Разработка технической документации с помощью DocBook/XML. Принцип единого источника

Обзор

Михаил Острогорский

Разработка технической документации с помощью DocBook/XML. Принцип единого источника: Обзор

by Михаил Острогорский

Аннотация

Предлагаемый обзор адресован в первую очередь руководителям, выбирающим инструментарий для разработки технической документации в своих организациях или структурных подразделениях. Он дает общее представление о принципе единого источника применительно к задачам документирования и показывает, как именно этот принцип реализуется средствами технологической платформы DocBook/XML. В обзоре отражен опыт, приобретенный компанией PhiloSoft Technical Communication при выполнении ряда крупных проектов по документированию, необходимо учитывать, что практика других организаций может сильно отличаться от описываемой здесь. Обзор также будет интересен техническим писателям и другим специалистам, участвующим в документировании программных средств, автоматизированных систем и компьютерной техники.

Содержание

1 Цели, задачи и компромиссы 5
2 Принцип единого источника 8
2.1 Основные понятия и определения 8
2.2 Содержание, структура, оформление 9
2.3 Единый источник как база знаний 10
3 Технологическая платформа DocBook/XML 13
3.1 Языки разметки 13
3.1.1 Мир XML 13
3.1.2 Язык разметки DocBook/XML 14
3.1.3 Спецификация XInclude. Языки XPointer и XPath 14
3.1.4 Язык описания графов пакета GRAPHVIZ 14
3.1.5 Открытость языков разметки 15
3.2 Инструментарий и порядок создания документов 15
3.2.1 Редакторы исходных документов 15
3.2.2 XSLT-процессоры и XSLT-стили 17
3.2.3 Стандартные стили DocBook XSL 18
3.2.4 Типовые преобразования 18
4 Прикладные решения 20
4.1 Документирование корпоративной информационной системы 20
4.2 Документирование комплектуемого решения 22
Литература 25

Список иллюстраций

1 Формирование конечного документа 8
2 Цепочка преобразований 8
3 Составляющие документов 10
4 Профилирование 11
5 Редактор XMLmind 16
6 Редактор <oXygen/> 17
7 Преобразования DocBook/XML в HTML, CHM и PDF 18
8 Документирование автоматизированной системы 22
9 Формирование руководства пользователя на компьютер 24

1 Цели, задачи и компромиссы

Технология *единого источника* нужна для того, чтобы выпускать качественную техническую документацию, даже если она достаточно сложно устроена, или ее объем достаточно велик. Теоретически любой комплект документов можно разрабатывать и сопровождать с помощью текстового процессора, т.е. по современным меркам вручную. Поручив это квалифицированным и добросовестным специалистам, мы избавимся от проблем с качеством, во всяком случае, до тех пор, пока у исполнителей будет хватать времени на контроль качества: тщательную проверку всех результатов, правку и повторную проверку. На практике времени обычно не хватает. Дело не только в том, что конкуренция, бюджетные ограничения, давление со стороны заказчика и другие внешние обстоятельства вынуждают разработчиков выдерживать сжатые сроки, в конце концов, изыскивать ресурсы — задача управленца. Подготовка текста требует ресурсов, которые не раздобыть ни одному менеджеру. Ограничен ресурс человеческого внимания, поэтому с ростом объема документов и количества взаимосвязей между ними затраты на проверку возрастают нелинейно. Люди не телепаты[1], поэтому совместная работа нескольких авторов приводит к разнобою в документах. Вопреки стараниями разработчиков в техническую документацию закрадываются ошибки. Нарушаются правила оформления, одни и те же сведения в разных документах излагаются по-разному, документы обновляются несвоевременно или обновленные части перемешиваются с устаревшими. Иными словами, теряется *устойчивость* качества: к нему можно стремиться, но за него невозможно ручаться. Особенно остро это проявляется при сопровождении больших комплектов технической документации.

Первая мера, на которую идут, чтобы все-таки удержать качество на должном уровне — упрощение комплекта. Если не получается эффективно исправлять ошибки, то надо сузить их «среду обитания». Всякое дублирование информации запрещается. Количество перекрестных ссылок сводится к минимуму. От информационно-поискового аппарата в линейных документах и навигации в гипертекстовых отказываются, ограничиваясь оглавлением. Это выход из положения, но плохой, потому что расплачивается за него пользователь. Он получает формально качественные документы, работать с которыми ему неудобно.

Важно

Теперь необходимо сделать две важные оговорки. *Большим* объемом будем называть такой, при котором моральные и профессиональные достоинства разработчиков перестают гарантировать качество результатов их труда. Здесь мы предполагаем, что с ростом объема текста и его структурной сложности (которая характеризуется количеством перекрестных ссылок, повторений, устойчивых формулировок, терминов, упоминаемых названий), любой проект рано или поздно перейдет эту грань. *Качество* будем сводить к набору измеримых показателей, для каждого из которых можно установить минимальный порог, и если он не достигнут, то документ (или комплект) считается неудовлетворительным. Например, все списки должны иметь быть оформлены с определенным отступом от левого поля, сведения только для службы тех. поддержки никогда не должны включаться в документацию конечного пользователя, выводимые программой сообщения должны цитироваться в документации точно и т.п. Измерять стилистику или логичность изложения мы не умеем, таким образом, на самом деле речь идет в важных, но не исчерпывающих характеристиках качества технической документации.

В противоположность упрощению комплекта подход единого источника предполагает автоматизацию труда разработчика, а не отказ от ручных операций заодно с их полезными результатами. Отлаженный (!) автомат не совершает случайных ошибок, свой-

[1] Телепаты, если они существуют, вряд ли занимаются разработкой технической документации.

ственных людям, поэтому проверять, править и перепроверять сгенерированные им документы не нужно вообще. Правда, современные автоматы не в состоянии превзойти даже человека средних способностей, посредственный верстальщик верстает лучше, чем самый совершенный автомат. В каком смысле лучше? Адекватнее нашему восприятию, естественнее. Автомат соблюдает правила всегда, человек же нарушает их, заметив, что здесь и сейчас они действуют против тех целей, на достижение которых изначально направлены. Скажем, особенности конкретной полосы таковы, что бездумное применение некоторого правила типографики делает ее менее эстетичной. Итак, первая цель — устойчивость качества, первый компромисс — согласие не известную формальность результата ради устранения формальных же ошибок.

Часто говорят, что единый источник ускоряет и удешевляет разработку технической документации. Спорное устверждение, ибо за ускорение и удешевление одних операций мы расплачиваемся появлением дополнительных. В любом случае это не самостоятельные его преимущества, а другое выражение устойчивости качества. Неудовлетворительная документация никому не нужна даже мгновенно и бесплатно, а комфорт исполнителя сам по себе, как правило, не является целью проекта или организации. Ускорение и удешевление достигаются по сравнению с непредсказуемо долгим исправлением ошибок при ручной работе над большим комплектом. Очевидно, разработка обозримого (не «большого» в нашем смысле) комплекта традиционными средствами обходится дешевле, чем с использованием сложной техники. Отсюда второй компромисс — согласие на управляемый рост затрат в начале проекта ради предотвращения неуправляемого в конце.

Автоматизация разработки технической документации вызвана к жизни борьбой с проблемами, которые поражают любой крупный проект. Вместе с тем она открывает возможности, которые позволяют ставить новые позитивные цели, как то:

- повышение информативности технической документации;
- сокращение сроков доставки актуальной технической документации пользователям;
- предоставление пользователям инструмента для фиксации своего опыта или установившейся практики работы.

