САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра информатики

Поиск клонов в XML-документации

Дипломная работа студента 542 группы

Копина Дмитрия Валерьевича

Научный руководитель	…… / подпись /	к. ф.-м.н., доцент Кознов Д.В.
Рецензент	…… / подпись /	ст. преподаватель Смирнов М.Н.
“Допустить к защите” заведующий кафедрой,	…… / подпись /	д.ф.-м.н., проф. Косовский Н.К.

Санкт-Петербург
2013

SAINT PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Chair of Computer science

Clone detection in XML documentation

by

Dmitrii Kopin

Graduation Paper

Supervisor	……	PhD, Associated Professor D. V. Koznov
Reviewer	……	Senior Lect. M. N. Smirnov
"Approved by" Head of Department	……	Professor N. K. Kosovskiy

Saint Petersburg
2013

Оглавление

Введение

В современных проектах по разработке промышленного программного обеспечения возникает множество задач, столь же важных, как и непосредственно программирование, и одна из них – это разработка технической документации. Существует целый спектр программных средств, обеспечивающих разработку, как одиночных документов, так и масштабных пакетов документов. При этом документация может представляться в различных форматах – в виде встроенных справочных систем (WinHelp, HTML), печатная (PDF, MS Word). Так же для ведения документации получили широкое распространение различные форматы, основанные на XML. Одним из популярных представителей таких технологий является формат DocBook [25], являющийся де-факто open-source стандартом в области разработки документации для Unix/Linux-приложений.

Часто встречается ситуация, когда в рамках ряда схожих программных приложений или их версий создаётся похожая документация. Многократные повторы одной и той же информации затрудняют дальнейшее сопровождение такой документации – необходимо отслеживать повторяющиеся фрагменты текста и изменять их синхронно. Рост числа похожих документов, а также их объёмов, приводит к нехватке ресурсов на качественное сопровождение документации и, соответственно, к снижению их качества. Поэтому для облегчения сопровождения и разработки документации необходимо применять подход повторного использования, активно задействованный и в разработке ПО.

Решению этой задачи посвящена технология DocLine, созданная и развиваемая на кафедре системного программирования СПбГУ [2], [4], [5], [6], [7], [8], [20], [23]. На основе формата DocBook был создан язык DRL для организации повторного использования фрагментов текста, а также разработан комплекс программных средств, включающий в себя средства рефакторинга документации [2]. Последние предназначаются для улучшения структуры повторного использования документации в процессе её сопровождения.

Для автоматического поиска повторяющихся фрагментов в DocLine, а также для дальнейшего использования этих фрагментов в автоматизированном рефакторинге было разработано специальное средство [3] на базе пакета по поиску клонов CloneMiner [11]. Однако созданное средство при использовании для XML-текстов приводило к потере значимых результатов (до одной трети повторов пропускалось), а также большому количеству ложных срабатываний. Кроме того имелись недостатки, связанные с недостаточной гибкостью операции (нет прямой возможности менять параметры поиска) и неудобным предоставлением пользователю полученных результатов.

Постановка задачи

Целью данной дипломной работы является доработка алгоритма и программных средств DocLine для поиска клонов в документации. Для достижения этой цели в рамках работы были сформулированы следующие задачи.

1. Знакомство с предметной областью (формат DocBook, DocLine, модуль поиска клонов, CloneMiner).
2. Уулучшение алгоритма поиска XML-клонов и фильтрации полученных результатов в DocLine для XML-текстов (DocBook и DRL).
3. Проведение экспериментов и тестирования разработанного алгоритма.

1.Обзор

1.1 Технология DocLine

Эта технология предназначена для разработки документации семейств программных продуктов и разрабатывается на кафедре системного программирования математико-механического факультета СПбГУ[1]. Технология состоит из следующих частей:

- процесс разработки документации;
- язык разработки документации DRL (Document Reuse Language);
- пакет инструментальных средств [2, 23].

Для разметки документов DocLine использует известный XML-язык DocBook [25]. DocBook является де-факто стандартом для разработки документации к свободно распространяемому программному обеспечению, в частности, широко применяясь в мире Linux. DocLine успешно показала себя для организации повторного использования документации ПО семейства телефонных станций [5].

Основной единицей языка DRL является *информационный элемент* – фрагмент документации, который может включаться в различные позиции других документов (с помощью ссылки на него). Другой важной конструкцией является *словарь* – структура, состоящая из набора пар «ключ – значение». С помощью этой структуры в тексте документа вместо некоторого небольшого участка текста можно использовать короткий ключ.

