Московский Государственный Университет имени М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Системного Программирования

Выпускная квалификационная работа

Методы и инструменты разметки требований в многоверсионных текстовых документах

Автор::
гр. 428
Зинченко Дмитрий Александрович

Научный руководитель:
профессор,
Петренко Александр Константинович

Москва, 2015

Аннотация

Методы и инструменты разметки требований
в многоверсионных текстовых документах

Зинченко Дмитрий Александрович

В данной работе исследуется и проектируется алгоритм поиска соответствий фрагментов текста, выделенных в старой версии документа, фрагментам новой версии того же документа. Проводится оценка работы алгоритма на тестовой базе и сравнение с существующим алгоритмом, использующимся в системе управления требованиями Requality

Abstract

Methods and instruments of requirement marking up
in multiversion documents

Zinchenko Dmitrii

Abstract in english

Содержание

1 Введение

Этот раздел на данный момент состоит только из набросков, и в дальнейшем изменится

Выявление и отслеживание требований является важной частью проектирования программных систем, при этом отслеживание выполнения требований остается важным на протяжении всего жизненного цикла системы.

Здесь можно привести диаграмму цикла разработки программной системы (например, итеративной), и указать, что согласование с требованиями, составленными на этапе планирования, происходит на всех последующих

При разработке программных систем и стандартов очень важную роль играет написание документации. Однако не менее важной является возможность проверить, удовлетворяет ли текущая версия программы документам, описывающим её, каково покрытие тестами требований, изложенных в документации, или же соответствует ли программный продукт стандарту.

Для этих целей и служат системы управления требованиями, позволяющие в удобной форме выделить участки текста, соответствующие требованиям, выполнение которых необходимо отслеживать, задать связи между различными фрагментами текста (сгруппировать требования), привязать конкретные тесты к требованиям и проверить, насколько множество тестов покрывает множество требований документа.

Одной из систем управления требованиями является Requality, разрабатываемая в Институте Системного Программирования РАН. Особенностью данной системы является возможность работать с многоверсионными текстовыми документами, и, в частности, осуществлять перенос требований между версиями одного и того же документа. Исследованию задачи переноса требований между версиями документа и посвящена эта работа.

1.1 Используемая терминология

Для проектирования разрабатываемой системы и постановки задачи введем следующие термины:

- **Документ** – текстовый файл, являющийся синтаксически корректным XHTML документом.

- **Теги требования** – пара тегов ** ** , где *** - идентификационный номер требования. Помимо этого, в Requality после открывающего тега требования и перед текстом внутри него, может находиться пара тегов **, позволяющая интерфейсу системы определять, на каком фрагменте текста центрировать редактор документов при выборе требования.

- **Фрагмент текста** – участок документа, ограниченный тегами требования. Фрагмент не может содержать XHTML теги внутри себя.

- **Требование** – логическое объединение фрагментов текста, ограниченных тегами требований с одинаковыми идентификационными номерами. Идентификационный номер требования – номер, использующийся в тегах требования, соответствующих ему. Идентификационный номер требования уникален, двух разных требований с одним номером не существует.

- **Исходный документ** – документ, содержащий содержащий по крайней мере один фрагмент текста. Неформально - документ, разметку требований из которого необходимо перенести.

- **Конечный документ** – документ, не содержащий разметку требований. В общем случае конечный документ может быть никак не связан с исходным.

- Часть конечного документа считается **полностью соответствуещей** фрагменту текста исходного документа, если содержимое фрагмента текста исходного документа совпадает с частью конечного документа с точностью до незначащих символов и тегов XHTML. В противном случае часть конечного документа**не соответствует** содержимому фрагмента исходного документа.

- Под **переносом фрагмента требования** мы будем понимать перенос тегов требования, ограничивающих этот фрагмент, из исходного документа в конечный.

- Под **переносом требования** мы будем понимать перенос всех его фрагментов, для которых в конечном документе было найдено полное соответствие.

- **Незначащими символами** будем считать пробелы, переносы строк и знаки препинания.

Используя введенные определения, формально поставим решаемую задачу.

2 Постановка задачи

В рамках дипломной работы необходимо разработать библиотеку, позволяющую осуществлять перенос требований между двумя версиями документа.

Здесь, по всей видимости, нужно добавить, что задача решается в предположении, что исходный и конечный документ имеют схожую структуру разделов и подразделов, а так же что большая часть заголовков в новой версии документа остается неизменной, но пока я не придумал, как грамотно это изложить в постановке задачи

В поставленной задаче не рассматривается возможность переноса требований, для которых не удалось найти полного соответствия, но которые возможно перенести, если ввести функцию, позволяющую определять меру частичной схожести фрагмента требования и участка конечного текста. Причиной этому является то, что лишь около 5% требований в тестовой базе документов подпадают под случай "частичного соответствия". Однако определенные наработки в этой области есть, и они приведены в отдельном разделе.

