РАЗРАБОТКА И РЕАЛИЗАЦИЯ КОНЦЕПЦИИ АДАПТИВНЫХ ДОКУМЕНТОВ

Мажник А.А., Кантор С.А.
Алтайский государственный технический университет им. И.И.Ползунова
(г. Барнаул)

В последнее время всё большую актуальность приобретают задачи создания всевозможных систем автоматизации делопроизводства и документооборота. Одно из наиболее сложных и неоднозначных направлений в этом вопросе – создание и последующая обработка слабоформализованных и слабоструктурированных документов с произвольной, меняющейся в зависимости от содержимого, структурой. Содержимое таких документов не ограничивается какой-либо фиксированной формой, шаблоном, а регулируется набором определенных правил, принятых в соответственной отрасли, либо набором логических выражений, определяющих подход к содержимому того или иного документа. Как правило, в среднестатистической компании документооборот состоит именно из таких документов [1].

Данная статья посвящена описанию работы по исследованию слабоструктурированных документов, описанию концепции для работы с ними, а также разработке и программной реализации универсального подхода к отображению логико-смысловых правил документа на его структуру.

Когда речь идет о документах со строго формализованной структурой, обычно используются стандартные решения с применением неких шаблонов, при этом зачастую встает задача по обработке некоего произвольного множества данных, связанных между собой логически, и создание на их основе документов [2]. Некоторые из них данных имеют форму произвольных полнотекстовых документов, некоторые – содержат только информацию, но с четко выраженными границами, а некоторые – наборы структурированных данных с разбиением по фрагментам.

Если рассматривать слабоструктурированные документы в общем случае, то найти в них общие черты не представляется возможным, но если распределить их по степени соответствия определенной тематике и степени использования информации из определенной области, то можно найти логические правила и следствия, по которым структура документа и строится [3]. Человек, поставивший перед собой задачу набрать определенный документ, заранее не заботится о том, каким образом согласовать предложения, как быть с грамматикой и синтаксисом языка, он выполняет это после либо во время написания текста. Но вот зачем человек должен следить, и следит – это за смыслом, чтобы из одного логически связанного фрагмента следовали другие, логически связанные с предыдущими фрагментами. Если же рассмотреть только логический процесс создания документа определенной тематики, то можно отобразить его на его же структуру и на формализацию фрагментов.

Сама по себе задача по отображению логики документа на его структуру и последующую обработку документов сейчас не имеет четко обоснованного и теоретического решения. Все универсальные подходы обладают теми или иными достоинствами и недостатками.

Взяв определенную предметную область, человек видит, что взаимодействие в ней происходит по определенным логическим правилам, если правила нарушаются, то процесс выходит за пределы рассматриваемой области. Так же и в слабоструктурированных документах – «взаимодействие» между частями документа происходит по определенным правилам.

Рассмотрим процесс подготовки документа не со стороны пользователя, а со стороны логики документа. Пусть известно, что к некоторому моменту создания документа в него необходимо включить информацию в определенном виде и в определенном месте, после чего в документ опять включается следующая часть информации, но уже в виде и месте, зависимом от предыдущей информации. И так, пока документ не будет содержать всей необходимой информации. В таком случае можно говорить об итерационном процессе пошаговой формализации документа в зависимости от информации на каждом шаге. Т.е. существует множество объектов информации, и существует множество вариантов оставшейся части документа.

Как пример отображения логики на структуру можно привести следующее:

Иванов И.И. имеет задолженность по кредиту в размере 10 тыс. руб.

Иванов И.И. не имеет задолженности по кредиту.

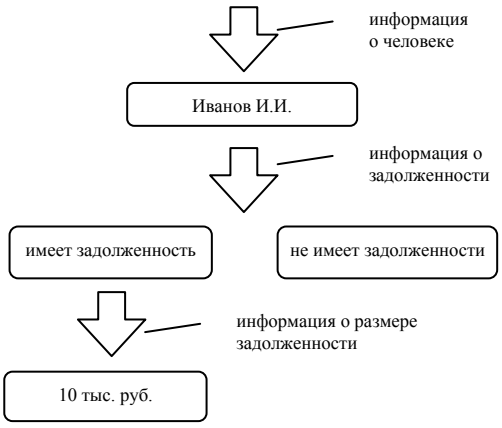

Рисунок 1 - Дерево структуры документа на основе поступающей информации

На рисунке 1 видно, что логика перехода от шага к шагу – это соответствие фрагмента документа и полученной информации.

