Московский государственный университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра алгоритмических языков

Введение в дипломную работу

«Система ведения документации в разнородных форматах логической разметки»

студент гр. 524 Потапенко Александр Александрович
научный руководитель:
к. ф.-м. н. Столяров Андрей Викторович

Москва
2007

Содержание

1 Постановка задачи **3**
1.1 Задача хранения и публикации документов . . . 3
1.2 Разметка текстов . . . 3
1.3 Существующие решения . . . 4
1.4 Естественные ограничения . . . 5
1.5 Уточненная постановка задачи . . . 5

2 Преобразование форматов **5**
2.1 Разбор и генерация разметки . . . 5
2.2 Промежуточный формат представления . . . 6
2.3 Структура документа . . . 6
2.4 Преобразование графовой модели . . . 7
2.5 Синтаксический разбор и уровень моделирования . . . 7

3 Архитектура системы **8**
3.1 Интеграция с MoinMoin . . . 8
3.2 Процесс использования . . . 9

4 Заключение **9**
4.1 Текущие результаты . . . 9
4.2 Дальнейшие направления деятельности . . . 10

1 Постановка задачи

1.1 Задача хранения и публикации документов

В современных сообществах разработчиков и пользователей свободного ПО важную роль играют обмен опытом и накопление информации о существующих решениях. Поэтому часто возникает необходимость создания общедоступных хранилищ текстов с возможностью публикации [1] свободной документации. Специфика упомянутых сообществ (в частности, отсутствие денежной мотивации) такова, что на публикуемые тексты должны накладываться минимальные ограничения: чем комфортнее процесс создания, редактирования и публикации документов, тем большее количество пользователей примет участие в разработке корпуса документации.

Под комфортным созданием документов в данном случае мы будем понимать любые средства написания и представления текста, которые были бы привычны для автора. Под комфортным редактированием — потенциальную возможность внесения изменений в любой из документов в хранилище, независимо от его формата, а также организации совместной разработки документации. Процесс публикации можно считать комфортным, если для выдачи документа в удобном для читателя виде требуется приложение минимума усилий. В идеале предполагается, что трудовые затраты на обновление и переиздание документа должны быть существенно ниже соответствующих затрат на внесение его в хранилище и издание.

Изначально задача, решаемая в данной работе, возникла в рамках сообщества пользователей ALT Linux, в котором документация пишется и публикуется разными людьми в довольно широком наборе форматов, а в ее редактировании могут принимать участие волонтеры. Однако потребность в подобных хранилищах появляется и в других случаях — например, при создании многопользовательских систем управления контентом (CMS). Кроме того, вопросы преобразования документов из одного формата в другой часто приходится решать и в отрыве от их хранения.

1.2 Разметка текстов

Под *форматом* текста здесь и далее мы подразумеваем формат его разметки, т.е. правила, по которым в тексте располагаются управляющие символы, описывающие способы его интерпретации.

Будем называть разметкой произвольный способ сделать явной интерпретацию текста. Можно выделить два типа разметки: *логическую* (или семантическую), определяющую структуру документа и *физическую* (или процедурную), определяющую способ обработки и отображения текста в данной точке документа. При создании документации чаще используется логическая разметка, так как на этапе написания текста способ отображения документа еще может быть не определен. Кроме того, описание структуры более естественно для авторов и не накладывает строгих ограничений на конечный вид документа.

Задача логической разметки заключается в разбиении исходного текста на блоки, каждый из которых связан с некоторым набором атрибутов, в соответствии с которыми блоки будут интерпретироваться. После этого в исходный текст вставляются некоторые управляющие последовательности символов, позволяющие каким-либо образом однозначно восстановить набор атрибутов для каждого из блоков.

Процесс разметки текстов может быть полностью или частично автоматизирован за счет применения специальных текстовых процессоров. Однако зачастую синтаксис

[1] Здесь мы называем публикацией произвольный способ доведения информации до читателя.

разметки выбирается с расчетом на то, что редактирование будет производиться вручную; в таких случаях размеченный текст обычно похож на ожидаемый в результате интерпретации, а количество управляющих символов очень мало. Примерами этих так называемых легковесных языков разметки являются reStructuredText([2]) и различные Wiki-форматы[2].

