образом, можно сказать, что каждый структурированный документ состоит из некоторого набора семантических элементов, связанных друг с другом по определенным правилам.

Синтаксическое представление элемента документа показано на рис. 1.

Тело элемента (содержательный текст) обрамляется открывающим и закрывающим маркерами. Каждый маркер состоит из имени элемента, уникального для элементов одинаковой семантики,

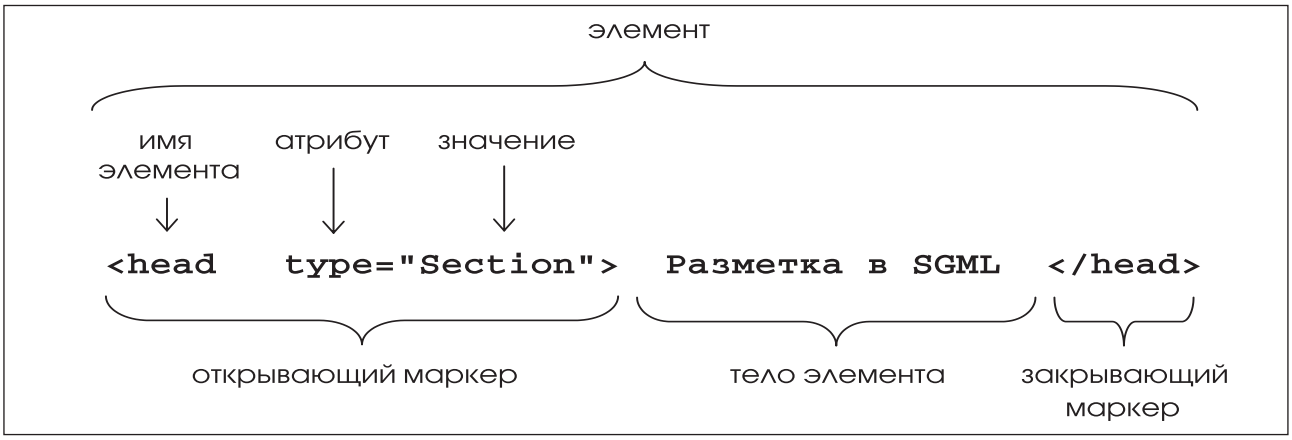

Рис. 1. Пример SGML-элемента.

и может иметь некоторое количество атрибутов. Атрибуты предназначены для более детального описания текста среди семантически однородных элементов.

Важным достоинством SGML является то, что он не определяет заранее имена элементов и их атрибуты. Например, если автор документа считает, что семантически корректнее определить в тексте два типа списков: список фамилий и список компаний, то он может ввести два элемента listofpeople и listofcompanies. В дальнейшем эти элементы могут обрабатываться как различные семантические единицы.

Чтобы документ являлся синтаксически корректным с точки зрения SGML, необходимо, чтобы его разметка подчинялась некоторому набору правил, определяемых стандартом ISO. Одно из правил состоит в том, что допускается лишь полная вложенность одного элемента в другой. Таким образом, в каждом документе всегда будет один корневой элемент и некоторое количество иерархически вложенных элементов. (Вообще говоря, допускается наложение на документ двух независимых разметок, элементы одной из которых могут не являться вложенными в другую, но это предмет отдельного обсуждения.) Вложенность является одним из видов связей между вершинами графа компонентов.

Размеченный документ предназначен для дальнейшей обработки различными программами, каждая из которых может применять свои правила обработки к тем или иным элементам документа. Одна программа может преобразовывать текст к виду, пригодному для печати на бумаге, а другая — лишь извлекать некоторые данные (например, названия терминов) и помещать их в таблицу или базу данных.

2.2. Типы документов

Структурная разметка не предназначена для обеспечения удобочитаемости документов. Для этого существует разметка представления и соответствующие программные средства, преобразующие структурную разметку в разметку представления. Эти и другие программы, обрабатывающие документ, должны уметь распознавать элементы структуры и атрибуты элементов и применять необходимые операции к определенным элементам.

В SGML это достигается с помощью определений типов документов (Document Type Definition, DTD), посредством конструкций языка, называемых декларациями элементов. В то время как разметка документа занимается описанием семантических единиц, DTD определяет набор всех возможных разметок документов описываемого типа.

Тип документа формально определяется его составными частями и их структурой. Например, письмо можно определить как документ, имеющий реквизиты отправителя и получателя, заголовок, несколько абзацев и дату отправления. Если документ не имеет реквизитов отправителя, то, в соответствии с нашим определением, письмом он не является.

DTD определяет допустимые элементы для данного типа документа на любом из уровней вложенности, допустимое содержание каждого из элементов и набор допустимых атрибутов. При этом наличие DTD является обязательным для любого документа. Можно сказать, что в рамках SGML имеют право на существование информационные объекты, состоящие из размеченного документа и его DTD.

Декларация элементов в DTD определяет допустимое содержание как тела элемента, так и его атрибутов. Предположим, например, что необходи-

```
<!--       ELEMENT     MIN    CONTENT (EXEPTIONS) -->
<!ELEMENT list         - -    (head?, item+)>
<!ELEMENT head         - 0    (#PCDATA)>
<!ELEMENT item         - 0    (p+)>
<!ELEMENT p             - 0    (#PCDATA)>
```

Листинг 3. Декларации списка и его элементов.

```
<list>
 <head>Перечень важных дел
 <item>
  <p>В 11-00 переговоры
  <p>Необходимо подготовить полный
     комплект документов
 <item>
  <p>В 14-00 совещание у руководства
</list>
```

Листинг 4. Пример использования списка.

мо дать определение элемента **<list>**, представляющего собой список. В этом случае декларации могут выглядеть так, как показано в листинге 3.

Первая декларация (вторая строка листинга, так как первая является комментарием) обозначает, что список может включать необязательный заголовок, но обязательно содержит один или несколько элементов списка.

Вторая декларация говорит, что заголовок содержит некоторое количество символов (текст).

Третья декларация указывает на то, что каждый элемент списка, в свою очередь, состоит из одного или более абзацев.

И, наконец, последняя декларация, как и вторая, говорит, что абзацы содержат символьный текст.

Символ «0» в колонке MIN обозначает, что закрывающий маркер в данном элементе может быть опущен без нарушения структуры документа. Следующий открывающий маркер такого же элемента или маркер внешнего элемента фактически будет выполнять ту же функцию.

Возможное использование списков приведено в листинге 4.

2.3. Работа с объектами

Одним из достоинств SGML является то, что он позволяет работать не только со структурированными текстами, но и с произвольными информационными объектами. Для этого вводится понятие объекта (**entity**).

Объектом может быть строка символов или файл (текстовый или бинарный). Для включения его в документ используется конструкция, известная в ряде языков программирования как ссылка на объект. Например, объявление

```
<!ENTITY SGML "Standard Generalized
Markup Language">
```

определяет объект, называющийся SGML, значением которого является строка «Standard Generalized Markup Language». Это пример декларации объекта (**entity declaration**), которая содержит внутренний объект (**internal entity**). Следующее объявление, напротив, вводит системный объект (**system entity**):

```
<!ENTITY picture SYSTEM "picture.gif">
```

В этом случае определен объект, являющийся рисунком, а не структурированным текстом. При обработке документа некоторой программой файл picture.gif может быть, например, выведен на экран монитора для иллюстрации соответствующего текста.

2.4. Возможности и недостатки SGML

В рамках данной статьи мы не ставим цель дать полное описание возможностей SGML и требований, которые в связи с этим накладываются на создаваемый документ. SGML представляет собой достаточно емкий и, в то же время, сложный метаязык. На его основе создаются языки разметки, используемые в различных областях: подготовка книг, документации, построение систем визуализации данных и т.д. Такие языки, как HTML, XML, MathML, CML и многие другие созданы на основе SGML и полностью ему соответствуют.

Широта охвата порождает вместе с тем и ряд недостатков. Так, например, создание единого DTD для подготовки документации в рамках одной организации, несомненно, имеет преимущества, такие как унификация исходного кода, возможность автоматического индексирования данных, ведение единого словаря терминов, написание стандартных средств обработки документов, получение стандартного бумажного представления и т.п. Но как только мы выходим за рамки организации, проекта или отрасли, то все упирается в утверждение данного DTD в качестве общего стандарта. Кроме того, как только принимается стандарт на некоторый DTD, сразу начинается борьба за его расширение, и так может продолжаться до бесконечности.

Другой недостаток проявляется при создании программ (например, для редактирования SGML-документов), которые должны позволять работать с любыми возможными DTD и учитывать все возможности, предоставляемые стандартом SGML. К сожалению, это возможно лишь теоретически, так как объем таких программ будет чрезвычайно велик.