Рассмотрим набор объектов и способов, для автоматического управления логико-смысловой нагрузкой документа, которые являются основой концепции адаптивных документов.

Если определить в документе новый объект: *поле*, у которого есть основные атрибуты: *название*, *позиция, функция* и *значение*, а также поле может состоять из других полей, то можно говорить о принадлежности некого множества данных этому полю.

Название поля идентифицирует его среди других в документе, значение характеризует информацию, которая принадлежит данному полю, а позиция определяет положение поля в тексте. Атрибут позиция – это единственная характеристика поля, изменение которой неподконтрольно пользователю, она может быть задана первоначально и в последствии меняется при изменении текста.

Значением будем называть ту информацию, которая вводиться в поле в виде текста, чисел, дат и в последствии преобразуется в текст для отображения в документе.

Основополагающим тезисом адаптивных документов будет то, что *поля с одинаковыми названиями имеют одинаковые значения*.

Понятие функции – определим как множество правил, по которым значение поля преобразуется в текст, отображаемый в документе. Данные преобразования производиться автоматически при изменении значения одного из полей.

Основными функциями адаптивных документов являются функции грамматики естественного языка, которые подразделяются на: морфологические, семантические и синтаксические. Морфологические функции обеспечивают словообразование через управление морфемами слова. В качестве примера можно привести функции склонения по падежам слов, фамилий, склонения предложений, определение родов и т.д. Синтаксические функции обеспечивают правила словосочетания и расстановку знаков препинания в документе. Семантические функции обеспечивают правила для корректности смысла одного фрагмента текста в зависимости от смысла другого.

Для обеспечения изменяемости отдельных частей документа в зависимости от данных используются разбиение его на *фрагменты*, которые подключаются к документу специальной разметкой либо применением поля с семантической функцией «фрагмент». Фрагмент может содержать варианты, в которых явно указано, при появлении какого рода данных в документе появиться заданная часть текста.

Поля, находящиеся во фрагментах, расположены в отдельном пространстве названий. В общем случае, если условия появления того или иного фрагмента в документе выполнены, то происходит переопределение названий полей фрагментом. Поле *A*, содержащее фрагмент, в котором присутствуют поля *B* и *C*, в остальном документе будет адресоваться как *A,* но поля фрагмента как *A.B* и *A.C*. Такие названия, учитывающие иерархическую структуру полей, названы *абсолютными названиями*, хотя в концепции применяется методика использования определенных правил для управления формированием абсолютного названия. Так использование символа «:» в названии поля обеспечивает фиксацию имени для иерархически более высокого уровня. Формат применения: *<название предка>:<собственное название>*. Пусть поле *A* содержит поля с названиями *B* и *D:C*, тогда будут иметь место следующие абсолютные названия: *A*, *A.B*, *D.C* (а не A.D.C) – т.к. отсутствует название предка для предыдущего уровня в иерархии. Данный подход используется для получения значений полей иерархически выходящих из фрагмента документа представленного полем.

Использовать фрагментацию, в предыдущем примере, можно несколькими способами.