Технологии единого источника позволяют решать следующие задачи:

- *Автоматизация оформления документов по заданному макету или стандарту.* Предполагается как публикация серии документов по одному хорошо отлаженному макету, так и публикация одного документа по нескольким макетам, допустим, руководство пользователя оформляется по ГОСТу для поставки гос. заказчику и в соответствии с корпоративным стандартом для выпуска продукта в розницу. Документы как таковые и описания макетов при этом создаются и хранятся раздельно.
- *Публикация документа в разных электронных форматах.* Описания продуктов размещаются на веб-сайте в формате HTML и передаются типографии для тиражирования в формате PostScript. Многоплатформенное решение может комплектоваться документацией в форматах, специфичных для разных сред: CHM для Microsoft Windows и man pages для Unix. Другой типичный случай — создание руководства пользователя и хелп-файла на основе одного и того же текста.
- *Автоматизация компоновки документов.* Устранение физического дублирования текста при сохранении его вхождений в разные документы. Условный текст, в том числе, *профилирование* текста по аудитории, платформам, операционным системам. Автоматизированное формирование структур документов.
- *Создание функциональных электронных документов.* В основном это веб-публикации с развитыми средствами навигации и поиска, базы знаний наподобие MSDN.

- *Формирование информационно-поискового аппарата.* К информационно-поисковому аппарату относятся оглавления, всевозможные указатели, перекрестные ссылки и т.п. В комплект, состоящий из нескольких документов, полезно включать объединенное оглавление и объединенный предметный указатель. Также бывают необходимы перекрестные ссылки между документами. Реализация этих возможностей средствами обычных текстовых процессоров отнимала бы у авторов слишком много времени.

- *Интеграция документирования с разработкой.* Будучи связаны административно, эти процессы нередко полностью изолированы друг от друга технологически. Управление требованиями, управление конфигурациями, версионный контроль, трассировка ошибок для технической документации и документируемого решения выполняются параллельно. Дублирование функций неэкономично и неудобно. Кроме того, документы и программы могут иметь общие компоненты, например, тексты сообщений, которые и отображаются в интерфейсе пользователя, и упоминаются в документации. Очевидно, разработчики могут время от времени редактировать их, поэтому, если не сделать соответствующий ресурс общим, то придется постоянно сверять документацию с программой.

2 Принцип единого источника

2.1 Основные понятия и определения

Конечным будем называть электронный документ, который поставляется пользователю. Случай, когда пользователь работает с твердой копией, даже если это книга, отпечатанная в типографии, не рассматривается как отдельный, потому что твердая копия производится на основе электронного документа. Формат файла конечного документа называется *целевым*. Внешний вид, графическое оформление конечного документа будем называть *макетом* [1]. Набор правил построения макета называется *принципиальным макетом*. Свойства принципиального макета и технические свойства конечного документа, связанные с целевым форматом (например, средства навигации в PDF-файле), обобщенно будем называть *оформлением*.

Взятую отдельно смысловую часть документа, т.е. конечный документ за исключением оформления, будем называть *исходным документом*. Если исходный документ и данные об оформлении хранятся раздельно, то конечный документ формируется в результате определенного *преобразования* (рис. 1).

Рисунок 1. Формирование конечного документа

Преобразование может представлять собой *цепочку* преобразований, выполняемых друг за другом. Тогда каждое из них кроме последнего формирует *промежуточный* документ, который подается на вход следующему в цепочке (рис. 2). При этом каждое преобразование может использовать относящуюся именно к нему часть данных об оформлении.

Рисунок 2. Цепочка преобразований

Большие объемы технической документации разрабатываются не в одиночку и не сразу. Поэтому единый источник не сводится к чистой технологии, это еще и определенный способ организации работы специалистов, участвующих в документировании. Всех, кто составляет текст технической документации, будем называть *авторами*. При этом, говоря о тексте, будем иметь в виду и собственно текст, и относящиеся к нему материалы, в том числе, рисунки. Авторами могут быть не только профессиональные технические писатели, но и специалисты, для которых документирование только одна из их функций. Принципиальный макет либо создается *дизайнером*, либо продиктован стандартом, скажем, ГОСТ 2.105-95 [1]. Технически оформление реализуется *програм-*

[1]Вообще оформление документа не обязательно должно быть графическим. Не секрет, что существуют синтезаторы речи, позволяющие озвучивать текст, и легко представить себе случаи, когда это полезно.

мистом, который умеет описывать принципиальные макеты тем способом, который «понятен» программам, выполняющим преобразования. Автора, ответственного за успех подготовки документов, будем называть *редактором* (он руководит авторами, но о его функциях и полномочиях разговор отдельный). Мы предполагаем, что документирование проводится в рамках некоторого *проекта*. У проекта обязательно есть *заказчик*, он может быть внутренним или внешним по отношению ко всей организации, но по отношению к самому проекту он в любом случае внешний. Заказчик диктует *требования* к технической документации, причем некоторые из них, например, соответствие документов определенному стандарту, подлежат безусловному выполнению. «Бесконечную» деятельность по развитию некоторого технического решения (автоматизированной системы, программного продукта) будем рассматривать как серию проектов.

2.2 Содержание, структура, оформление

Единый источник вне зависимости от его технической реализации содержит все *фрагменты*, которые могут быть включены в разрабатываемые документы. Объем фрагмента определяется его предполагаемой функцией, фрагментом может быть целая глава, а может ячейка таблицы. Обычно от фрагментов, несущих одну функцию, требуют единообразной структуры, например, все функции API должны быть описаны по одному плану. Для каждого фрагмента в едином источнике должна храниться ровно одна копия. Фрагмент обязательно снабжается *уникальным идентификатором*, по которому к нему можно получить доступ. Допускается выделение внутри фрагмента других фрагментов. Последнее не означает, что фрагмент, содержащий другие фрагменты, целиком из них состоит, просто его частям, представляющим самостоятельный интерес, присваиваются уникальные идентификаторы. В крупных проектах фрагменты чаще образуют лес, а не дерево. Последовательность разработки фрагментов и их загрузки в единый источник не имеет значения. Не существенен и способ хранения фрагмента, такие детали, как число и взаимное расположение XML-файлов, которыми представлен фрагмент, должны касаться только его автора.

Шаблон документа задает структуру его разделов и состав включаемых в него фрагментов. Кроме заголовков и метаданных шаблон не содержит собственного текста. Его место занимают *включающие ссылки* на фрагменты единого источника. Включающая ссылка может явно указывать на фрагмент с определенным значением уникального идентификатора или содержать запрос на выборку фрагментов, удовлетворяющих определенным условиям. Во избежание терминологической путаницы напомним, что применительно к текстовым процессорам шаблоном обычно называют образец оформления любых документов некоторого типа, например, в поставку Microsoft Word входят шаблоны писем, резюме, служебных записок и даже технической документации. В нашем случае у каждого конкретного документа должен быть свой шаблон, причем к оформлению он как раз не имеет отношения.

Шаблон и все фрагменты, на которые он прямо или косвенно ссылается, составляют исходный документ.

Замечание

В общем случае структура исходного документа «глубже» структуры шаблона, потому что любой фрагмент может иметь собственную структуру.

Таким образом, каждый документ можно разложить на три обязательных составляющих: содержание, структуру и оформление (рис. 3).