Вот почему со временем возникла тенденция создания языков разметки с более простым синтаксисом, которые, в то же время, подчинялись бы требованиям стандарта SGML.

3. HTML

Язык разметки HTML родился в Лаборатории физики высоких энергий (CERN) в Женеве в 1990 году [2]. Первоначально HTML был предназначен для разметки научных документов и их последующего совместного использования сотрудниками разных институтов и лабораторий. HTML состоял из небольшого фиксированного набора элементов — заголовков нескольких уровней, абзацев, списков

и др., но главной его особенностью было использование гиперссылок и специальных меток (anchors) для указания точек перехода. Все вместе позволяло достаточно легко размечать простые документы и устанавливать связи как между ними, так и между компонентами одного документа. Человек всегда обрабатывает и анализирует информацию нелинейным образом. Поэтому возможности нелинейного хранения информации, простота использования языка разметки и широкие возможности применения привели к тому, что популярность HTML стала быстро расти и вне академических рамок. Как это часто бывает с любыми гениальными открытиями, успех превзошел все ожидания создателей.

3.1. Гонка стандартов

В 1992 году HTML был формализован в качестве SGML DTD, при этом в его спецификацию была заложена возможность дальнейшего расширения. Простой синтаксис языка, в отличие от SGML, позволял создавать простые программы для анализа размеченного текста и его отображения. Начался бурный рост публикаций в HTML-формате и рост числа приложений, поддерживающих этот формат. Потребности пользователей, а также конкурентная борьба производителей программного обеспечения привели к тому, что в HTML стали добавляться неспецифицированные элементы разметки. Отсутствие строгих синтаксических правил и использование нестандартных элементов вынудили производителей программного обеспечения допускать использование синтаксически некорректных конструкций. Отметим, что в WWW найдется не так много документов, полностью удовлетворяющих общепринятым спецификациям.

В целях регулирования процесса роста и стандартизации предлагаемых решений для WWW в октябре 1994 года была создана координирующая рабочая группа — World Wide Web Consortium (W3C), которая сегодня объединяет представителей более чем 370 организаций. Основными задачами W3C являются накопление информации о WWW, необходимой как разработчикам, так и пользователям, подготовка и утверждение стандартов (технических спецификаций) на технологии, связанные с WWW, и создание прототипов и образцов приложений для демонстрации использования новых технологий.

Положительная роль W3C в судьбе HTML очевидна — этот язык удалось сохранить от разделения на несколько диалектов, правда, ценой постоянного принятия все новых и новых расширенных спецификаций, которые сменяют друг друга с периодичностью раз в два года. Но нельзя же до бесконечности расширять язык, изначально предназначенный совсем для других целей! Борьба за перетягивание одеяла на свою сторону двумя крупнейшими разработчиками Web-навигаторов в конце концов привела к тому, что стандарты начали плыть, а пользователям приходилось очень часто менять программное обеспечение. Сами же пользователи все больше и больше становились зависимыми от разработчиков программных продуктов — у них не было возможности добавлять собственные расширения в языки разметки.

3.2. Структура или представление

За время своего существования HTML претерпел множество изменений, что весьма неприятно для создателей документов и разработчиков программ. Но гораздо большей неприятностью стало то, что, изначально задуманный как язык структурной разметки, в результате своего развития HTML превратился в язык разметки представления. Чего стоит, например, форматирование документа для улучшения его внешнего вида с помощью таблиц! Исходный текст таких документов становится практически нечитаемым, а доля полезной информации составляет лишь несколько процентов.

К счастью, ситуация постепенно начинает улучшаться. В последней на момент написания статьи версии языка HTML 4.0 содержится около 80 элементов. Темп роста их числа заметно уменьшился. Этому способствовало, прежде всего, введение атрибута CLASS во все элементы. Используя этот атрибут, можно определить новые семантические единицы без изменения синтаксиса языка в целом. (см. рис. 2) Немного неуклюжее, но все же решение. Кроме того, несомненным шагом вперед (или назад) по направлению к структуризации языка стало удаление ряда элементов, отвечающих только за внешнее представление, и декларирование строгой необходимости использования таблиц стилей (style sheets) для целей внешнего представления.

3.3. Недостатки HTML

Несмотря на массовое признание и использование HTML, а также на ряд разумных шагов, предпринятых W3C, в HTML все еще имеются существенные недостатки.

Отсутствие жесткой иерархии элементов приводит к тому, что один и тот же документ может быть размечен и, соответственно, будет интерпретироваться программным обеспечением различными способами. Так, например, текст HTML-документа или любая его часть может предваряться заголовком любого уровня, что оставляет автору слишком большую свободу выбора, а читателю создает некоторые трудности при работе с документами разных авторов.

Далеко не всякая метаинформация может быть простым и корректным образом вставлена в документ, поэтому при преобразовании произвольного документа в формат HTML часть информации может быть потеряна. Использование атри-

```
<div CLASS="author">
  <div CLASS="name">  Дуров Илья  </div>
  <div CLASS="email"> durov@jet.msk.su </div>
</div>
```

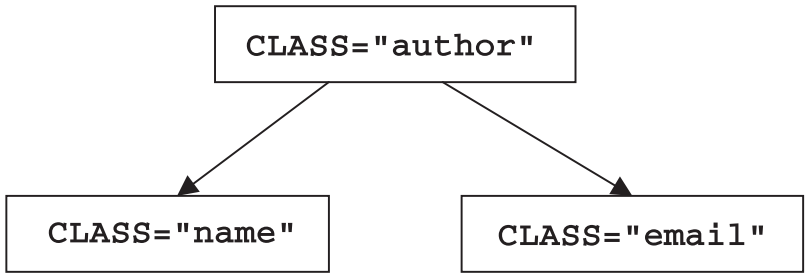

Рис. 2. Пример использования атрибута CLASS.

бута CLASS только частично решает эту проблему.

Для некоторых областей деятельности HTML не предоставляет возможностей ни структурно размечать требуемые элементы, ни правильным образом выводить их на экран или принтер. Математикам необходима возможность работы с формулами, химикам нужно отображать структуру химических соединений, и, вместе с тем, всем разработчикам и пользователям WWW необходимо наличие единых принципов разметки документов, универсальность их обработки и отображения.

4. XML – мода или необходимость?

В предыдущих разделах мы попытались проанализировать достоинства и недостатки двух «идейных антиподов» среди языков разметки — SGML и HTML. С одной стороны, максималистский подход при создании SGML привел к чрезмерной сложности языка и соответствующих программных продуктов, что неприемлемо для массового потребления. С другой стороны, простота и ограниченность HTML создавала трудности при описании сложных информационных объектов, поиске необходимой информации, создании приложений, обменивающих данными через Интернет. Поэтому в 1996 году была сформирована рабочая группа W3C, основной задачей которой являлось создание нового языка разметки. Этот язык должен был включать в себя гораздо больше возможностей SGML, чем HTML, но, в то же время, оставаться подходящим для использования в WWW. Чуть позже этот язык стал известен как XML (eXtensible Markup Language, расширяемый язык разметки). Разработка нового языка разметки велась около двух лет, и в начале февраля 1998 года W3C утвердила в качестве рекомендации первую спецификацию XML — XML версии 1.0 [3].

За сравнительно недолгий срок с момента своего появления на свет XML сумел завоевать огромную популярность среди разработчиков Интернет-технологий. Число созданных и разрабатываемых программных продуктов на основе XML, число компаний, включающих поддержку XML в свои уже готовые продукты, количество публикаций в компьютерной прессе уже весьма велико и продолжает расти. Что это — дань моде или естественное желание сохранить конкурентоспособность, используя современные и прогрессивные технологии?

4.1. Основные понятия

Как и SGML, XML является метаязыком для формального описания прикладных языков разметки, предназначенных для кодирования структурированных документов. Спецификация XML определяет, как стандартным способом разметить документ, выделяя все семантически значимые компоненты [4].

При разработке нового языка разметки учитывались достоинства и недостатки уже существующих языков, а также то, что основным местом применения XML является Интернет. Основные требования к создаваемому языку были сформулированы следующим образом:

- XML должен быть годен к непосредственному применению в Интернет.
- XML должен быть совместимым с SGML (XML-документ должен одновременно являться и SGML-документом без внесения каких-либо изменений или дополнений).
- Число необязательных свойств в XML должно быть минимальным, в идеале нулевым (любая XML-программа должна уметь читать любой XML-документ).
- XML-документы должны быть легко читаемы с помощью простейших текстовых процессоров.