В Java-версии все инструменты встроены в среду разработки Eclipse в виде модулей расширения (plug-ins). Графический редактор реализован с помощью технологии Eclipse GMF. GMF используется для создания инструментария проблемно-ориентированных языков. Осуществляется поддержка форматов HTML и PDF. Также в пакет входят вспомогательные модули, которые обеспечивают интеграцию в среду разработки Eclipse.

С помощью DocLine можно располагать в отдельных файлах различные элементы документации, например: информационные продукты и финальные информационные продукты, информационные элементы, каталоги и словари. Среда Eclipse имеет стандартные возможности интеграции с системами контроля

[1] Данный раздел использует работы [2], где были изложены данные результаты.

версий, в частности Subversion, Perforce, Microsoft Visual SourceSafe, и другими. Таким образом осуществляется интеграция DocLine с системами контроля версий, что позволяет организовать коллективную работу над документацией.

1.2 РЕФАКТОРИНГ ДОКУМЕНТАЦИИ В DOCLINE

В технологии DocLine был создан ряд операций рефакторинга, с помощью которых можно изменить представление документации для улучшения структуры, что необходимо для повторного использования. Основные операции рефакторинга DocLine можно сгруппировать следующим образом[2].

Вспомогательные операции. Это группа содержит одну операцию – переход к документации с повторным использованием. При этом осуществляет переход от DocBook формата документа к DRL.

Операции выделения крупных фрагментов текста. Эта группа состоит из операций, позволяющих создавать крупные компоненты повторного использования. Назначение – обеспечить работы с информационными элементами.

Операции выделения небольших фрагментов текст. Данная группа предоставляет функционал для работы с каталогом и словарем.

Операции настройки повторного использования. Эти операции предназначены для подготовки фрагментов текста к повторному использованию в различных контекстах. Обеспечивают работу с точками расширения и ссылками на информационные элементы.

1.3 СРЕДСТВА ПОИСКА КЛОНОВ В DOCLINE

В данном разделе описаны средства поиска клонов в DocLine, которые улучшались в рамках данной дипломной работы[3].

В качестве основного инструментария для поиска клонов использовался инструмент CloneMiner [11]. Работа с данным инструментом может осуществляться из командной строки. Для работы можно задавать следующие параметры: путь к файлу(или файлам),содержащему входные параметры, путь к файлу для записи результатов поиска, минимальный размер клона. Результаты поиска выводятся в указанный файл в виде списка групп клонов. CloneMiner

[2] Данный раздел использует материалы работы [2], где были изложены эти результаты.
[3] Этот раздел использует материалы работы [3], где были изложенные данные результаты.

поддерживает формат Unicode, что позволяет применять инструмент к русскоязычной документации.

В версии, использованной в DocLine, CloneMiner искал клоны в плоском тексте (здесь и далее плоский текст – это текст без учёта XML-разметки). Для того, чтобы выдаваемый результат был корректен с точки зрения XML (а поиск происходил по XML-текстам) была реализована фильтрация. XML-фильтрация позволяет выдавать в качестве повторно используемых фрагментов только корректные, с точки зрения DocBook/DRL конструкции.

Алгоритм работы средств поиска клонов в DocLine представлен на рис 1.

Рис. 1. Алгоритм поиска клонов в старой версии. Рис. взят из работы [3].

Сначала пользователь выбирает в DRL-документе информационный элемент, в котором он хочет искать клоны. Выбранный информационный элемент записывается в отдельный файл, который и подаётся в качестве входного параметра CloneMiner. Результаты работы CloneMiner проходят XML-фильтрацию, чтобы стать корректными с точки зрения DRL и DocBook.

