



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, используемым в системе управления требованиями Requality

## **Abstract**

### **Methods and instruments of requirement marking up in multiversion documents**

*Zinchenko Dmitrii*

Abstract in english

# Содержание

<b>1</b>	<b>Введение</b>	<b>5</b>
1.1	Используемая терминология . . . . .	5
<b>2</b>	<b>Постановка задачи</b>	<b>8</b>
<b>3</b>	<b>Проектирование системы</b>	<b>9</b>
3.1	Возможные сценарии использования библиотеки . . . . .	9
3.2	Использование библиотеки JDOM . . . . .	10
3.3	Обзор решения, применяемого в Requality . . . . .	11
3.3.1	Основная идея . . . . .	11
3.3.2	Алгоритм . . . . .	11
3.3.3	Преимущества и недостатки . . . . .	12
3.4	Предлагаемое решение . . . . .	12
3.5	Внутреннее представление документа . . . . .	13
3.6	Алгоритм . . . . .	13
3.7	Преимущества и недостатки . . . . .	14
<b>4</b>	<b>Устройство системы</b>	<b>15</b>
4.1	Дерево структурной разметки документа . . . . .	15
4.2	Построение дерева структурной разметки по DOM модели . . . . .	15
4.3	Устройство требования . . . . .	15
4.3.1	Location . . . . .	15
4.3.2	ActualLocation . . . . .	15
4.3.3	Построение ActualLocation-ов . . . . .	15
4.4	Извлечение текста по вершине дерева структурной разметки . . . . .	15
4.4.1	Текст заголовка . . . . .	15
4.4.2	Текст раздела . . . . .	15
4.5	Устройство переноса разметки требований . . . . .	15
4.5.1	Поиск соответствующего пути в конечном документе . . . . .	15
4.5.2	Использование регулярных выражений . . . . .	15
4.6	Добавление разметки требований дерево разметки . . . . .	15
4.7	Добавление разметки требований в DOM модель . . . . .	15

<b>5</b>	<b>Анализ полученных результатов</b>	<b>16</b>
<b>6</b>	<b>Заключение</b>	<b>17</b>

# 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 `<span class="requality_text id_***"> </span>`, где \*\*\* - идентификационный номер требования. Помимо этого, в Requality после открывающего тега требования и перед текстом внутри него, может находиться пара тегов `<a name="***" id="***" class="requality_id"></a>`, 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 по крайней мере один фрагмент текста. Неформально - документ, разметку требований из которого необходимо перенести.
- **Конечный документ** – документ, для которого разметка требований не определена. В общем случае конечный документ может быть никак не связан с исходным.
- Часть конечного документа считается **полностью соответствующей** 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.
- **Переносом требования** - перенос всех его фрагментов, для которых в конечном документе было найдено полное соответствие.
- **Незначащие символы** - пробелы, переносы строк и знаки препинания.

- Структурная разметка документа - теги заголовков и абзацев

## 2 Постановка задачи

В рамках дипломной работы необходимо разработать библиотеку, позволяющую осуществлять перенос требований из исходного документа в конечный.

Стоит отметить, что в поставленной задаче считается, что оба документа имеют схожую структурную разметку и название подразделов, в которых находятся фрагменты требований.

Перенос требований для которых не найдено полного соответствия выходит за рамки данной работы, хотя и может быть осуществлен путем построения функции схожести фрагментов текстов. Одна из причин этого - то что только около 5% требований в тестовой базе документов могут быть перенесены подобным образом.



## 3 Проектирование системы

### 3.1 Возможные сценарии использования библиотеки

- Получение объектной модели документа Одним из инструментов, позволяющих извлекать DOM модель документа из текстового файла, является библиотека JDOM (в работе используется версия 2.0.6).

Получение по имени файла исходного документа структуры данных (дерева), соответствующего объектной модели документа.