- XML-разметка должна быть простой для понимания.

Формальное описание нового языка разметки состоит из нескольких взаимосвязанных частей:

- Спецификации eXtensible Markup Language (XML) 1.0, которая определяет синтаксис языка.
- Спецификаций XML Pointer Language (XPointer) и XML Linking Language (XLink), которые определяют стандартные механизмы установления связей между компонентами XML-документов.
- Спецификации eXtensible Style Language (XSL), которая определяет механизмы для внешнего представления XML-документов.

4.2. Структура

По своей структуре XML-документ очень похож на SGML- или HTML-документ. В качестве примера на листинге 5 приведено уже знакомое нам начало статьи.

Существует несколько основных правил составления XML-документа.

Каждый документ начинается с пролога. В данном случае — это инструкция **<?xml version="1.0"?>**, которая является XML-декларацией. Ее наличие идентифицирует XML-документ и указывает, какой версии XML он соответствует.

В данном листинге нет указания на используемое определение типа документа (DTD), так как, в отличие от SGML, XML не требует обязательного определения DTD для каждого документа. При необходимости описание или указание на месторасположение DTD также помещается в прологе документа.

За прологом следует тело документа, которое представляет собой жесткую структуру элементов, подчиняющихся принципу вложенности.

Именование элементов либо соответствует объявленному DTD, либо произвольно.

Обязательным является наличие как открывающего, так и закрывающего маркера в каждом элементе, ибо без этого при отсутствии DTD определить структуру документа невозможно.

Каждый из элементов может по аналогии с SGML содержать атрибуты, предназначенные для более детального описания семантически однородных элементов.

Возможно наличие пустых элементов, то есть элементов без содержимого. Такие элементы обозначаются с помощью символа '/' перед закрывающей угловой скобкой, например:

<Empty-Marker/>.

В общем случае XML-документ может иметь шесть типов компонент:

- элементы;
- ссылки на текстовые или бинарные объекты (entity references);
- комментарии;
- инструкции обработки;
- отмеченные разделы данных (CDATA sections);
- декларация типа документов.

Мы не будем подробно останавливаться на всех типах компонентов. Отметим лишь, что инструкции обработки, в соответствии со своим названием, предназначены для предоставления информации программам, которые будут в дальнейшем обрабатывать документ. Тип документа определяются тем же способом, что и в SGML, а отмеченные разделы данных позволяют передавать размещенные в них данные или текст «как есть», без анализа его структуры.

4.3. Структурная и синтаксическая корректность

Необязательность определения DTD, с одной стороны, существенно облегчает XML-разметку, но, с другой стороны, может значительно усложнить программы обработки. Каким образом определить в данном случае корректность XML-документа?

Чтобы определить класс правильно составленных (с точки зрения XML) документов, вводятся понятия структурной и синтаксической корректности. XML-документ является структурно корректным, если он отвечает следующим требованиям:

- Конструкция документа должна отвечать общим правилам составления документа, определенным в спецификации. В частности, некоторые конструкции (например, инструкция **<?xml version="1.0"?>**) могут присутствовать только в определенных местах документа.
- Никакой атрибут не используется более одного раза в одном маркере элемента.
- Значения атрибутов не ссылаются на внешние объекты.

```
<?xml version="1.0"?>
<Section>
 <head-of-section>Введение</head-of-section>
 <paragraph>Так уж сложилось, что большую часть информации человек
  предпочитает хранить в виде документов...</paragraph>
</Section>
```

Листинг 5. XML-разметка начала статьи.

- Все непустые элементы удовлетворяют принципу вложенности.
- Все используемые объекты продекларированы.
- Нет ссылок на бинарные объекты непосредственно из текста. Такие ссылки возможны лишь в момент декларации объекта.
- Текстовые объекты не являются рекурсивными.

По определению, если документ не является структурно корректным, то он не является и XML-документом.

При наличии у документа DTD возможна его проверка на синтаксическую корректность. При этом XML-документ считается синтаксически корректным, если он, во-первых, является структурно корректным, а, во-вторых, полностью соответствует всем правилам, изложенным в соответствующем DTD.

4.4. Ссылки в XML-документах

Для языка разметки с непредопределенными названиями элементов и даже отсутствующим иногда DTD невозможно определить стандарт на механизм связывания через элементы. Напротив, ссылающиеся и указуемые объекты должны иметь специальные атрибуты, которые идентифицируют их в этом качестве.

Все элементы XML имеют специально зарезервированный атрибут **XML-LINK**. Присутствие этого атрибута в элементе определяет наличие ссылки, а значение атрибута указывает, какой тип ссылки в данном месте используется. В XML, в отличие от HTML, возможно создание не только однонаправленных гипертекстовых ссылок по типу «один-к-одному», но и:

- Двунаправленных ссылок. Используя HTML и перейдя по стандартной гипертекстовой ссылке на новую страницу, пользователь имеет только одну возможность перехода назад — нажатием кнопки «Back» в Web-навигаторе. При использовании двунаправленных ссылок пользователь не только может вернуться по ссылке в то место, откуда пришел, но и перейти на те страницы, которые ссылаются на указуемый объект.
- Ссылок с возможностью перехода к одному или нескольким объектам («один-ко-многим»). У пользователя появляется возможность выбора между несколькими точками назначения при использовании всего одной ссылки.
- Ссылок к объектам, размещенным в базах данных. Гипертекстовые ссылки в HTML опираются только на файловую адресацию информации, а XLink предоставляет возможность динамического изменения адресов местонахождения информации с помощью баз данных.

То, что произойдет при переходе по ссылке, определяется атрибутом **SHOW**, который может иметь одно из следующих значений: **EMBED**, **REPLACE**, **NEW** [5].

В первом случае указуемый объект будет импортирован в то место, откуда идет ссылка. Это произойдет либо при показе документа, либо при его обработке. Такой подход может быть полезен при вставке некоторого текста из другого файла, или для вставки картинки внутрь текста. При этом возможна как автоматическая подстановка объекта, так и ручная, требующая от пользователя некоторых действий.

Во втором случае ссылающийся объект будет заменен на указуемый. Это может быть полезным, например, при наличии двух вариантов некоторого компонента. При помощи этого механизма возможен просмотр обеих версий или обработка по выбору, в зависимости от наличия тех или иных инструкций обработки.

В последнем случае исходный объект исчезает, и происходит полный переход к указуемому объекту. Такой механизм реализован в обычных гипертекстовых ссылках, когда при переходе по ссылке на экране отображается новая HTML-страница.

Механизмы ссылок и адресации в XML описываются в трех спецификациях W3C: XPath, XPointer и Xlink. Xlink описывает механизмы связывания: организацию многонаправленных и однонаправленных ссылок между ресурсами, аннотированных ссылок и внешних наборов ссылок [6].

XPath поддерживает спецификацию месторасположений в XML-документах в терминах элементов и списков элементов и является языком для адресации частей XML-документа. Xpath использует компактный не-XML-синтаксис для возможности применения XPath внутри указателей ресурсов и значений атрибутов XML. Он оперирует с абстрактной структурной моделью документа, а не с его синтаксическим выражением, и моделирует документ как некое дерево элементов для осуществления навигации по иерархической структуре XML-документа.

XPointer, построенный на основе XPath, определяет язык для поддержки адресации внутренних структур XML-документа. В частности, он позволяет определять ссылки к элементам, символьным строкам и т.п. вне зависимости от того, имеют ли они универсальный атрибут идентификации (ID). Для этого XPointer использует структуру документа и возможность выбора его частей на основе типов элементов, значений атрибутов, относительного расположения, содержания или порядка следования элемента. Так, например, возможно ссылаться на первый абзац третьей главы. И даже после осуществления другой разбивки на главы и абзацы или добавления нескольких абзацев впереди указуемого эта ссылка все равно приведет нас именно к первому абзацу третьей главы. Это свойство чрезвычайно полезно, если нужно сослаться на некото-

рое место в чужом документе, к которому нет прав на редактирование, а автор не расставил меток на требуемом участке документа.

XLink определяет конструкции, которые могут быть использованы в XML-документах для описания связей между объектами. XLink может описывать простые ссылки (simple links), которые очень похожи на гипертекстовые ссылки в HTML, но при этом они могут быть более детально классифицированы, например, по типу приложения, которое будет использовать данный документ. Кроме этого, XLink в комбинации с XPointer может описывать расширенные ссылки (extended links) и указуемые точки объекта, являющиеся многонаправленными по своей природе.