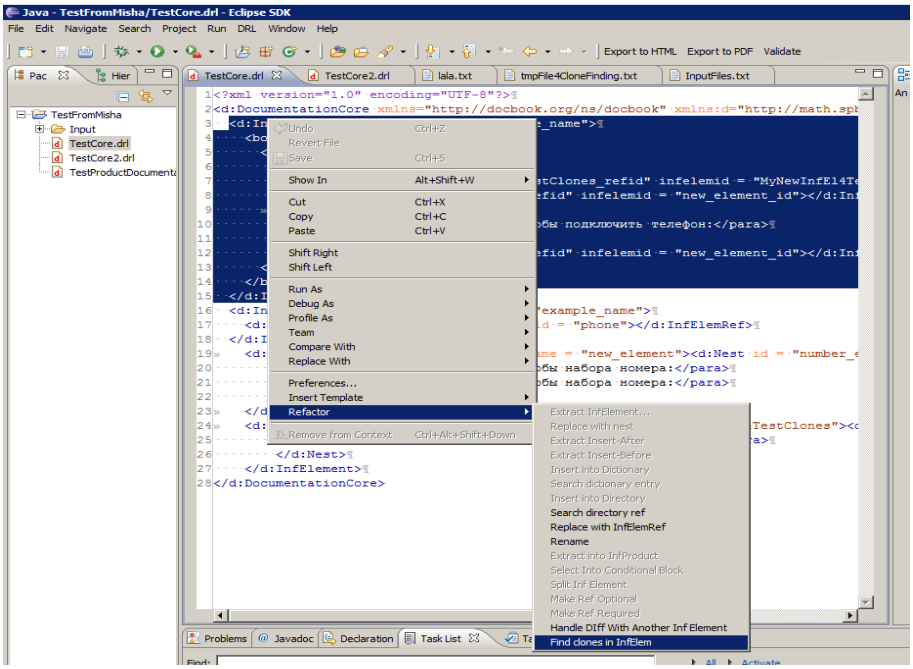

Рис. 2.Окно операций рефакторинга. Рис. взят из работы [3].

На рис. 2 представлено окно редактора DocLine, позволяющее применить операцию поиска клонов для выделенного информационного элемента. Если границы информационного элемента выбраны неправильно, данная операция будет неактивной. После осуществления операции поиска клонов в нижней части экрана появляется вкладка «DocLine clones» (см. рис. 3). Результаты операции отображаются в виде дерева, состоящего из двух уровней. Первый уровень содержит краткое описание найденных групп клонов (текст клона, количество термов в клоне, количество представителей данного клона). Для каждой группы клонов второй уровень содержит список ее представителей (с указанием положения границ в тексте).

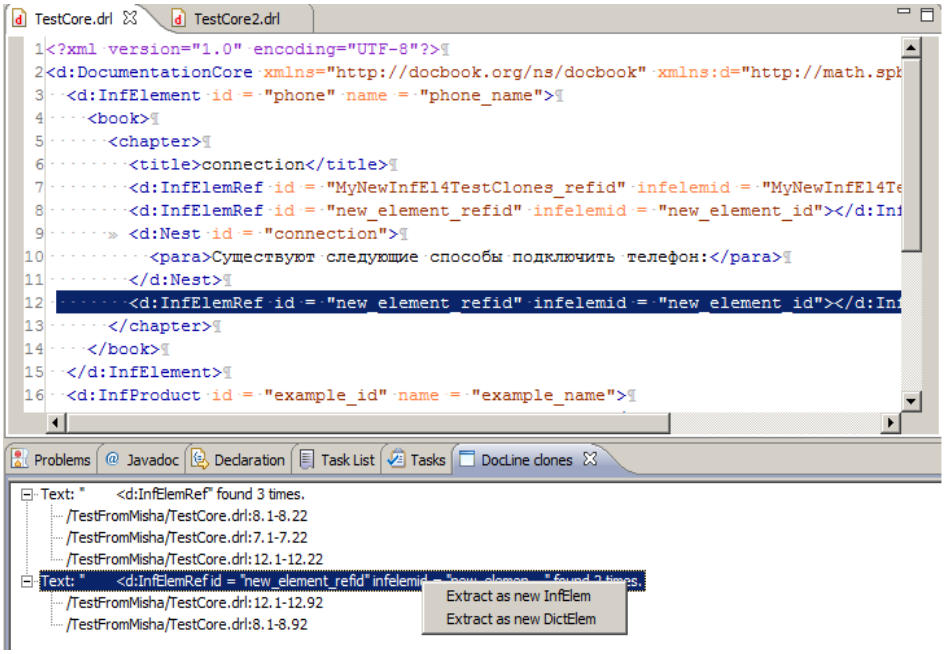

Рис. 3. Работа с выделенными клонами. Рис. взят из работы [3].

При обращении к конкретному представителю данной группы клонов (двойной щелчок мышью на элемент дерева второго уровня) его текст выделяется в окне DRL-редактора.

Дальнейшие действия пользователя направлены на создание элементов повторного использования. Есть возможность выделить сразу всю группу в общий актив. Для этого нужно вызвать всплывающее меню (правый щелчок мыши на элементе первого уровня дерева результатов). Всплывающее меню содержит два вида действий: «Выделить в новый информационный элемент» (пункт меню «Extract as new InfElem») и «Выделить в новый элемент словаря» (пункт меню «Extract as new DictElem»).