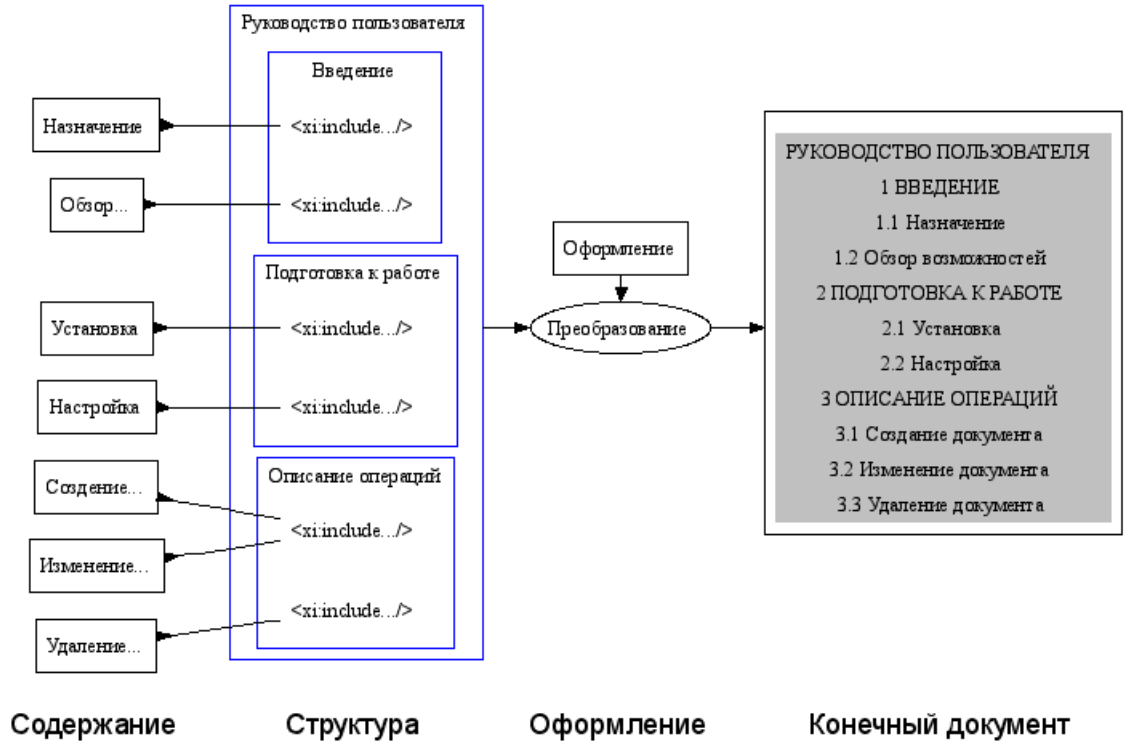

Рисунок 3. Составляющие документов

Назвать эти составляющие абсолютно независимыми друг от друга было бы преувеличением, но выделить разработку каждой из них в самостоятельную задачу внутри проекта, как правило, удается. Что это дает?

- Содержание и структура изолированы от оформления. Их можно разрабатывать параллельно, сокращая критический путь проекта. Вдобавок модификация одной из этих составляющих никогда не приводят к сбоям в другой: авторы заведомо не испортят оформление, а «оформитель» текст.
- Разработка содержания легко распределяется между несколькими авторами. При обновлении фрагментов не требуется повторно вставлять их в исходный документ. Если фрагмент входит в несколько исходных документов, то расхождения между последними по этому фрагменту заведомо исключены.
- Полученные результаты можно использовать многократно. Из одних и тех же выверенных фрагментов формировать документы, предусмотренные ГОСТами, RUP или MSF, одни и те же документы оформлять по-разному и распространять в разных форматах, одни и те же отлаженные правила оформления использовать для разных документов и даже в разных проектах и т.д.

В документировании, как и в программировании, изоляция участков работы и повторное использование отлаженных результатов обеспечивают устойчивость качества.

2.3 Единый источник как база знаний

Атрибутизация фрагментов и создание механизмов их поиска по значениям *полей* позволяет рассматривать единый источник как базу данных. Тогда конечный документ можно сравнить с отчетом, а шаблон документа — с формой отчета. Поля могут содержать данные об аудитории, теме или состоянии фрагмента. Например, фрагмент помечают как адресованный только квалифицированным или только начинающим пользо-

вателям, относящийся только к версии продукта для той или иной платформы, завершенный или незавершенный. Также в поля можно выносить важные сведения о предмете описания, разумеется, когда их удается формализовать. Допустим, если фрагмент говорит о некоторой функции автоматизированной системы, то в специальное поле можно поместить список пользовательских ролей, которые к этой функции обращаются.

Атрибутизация фрагментов позволяет:

- профилировать документы по разным признакам;
- организовать проблемно-ориентированный поиск;
- организовать проблемно-ориентированную навигацию.

Профилированием называется автоматический отсев фрагментов по заданным признакам при формировании конечного документа (рис. 4). Обычно профилирование применяется, когда необходимо подготовить несколько редакций одного и того же документа для разных аудиторий, причем различия между редакциями заключаются во множестве небольших по объему фрагментах, касающихся важных деталей. Например, руководство пользователя адресовано и службе технической поддержки, и конечным пользователям, но последних лучше не знакомить с особенностями защиты от копирования или редко возникающими проблемами. Другой типичный случай профилирования — руководства по приложениям, имеющим параллельные версии для разных аппаратных платформ и операционных систем. Встречается профилирование по последовательным версиям самого документируемого технического решения, если они выпускаются часто и отличаются друг от друга незначительно.

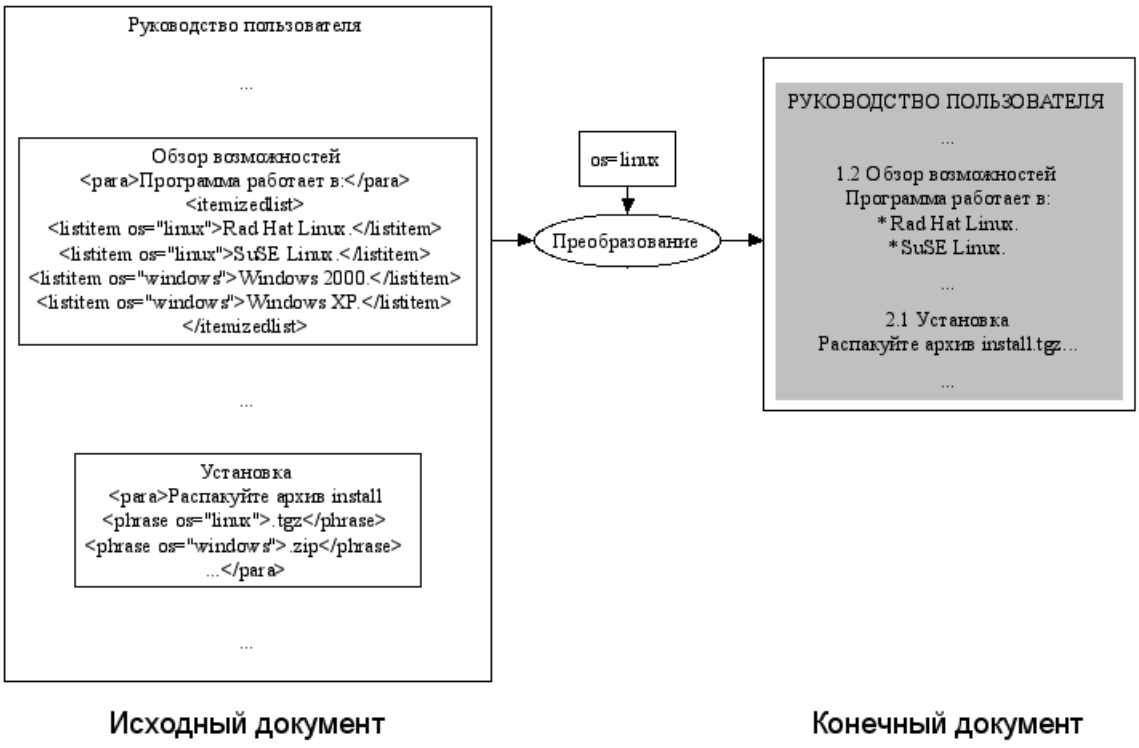

Рисунок 4. Профилирование

Проблемно-ориентированный поиск обеспечивает пользователю возможность быстро ориентироваться во взаимосвязях между частями или свойствами технического решения. Распространенный пример проблемно-ориентированного поиска — размещаемый в конце каждого раздела список перекрестных ссылок на ассоциативно связанные раз-

делы, обычно он имеет подзаголовок «См. также». Такие списки могут формироваться авторами вручную, а могут автоматически по общим для связываемых разделов ключевым словам. К средствам проблемно-ориентированного поиска мы относим указатели специального вида, предположим, указатель функций API в руководстве программиста или указатель документов в руководстве по ERP-системе.

Проблемно-ориентированная навигация не только позволяет быстро найти нужные сведения, но и сама по себе информативна. Если техническое решение состоит из большого количества компонент, то можно построить схему, иллюстрирующую характер взаимосвязей между ними, допустим, дерево зависимостей или диаграмму потоков данных, снабдив каждый элемент этой схемы гипертекстовой ссылкой на соответствующий раздел. Если система автоматизирует какой-то бизнес-процесс, то можно построить диаграмму бизнес-процесса, разметив ее гипертекстовыми ссылками на инструкции по выполнению отдельных операций.