Косвенные ссылки (indirect links) позволяют упростить сопровождение большого количества документов. Рассмотрим случай, когда необходимо изменить месторасположение какого-нибудь документа, скажем, перенести его в другой каталог или на другой сервер. При использовании простых гипертекстовых ссылок это привело бы к необходимости найти все места во всех документах, откуда определены ссылки на перемещаемый документ, и внести соответствующие исправления. При этом разрешения на редактирование ко всем документам получить практически нереально. Потребуются просто титанические усилия, чтобы осуществить задуманное, поэтому документ, на который ссылаются другие документы, становится неперемещаемым по файловой системе. С использованием косвенных ссылок все оказывается гораздо проще. Связывание документов происходит не напрямую, а через некоторый промежуточный файл ссылок. При изменении месторасположения документа необходимо лишь внести изменения в этот промежуточный файл, а все остальные документы при этом останутся нетронутыми.

4.5. Отображение документов

Используя XML, автор документа может самостоятельно определять тот набор элементов, который наиболее точным образом будет соответствовать его структурным компонентам. Но свобода выбора имеет свою цену — набор используемых элементов не обладает предопределенной семантикой. Для совместной работы с XML-документами необходим стандартный механизм получения внешнего представления. Таким механизмом для XML является XSL (eXtensible Style Language, расширяемый язык стилей).

Обычные таблицы стилей, используемые, например, для работы с HTML, содержат набор инструкций, которые говорят программе (Web-навигатору, текстовому редактору или процессору печати), каким образом преобразовывать структуру документа во внешнее представление [7]. При этом таблицы стилей, как правило, содержат такие инструкции, как:

- отображать гипертекстовые связи синим цветом;
- начинать главу с новой страницы;
- вести сквозную нумерацию рисунков по всему документу.

Необходимо понимать, что использование или наложение стиля — это не что иное, как преобразование исходного документа к требуемому виду. Документ, отображаемый на экране, и документ, написанный и размеченный автором — это совсем не одно и тоже. Степень трансформации может меняться в зависимости от презентационных целей — страница документа для публикации в Интернет и для высококачественной полноцветной полиграфической печати должна обрабатываться по-разному, но в любом случае требуется некоторое преобразование [8].

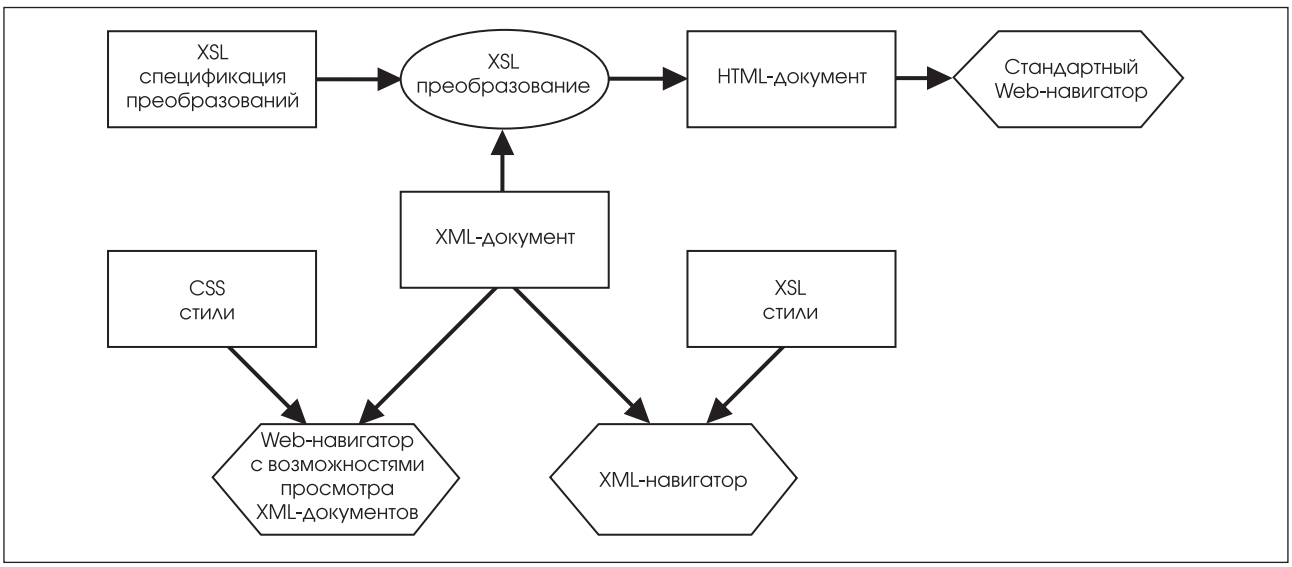

Рис. 3. Возможные преобразования XML-документа.

Использование языков разметки с предопределенной семантикой позволяет существенно упростить реализацию таблицы стилей. Программа, обрабатывающая, например, размеченную таблицу, может отобразить ее различным способом, но она заранее, даже без использования таблицы стилей, знает, что обрабатываемый объект является таблицей.

В случае использования XML-разметки XSL не только должен определять, каким образом тот или иной элемент будет отображаться, скажем, на экране, но и каким объектом он будет в итоге являться. Для того чтобы передать содержание XML-документа наиболее эффективным образом, необходимо две вещи: стандартный язык, описывающий требуемую разметку на выходе (в XSL это форматирующие объекты — formatting objects), и средство для преобразования исходного документа к требуемой разметке (в XSL это язык трансформации — transformation language). XSL включает стандартный словарь форматирующих объектов с хорошо определенными свойствами для осуществления контроля. Эти форматирующие объекты, такие как страница, блок текста, таблица, список и другие, позволяют авторам стилей получать высококачественное внешнее представление.

Работа с XML начинается с обработки исходного текста программой-анализатором (parser). Эта программа проверяет структурную и синтаксическую корректность XML-документа и создает дерево элементов исходного документа. Далее вступает в действие XSL-процессор, который в качестве исходных данных берет построенное дерево и соответствующий стиль. Шаг за шагом, начиная с корневого элемента, XML-процессор по шаблону, определенному в таблице стилей, обрабатывает всю структуру документа. Получающееся в результате дерево элементов может состоять из форматирующих объектов, которые и описывают внешнее представление документа. Форматирующие объекты представляют собой описание, независимое от устройства представления, и, следовательно, конечный документ может быть использован различными устройствами вывода.

Возможна и альтернатива форматирующим объектам. Так, в случае необходимости преобразования к HTML-виду, вместо форматирующих объектов будут использованы элементы языка разметки HTML. При этом результирующий документ будет выглядеть очень похожим на HTML-документ и обрабатываться стандартными Web-навигаторами. Однако следует понимать, что любое XSL-преобразование XML-документа в результате даст тоже XML-документ.

Основными преимуществами XSL над другими механизмами наложения стилей, помимо возможности работы с элементами непредопределенной семантики, являются:

- возможность изменения порядка следования элементов в результирующем документе;
- возможность сортировки и сравнения элементов текста (список используемых терминов, упомянутых авторов);
- повторная обработка некоторых элементов (например, для печати разными стилями названия главы в начале страницы, в колонтитуле, оглавлении);
- возможность генерации вспомогательного текста («Глава», «Оглавление», «Список иллюстраций» и т.п.);
- подавления вывода некоторого текста (удаление редакторских примечаний или вывод только предисловия, а не полного документа).

5. Средства работы с языками разметки

5.1. Анализаторы структурированных документов

Основное назначение SGML-документов — хранить структурированную информацию. Для извлечения информации из документов и передачи ее прикладной программе используются программы-анализаторы (parsers).

Программа-анализатор по исходному документу строит древовидную структуру с ассоциированной метаинформацией о назначении элементов-узлов. Эта структура используется в дальнейшем прикладными программами. В качестве примера приложения можно привести программу просмотра, которая, применяя к дереву элементов некоторый стиль отображения, выводит документ на экран в удобном для чтения виде. Другим примером является программа преобразования дерева в HTML-формат, понятный обычному Web-навигатору.

Анализаторы используются также для проверки структурной и синтаксической корректности XML-документов. В последнем случае необходимо наличие соответствующего DTD.

Одним из лучших SGML-анализаторов является свободно распространяемый, написанный на языке C анализатор SP, который используется как для построения древовидной структуры элементов, так и для проверки синтаксической корректности разметки. Основными его достоинствами являются поддержка практически всех используемых в настоящее время аппаратно-программных платформ, быстрота работы и доступность в исходных текстах, что позволяет при необходимости модифицировать и встраивать его в новые приложения.