2.Улучшение алгоритма поиска клонов

2.1 Отбрасывание пересечений

В ходе знакомства с CloneMiner было замечено, что клоны и группы клонов, найденные CloneMiner, могут содержать пересечения (т.е. два или более представителей содержат общий элемент текста). Данная ситуация может приводит к появлению ошибок – пользователь редактирует один элемент, вынесенный в общие активы, при этом меняется другой элемент. После обработки и XML-фильтрации результаты по-прежнему содержали пересечения. В связи с этим возникла необходимость реализовать функционал, устраняющий пересечения.

Для этого был выбран следующий подход. После окончания обработки результатов (рис 4.) происходит процесс удаления пересечений, путем отбрасывания клонов из результирующего множества. Все найденные клоны формируются в один список, который затем сортируется по порядку вхождения в текст. Затем происходит поиск пересечений. Как только два клона попадают в конфликтную ситуацию (содержат пересечение) происходит удаление одного из них. Для определения того, какой из элементов должен быть исключен из итогового результата была введена функция значимости (полезности). Наиболее содержательных результатов удалось добиться при функции значимости клона определяемой как $m*n^2$, где m– мощность группы клона (количество представителей группы, к которой принадлежит данный элемент), а n – мощность самого клона (количество символов, из которых он состоит). Конечно, предложенная реализация не подразумевает нахождения лучшего (с точки зрения нахождения максимального количества повторно используемой информации) разбиения на группы без пересечений. Однако предложенный алгоритм позволяет выполнить поставленную задачу за разумное время и с приемлемым качеством.

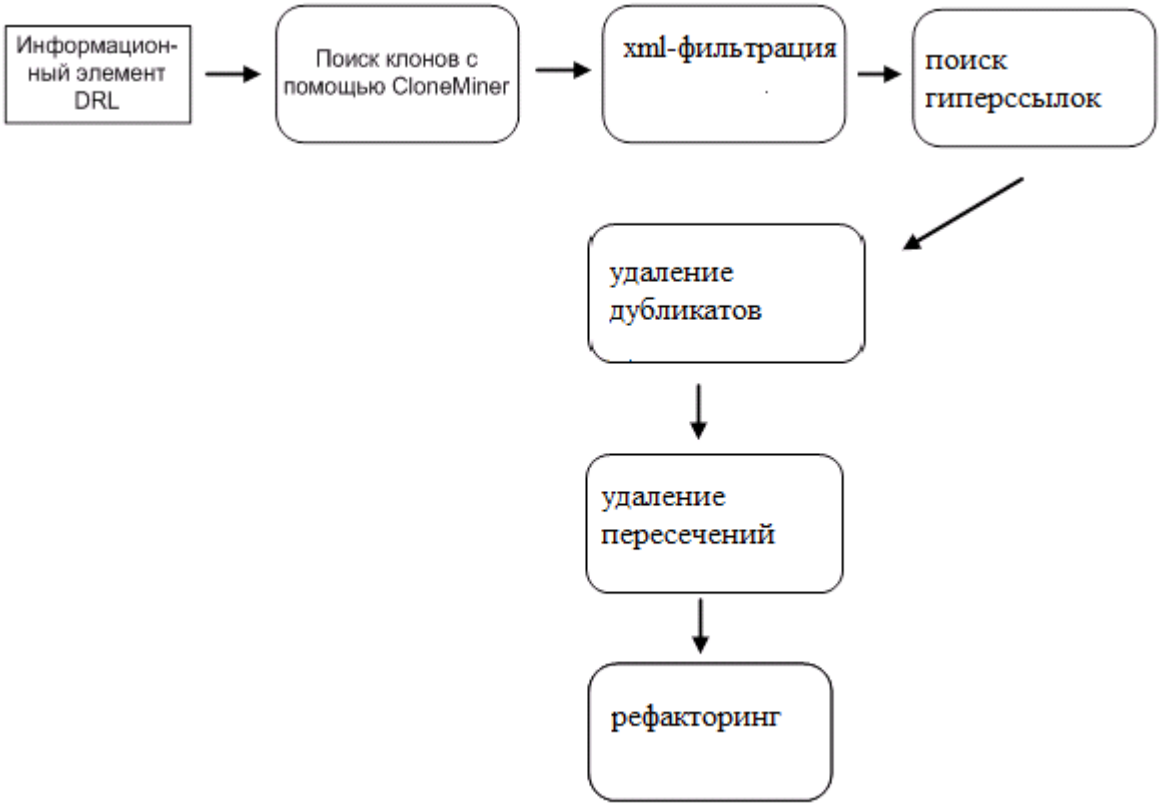

Рис. 4. Модифицированный алгоритм поиска клонов.