Перечисленные возможности в совокупности позволяют обращаться с единым источником как с базой знаний о техническом решении. Поддержание полноты и актуальности базы знаний выходит на первый план, а формирование конечных документов становится вспомогательной задачей, быстро решаемой в случае необходимости.

3 Технологическая платформа DocBook/XML

3.1 Языки разметки

3.1.1 Мир XML

Об XML-технологиях написаны библиотеки учебников и монографий [6, 4, 5], не счесть посвященных им статей в периодике и ресурсов в Интернете. Здесь мы расскажем о них предельно кратко для тех, кто совсем не представляет себе, что это такое, а также чтобы далее не засорять текст ссылками на известные факты.

Проблема унификации форматов данных, вероятно, возникла вместе с первыми ЭВМ, это «родовая травма» информационных технологий. Противоборствуют две тенденции: с одной стороны, каждая задача требует средств обработки и хранения специфичных для нее данных, с другой стороны, необходимость обмена данными требует эти средства унифицировать. Поиски разумного компромисса на этом поле время от времени приводили к появлению более или менее удачных универсальных форматов. В основном они предназначались для представления данных определенного характера: графики, текста, звука, географических карт, баз данных и т.п. Поскольку разработчики действовали разрозненно, форматы не только получались разными, но и в основе имели разные принципы организации данных. Приходилось для каждого формата разрабатывать отдельные механизмы обработки, что неэкономично само по себе и вдвойне неэкономично, если в одном приложении обрабатываются данные разных форматов.

Вряд ли кто-то строил иллюзии относительно возможности разработки универсального, энциклопедического формата, пригодного для представления любых мыслимых данных. Выходом из положения виделось создание абстрактного формата форматов, который позволял бы описывать сколь угодно специфичные форматы, но такие, чтобы синтаксическое родство между ними было достаточно сильно и допускало реализацию общих механизмов обработки. В качестве решения этой задачи в 1989 г. был предложен язык *Standard Generalized Markup Language (SGML)*, а позже, в 1998 г., более простой и технологичный *eXtensible Markup Language (XML)* [1].

Не вдаваясь в подробности, скажем, что XML — это язык описания языков разметки с общим синтаксисом и разными словарями. Свобода словаря позволяет создавать конкретные специализированные XML-языки, а общность синтаксиса обрабатывать произвольные XML-данные одними и теми же программами. Если вы изучаете лягушек, то вам никто не мешает придумать специальный формат структурированного представления информации о них. Для ввода данных вы сможете использовать любой понравившейся вам XML-редактор, а для публикации на своем веб-сайте любой XSLT-процессор. Одновременно у вас есть возможность автоматически загружать данные в базу данных и находить там нужные сведения по существенным для предметной области признакам.

Хороший принцип в сочетании с удачной реализацией привели к взрывообразному росту XML-технологий. Количество программ для работы с XML-данными и специалистов, владеющих этой областью, увеличивается ежедневно. Во многие прикладные пакеты встраивается возможность импорта и экспорта XML-данных. Не возьмемся делать прогнозы, но, судя по всему, решениям, основанным на XML, технологическая и кадровая изоляция в обозримом будущем не угрожает.

[1] Вынуждены признаться, что этот абзац написан наугад. Не исключено, что расширяемые форматы данных существовали до SGML, значительное явление редко возникает на пустом месте без затмеваемых им прототипов.

3.1.2 Язык разметки DocBook/XML

Основой технологической платформы DocBook/XML служит одноименный проблемно-ориентированный язык разметки [9]. Он предназначен для записи текста технической документации на программы, алгоритмические языки, компьютерное оборудование и другие решения в области информационных технологий, чем принципиально отличается от большинства форматов хранения текстовых данных (но не XML-языков!). В языке DocBook/XML предусмотрены средства описания фрагментов, свойственных технической документации, например, специальными тегами полагается выделять названия элементов интерфейса, обозначения клавиш, имена переменных, термины, различные врезки (замечания, подсказки, предупреждения), листинги, описания выполняемых пользователем процедур. Разметка, задаваемая языком DocBook/XML, носит преимущественно функциональный характер: автору предписано указывать роль, которую тот или иной фрагмент играет в тексте, а не способ его внешнего оформления. Такой подход сковывает автора, зато позволяет добиться заведомой независимости содержания от оформления конечного документа и унифицировать некоторые важные качества стиля текста при работе нескольких авторов в одном проекте.

Язык разметки DocBook имеет долгую по «компьютерным» меркам историю. Его первая версия, опиравшаяся на SGML, была создана еще в 1991 г. совместными усилиями компаний O'Reilly & Associates[2] [3] и HaL Computer Systems. Версия на базе XML увидела свет в 1998 г., тогда же язык перешел в ведение специально образованного технического комитета консорциума OASIS[4]. В состав этого технического комитета входят всемирно известные компании, в том числе, Hewlett-Packard и IBM. Сегодня язык разметки DocBook/XML применяется разработчиками технической документации во всем мире.

3.1.3 Спецификация XInclude. Языки XPointer и XPath

Язык разметки XInclude [10], основанный на XML, предназначен для создания внутри XML-документа включающих ссылок на другие XML-документы или их узлы. Адресоваться к включаемым документам можно по URI. Для указания на узлы XML-документов используются языки XPointer и XPath [13, 11, 12, 4, 5], позволяющие формулировать запросы к XML-данным. Запросы могут содержать разнообразные условия: навигационные указания, проверки и логические функции. Результатом запроса в общем случае является набор узлов, удовлетворяющих его условиям. Средствами языка XInclude можно также потребовать включения в XML-документ «плоского» текстового файла, что особенно полезно при цитировании листингов.

3.1.4 Язык описания графов пакета GRAPHVIZ

Пакет GRAPHVIZ[5] (проект компании AT&T) интересен тем, что позволяет применить принцип единого источника к графическим схемам. Автор описывает схему на специальном языке разметки, а потом с помощью специального конвертора генерирует графический файл. По синтаксису этот язык напоминает язык программирования Си, но проще и менее строг. Лучше всего он приспособлен для описания графов, так как предлагает оперировать вершинами и соединительными стрелками. Вершинам и стрелкам можно придавать разные формы, предусмотрены средства для взаимного позиционирования вершин.

[2] http://www.oreilly.com
[3] В нашей стране компания O'Railly известна прежде всего серией книг по программированию с фотографиями животных на обложках.
[4] http://www.oasis-open.org
[5] http://www.graphviz.org

3.1.5 Открытость языков разметки

Мы упомянули несколько языков, наиболее интересных и важных с точки зрения настоящего обзора. Многообразие языков разметки невероятно велико. Так, для векторной графики существует язык SVG, для математических формул MathML, для отформатированного текста XSL-FO; перечислять здесь все языки разметки нет ни возможности, ни надобности. Главное, что они складываются в открытую технологию, допускающую решение любой осмысленной (не все-таки не фантастической) задачи за соразмерное время при соразмерных затратах. В этом их преимущество перед проприетарными продуктами, которые могут предоставлять богатые возможности, но не допускать выхода за их пределы. Даже если сегодня их возможности кажутся нам избыточными, нет гарантии, что завтра, через месяц или через год нам не потребуется какая-нибудь важная мелочь, которую разработчики не реализовали и даже не собираются?

3.2 Инструментарий и порядок создания документов

3.2.1 Редакторы исходных документов

Для набора текста исходного документа могут применяться разные программы, от обыкновенного «Блокнота» до развитых XML-редакторов. Поскольку требования к формату исходного документа определяются спецификациями используемых языков разметки, выбор редактора перестает быть важным вопросом, который необходимо решить на уровне проекта. Каждый автор может работать в том редакторе, который ему удобен.