Для обеспечения мобильности многие анализаторы написаны на языке Java. Это, например,

msxml, предназначенный для разбора XML-документов с последующим просмотром его структуры в Internet Explorer 5, а также XML for Java, созданный в корпорации IBM.

5.2. Редакторы структурированных документов

5.2.1. Adept Editor

Редактор Adept Editor компании Arbortext предназначен для создания и редактирования SGML- и XML-документов [9].

Adept Editor состоит из собственно редактора текста документа, модуля для редактирования таблиц, редактора математических формул. Дополнительно Adept Editor может включать в себя модули проверки орфографии на 15 языках.

В комплект поставки Adept Editor входит стандартный набор DTD для создания деловых документов, файлов в формате HTML и технической документации. При необходимости возможно использование любых имеющихся у пользователя корректных SGML- или XML-DTD или определение собственных типов документов с помощью дополнительного продукта Arbortext Architect.

Внешний вид редактора и его функциональное наполнение соответствует тому, что пользователи привыкли получать от традиционных текстовых редакторов. Многооконность, многофункциональная кнопочная панель, использование режима «cut-and-paste» дополнено, однако, специфическими особенностями, связанными с работой со структурированными данными. Так, например, Adept Editor позволяет редактировать документ с одновременным предварительным просмотром его внешнего представления (см. рис. 4). Для этого используется набор стилей и возможность быстрой фиксации произведенных изменений. Существует возможность редактирования как в режиме WYSIWYG (в том числе при работе с таблицами и формулами), так и в режиме полного показа маркеров разметки.

При вставке блока информации в редактируемый текст Adept Editor проводит автоматическую проверку правильности получаемой структуры. В случае, если будет обнаружено несоответствие структуры получаемого текста заданному DTD, то либо недостающая структура будет сформирована автоматически, либо в копировании информации будет отказано.

Предусмотрены специальные функции Quick Tag и Tag Prompt. Quick Tag позволяет в любом месте редактируемого документа получить список допустимых в данном месте элементов (в соответствии с используемым DTD). Tag Prompt автоматически выделяет пустые элементы в документе, сообщая о пропуске смысловой информации.

Отдельно стоит отметить возможность использования командной строки для осуществления операций над редактируемым текстом, поддержку Unicode, поддержку таблицы формата CALS, SGML Open Exchange и ATI, а также наличие встроенного языка Adept Common Language (ACL). С помощью ACL возможно автоматизировать работу пользователя с самим редактором, а также интегрировать Adept Editor с другими офисными приложениями.

Adept Editor позволяет открывать для редактирования файлы с некорректной разметкой для устранения ошибок и приведения текста к виду, соответствующему используемому DTD.

На момент написания статьи Adept Editor был доступен на платформах Sun Solaris 2.6, HP-UX 10.20, Windows NT 4.0, Windows 95 и Windows 98.

Недостатками Adept Editor являются его высокая стоимость, требовательность к ресурсам и отсутствие поддержки операционной системы Linux.

5.2.2. XMetaL

Продукт XMetaL компании SoftQuad представляет собой весьма изящный редактор XML-документов, очень простой в использовании. Его внешняя схожесть со стандартными редакторами текстов делает очень простым процессы обучения пользователей и внедрения нового программного средства в качестве рабочей среды [10].

Широкие возможности настройки свойств редактора под любой используемый DTD без какого-либо программирования позволяют превратить XMetaL в удобный инструментарий автора. Проверка соответствия используемому DTD происходит «на лету», и в случае обнаружении ошибки в разметке происходит автоматическое переключение в режим плоского текста для предоставления автору возможности коррекции разметки.

Несомненным удобством является возможность работы с тремя видами отображения документа. Стандартным и наиболее принятым для обычных текстовых редакторов является отображение WYSIWYG. Редактирование в режиме плоского текста может быть особенно удобным для авторов, привыкших к возможности все править своими руками. И, наконец, режим работы с маркированным текстом сочетает возможности стандартных тестовых редакторов и быстроты доступа к метаинформации: элементам и их атрибутам.

Встроенный в редактор Инспектор Атрибутов, доступный при любом отображении текста, позволяет проверять корректность разметки и предоставляет список атрибутов, доступных в точке редактирования. При удалении или вставке некоторого блока информации пользователь автоматически обращается к Инспектору Атрибутов, который проверяет корректность действия с учетом DTD.

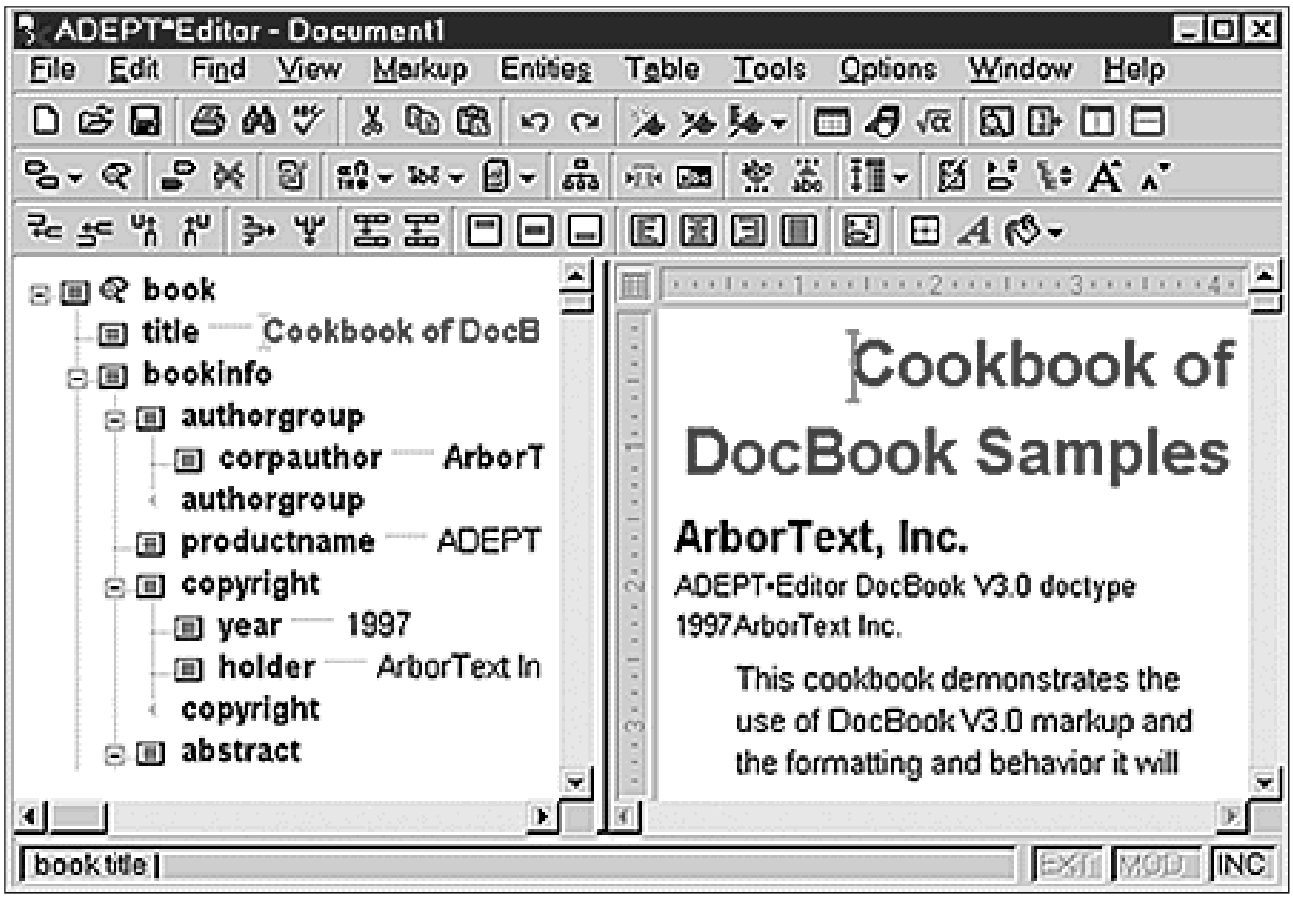

Рис. 4. Пример работы с редактором Adept Editor.

Весьма полезными свойствами XMetaL являются возможности проверки орфографии и расширенные механизмы поиска и замены — как текста, так и элементов документа.

Менеджер ресурсов предоставляет автору документа расширенные возможности объектной работы с изображениями, текстовыми фрагментами, стилями представления, шаблонами документов и т.п. вне зависимости от того, где расположены эти объекты — на жестком диске, в локальной сети или Интернет.