3 Проектирование системы

3.1 Возможные сценарии использования библиотеки

3.1.1 Извлечение требований из исходного документа

- Получение разметки по документу Под разметкой документа мы будем понимать XHTML структуру документа, необходимую для его визуализации. Одним из инструментов, позволяющих извлекать такую структуру из текстового файла, является библиотека JDOM.

  Входные данные: имя файла исходного документа

  Выходные данные: дерево, соответствующее разметке документа, содержащее в вершинах всю информацию, необходимую для визуализации и дальнейшей работы с ним

  Наверное, здесь хорошо бы добавить картинку

- Получение списка требований по дереву документа Необходимо для визуализации требований, и навигации по документу.

  Входные данные: дерево, соответствующее разметке документа

  Выходные данные: Список требований(объектов), каждое требование содержит идентификационный номер и список фрагментов текста, содержащихся в нем

3.1.2 Синхронизация объектов

Под синхронизацией подразумевается возможность получения объекта одного типа данных по объекту другого типа

- Синхронизация по фрагменту требования

  Входные данные: требование, содержащее искомый фрагмент; порядковый номер фрагмента в требовании; дерево, соответствующее разметке документа

  Выходные данные: вершина дерева разметки, соответствующая структурному элементу (разделу, секции) исходного документа, содержащему данный фрагмент.

- Синхронизация двух документов по элементу дерева исходного документа

  Входные данные: дерево разметки исходного документа; дерево разметки конечного документа; вершина дерева разметки исходного документа.

  Выходные данные: вершина дерева разметки конечного документа, соответствующая исходной, если таковая была найдена.

- Получение пути к вершине исходного дерева

  Входные данные: дерево разметки исходного документа; вершина дерева разметки исходного документа.

  Выходные данные: Последовательность вершин дерева исходного документа, определяющая путь к заданной вершине.

3.1.3 Перенос требований и фрагментов текста между версиями документа

- Перенос фрагмента требования

  Входные данные: исходный документ; конечный документ; деревья разметки исходного и конечного документов; фрагмента текста требования.

  Выходные данные: сообщение с диагностикой переноса; список позиций, в которые возможен перенос.

  Список позиций, в которые возможен перенос, задает множество мест, в которые фрагмент требования может быть перенесен. При этом, если позиций не найдено, то теги фрагмента не переносится, если найдена ровно одна позиция, то теги фрагмента переносятся в автоматическом режиме, если позиций найдено несколько, то для переноса тегов требуется подтверждение пользователя.

- Перенос требования

  Входные данные: исходный документ; конечный документ; деревья разметки документов; требование.

  Выходные данные: сообщение о результате переноса требования; список результатов переноса для фрагментов требования.

3.2 Использование библиотеки JDOM

Важным этапом работы любого алгоритма, решающего поставленную задачу, является получение и хранение содержимого исходного и конечного документов. Одним из инструментов, позволяющих это сделать, является библиотека JDOM, используя которую можно получить дерево Java-объектов по XML документу.

Помимо этого, JDOM предоставляет средства для редактирования XML документов, что является лучшим способом добавить разметку в конечную версию документа после нахождения соответствий фрагментам требований.

Пример работы? Что еще хорошо бы здесь рассказать про JDOM?

Статья http://www.ibm.com/developerworks/ru/library/j-jdom/

3.3 Обзор решения, применяемого в Requality

3.3.1 Основная идея решения

Решение проблемы переноса разметки требований, использующееся на данный момент в Requality, основанно на сравнении исходного и конечного документа без XHTML разметки, как строк текста. Для сравнения используется библиотека google-diff, в качестве результата сравнения двух текстов возвращающая список объектов Diff, каждый из которых содержит описание операции, которую нужно выполнить с текстом исходного документа.

Статья https://neil.fraser.name/software/diff$_m$atch$_p$atch/myers.pdf описывает ключевой алгоритм, исользующийся в google-diff

Операции имеют формат *(<Тип операции>, <Текст>)*, где тип операции - одна из трех команд "EQUAL" "DELETE" "INSERT" , а Текст - часть текста исходного документа, над которой операцию необходимо совершить. Гарантируется, что при последовательном выполнении операций из списка из текста исходного документа получится текст конечного.

Здесь должен быть пример работы Diff

3.3.2 Алгоритм

Первоначально исходный документ преобразуется в JDOM дерево, и из него извлекаются все выделенные фрагменты требований, которые нужно перенести. Затем осу-

ществляется преобразование исходного и конечного документа в текстовые данные без XHTML разметки, при этом позиции в исходном тексте извлеченных ранее фрагментов требований запоминаются. Два полученных текста подаются на вход Diff алгоритму, который возвращает список операций, описывающих разницу между исходным и конечным документом.