По отношению к задаче набора текста в формате DocBook/XML существующие сегодня редакторы можно разделить на следующие группы:

- обычные текстовые редакторы;
- XML-редакторы с интерфейсом текстового процессора;
- XML-редакторы, упрощающие набор разметки.

Обычные текстовые редакторы позволяют набирать текст, интерпретируемый ими как простая последовательность символов и строк. Все заботы о его структуре перекладываются на плечи автора. Функциональность текстовых редакторов, если она шире, чем у редактора «Блокнот», обычно связана с набором символов в разных кодировках, поиском и заменой подстрок, обработкой нескольких текстовых файлов, а также макросами, ускоряющими ввод часто повторяющихся фрагментов.

Замечание

Среди текстовых редакторов особняком стоит Emacs , пользовательский интерфейс которого не назовешь дружественным, зато средства автоматизации работы с текстом чрезвычайно богаты и гибки.

XML-редакторы с интерфейсом текстового процессора предлагают пользователям, не желающим вникать в синтаксис языков разметки, более-менее привычную среду для ввода и правки текста (рис. 5). Обычно они позволяют редактировать любые XML-документы, но ориентированы на популярные форматы, в том числе, DocBook/XML, XHTML, TEI.

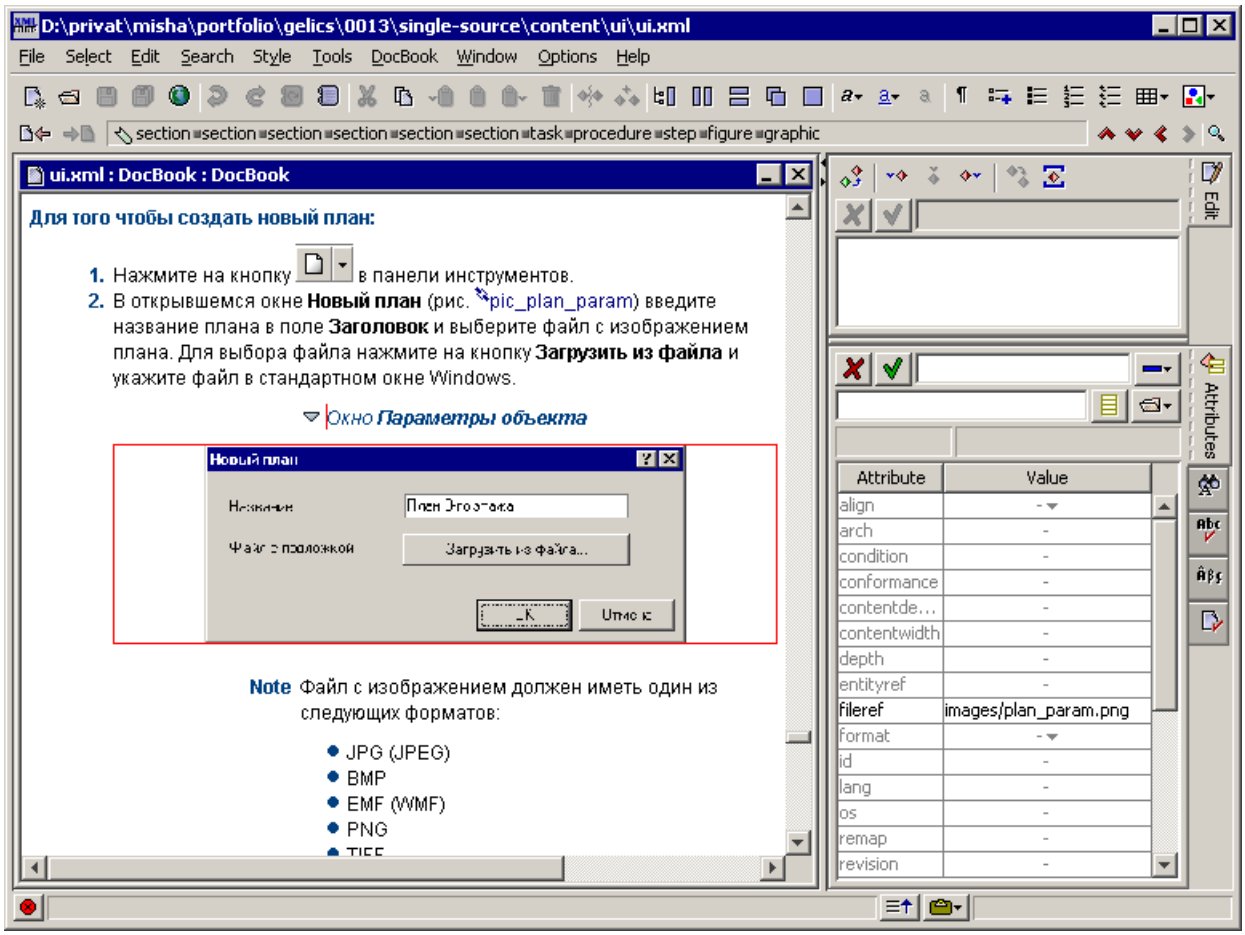

Рисунок 5. Редактор XMLmind[6]

Такие редакторы — большое подспорье для авторов, начинающих пользоватся языками разметки, поскольку автоматически расставляют теги и порождают только хорошо оформленные (или даже валидные)[7] исходные документы, которые сразу без ошибок можно преобразовывать в конечные. Обратная сторона состоит в том, что генерируемую ими разметку сложно контролировать, а делать это приходится, поскольку конверторы, преобразующие исходные документы в конечные, обрабатывают именно XML-файл, а не с изображение текста, которое видит автор. Результат обработки зависит от того, какие в каждом конкретном случае поставлены теги, какие значения атрибутов присвоены узлам документа, и т.п. Наконец, работу опытного автора они скорее замедляют, чем ускоряют.

XML-редакторы, упрощающие набор разметки, интерфейсом напоминают обычные текстовые, но учитывают структуру набираемого документа, заданную тегами (рис. 6).

[6] http://www.xmlmind.com/

[7] *Хорошо оформленным* (*well formed*) называется документ, соответствующий правилам синтаксиса XML. Каждому открывающему тегу в нем должен быть сопоставлен один и только один закрывающий, все значения атрибутов должны быть заключены в кавычки, корневой элемент должен быть единственным и т.п. *Валидным* (*valid*) называется документ, соответствующий правилам определенного XML-языка. Например, правила языка DocBook/XML в числе прочего требуют, чтобы элемент section содержал ровно один элемент title. Строгие определения хорошо оформленного и валидного документа приведены в [6].