Использование Database Import Wizard дает возможность с помощью ODBC получать доступ к базам данных. XMetaL позволяет осуществлять запросы к базам данных, создавать персональную библиотеку таких запросов для их повторного применения. Он поддерживает сложные запросы к объединениям таблиц данных. Вставка запрошенной информации осуществляется в виде CALS- или HTML-таблиц, а также в виде отдельных элементов XML.

Поддержка каскадных таблиц стилей (Cascading Style Sheets, CSS) заключается в возможности не только немедленного отображения текста с использованием стиля во время редактирования, но и, при необходимости, редактирования самих CSS-файлов с помощью встроенного редактора.

Несомненными достоинствами XMetaL являются нетребовательность к ресурсам, возможность получения пробной версии редактора по Интернет. Однако XMetaL работает только под Windows (95/98/NT), и на момент написания статьи не было известно, планируется ли его перенос на другие платформы.

5.3. Просмотрщики

При создании XML-документов с помощью специализированных редакторов, как правило, удобно пользоваться специальными средствами просмотра, встроенными в сам редактор. Если же по каким-либо причинам подобный подход не годится, то для просмотра можно найти отдельный программный продукт. Правда, выбор тут не особенно велик, что, в первую очередь, связано с отсутствием ряда утвержденных стандартов на стили отображения (например, спецификация XSL все еще находится в стадии разработки).

5.3.1. Mozilla

Mozilla создается и распространяется по модели Open Source, что означает доступность исходных текстов продукта для всех желающих. В текущей версии M11 включена полная поддержка основных стандартов (HTML 4.0, XML, CSS, DOM). Кроме того, сама программа является небольшой по объему, быстрой и модульной. Mozilla включает XML-анализатор Expat, поддерживает просмотр XML + CSS, а также простые ссылки XLink.

5.3.2. XML Viewer

XML Viewer for Java является приложением, целиком написанным на языке Java, которое позволяет просматривать любые XML-документы. Пользователь может перемещаться по документу, представленному в виде дерева, находить атрибуты отдельных элементов и просматривать как исходные тексты XML-документов, так и соответствующие DTD. Возможно также одновременное отображение некоторого элемента исходного текста XML-документа и определения этого элемента, указанного в DTD.

5.3.3. IE5

Internet Explorer 5.0 является программным продуктом, который может отображать XML-документы с использованием стилей CSS. XML-документы могут быть показаны и в виде иерархической структуры элементов, и без использования специальных стилей. Поддержка XSL заключается в возможности осуществления преобразований и пока не включает поддержку словаря форматирующих объектов. IE 5.0 также осуществляет поддержку векторного языка разметки (Vector Markup Language, VML).

5.4. Пакетные средства

Одним из достоинств работы с размеченными документами является возможность их пакетной обработки. В качестве примера можно привести программный продукт Jade. Он представляет собой реализацию обработки разобранных структурированных документов с помощью языка описания стилей DSSSL (Document Style Semantics and Specification Language). Это первый (и, кстати, свободно распространяемый с исходными текстами) продукт подобного класса.

Jade доступен для большинства UNIX-платформ, существуют его реализации под Windows. На выходе он позволяет получать файлы в форматах RTF, TeX, HTML и MIF.

5.5. XML и Java

Данный раздел посвящен вопросам взаимодействия двух миров, двух систем: XML и Java.

Эти миры объединяет изначально заложенная в них независимость от аппаратных и программных платформ, национальных языков и т.п. Миры XML и Java дополняют друг друга. Java реализует процедурный подход, XML — декларативный. По идее, взаимодействие Java и XML должно привести к созданию функционально полной, универсальной среды обработки структурированных документов, что необходимо для организации электронного документооборота, ведения электронного бизнеса, движения по другим перспективным направлениям.

Вероятно, наиболее кардинальным решением проблемы организации взаимодействия миров XML и Java стала бы автоматическая генерация Java-классов и объектов по XML-схеме и документам, удовлетворяющим схеме. На достижение этой цели направлен проект XML Data Binding.

Схема XML-документа — относительно новое понятие. В первом приближении можно считать, что схема играет ту же роль, что и DTD, задавая синтаксис и семантику множества XML-документов, однако обладает большей, чем у DTD, выразительной силой. Главное отличие от DTD состоит в том, что схемы могут содержать определения типов данных и сложных связей между компонентами документов.

Если выделить в качестве понятий верхнего уровня схему и документ (для XML-мира), а также класс и объект (для Java), то связи между ними, которые предполагается реализовать в рамках проекта XML Data Binding, сводятся к компиляции схем (то есть к автоматической генерации Java-классов, соответствующих элементам схемы) и к реализации экспорта/импорта XML-документов (то есть к преобразованию XML-документов, удовлетворяющих заданной схеме, в Java-классы и наоборот).

Следуя [12], рассмотрим в качестве примера обработку заказов на обувь. Пусть заказ должен выглядеть примерно так, как показано на листинге 6. Соответствующая схема (с некоторыми изъятиями) приведена на листинге 7. Язык схем достаточно прост, так что листинги не требуют пояснений. (Исчерпывающие сведения о схемах можно найти по адресу http://www.w3.org/XML/Group /Schemas.html.)

Разрабатываемый компилятор XML-схем должен преобразовать приведенную на листинге 7 схему в Java-класс, показанный на листинге 8.

Остановимся на некоторых интересных моментах, связанных с компиляцией схем.

Во-первых, результатом компиляции схемы, вообще говоря, является не один класс, а несколько. В данном случае должен быть сгенерирован еще и класс Style.

Во-вторых, set-методы должны осуществлять проверки (например, проверки размеров обуви), предписанные схемой.

В-третьих (и это самое существенное), должны быть реализованы методы как для порождения Java-объектов, представляющих заданный XML-документ (метод umarshal — экспорт XML-докумен-

```
<ShoeOrder id="12345" style="Sandal">
  <size>9 1/2</size>
  <color>Brown</color>
  . . .
</ShoeOrder>
```

Листинг 6. XML-документ, представляющий заказ на обувь.

```
<schema name="ShoeOrder">
 <elementType name="ShoeOrder">
  <attrDecl name="id" required="true">
    <datatypeRef name="ID"/>
  </attrDecl>

  . . .

  <model>
    <sequence>
     <elementTypeRef name="size"/>
     <elementTypeRef name="color"/>
     . . .
    </sequence>
  </model>

  <datatype name="Size">
    <basetype name="string"/>
    <lexicalRepresentation>
     <lexical>[1-9][0-9]?( 1/2)?</lexical>
    </lexicalRepresentation>
    <minInclusive>3 1/2</minInclusive>
    <maxInclusive>13</maxInclusive>
  </datatype>

  <elementType name="size">
    <datatypeRef name="Size"/>
  </elementType>

  . . .

 </elementType>
</schema>
```

Листинг 7. Схема, описывающая заказы на обувь.

тов в Java-среду), так и для генерации соответствующих XML-документов (метод marshal, импорт XML-документов).

Проект XML Data Binding — это весьма амбициозное, но крайне соблазнительное предприятие. После преобразования XML-документа в совокупность Java-объектов работа с ним становится естественной, прямой, а не опосредованной. Это сулит существенный выигрыш по времени, стоимости и качеству XML-приложений.

В недавно созданную экспертную группу проекта XML Data Binding вошли представители крупнейших компаний: Sun Microsystems, AOL/Netscape, IBM, Oracle и некоторых других. Будет весьма интересно проследить, как пойдет процесс интеграции XML и Java.

У компании Sun Microsystems есть и более приземленные планы по интеграции XML и Java. В рамках развития окружения времени выполнения Java-программ реализована пробная версия дополнительного (то есть не входящего в число стандартных) пакета для анализа XML-документов. Соответствующий программный интерфейс получил наименование JAXP (Java API for XML Parsing).

```
public class ShoeOrder {

  public ShoeOrder (String id, Style
  style, String size, ...);

  public String getId ();
  public void setId (String id);

  . . .

  public String getSize ();
  public void setSize (String size);

  . . .

  public void marshal (OutputStream
  out)
    throws IOException;
  public static ShoeOrder umarshal
  (InputStream in)
    throws IOException;
}
```

Листинг 8. Java-класс, полученный в результате компиляции XML-схемы.

JAXP предоставляет базовые средства для чтения, обработки и генерации XML-документов. С помощью этого интерфейса Java-программы смогут взаимодействовать не только с предложенной Sun эталонной реализацией XML-анализатора, но и с другими продуктами, такими как Xerces (см. http://xml.apache.org/). Программы, использующие интерфейс JAXP, работают под управлением событий, возникающих в процессе синтаксического разбора документов.