1.3 Существующие решения

Системы хранения и преобразования документов устроены по одному из следующих принципов:

- Single-source, single-target. Поддерживаются единственный входной и единственный выходной форматы. Этот класс систем наименее удобен для пользователя, поскольку возможность коллективной работы над текстами не компенсирует ограничений, накладываемых на создание и публикацию.

- Single-source, multi-target. Поддерживается один входной формат, из которого можно изготовить несколько выходных. Представителями данного класса являются системы, основанные на форматах DocBook, reST и различных Wiki-разметках. Их основной недостаток — необходимость ручного перенабора входных текстов в других форматах, невозможность обратной связи с автором в его формате. Затраты на обновление из авторского формата оказываются в этом случае не меньше затрат на первоначальное вхождение документа в систему, и те, и другие велики.

- Multi-source, single-target. Поддерживается несколько входных форматов, преобразуемых к единому выходному (чаще всего к HTML). Недостатки: ограниченный список входных форматов, несовместимость дизайна документов, полученных из разных форматов, необходимость ручного перевода в другие выходные форматы. Затраты на переиздание в другом выходном формате и в этом случае оказываются не меньше затрат на первоначальное издание, и те, и другие велики.

- Multi-source, multi-target. Поддерживается несколько входных и выходных форматов, возможно преобразование некоторых входных форматов в некоторые выходные. Подобные системы реализуют два подхода к хранению и публикации документов:

  - входные документы принимаются в любом формате и публикуются без изменений, система организует лишь хранение и простейшую подготовку к публикации. Этот подход часто применяется при организации хранилищ ПО, поэтому привычен для пользователей, но создает проблемы при необходимости переиздания документов в другом формате. В частности, он реализован в репозитории ALT Linux Docs Heap[3], на совершенствование функциональности которого направлена настоящая работа.
  - входные форматы конвертируются в один из выходных форматов, который затем может быть преобразован в другие выходные форматы. Так устроены некоторые Wiki-системы (например, `MoinMoin` [3]). К сожалению, обычно они не предоставляют средств для технологичного преобразования выходных форматов, которое осложняется использованием одного из этих форматов в качестве промежуточного.

[2] См., напр., сайт рабочей группы Wiki Creole: `http://www.wikicreole.org/`

[3] Информационный сервер тематических проектов ALT Linux, `http://heap.altlinux.ru/`

Распространенной проблемой современных систем документооборота являются также трудности с модификацией существующих и введением новых форматов входных документов. Довольно часто этот вопрос решается в пользу удобства разработчиков, а не авторов текстов.

1.4 Естественные ограничения

Создание системы, максимально удовлетворяющей требованиям пользовательского комфорта при написании, редактировании и публикации документов невозможно из-за разнородности существующих форматов их представления. Однако задачу можно сильно упростить, наложив не очень серьезные ограничения на тексты, попадающие в хранилище.

Во-первых, есть смысл ограничить поддержку бинарных документов возможностью их преобразования к удобному для чтения виду. Такие документы чаще всего являются результатами преобразования некоторых текстовых данных (например, PDF можно получить из LaTeX), поэтому гораздо правильнее будет хранить в системе эти исходные тексты. Допускается хранение в бинарном виде объектов, включаемых в документы — например, изображений.

Во-вторых, поддержка большого количества выходных форматов требует отказа от визуальной разметки текстов в пользу логической. Это упростит автоматическую обработку документа и его отображение в различных условиях, но не затронет интересов автора (задачей которого, напомним, является создание и редактирование текста документа, а не подготовка его к публикации). Пользуясь тем, что речь идет о написании технической документации, можно также выделить конечный набор допустимых элементов логической разметки.

В-третьих, при совместном редактировании текстов следует учитывать различия в выразительных средствах разных форматов. Если изменения, внесенные в преобразованный из некоторого формата текст, не могут быть корректно сконвертированы обратно, они должны не вноситься в оригинальный документ, а оформляться в виде патчей (исправлений) к версии в новом формате.

1.5 Уточненная постановка задачи

Принимая во внимание требования пользователей, а также вышеизложенные ограничения, можно сформулировать более реалистичную задачу: создать инструментарий и инфраструктуру для хранения и переработки технической документации. Система должна уметь распознавать различные форматы логической разметки текста (входные форматы) и выдавать эти тексты в требуемых выходных форматах, при этом желательно, чтобы затраты на их обновление и переиздание были существенно ниже затрат на соответственно вхождение и издание.