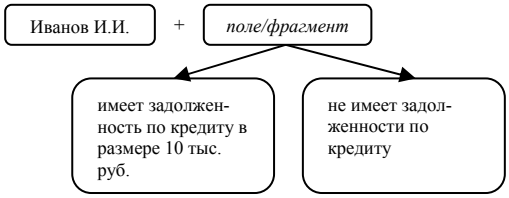

Рисунок 2 - Способ фрагментации текста документа по фрагментам

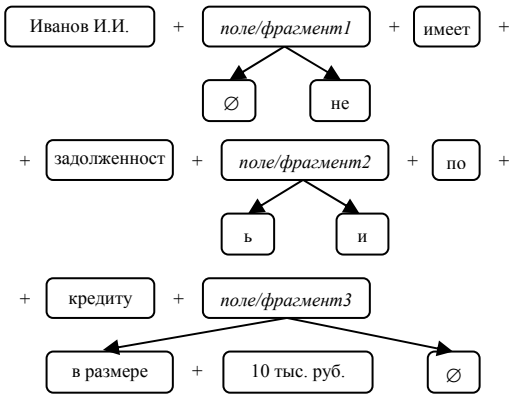

Рисунок 3 - Способ фрагментации текста документа по словам

Способы фрагментации, представленные на рисунках 2 и 3 имеют как свои преимущества, так и свои недостатки. Первый способ отличает то, что число используемых фрагментов минимально и более четко прослеживается логика появления одного из вариантов. Во втором способе фрагментация на уровне морфем слова, а управление морфемами можно обеспечить с помощью наложения морфологических функций, тем самым оптимизировать использование фрагментов.

Одним из центральных объектов адаптивного документа является понятие *итератор*. Итератор – это поле, но с заранее определенным поведением. Он имеет две основные цели: обеспечить группировку множества данных по смысловой нагрузке – итератор *прямого* формирования значения, так и обеспечить управление размером множества данных – итератор *обратного* значения.

Такой тип полей имеет характерное отличие от обычных – символ «*» в конце названия. Т.е. название итератора может рассматриваться как маска для отбора полей по названию.

Итераторы прямого формирования значения имеют два подтипа: *перечислимый* и *количественный* (хотя эта характеристика более относится к значениям итератора). Если определить название итерационного поля как *поле**, то в таком случае, его значение будет состоять из значений полей вида: *поле, поле1, поле2*, разделенных, специальным текстом, являющимся атрибутом поля, а сам итератор будет называться перечислимым. Но если итератор количественный, то его значение будет равно числу полей, подходящих под его маску (рис. 4).

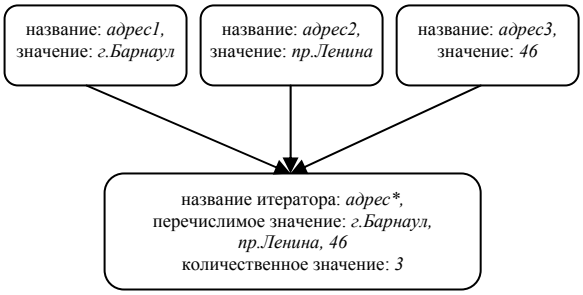

Рисунок 4 - Пример формирование значений итератора

На значение итератора, и к значению обычного поля, может быть применена функция. В случае с перечислимым итератором функция будет вычисляться для каждого из перечисленных значений в отдельности. А функция у количественного итератора вычисляется на основе его цифрового значения и является характеристикой количества полей.

Итераторы обратного значения формируются как и итераторы прямого значения, только сами влияют на этот процесс. Они одновременно являются перечислимыми и количественными. У такого итератора пользователю доступен интерфейс для изменения количественного значения.

Принцип работы итератора обратного значения основан на понятиях: *синхронизируемость* и *управление*. Поле – итератор обратного значения должен обладать атрибутом управление, для того чтобы он мог обеспечивать управление количеством полей в документе. А те поля, количеством которых итератор управляет должны иметь атрибут синхронизируемость. Такая связка двух атрибутов обеспечивает следующее: пусть итератор имеет название *поле**. А в документе присутствуют поле1 и *поле2*. В этом случае перечислимое значение итератора будет: «поле1, поле2»; количественное: «2». Пользователь изменяет количественное значение на «4», после чего в документе появятся новые поля *поле3*, *поле4*, а перечислимое значение итератора становиться: «поле1, поле2, поле3, поле4». Если пользователь еще раз изменит значение, скажем на «3», то *поле4* будет удалено из документа.

Включение итераторов в подсистему текстового редактора по работе с буфером

обмена позволяет привычным для пользователя способами - копирование и вставку, обеспечивать изменение зависимых по смыслу частей текста.

Таким образом, можно сформулировать следующие принципы, которым должны удовлетворять адаптивные документы:

1. Все множество документов разделено на однозначно идентифицируемые документы по определенным признакам области их применения.