Выпуск окончательной версии пакета запланирован на первый квартал 2000 года.

6. Области применения

В этом разделе мы коротко опишем некоторые проекты, где используются либо XML, либо производные от него языки.

Компания Zedak Corp., являясь агентом The New York Times, разработала систему автоматической подписки на рекламу. Эта система использует специально разработанный на основе XML язык разметки AdMarkup и большое количество стандартизированных DTD. Система позволяет рекламным агентствам размещать заказы на рекламу через Интернет, предоставляя издателю информацию о тексте и изображениях рекламы, месте размещения рекламы, уникальный номер рекламодателя, предварительную дату устаревания рекламы, а также назначение данного заказа (создание нового реклам-

ного модуля, отмена существующего, обновление или предварительное согласование). Кроме того, система позволяет получать информацию о текущем состоянии счета рекламодателя, специальные требования по размещению рекламы, географические регионы публикации и т.п. Система была с успехом внедрена в большом количестве рекламных агентств, сотрудничающих с The New York Times.

Одним из направлений использования языков разметки является электронная коммерция. Во-первых, необходимость предоставления потенциальному покупателю информации о товаре в структурированном виде сразу наводит на мысль об использовании языков разметки. Так, например, специально созданная библиотека Commerce One's Common Business Library представляет собой набор стандартизованных описаний заказов на покупку, счетов, описаний товаров и графиков поставки. Во-вторых, осуществление финансовых транзакций через Интернет должно основываться на полной стандартизации используемых механизмов. Чтобы помочь развитию в этом направлении, был учрежден проект Bank Internet Payment System, поддерживаемый консорциумом ведущих банков, направленный на поощрение компаний, использующих платежные транзакции через Интернет. Основная часть финансовой информации в системах, создаваемых в рамках данного проекта, передается с использованием структурно размеченных документов.

J.P. Morgan&Co., PricewaterhouseCoopers и IBM ведут совместные разработки по созданию FpML — основанного на XML языка разметки для финансовых применений (XML-based Financial Products Markup Language). Спецификация FpML позволит осуществлять интеграцию широкого спектра финансовых услуг — от электронной торговли до анализа рисков.

Для предоставления информации об объектах недвижимости и осуществления сделок на рынке недвижимости OpenMLS и 4thWORLD Telecom разработали набор DTD, помогающий стандартизировать информационный обмен между покупателем и продавцом. Этот набор включает Commercial DTD, Vacant Land DTD, Working Land DTD и Residential Listing DTD.

Большое количество проектов ведется в научных кругах. Например, для осуществления обмена стандартизованными данными на основе XML в астрономии был создан язык разметки AML (Astronomical Markup Language). AML поддерживает работу со следующими объектами: статьи, таблицы, наборы таблиц, изображения астрономических объектов, персоналии. Это означает, что все эти объекты могут быть описаны с помощью единого языка разметки, что облегчает установление связей между объектами и создание программных продуктов, поддерживающих все перечисленные объекты в рамках единого пользовательского интерфейса. Другим примером является создание математического языка разметки MathML (Mathematical Markup Language).

7. Создание Jet Info Online

В качестве примера использования структурного языка разметки рассмотрим проект создания Web-версии информационного бюллетеня Jet Info — Jet Info online.

7.1. Постановка задачи

Идея создания Web-версии технического издания, разумеется, не нова. Однако реализация этой идеи оказалась не совсем простым делом в связи с задачами, которые были поставлены перед участниками проекта.

Во-первых, необходимо было обработать весь накопленный материал (а Jet Info издается с 1993 года) и привести его к единому виду.

Во-вторых, требовалось реализовать функциональные модули электронного издания, отвечающие за контекстный и атрибутный поиск, организацию тематической и хронологической иерархии материала. Большого размера статьи и разделы должны быть представлены в удобном для «пролистывания» виде.

В-третьих, предстояло создать единый дизайн журнала.

И, наконец, в проекте должна быть реализована технология, позволяющая минимизировать расходы на сопровождение журнала.

Все поставленные задачи требовалось решить в сжатые сроки, поэтому возникла еще одна организационно-техническая задача — распараллеливание работы участников проекта. Это означало, что стандартный подход к реализации подобных проектов — создание и утверждение дизайна, написание функциональных модулей и лишь затем HTML-разметка документов — не подходил. Наиболее объемную часть проекта — разметку накопленного материала — необходимо было начать как можно раньше.

Было принято естественное решение — разделить вопросы HTML-представления (дизайн издания) и структурную разметку материала, используя для последней SGML-средства. Страницы, передаваемые Web-навигатору, должны порождаться по определенным правилам путем преобразования исходных SGML-документов.

7.2. Шаг за шагом

Выбор средств структурной разметки предопределил построение всей дальнейшей работы над проектом. Все было осуществлено в рамках традиционного использования SGML-технологии:

- выбирается DTD, который определяется типом размечаемых документов;
- исходные тексты размечаются в соответствии с выбранным DTD;
- корректность осуществляемой разметки контролируется с помощью стандартных SGML-анализаторов;
- параллельно определяется дизайн издания;
- происходит выбор механизма преобразования (процессора) и стиля преобразования SGML-документов в HTML-формат;
- реализуется преобразование структурно размеченных документов к HTML-представлению.

Выбранный подход, опирающийся на SGML-технологию имеет ряд важных достоинств.

Во-первых, структурная разметка имеет достаточно формальный характер и позволяет описывать все компоненты документов наиболее естественным образом. Статья может быть обозначена маркером article, автор — author, термин — term и т.д. Кроме того, корректность размеченного документа может быть автоматически проверена с помощью программ-анализаторов, что дает первый рубеж контроля качества разметки.

Во-вторых, преобразование документов в формат HTML производится программно. Следовательно, качество внешнего представления не зависит от художественных способностей «разметчиков». К тому же, внешним представлением можно управлять с помощью стилей.

В-третьих, в результате структурной разметки получается полноценный электронный архив документов, имеющих формальную структуру и пригодных для любого автоматического преобразования, в том числе и для преобразования к виду, близкому к требуемому типографией при издании печатной версии.

Наконец, решающим соображением в пользу использования SGML-технологии стало то, что при таком подходе можно формализовать всю технологическую цепочку публикации. При этом существенны как доступность средств реализации всех операций, так и наличие опыта работы с SGML у участников проекта.

Поскольку решаемая задача и используемые подходы являются типичными, полезно проследить, как проходила работа по реализации проекта.

7.2.1. Выбор DTD

Выбор DTD имеет очень большое значение для всей последующей работы. Неудачный выбор DTD может привести к тому, что семантика документов будет неадекватно отражаться в их структуре, разметка будет неочевидной, искусственной, а преобразование в форматы для внешнего представления — некачественным и негибким.

При выборе DTD для Jet Info Online рассматривалось два варианта:

- создание собственного DTD;
- выбор из DocBook и TEILite.

В нашем случае на создание собственного DTD не было ни времени, ни сил. DTD легко и просто создавать тогда, когда документы имеют простую структуру, которая к тому же является фиксированной и в дальнейшем не будет претерпевать изменений. «К сожалению», Jet Info — издание развивающееся, поэтому предусмотреть характер будущих публикаций и составляющих их элементов невозможно.

Статьи, публикуемые в Jet Info, имеют весьма развитую структуру. В них присутствуют многоуровневые разделы, приложения, библиографии, эпиграфы, листинги, списки терминов, рисунки, таблицы и многое другое. Естественно, хотелось, чтобы вся эта структура была отражена без потери семантики.

Не стоит забывать и о том, что, если встать на путь создания собственного DTD, придется разрабатывать новые стили для различных способов отображения структурированной информации, что займет немало времени.

В силу перечисленных причин было принято решение сориентироваться на один из двух наиболее часто используемых DTD общего назначения — DocBook и TEILite. Они длительное время применяются в компании «Инфосистемы Джет» для подготовки различных документов, для них существуют процессоры и стили, как стандартные, так и специально разработанные для нужд Компании.

Было учтено, что DocBook больше ориентирован на документы технического характера, а TEILite — на художественную прозу. К тому же личные симпатии большинства участников проекта были на стороне DocBook. Все это и предопределило сделанный выбор.

7.2.2. Стиль и процессор

Выбрав DocBook в качестве DTD, мы практически лишились вариантов при выборе стиля и процессора, так как разработка собственных стилей для DocBook — задача весьма трудоемкая. К счастью, существует готовый, качественный и сопровождаемый автором стиль — Modular DocBook StyleSheet [11]. Этот стиль реализован с помощью языка спецификации стиля и семантики документов (DSSSL). Он позволяет преобразовывать исходные документы как в HTML, так и в TeX или RTF, что дает дополнительную возможность получения качественного представления бумажных копий размеченного материала.