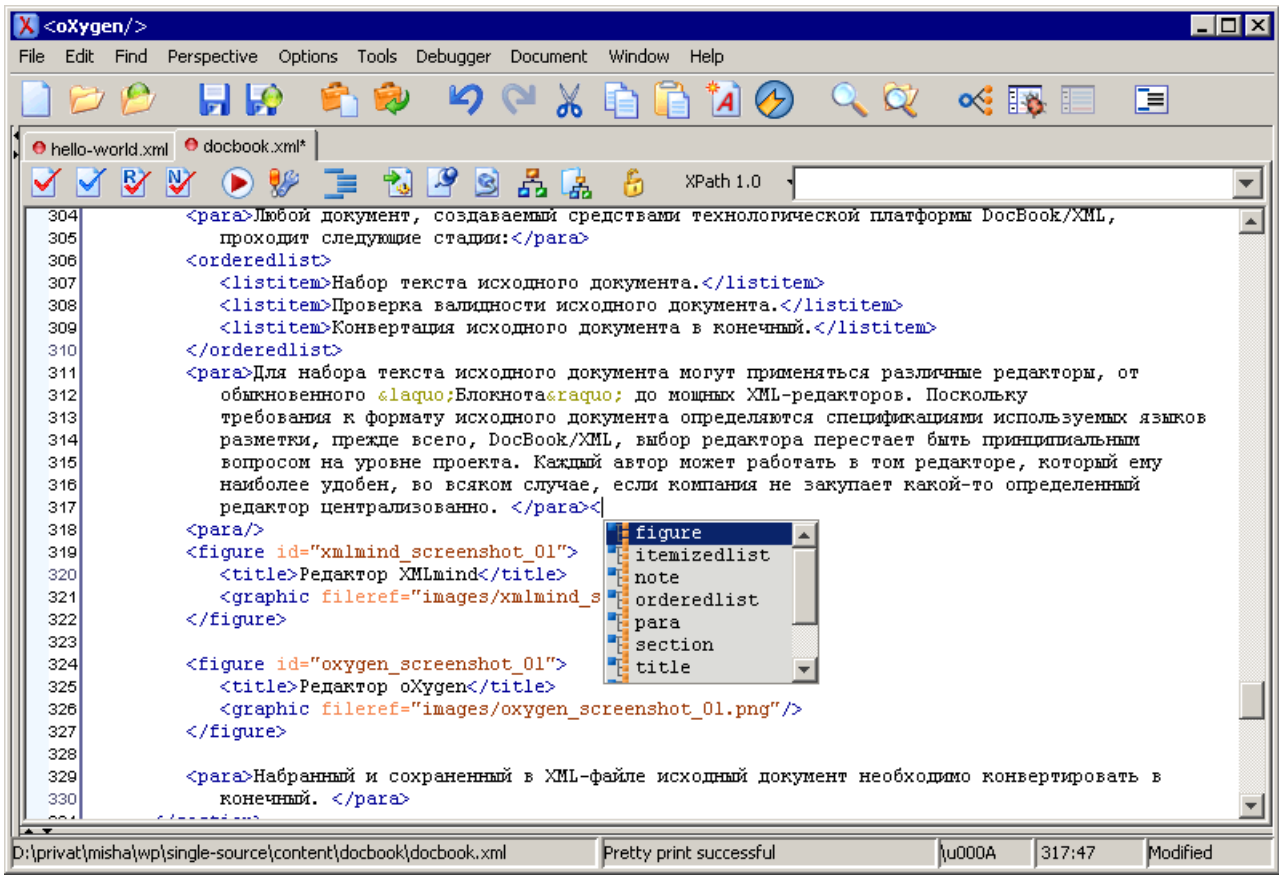

Рисунок 6. Редактор <oXygen/>[8]

В XML-редакторах данного типа автор получает разнообразный сервис для работы с разметкой. Подсветка тегов и других частей синтаксиса, автоматическая поддержка отступов для вложенных элементов, автоматическая вставка закрывающего тега при вводе открывающего, выбор нужного тега в меню, встроенная проверка целостности структуры и валидация — все это облегчает и ускоряет работу автора.

3.2.2 XSLT-процессоры и XSLT-стили

Мы храним данные ради того, чтобы пользоваться ими, обрабатывать или отображать их всевозможными способами. Данные, выведенные на печать непосредственно в формате XML, как правило, прочитать удается, но пользователю этого не предложишь. Во-первых, он не обязан знать язык разметки. Во-вторых, в XML-файле данные, которые должны оботражаться рядом, могут быть разделены. В-третьих, это неудобно и не соответствует никаким стандартам и традициям оформления документов. Следовательно, должны существовать механизмы, позволяющие реструктурировать XML-данные и переводить их в другие форматы, в том числе, в такие, для которых уже разработаны программы просмотра.

Один из основных механизмов обработки XML-данных основан на языке *XSL Transformations* [9], синтаксис которого основан на XML, однако, назвать его языком разметки было бы неправильно из-за принципиально иной семантики. XSLT — декларативный язык, приспособленный для записи правил преобразования XML-документов [15, 16, 4]. Они могут быть как совсем простыми, так и сложными, многоступенчатыми,

[8] http://www.oxygenxml.com/

[9] Аббревиатура XSL означает eXtensible Stylesheet Language. «Двухступенчатая» расшифровка аббревиатуры XSLT связана с историей развития этих языков.

с параметрами, ветвлениями и обращениями, в частности, рекурсивными, к другим правилам. Набор правил для решения некоторой прикладной задачи (фактически, программа на языке XSLT), называется *XSLT-стилем*. Программа, интерпретирующая XSLT-стили, называется *XSLT-процессором*. Входными данными для XSLT-процессора служат XML-файл и XSLT-стиль, на выходе формируются результаты преобразования.

Важнейшее преимущество XSLT в том, что с разработчика стилей снимается обязанность решать по-настоящему сложные программистские задачи вроде синтаксического разбора XML-кода и анализа структуры XML-документа. Создание прямого конвертора, допустим, из RTF в TeX, даже у квалифицированного программиста отняло бы гораздо больше времени, чем составление стиля для работы с XML-форматами сопоставимых описательных возможностей.

3.2.3 Стандартные стили DocBook XSL

Требования к оформлению конечных документов в разных проектах существенно различны, поэтому попытки выпустить исчерпывающий набор XSLT-стилей обречены на провал. Создавать XSLT-стили заново в каждом проекте тоже нельзя, потому что сроки и стоимость этой работы получатся неприемлемыми. На практике обычно применяется свободно распространяемый комплект стандартных стилей *DocBook XSL* [8]. Он поддерживает основные элементы макетов и целевые форматы, а также хорошо приспособлен к доработке для нужд конкретного проекта. Благодаря архитектуре XSLT адаптация стандартных стилей не требует модификации их кода. Новые правила, располагаемые в отдельных файлах, дополняют или замещают стандартные. Выполненные доработки не привязаны к конкретной копии стандартных стилей, следовательно:

- доработки не конфликтуют друг с другом, и ими легко обмениваться;
- возможность обновления версий DocBook XSL ограничена только их обратной совместимостью.

3.2.4 Типовые преобразования

Рассмотрим преобразования исходных документов формата DocBook/XML в целевые форматы HTML, CHM и PDF. Соответствующие типовые цепочки схематически показаны на рис. 7.

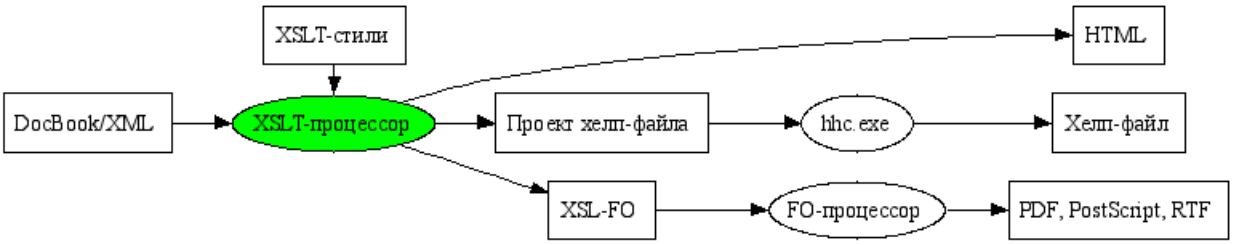

Рисунок 7. Преобразования DocBook/XML в HTML, CHM и PDF

Порядок формирования HTML-файлов:

1. С помощью XSLT-преобразования по исходному документу формируется один HTML-файл или каталог взаимосвязанных HTML-файлов.

Порядок формирования хелп-файлов формата CHM:

1. С помощью XSLT-преобразования по исходному документу формируется проект хелп-файла. Его состав и форматы входящих в него файлов диктуются требованиями инструментария HTML Help Workshop, стандартного для среды Microsoft Windows.

2. Проект обрабатывается компилятором hhc.exe, который входит в HTML Help Workshop. В результате формируется CHM-файл.

Порядок формирования файлов формата PDF:

1. С помощью XSLT-преобразования по исходному документу формируется файл формата XSL-FO [14]. Он представляет собой XML-разметку текста, оформленного для вывода на печать, но не разбитого на страницы.

2. Файл формата XSL-FO обрабатывается программой *FO-процессором*. В результате формируется PDF-файл.

Замечание

FO-процессором могут формироваться не только PDF-файлы, но и файлы в формате PostScript[10]. Существуют FO-процессоры, предназначенные для формирования RTF-файлов.