2. Документы разделены на смысловые фрагменты.

3. Фрагменты делятся на варианты.

4. Элементарной единицей управляемой части документа является поле.

5. Для управления поведением документа используются несложный, понятный язык разметки и подсистема полей, позволяющие легко отделить логику от содержимого и оформления.

6. Сам документ либо его любая составляющая могут быть произвольно изменены пользователем, не знакомым с языком разметки документа и принципами построения этого документа.

7. Адаптивный документ формирует некую структуру машинно-обрабатываемых данных, которые в дальнейшем могут индексироваться, их можно анализировать, создавать на их основе другие документы, строить по ним статистические отчеты и т. п.

8. Для редактирования и формирования документов используется WYSIWYG (What You See Is What You Get) [4] редактор.

9. Текст документа может быть открыт и изменен любым текстовым редактором.

Основная сложность в реализации адаптивных документов заключается в том, что они должны работать в WYSIWYG режиме с сохранением сложного форматирования, использования существующей объектной модели документа, состоящей из рисунков, ссылок, списков и т.д. [5], а также необходимостью придерживаться стандартов: Rich Text Format Specification 1.7:2001 и PDF/A (ISO 19005-1:2005) для архивного хранения документов.

Для реализации таких документов используется древовидная структура, листьями которой являются поля, синхронизированные через подсистему управления поведением – набор методов по обеспечению работы функций. Т.к. данная структура должна помнить все свои изменения и быстро «самоперестраиваться» в соответствии с заданной логикой, используются структуры типа VList и Hashed Array Tree [6] для каждой группы документа и каждого фрагмента. Для обеспечения морфологии естественного языка (русского) применяются различные функции по склонению слов, сложных предложений, фамилий, работе с числительными. Для совместимости документов с другими программами структура представлена в XML-формате и располагается в комментариях RTF – документа ({*\fields … }, {*\fragments … }. Тем самым можно обеспечить экспорт текста документа для использования его в других текстовых редакторах.

Для того, чтобы обеспечить простоту восприятия документа с полями, а также чтобы подчеркнуть то, что текущая часть документа является основной и может влиять на остальной документ, применяются специальные маркеры (‹ Unicode 0x2039: Single Left-Pointing Angle Quotation Mark и › Unicode 0x203A: Single Right-Pointing Angle Quotation Mark). Кроме того применяется цветовая разметка полей, чтобы пользователь мог определить может ли он изменить ту или иную часть документа, либо ее изменение возложено на программную часть. Безусловно, изначально логика документа должна быть задана расстановкой полей и настройкой фрагментов, что производиться во встроенном редакторе.

Реализация описанного выше подхода представляет собой клиент-серверное приложение и WYSIWYG редактор, разработанные в IDE Embarcadero Developer Studio 2007. Библиотеки, отвечающие за хранение базы данных и отрисовку RTF-документа (продукт компании The Imaging Source Europe GmbH), написаны на Microsoft Visual Studio C++). Для доступа к данным и реализации распределенной базы данных используется СУБД Firebird 2.1 либо Borland Interbase 2009.

Таким образом, предложенная концепция адаптивных документов может использоваться в системах автоматизации документооборота и делопроизводства, где требуется организовать работу со слабоструктурированными данными. Также возможно применение в системах синхронного перевода текста документа. Во время заполнения документа производиться автоматический перевод введенной информации на иностранные языки, с обеспечением грамматики языка, на который производится перевод. Перспективным направлением для дальнейшей работы представляется применение модели SaaS (Software as a service [7]), где любой пользователь вне зависимости от используемого

настольного решения для подготовки документов сможет использовать подсистему адаптивных документов.

В качестве простого примера адаптивного документа можно привести следующее (рис 5,6,7).

Ввод ФИО участников производиться в именительном падеже, а количество указывается только числом. Подчеркнутый текст был автоматически изменен после ввода.