В данной работе будет рассмотрена система, направленная на решение этой задачи; она получила название Babylon.

2 Преобразование форматов

2.1 Разбор и генерация разметки

Разбором разметки назовем процесс восстановления по размеченному тексту синтаксической структуры документа, то есть набора описанных выше блоков исходного текста

с атрибутами. Считается, что блоки и их атрибуты определяются только управляющими последовательностями символов, встречающимися в тексте. Программы, осуществляющие автоматический разбор, называются синтаксическими анализаторами (парсерами).

В общем случае разбор разметки является довольно трудоемкой задачей, поскольку набор управляющих символов может быть выбран сколь угодно большим, а их семантика — сколь угодно сложной и не поддающейся классификации. Однако в прикладных задачах, связанных с разбором логической разметки, семантика управляющих конструкций обычно подчиняется некоторым закономерностям. Это сильно упрощает процесс разбора и дает возможность автоматизировать написание парсеров для конкретных логических разметок.

Обратный процесс построения по структуре текста одной из эквивалентных разметок логично называть генерацией, а программы, выполняющие его — генераторами текста. Задача генерации разметки текста по его синтаксической структуре обычно является более простой, чем задача разбора.

2.2 Промежуточный формат представления

Очевидно, простейшим решением задачи публикации размеченных документов будет написание программ-преобразователей для каждой пары форматов. Но в этом случае неоправданно возрастут затраты на добавление в систему нового формата разметки: для него придется описывать преобразования во все входные форматы, из которых его планируется получать, и все выходные форматы, в которые его придется конвертировать. При этом понадобится многократно реализовывать практически не отличающиеся друг от друга парсеры и генераторы одних и тех же форматов.

Поскольку мы договорились считать, что преобразования между форматами имеют смысл только при совместимости их выразительных средств, можно конвертировать документы в два этапа: сначала извлекать некоторую «усредненную» структуру с помощью парсера входного формата, а затем с помощью генератора получать выходной текст. Этот подход требует написания лишь одного парсера и одного генератора для каждого нового формата в системе. Промежуточную структуру документа, речь о которой пойдет ниже, будем называть форматом представления документа.

2.3 Структура документа

Наиболее распространенным способом представления нелинейной (содержащей более одного уровня иерархии)[4] структуры документа являются деревья. Обычно в качестве корня выбирается верхний элемент иерархии (книга, статья и т. п.), поддеревья узлов задают вложенные элементы более нижних уровней, а листья — текст документа. На поддеревьях задается транзитивное отношение порядка, позволяющее однозначно определить последовательность блоков текста.

Примером языка, представимого в виде дерева, является (с незначительными ограничениями) язык GML, разработанный фирмой IBM в 1970 году, а также развившиеся из него и широко используемые в настоящее время SGML ([6]), HTML и другие XML-подобные форматы.

Структуры данных для поддержания древесного представления довольно просты, но подход имеет существенный недостаток — невозможность работы с текстами, содержащими параллельные иерархии логических элементов (см. примеры в [11]). Эту проблему

[4]В этом смысле большая часть документов нелинейна, оставшиеся же проще рассматривать как нелинейные, чем вводить для них отдельный способ представления.

решают объединением нескольких деревьев иерархии в лес, либо применением языков, поддерживающих мультииерархическую структуру на уровне модели данных (например, [9] или [1]).

Кроме того, в технических документах часто появляется необходимость указать связь между двумя элементами, находящимися на расстоянии друг от друга (это может быть ссылка на библиографический источник, пункт оглавления или внешний объект) или пометить некоторую сущность в документе участком размеченного текста (возможно, также содержащим иерархическую структуру, ссылки и т.п.). В древесной модели в таких случаях пользуются атрибутами узлов, но при этом обход дерева становится неоднородным — в зависимости от типа связи между элементами разметки приходится выполнять различные действия для перехода к другому концу связи.

Обобщая понятие связи между элементами разметки, перейдем к графовой модели представления структуры документа. В ней узлам графа соответствуют участки исходного текста и элементы логической разметки (теги), а ориентированные ребра, помеченные типами связей, отражают отношения между этими узлами. В отличие от древесной модели, в которой предпочтение отдается отношениям вложенности и следования, здесь все связи равноправны и могут обрабатываться универсальным образом.