После окончания работы diff алгоритма для каждого фрагмента требования из исходного текста определяется, находился ли он в блоке EQUAL результата работа Diff, и если да, то осуществляется перенос соответствующих тегов требования в JDOM дерево конечного документа. В противном случае считается, что фрагмент перенести не удалось.

Нужно ли описывать, как происходит добавление тегов требования? Скорее всего здесь - нет, поскольку я не углуюляюсь в детали реализации. Но пи описании моего алгоритма о восстановлении разметки в конечном документе нужно сказать больше, потому чо это оказалось довольно нетривиальной частью.

Таким образом, алгоритм можно представить следующим обзаром:

Здесь, предположительно, должна быть диаграмма, наглядно показывающая то, что описано ранее. Или текстового описания достаточно?

3.3.3 Преимущества и недостатки

Преимуществом по сравнению с предложенным мной решением является независимость получения результата от разметки документов. То есть, если в новой версии документа используется другая разметка, на работу алгоритма она не повлияет.

Недостатками, по всей видимости, являются две вещи - неидеальность работы diff алгоритма в целом, за счет использования эвристик, ускоряющих алгоритм (к сожалению, пока информации по этой теме мне найти не удалось), и то, что такие вещи, как перестановка разделов/текста внутри разделов гарантировано испортит работу этого алгоритма.

Помимо этого, есть фрагменты, на которых алгоритм должен работать, но по каким-то причинам, которые мне не известны, не работает. Примером такой проблемы является отсутствие переноса требований в заголовках разделов в некоторых документах

3.4 Идея предлагаемого решения

Основной идеей решения поставленной задачи, которое описывается в данной работе, является локализация места конечного документа, в котором осуществляется поиск соответствия фрагменту, выделенному в исходном документе. В качестве места, в котором нужно осуществлять поиск, выбирается подраздел максимальной глубины такой, что список заголовков, позволяющий однозначно определить его в конечной версии документа, совпадает с точностью до незначащих символов и цифр с списком заголовком, позволяющим найти в исходном документе подраздел, содержащий фрагмент, для которого осуществляется перенос.

После проведения этой операции размер текста, в котором нужно найти соответствие фрагменту требования, считается достаточно маленьким для применения алгоритма прямого поиска. В том случае, если совпадение с текстом фрагмента было найдено, осуществляется перенос разметки в соответствующее место конечного документа.

3.5 Внутреннее представление документа

Однако DOM-дерево, полученное из исходного и конечного документов, не полностью удовлетворяет потребностям такого алгоритма переноса - текст раздела в таком дереве не привязан к заголовку, а заголовки, являющиеся вложенными в документе, в DOM-дереве находятся на одном уровне иерархии.

Поэтому для удобства работы с иерархией разделов документов было решено осуществлять преобразование DOM-деревьев документов в деревья другого типа, позволяющие проще отслеживать вложенность разделов и связь текстов разделов с их заголовками.

Здесь должна быть картинка, демонстрирующая разницу между документом, представленным в виде DOM-дерева, и в виде другого дерева

Основываясь на введенном новом внетреннем представлении документа, введем следующие понятия:

Путь до раздела документа - список вершин пути в дереве внутреннего представления документа до вершины, соответствующей разделу.

Путь до фрагмента требования - путь до раздела максимальной глубины, в котором содержится этот фрагмент.

3.6 Алгоритм

Таким образом, алгоритм переноса разметки требований между версиями документа выглядит следующим образом:

1. Извлечь содержимое исходного и конечного документов при помощи JDOM
2. Преобразовать DOM-деревья документов в деревья внутреннего представления
3. Извлечь требования из дерева исходного документа
4. Для каждого фрагмента требования:
   (a) Получить путь до фрагмента требования
   (b) Получить по дереву конечного документа путь до соответствующего раздела
   (c) Найти вхождение текста фрагмента в полученном разделе.
   (d) Если путь в конечном документе или вхождение текста фрагмента требования в разделе конечного документа не было найдено, считать, что перенос фрагмента осуществить не удалось.
   (e) В противном случае - изменить структуру раздела конечного документа, добавив в неё требование
   (f) Изменить структуру соответствующей разделу части DOM-дерева конечного документа
5. Изменить конечный документ в соответствии с изменением структуры его DOM-дерева

3.7 Преимущества и недостатки

4 Устройство системы

По всей видимости, здесь нужно описывать конкретно мою реализацию - устройство модулей, формат используемых объектов

5 Анализ полученных результатов

6 Заключение