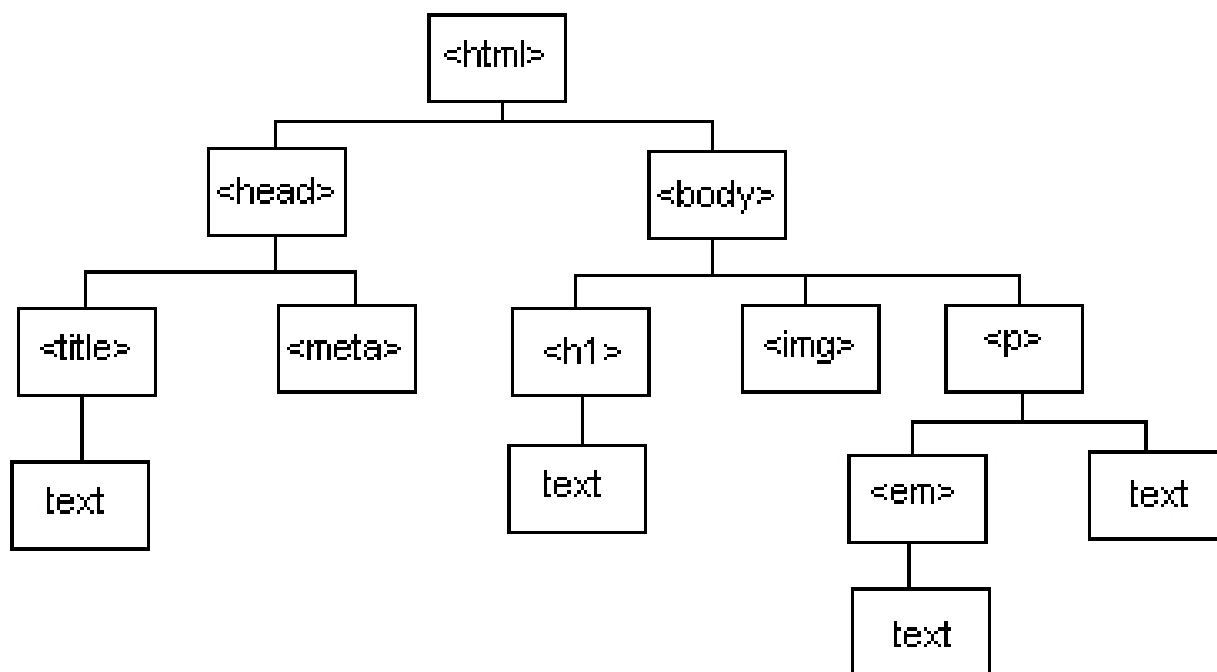


Рис. 1: Пример DOM дерева документа

- Выделение структурной разметки документа из его DOM дерева

Осуществляется преобразование DOM модели документа в дерево, отражающее его структурную разметку.

- Получение списка требований по дереву документа Выделение фрагментов требований из дерева с дальнейшей группировкой по id.

Получение по дереву структурной разметки документа списка требований(объектов), каждое требование содержит идентификационный номер и список фрагментов текста, содержащихся в нем.

- Синхронизация по фрагменту требования

По фрагменту требования находится вершина дерева разметки исходного документа, в тексте которой содержится данный фрагмент.

- Получение пути в дереве разметки от корня до элемента

По дереву разметки документа осуществляется поиск последовательности вершин - пути в дереве от корня до данной вершины

- Поиск элемента, соответствующего данному, в дереве другого документа

По элементу дерева разметки одного документа осуществляется поиск элемента разметки другого документа так, что пути из корней деревьев до элементов имеют одинаковые типы вершин и схожий текст.

- Перенос фрагмента требования
- Перенос 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 DOM-объектов по XML документу. Также JDOM позволяет эффективно получить все фрагменты требований из исходного документа.

Помимо этого, JDOM предоставляет средства для редактирования XML документов, что является лучшим способом добавить разметку в конечную версию документа после осуществления переноса фрагментов требований.

*Статья <http://www.ibm.com/developerworks/ru/library/j-jdom/> описывает возможности JDOM*

### 3.3 Обзор решения, применяемого в Requality

#### 3.3.1 Основная идея

Решение проблемы переноса разметки требований, использующееся на данный момент в Requality, основанно на сравнении исходного и конечного документа без XHTML разметки, как строк текста (plain text). 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\\_match\\_patch/myers.pdf](https://neil.fraser.name/software/diff_match_patch/myers.pdf) описывает ключевой алгоритм, использующийся 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 DOM дерево, и из него извлекаются все выделенные фрагменты требований, которые нужно перенести. Затем осуществляется 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 работы Diff, и если да, то осуществляется перенос соответствующих тегов требования в DOM дерево конечного документа. В противном случае считается, что фрагмент перенести не удалось. Обновленный конечный документ восстанавливается по измененному DOM дереву.

Таким образом, алгоритм можно представить следующим образом:

*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 Предлагаемое решение

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, в конечном документе выбирается подраздел максимальной глубины такой, что тип и текст вершин пути до вершины подраздела, содержащего фрагмент в исходном документе, совпадает с путем до него (текст - с точностью до незначащих символов и цифр).

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 удовлетворяет потребностям такого алгоритма переноса полностью - текст раздела в таком дереве не привязан к заголовку, а заголовки, являющиеся вложенными в документе, в DOM дереве находятся на одном уровне иерархии. Таким образом, DOM модель не отражает структурную разметку документа.

Поэтому для удобства работы с иерархией разделов документов было решено осуществлять построение на основе DOM моделей документов дерева, 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 представлении документа, определим следующие понятия:

**Путь до раздела документа** - список вершин пути в дереве разметки документа до вершины, соответствующей разделу.

**Путь до фрагмента требования** - путь до вершины дерева разметки максимальной глубины, соответствующей разделу, 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. Построить на основе DOM деревьев документов деревья структурной разметки
3. Извлечь требования из дерева исходного документа
4. Для каждого фрагмента требования:
  - (a) Получить путь до фрагмента требования
  - (b) Получить по дереву разметки конечного документа путь до соответствующего раздела

- (с) Найти вхождение текста фрагмента в полученном разделе.
  - (d) Если путь в дереве разметки конечного документа или вхождение текста фрагмента требования в разделе конечного документа не было найден, считать, что перенос фрагмента осуществить не удалось. В противном случае - изменить структуру части дерева разметки, соответствующей разделу конечного документа, добавив в неё требование
  - (е) 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

### 4.1 Дерево структурной разметки документа

### 4.2 Построение дерева структурной разметки по DOM модели

### 4.3 Устройство требования

#### 4.3.1 Location

#### 4.3.2 ActualLocation

#### 4.3.3 Построение ActualLocation-ов

### 4.4 Извлечение текста по вершине дерева структурной разметки

#### 4.4.1 Текст заголовка

#### 4.4.2 Текст раздела

### 4.5 Устройство переноса разметки требований

#### 4.5.1 Поиск соответствующего пути в конечном документе

#### 4.5.2 Использование регулярных выражений

### 4.6 Добавление разметки требований дерево разметки

### 4.7 Добавление разметки требований в DOM модель

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

## 5 Анализ полученных результатов



## 6 Заключение