2.4 Преобразование графовой модели

Большинство современных подходов к преобразованию графов используют так называемый метод *переписывания* (graph rewriting, см. [4]). Он заключается в последовательном применении к графу правил, в левых частях которых некоторым формальным образом описываются условия на подграф преобразуемого графа, а в правых указываются действия по его преобразованию. Процесс завершается, когда граф оказывается сведен к нормальной форме, т.е. к нему не может быть применено ни одно из правил.

При поиске и модификации подграфов над структурой документа можно оперировать следующими способами:

- декларативным (задавая пред- и постусловия, которым должны удовлетворять правила преобразования);
- императивным, при котором описывается последовательность действий, производимых системой.

Возможность сочетания обоих способов обеспечивает применение в работе языка Пролог ([8]). Граф документа представляется в виде реляционной базы данных, содержащей прологовские термы, а правила преобразования могут содержать как логические условия, так и функции, изменяющие содержимое базы данных и управляющие перебором. Кроме того, появляется возможность контролировать *схему документа*, т.е. набор допустимых типов элементов разметки и отношений между ними.

2.5 Синтаксический разбор и уровень моделирования

Большой интерес при описании языка разметки представляет тип грамматики Хомского (см. [12]), которой описывается язык. Чаще всего это либо контекстно-свободные, либо контекстно-зависимые грамматики, и если задача разбора первых сводится к автоматической генерации парсера по описанию грамматики ([5]), то для вторых построить адекватную грамматику не всегда представляется возможным. Однако, поскольку в данном случае речь идет о разборе технических документов, можно воспользоваться подходом, базирующимся на *островных грамматиках* (island grammars, см. [7]).

Островными называются контекстно-свободные грамматики с выделенным множество терминалов, которые представляют интерес при разборе. На остальные терминалы накладываются нежесткие условия, позволяющие не рассматривать эти символы. Такие грамматики могут распознать верхние уровни синтаксической структуры текста («острова»), после чего неразобранная «вода» обрабатывается пользовательскими функциями для получения недостающих поддеревьев и листьев.

Полученное дерево синтаксического разбора еще не содержит информации о семантике документа, но, в отличие от текста, уже представляет собой частный случай графа логической структуры текста, и может быть легко сконвертировано в прологовскую реляционную базу данных путем обхода. Назовем это промежуточное представление форматом моделирования. Таким же образом, выбрав в базе данных два типа отношений между объектами, задающие дерево, можно получить из него выходной формат моделирования. Отображение тегов формата моделирования в формат представления и обратно осуществляется с помощью правил переписывания графов.

Итак, процесс работы преобразователя можно разбить на четыре этапа:

- синтаксический разбор входного текста, в результате которого получается дерево моделирования входной разметки;
- преобразование дерева моделирования в формат представления в соответствии со схемой документа: достраивание отношений, выводимых из древесной структуры, переименование вершин, удаление избыточных элементов разметки;
- вывод дерева моделирования выходного документа из формата представления;
- генерация по нему документа в выходном формате.

3 Архитектура системы

3.1 Интеграция с MoinMoin

Кроме преобразователя форматов, важной частью системы Babylon является инфраструктура хранилища документов. От него требуются следующие качества:

- возможность управления версиями документов и правами доступа к ним;
- простой интерфейс добавления и редактирования (в том числе коллективного) документов в хранилище;
- поддержка бинарных приложений к текстам;
- не требующий вмешательства администратора хранилища способ преобразования документа из одного поддерживаемого формата в другой;
- требующий минимального вмешательства администратора способ добавления нового формата.

Система управления контентом [3] имеет встроенные механизмы контроля версий и разграничения прав доступа, поддерживает хранение приложений к документам, имеет веб-интерфейс для добавления и редактирования текстов. Кроме того, прозрачное хранение информации в файловой системе и открытые программные интерфейсы дают возможность использовать модули MoinMoin из Babylon и наоборот.

Эти достоинства, а также открытость исходного кода MoinMoin позволяют нам создать хранилище на ее базе. Дополнительно необходимо реализовать редактор связей, от которого требуется своевременно приводить приложения к совместимому с выходным форматом виду и исправлять ссылки в формате представления, и среду сборки, которая по заданному выходному формату будет выбирать и исполнять нужную последовательность преобразований.