В качестве процессора был выбран Jade — свободно распространяемая реализация DSSSL.

```
<article id="article1.1.2000"
arch="corp-systems">
 <artheader>
  <title>Современное состояние языков и
  средств разметки документов </title>
  <authorgroup>
   <author>
    <surname>Дуров</surname>
    <firstname>Илья</firstname>
   </author>
  </authorgroup>
 </artheader>
 <sect1>
  <title>Введение</title>
  <para>Так уж сложилось, что большую
  часть информации человек
  предпочитает хранить в виде
  документов...</para>
```

Листинг 9. Пример разметки в соответствии с DocBook DTD.

```
<html version="-//W3C//DTD HTML 3.2
Final//EN">
 <head>
  <title>Современное состояние языков и
  средств разметки документов</title>
 </head>
 <body text="black" bgcolor="white">
...
 <hr>
  <div class="article">
   <h1 class="title">Современное
   состояние языков и средств
   разметки</h1>
   <i>Дуров Илья</i>
   <br>
   <hr>
...
  </div>
  <h1 class="sect1">
  <a name="aen8">Введение</a></h1>
  <p>Так уж сложилось, что большую
  часть информации человек
  предпочитает хранить в виде
  документов...</p>
```

Листинг 10. Пример HTML-разметки после преобразования.

7.2.3. Разметка материалов

На первом этапе была выполнена рутинная работа по «извлечению» текста из внутреннего формата существующих электронных версий статей и набор тех материалов, которые сохранились только в печатном виде.

Затем была проведена разметка полученных текстовых файлов в соответствии с DocBook DTD. В качестве примера на листинге 9 представлен размеченный текст начала нашей статьи.

При этом оказалось, что некоторые документы, например, официальные Руководящие документы Гостехкомиссии России, не укладываются в выбранную модель. Для преодоления этой трудности была проведена модификация DocBook, который имеет соответствующие встроенные возможности. Так, например, был добавлен новый тип элемента **"clause"**, который представляет собой нумерованную статью в разделе.

После осуществления разметки, с помощью Jade и Modular DocBook StyleSheet было произведено преобразование материалов в HTML. В качестве примера на листинге 10 представлен преобразованный в HTML текст начала нашей статьи (для экономии места из листинга исключены разметка внешнего представления верхней части HTML-страницы и разметка оглавления статьи).

На этом этапе выявлялись смысловые ошибки, формальные ошибки разметки (несоответствия DTD) и недостатки во внешнем представлении документов. Постепенно, шаг за шагом, большинство из них было исправлено.

7.2.4. Преодоление трудностей

Не обошлось и без некоторых трудностей. В первую очередь, к ним можно отнести фиксированную структуру получаемого журнала, которую диктует DocBook. Дело в том, что в рамках используемого DTD статьи можно объединить в книги, которые, в свою очередь, можно объединить в наборы книг (Bookset). Это, с одной стороны, удобно, так как возможно оставаться в рамках одного DTD и реализовывать необходимую иерархию материалов. С другой стороны, используемый стиль, хотя и обладает большим количеством настраиваемых параметров (что позволяет управлять внешним видом получаемых HTML-документов) все же недостаточно гибок для того, чтобы, используя упомянутую иерархию, получить традиционный, привычного вида электронный журнал.

Один из вариантов решения этой проблемы заключался в том, чтобы на верхних уровнях иерархии отказаться от DocBook в пользу HTML, обеспечивая для них традиционный внешний вид при сохранении всех преимуществ структурной разметки и автоматического форматирования содержательных материалов.

Технически это было реализовано в виде набора HTML-шаблонов, в которые с помощью специальных (не входящих в HTML) элементов и атрибутов вставляются ссылки на размеченные статьи.

Для обработки этих шаблонов был разработан специальный стиль, назначение которого состояло в том, чтобы, оставляя неизменным содержание шаблона, обрабатывать ссылки на статьи и подставлять вместо них правильные ссылки на HTML-вер-

сии статей. С точки зрения SGML-технологий — это стандартный прием включения в основной документ подчиненных документов, имеющих, возможно, другой DTD.

7.3. Некоторые итоги

Опыт осуществления проекта Jet Info Online показывает, что выбранная SGML-технология хорошо подходит как для Web-публикации большого количества накопленного материала, так и для организации работы по подготовке новых статей к публикации и типографским способом, и в WWW.

Использование набора HTML-шаблонов позволяет при необходимости изменять дизайн Web-издания, заменив лишь несколько файлов и не затрагивая самого сложного — технологии подготовки материалов.

Использование структурной разметки позволяет создать полноценный архив документов — текстовую базу данных. Как следствие, создание подходящей поисковой машины и встраивание ее в Web-издание становится делом несложным. Ее возможности не ограничиваются только контекстным поиском. Можно осуществлять поиск по таким параметрам, как автор, тема, год издания. Причем, перечень параметров поиска в дальнейшем может быть легко расширен.

Сопровождение Web-издания из искусства превращается в рутинное занятие, благодаря тому, что формирование публикуемой электронной версии по размеченному документу осуществляется автоматически. Также автоматически оказываются «дополнительные услуги»: первые абзацы новых статей выносятся на корневую страницу, соответствующим образом модифицируются тематический и хронологический каталоги.

Объективную, взвешенную оценку того, что получилось в Jet Info Online, а что нет, насколько стабильной окажется выбранная технология, сможет дать только время. Пока все идет по плану.

8. Заключение

История развития языков разметки наглядно демонстрирует стремление найти золотую середину между желаниями и возможностями.

Общность и мощь SGML не только предоставляют широкие возможности для структурной разметки документов, но и создают значительные трудности при реализации приложений и, как следствие, использования языка разметки в Интернет. Напротив, HTML идеально приспособлен для работы в Интернет, но его простота и легкость не позволяют в полной мере реализовать всю необходимую на сегодняшний день функциональность.

Разработка XML стала важным этапом поиска разумного сочетания полноты структурного описания и широких возможностей создания прикладных программ на его основе. Окажется ли этот язык разметки искомой золотой серединой? На наш взгляд, ответ должен быть положительным, хотя истинным судьей является только время.

9. Литература

1. ISO 8879. Information Processing-Text and Office Systems. Standard Generalized Markup Language SGML, 1986.
2. Lie H., Saarela J. Multipurpose Web Publishing — Communications of the ACM, vol. 42. No. 10, October 1999.
3. Extensible Markup Language (XML) 1.0. W3C Recommendation — The World Wide Web Consortium, 10 February 1998 — http://www.w3c.org/TR/1998/REC-xml-19980210.
4. Walsh N. What is XML? — XML.com, 8 октября 1998 — http://www.xml.com/pub/1998/10/guide0.html.
5. Garshol L. Introduction to XML. — http://www.stud.ifi.uio.no/~lmariusg/download/xml/xml_eng.html.
6. Cover R. XML Linking and Addressing Languages (XPath, XPointer, XLink) — http://www.oasis-open.org.
7. Walsh N. XSL. The Extensible Style Language. Styling XML Documents — Webtechniques.com, январь 1999 — http://www.webtechniques.com/archives/1999/01/walsh/.
8. Walsh N. The XSL Debate: One Expert's View — XML.com, 8 июня 1999 — http://www.xml.com/xml/pub/1999/06/xml_edit.html.
9. 9. Brogden W., Tittel E. Arbortext Adept 8 Editor Review. — XML.com, 1999. http://www.xml.com/pub/1999/09/adept/AdeptRvw.htm.
10. Материалы о продукции фирмы SoftQuad. — http://www.softquad.com.
11. Материалы о Modular DocBook StyleSheets. — http://nwalsh.com/docbook/dsssl/index.html.
12. Reinhold M. An XML Data-Binding Facility for the Java Platform. — Sun Micorsystems, 1999. http://java.sun.com/xml/docs/bind.pdf.

Jet Info
ИНФОРМАЦИОННЫЙ БЮЛЛЕТЕНЬ
Издается с 1995 года
Издатель: компания Джет Инфо Паблишер

Главный редактор: Галатенко В.А. *(galat@jet.msk.su)*
Технический редактор: Антонов А.Н. *(silver@jet.msk.su)*

Россия, 103006, Москва, Краснопролетарская, 6
тел. (095) 972 11 82, 972 13 32
факс (095) 972 07 91
e-mail: JetInfo@jet.msk.su, http://www.jetinfo.ru

Подписной индекс по каталогу Роспечати **32555**