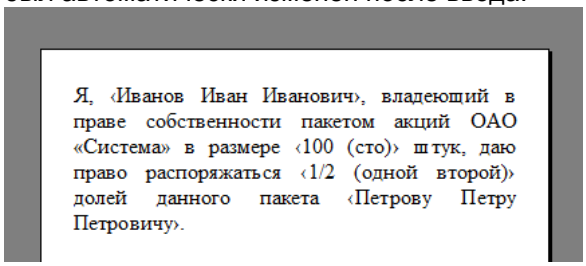
Я, ‹Иванов Иван Иванович›, владеющий в праве собственности пакетом акций ОАО «Система» в размере ‹100 (сто)› штук, даю право распоряжаться ‹1/2 (одной второй)› долей данного пакета ‹Петрову Петру Петровичу›.

Рисунок 5 - Пример исходного документа с введенной информацией

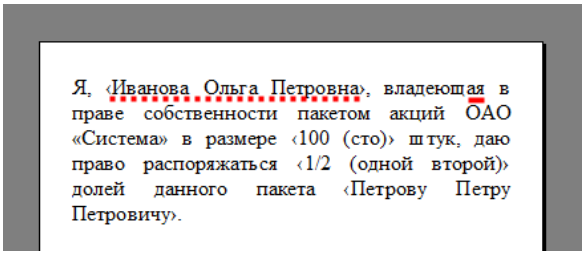
Я, ‹Иванова Ольга Петровна›, владеющая в праве собственности пакетом акций ОАО «Система» в размере ‹100 (сто)› штук, даю право распоряжаться ‹1/2 (одной второй)› долей данного пакета ‹Петрову Петру Петровичу›.

Рисунок 6 - Изменение ФИО приведет изменению окончаний

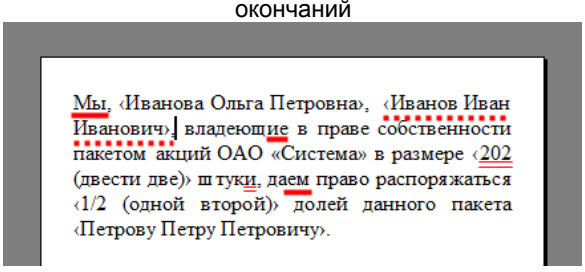
Мы, ‹Иванова Ольга Петровна›, ‹Иванов Иван Иванович›, владеющие в праве собственности пакетом акций ОАО «Система» в размере ‹202 (двести две)› штуки, даем право распоряжаться ‹1/2 (одной второй)› долей данного пакета ‹Петрову Петру Петровичу›.

Рисунок 7 - Копирование в буфер обмена, вставка изменение ФИО и количества акций также приведет к изменению зависимых по смыслу окончаний и слов в документе

Описанный подход к созданию адаптивных документов был применен в сфере автоматизации нотариального делопроизводства, т. к. в ней образовалась большая ниша из-за малого количества программных продуктов, полностью отвечающих требованиям работы с нотариальными документами. Она была реализована в программном продукте «Нотариат» (Свидетельство об официальной регистрации программы для ЭВМ № 2006611193 от 4 апреля 2006 г.). «Нотариат» – это программа для автоматизации делопроизводства и документооборота нотариуса, представляющая собой сочетание полнофункционального текстового редактора, системы генерации и хранения документов.

Программа «Нотариат» уже используются в большинстве нотариальных контор Алтайского края, Республики Алтай для повседневной работы по подготовке документов.

СПИСОК ЛИТЕРАТУРЫ

1. Miles L. Mathieu, Ernest A. Capozzoli, The Paperless Office: Accepting Digitized data - Troy State University, 2002

2. Kevin Craine Excerpts from Designing a Document Strategy - Craine Communications Group, 2006

3. Pedauque, R. T. Document: Form, Sign and Medium, as Reformulated for Electronic Documents - http://archivesic.ccsd.cnrs.fr

4. Chamberlin, Donald D. Document convergence in an interactive formatting system - IBM Journal of Research and Development, 1987

5. Le Hégaret, Philippe The W3C Document Object Model (DOM), World Wide Web Consortium, 2002

6. Bagwell, Phil, Fast Functional Lists, Hash-Lists, Deques and Variable Length Arrays – EPFL, 2002

7. Finch, Curt The Benefits of the Software-as-a-Service Model - http://www.computerworld.com, 2006