3.2 Процесс использования

Документ в одном из входных форматов пишется автором и добавляется в систему. При подготовке текста ему уже известны окончательная схема документа[5], поэтому при необходимости автор сможет описать устройство своего формата разметки и семантику его тегов в терминах схемы. Возможно, вместе с текстом в хранилище будут добавлены приложения, ссылки на которые находятся в документе.

Если в системе отсутствуют парсер и генератор данного формата, администратор автоматически создает их по предоставленному пользователем описанию. Возможно, отдельные функции для синтаксического анализа и/или генерации специфических участков текста будут написаны вручную. Также администратор описывает преобразования по схеме между форматом моделирования и форматом представления.

По запросу читателя на публикацию документа вызывается пара преобразований из входного формата в формат представления. После этого программа-редактор связей находит и модифицирует ссылки на приложения и одновременно приводит последние к нужному виду. Затем происходит преобразование в выходной формат моделирования, генерация текста и рендеринг внешних объектов. Полученный в итоге документ доставляется читателю.

Если читатель планирует вносить изменения в полученный текст и обновить содержимое авторского документа, то стиль разметки этого документа, возможно, будет отличаться от использованного автором (хотя и останется совместимым). Чтобы подобного не произошло, нужно более жестко зафиксировать схему документа, но чаще всего в этом нет нужды.

4 Заключение

4.1 Текущие результаты

В настоящий момент достигнуты следующие результаты:

- разработана архитектура системы ведения документации в разнородных форматах логической разметки;
- реализованы механизмы переписывания графов и анализа островных грамматик рекурсивным спуском с откатом;
- созданы библиотеки для облегчения написания парсеров и генераторов текста;

В июне 2007 года в соавторстве с Г. В. Курячим был сделан доклад о состоянии проекта на Четвертой конференции разработчиков свободных программ на Протве (см. [10]).

[5] То есть *рекомендованный* набор выразительных средств формата

4.2 Дальнейшие направления деятельности

В рамках дальнейшей работы над дипломом автор намерен автоматизировать написание парсеров и генераторов текста и внедрить систему в качестве замены репозитория ALT Linux Docs Heap. Впоследствии планируется более детальное изучение способов разбора легковесной разметки и преобразования структуры документов, а также ослабление ограничений, наложенных при постановке задачи.

Список литературы

[1] LMNL: the Layered Markup Annotation Language. http://lmnl.net/.

[2] reStructuredText – Markup Syntax and Parser Component of Docutils. http://docutils.sourceforge.net/rst.html.

[3] The MoinMoin Wiki Engine. http://moinmo.in/.

[4] Nachum Dershowitz. A Taste of Rewrite Systems. In *Functional Programming, Concurrency, Simulation and Automated Reasoning*, pages 199–228. Springer, 1993.

[5] Ceriel J.H. Jacobs Dick Grune. *Parsing Techniques - A Practical Guide*. Ellis Horwood, Chichester, England, 1990.

[6] ISO/IEC. ISO 8879:1986: Information processing – Text and office systems – Standard Generalized Markup Language (SGML). Technical report, International Organization for Standardization, Geneva, Switzerland., 1986.

[7] Rob van der Leek. Implementation Strategies for Island Grammars, Thesis Report. Faculty EWI, Delft University of Technology Delft, The Netherlands, 2005.

[8] Иван Братко. *Программирование на языке Пролог для искусственного интеллекта.* Москва, «Мир», 1990.

[9] П. В. Воздвиженский. Язык разметки мультииерархических документов. In *Компьютерная лингвистика и интеллектуальные технологии: Труды международной конференции «Диалог'2005» (Звенигород, 1-6 июня, 2005 г.).* М.:Наука, 2005.

[10] Г. В. Курячий, А. А. Потапенко. Система хранения и публикации документации Babylon. In *Четвёртая конференция разработчиков свободных программ на Протве: тезисы докладов.* Москва, Институт Логики, 2007.

[11] А. А. Реформатский. Архитектоника и шрифтовая система текстов (из монографии «Техническая редакция книги»). *Еженедельник «Русский язык», №01/2001.*

[12] А. Ахо, Дж. Ульман. *Теория синтаксического анализа, перевода и компиляции, т.1.* Москва, «Мир», 1978.