Возможности публикации исходных документов формата DocBook/XML не исчерпываются тремя перечисленными вариантами. Существуют XSLT-стили, поддерживающие форматы XHTML, Unix man pages, JavaHelp, TeX. Нет принципиальных препятствий к разработке стилей для формирования документов любого формата, во всяком случае, если его спецификация опубликована.

[10] http://www.adobe.com/products/postscript/overview.html

4 Прикладные решения

4.1 Документирование корпоративной информационной системы

Под *корпоративной информационной системой (КИС)* или *автоматизированной системой (АС)* [1] здесь понимается решение, полностью или частично автоматизирующее деятельность некоторой организации. Подчеркнем, что АС включает в себя и программы, и данные, и аппаратные средства, и участвующий в автоматизированных бизнес-процессах персонал [3, 7]. Иными словами, завершенная АС обязательно внедрена, она не продается «в коробке», хотя может создаваться на основе тиражируемого программного продукта.

Документирование АС обычно обусловлено следующими обстоятельствами:

- Разные компоненты АС (модули, функциональные подсистемы) могут создаваться в разное время разными командами, что объясняется постепенным изменением потребностей бизнеса, необходимостью планирования бюджета и загруженностью разработчиков.

- Документация на АС не сводится к технической документации на входящие в ее состав программы, обязательно должна быть описана технология выполнения бизнес-процессов с помощью системы.

- Постоянное изменение обстановки, в которой действует организация, вынуждает модифицировать АС и автоматизированные бизнес-процессы, при этом необходимо поддерживать актуальность документации.

- Аудитория документации на АС, т.е. сотрудники, которые непосредственно работают с ней, а также обеспечивают ее эксплаутацию и сопровождение, могут быть распределены по территориально удаленным друг от друга офисам организации.

- Документация на АС включена в документооборот организации, она может проходить согласование и утверждение, отдельные документы могут приобретать нормативное значение, выдаваться сотрудникам для ознакомления под роспись, предоставляться внешним и корпоративным аудиторам и т.п.

Это заставляет предъявлять к документации и процессу документирования противоречивые требования:

- Такие документы, как формуляр, руководство администратора или руководство программиста, должны содержать необходимые сведения обо всех компонентах АС и взаимосвязах между ними. Если АС достаточно сложна, для каждой пользовательской роли разрабатывают отдельное руководство, причем одна и та же пользовательская роль может требовать обращения к разным компонентам. Вместе с тем, документирование компонентов АС — часть их разработки, а разрабатываются они поотдельности.

- Пользователям должно быть удобно работать с документацией, а разработчикам сопровождать ее. В интересах первых руководства пользователя следовало бы объединить с описаниями бизнес-процессов, а в интересах вторых вынести описания бизнес-процессов в отдельный документ.

[1] Налицо терминологическая проблема: в отечественной «компьютерной» периодике укоренился первый термин, а в ГОСТ серии 34 используется второй.

- Чтобы сотрудники всегда имели доступ к самой последней версии докуменации, ее нужно публиковать на интранет-сайте, однако, нормативные и отчетные документы должны быть представлены твердыми копиями, оформленными согласно традиции или определенному стандарту.

Информация возникает и фиксируется в одном порядке, а потребляется в другом, причем разные аудитории нуждаются в разных форматах ее представления. Необходим механизм перегруппировки и дублирования информации. Реализовать его можно, в частности, с помощью технологии единого источника:

- Документация составляется по *предметам*, а публикуется по *аспектам*. Вместе с каждым компонентом разрабатывается техническая документация на него. Сведения об устройстве и порядке администрирования компонентов включаются в соответствующие сводные документы. Если между пользовательскими ролями и компонентами нет взаимнооднозначного соответствия, то для каждой роли формируется руководство, включающее инструкции по работе со всеми необходимыми компонентами.

- Техническая документация публикуется в нескольких форматах. Для поддержки ежедневной работы сотрудников создается HTML-публикация, а для изготовления твердых копий формируются PDF-документы.

- Описания бизнес-процессов (технологические инструкции) разрабатываются отдельно от руководств пользователя. Для удобства чтения их связывают:

    - перекрестными ссылками, преобразуемыми в гипертекстовые при формировании HTML-публикации;

    - включающими ссылками, преобразуемыми непосредственно в текст при формировании PDF-документов.

Схематически предложенное решение показано на рис. 8.

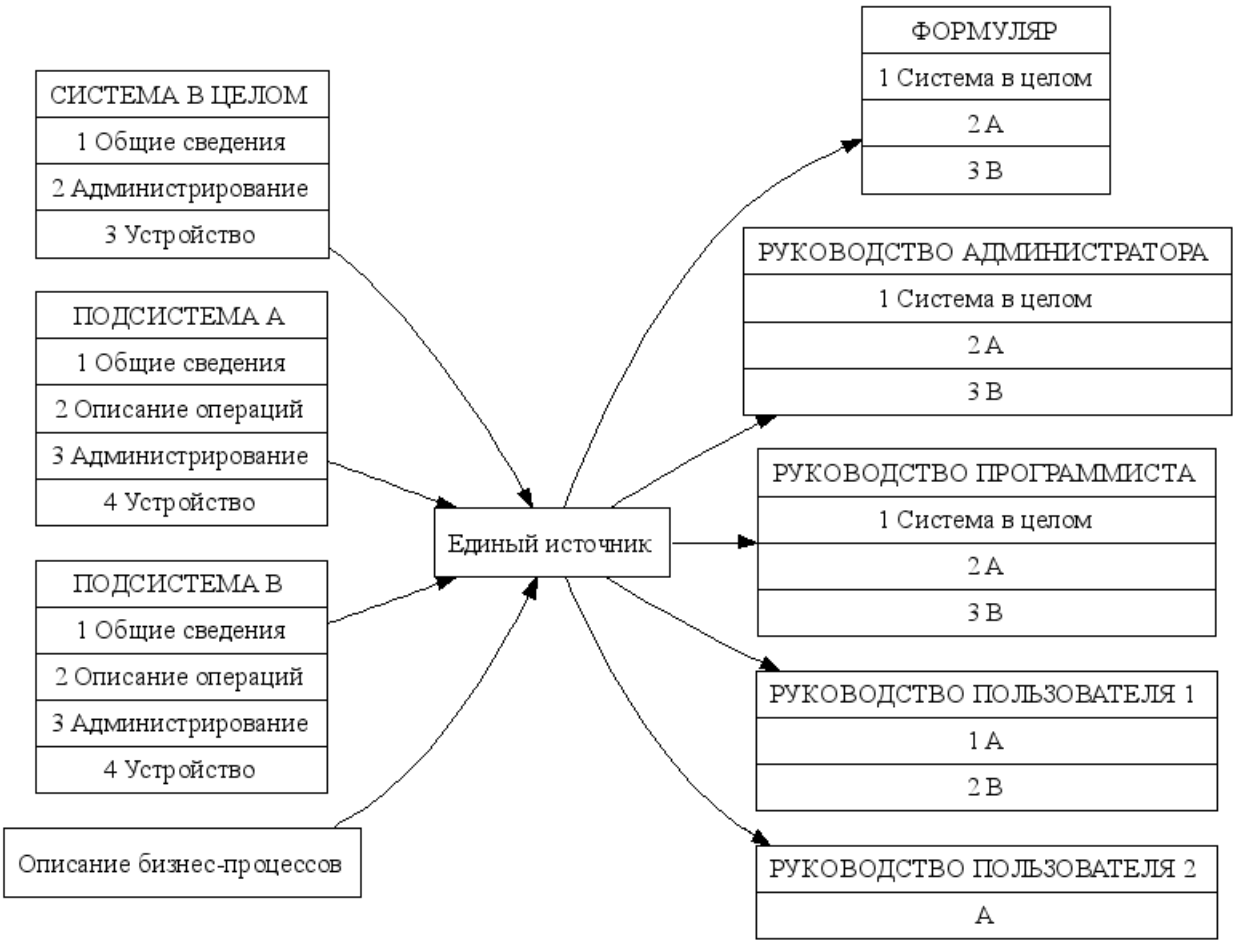

Рисунок 8. Документирование автоматизированной системы

4.2 Документирование комплектуемого решения

Комплектуемое решение поставляется потребителю в одном из возможных вариантов, количество которых комбинаторно. Типичный пример комплектуемого решения — PC-совместимый компьютер. Он состоит из определенных функциональных блоков (корпуса, системной платы, процессора, модулей оперативной памяти, жесткого диска и др.), причем их модели и технические характеристики варьируются от поставки к поставке. В действительности сочетаемость блоков ограничена: одни должны присутствовать обязательно, другие могут быть взаимоисключающими или, наоборот, устанавливаться только совместно, но для нас это сейчас не существенно. Важно, что эксплуатационная документация на такое решение тоже должна комплектоваться, поставлять ее в избытке практически невозможно, потому что избыточность получится огромной.

Далее для простоты изложения будем предполагать, что комплект эксплуатационной документации состоит из единственного документа, руководства пользователя. Другие документы, паспорт или спецификация [2], формируются аналогично.

Простейший способ комплектовать руководство пользователя — создать шаблон с унифицированной структурой разделов и вручную заполнять его включающими ссылками на фрагменты единого источника, содержащие описания моделей блоков, поставляемых в составе конкретной конфигурации. У этого способа есть два весомых недостатка. Во-первых, для каждой конфигурации фактически придется разрабатывать отдельный шаблон. Во-вторых, технические характеристики конфигурации тоже придется указывать вручную, что ведет к тиражированию разделов, в которых они приво-

дятся. Очевидно, этот способ трудоемок и чреват многочисленными ошибками, особенно учитывая, что некоторые технические характеристики конфигурации зависят от моделей блоков, например, максимальный объем оперативной памяти — свойство системной платы.

Технически сложнее, но намного эффективнее сделать руководство пользователя параметрическим. Автор, публикующий руководство пользователя на очередную конфигурацию, не будет редактировать ни шаблон, ни фрагменты единого источника, вместо этого он укажет значения *параметров конфигурации*. При формировании конечного документа они автоматически подставятся в него.

Предусмотрены три категории параметров конфигурации:

- *структурные*;
- *свободные*;
- *связанные*.

Структурный параметр содержит идентификатор (условное обозначение, принятое в рамках проекта) модели включаемого в конфигурацию блока определенного типа, допустим, корпуса или системной платы. По идентификатору фрагмент с описанием блока извлекается из единого источника при обработке исходного документа. Если блок некоторого типа отсутствует в конфигурации, то соответствующему структурному параметру следует присвоить пустое значение.

Свободный параметр содержит значение, которое варьируется от конфигурации к конфигурации. Свободными параметрами могут быть, например, объем установленной оперативной памяти, емкость жесткого диска, тактовая частота процессора, гарантийный срок.

Связанный параметр содержит значение, определяемое моделью какого-нибудь блока. Так, максимальный объем оперативной памяти, максимальное количество процессоров, габариты корпуса — связанные параметры.

Все значения структурных и свободных параметров указываются в едином *файле конфигурации*. Значения связанных параметров указываются непосредственно во фрагментах с описаниями блоков. Шаблон руководства, как обычно, содержит включающие ссылки на фрагменты единого источника, однако, они выражены через значения структурных параметров. Фрагменты единого источника могут содержать ссылки на параметры, иными словами, они «экспортируют» значения связанных параметров и «импортируют» значения свободных и связанных параметров.

Схема формирования параметрического руководства пользователя приведена на рис. 9.

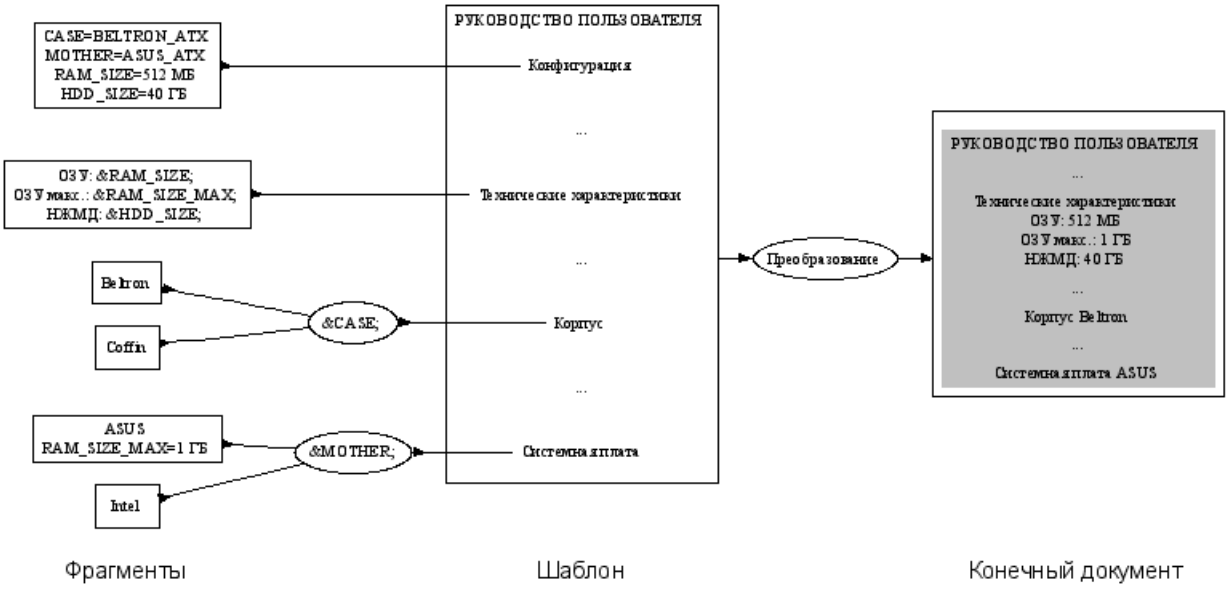

Рисунок 9. Формирование руководства пользователя на компьютер

Литература

1. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам
2. ГОСТ 2.601-95. Единая система конструкторской документации. Эксплуатационные документы
3. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения
4. А. Н. Валиков. Технология XSLT - СПб.: БХВ-Петербург, 2002 - 544 с.: ил.
5. И. Ш. Хабибуллин. Самоучитель XML - СПб.: БХВ-Петербург, 2003 - 336 с.: ил.
6. Extensible Markup Language (XML) 1.0 (Third Edition)[1]. W3C Recommendation 04 February 2004
7. ISO/IEC 15288. System Engineering - System Life Cycle Processes. - 2001
8. Bob Stayton. DocBook XSL: The Complete Guide[2] - Sagehill Enterprises, 2005 - 560 с.
9. Norman Walsh, Leonard Muellner. DocBook: The Definitive Guide[3] - O'Reilly & Associates, Inc., 1999 - 644 с.
10. XML Inclusions (XInclude) Version 1.0[4]. W3C Recommendation 20 December 2004
11. XML Path Language (XPath) Version 1.0[5]. W3C Recommendation 16 November 1999
12. XML Path Language (XPath) 2.0[6]. W3C Candidate Recommendation 3 November 2005
13. XML Pointer Language (XPointer) Version 1.0[7]. W3C Last Call Working Draft 8 January 2001
14. Extensible Stylesheet Language (XSL) Version 1.0[8]. W3C Recommendation 15 October 2001
15. XSL Transformations (XSLT) Version 1.0[9]. W3C Recommendation 16 November 1999
16. XSL Transformations (XSLT) Version 2.0[10]. W3C Candidate Recommendation 3 November 2005

[1] http://www.w3.org/TR/2004/REC-xml-20040204/
[2] http://www.sagehill.net/docbookxsl/index.html
[3] http://www.docbook.org
[4] http://www.w3.org/TR/xinclude/
[5] http://www.w3.org/TR/xpath
[6] http://www.w3.org/TR/xpath20/
[7] http://www.w3.org/TR/WD-xptr
[8] http://www.w3.org/TR/xsl/
[9] http://www.w3.org/TR/xslt
[10] http://www.w3.org/TR/xslt20/