2.2 Удаление дубликатов групп

В результате XML-фильтрации в группах может измениться текст представителей или они даже могут распасться на подгруппы. Таким образом, в итоговом результате может оказаться набор групп, содержащих разных представителей с одинаковым текстом. Такой набор целесообразно объединить в одну группу. Для этого был реализован функционал соединяющий группы с одинаковыми представителями в одну. Удаление дубликатов (рис 4.) запускается после формирования первичных результатов (проведена XML-фильтрация и добавлены гиперссылки). Данная операция осуществляется обычной маркировкой списка групп и параллельным формированием результирующего множества.

2.3 Изменение механизма XML-фильтрации

Идея, заложенная в механизм XML-фильтрации, заключалась в том, чтобы выделить максимальные корректные куски XML и сформировать таким образом новые наборы групп.

В результате приведения к валидному XML (все открывающие теги имеют соответствующие соответствующие им закрывающие), а так же в связи с особенностями реализации терялась часть значимой информации. Так, например, в тексте

```
      <screen>  java \
    -Dencoding.windows-1252=com.nwalsh.saxon.Windows1252 \
  com.icl.saxon.StyleSheet \
  mydoc.xml mystylesheet.xsl</screen>

              Or, for a more complete "real world" case showing other
              options you'll typically want to use:
                                                    <screen>  java \
    -Dencoding.windows-1252=com.nwalsh.saxon.Windows1252 \
  mydoc.xml mystylesheet.xsl</screen>
```

не находилась группа клонов, состоящая из фрагментов

```
                                java \
    -Dencoding.windows-1252=com.nwalsh.saxon.Windows1252 \
```

В связи с этим механизм фильтрации был изменен. Так же была добавлена поддержка тегов в сокращенном виде, а также конструкций вида

```
<![CDATA[<Arg type="boolean">False</Arg>]]>
```

2.4 Поиск гиперссылок

Рассмотрим строку:

<ulink url="http://www.w3.org/XML/">http://www.w3.org/XML/</ulink> .

CloneMiner осуществляет поиска клонов разбивая текст на токены. При этом разделителями между токенами считаются символы :

« » , «.» , «,» , «\t» , «\n» , «*» , «\"» , «\'» , «(» , «)» , «:» , «;» .

|<ulink| |url=| " |http| : |//www| . |w3| . |org/XML/| " |>http| : |//www| . |w3| . |org/XML/</ulink>|

Вместо того, чтобы найти строку http://www.w3.org/XML/ в качестве клона, CloneMiner предложит ://www.w3. , что не соответствует желаемому результату. К тому же данный результат содержит всего 2 токена, и при соответствующем параметре может быть утерян. Однако, следует заметить, что искомый фрагмент

является семантически значимой единицей и удачным кандидатом для вынесения в словарь. Поэтому было принято решение осуществлять дополнительный поиск клонов без помощи CloneMiner,а используя стандартные средства, предоставляемые Java.

Алгоритм основан на следующих принципах:

- группы меньшего размера, но содержащие клоны большей длины являются предпочтительными;
- при осуществлении поиска клонов максимального размера не должны теряться клоны, имеющие меньшую длину;
- время работы алгоритма важно: нет смысла ждать десятки секунд при поиске в больших документах, ради потенциального результата в несколько небольших клонов.

Алгоритм работает следующим образом. Сначала формируется стандартный набор префиксов (таких как http://). Затем осуществляется поиск всех вхождений каждого из префиксов. Найденные вхождения формируют набор потенциальных клонов. После этого осуществляется рекурсивный процесс. Для каждого из элемента набора делается просмотр следующего токена (при этом в качестве разделителей берется уже свой, расширенный по сравнению с разделителями CloneMiner набор символов). Те из элементов, которые совпали по новому токену, формируют новую группу и для них процесс поиска продолжается. Оставшиеся элементы группы делятся на два вида: те, которые могут образовать группу клонов с другими элементами текущего уровня (группа формируется и добавляется в итоговый результат) и те, что остались без групп (от них “отщепляется” с конца один токен и они возвращаются на предыдущий уровень). Итоговый результат добавляется к результату работы CloneMiner для последующей обработки.

2.5 Поиск по остаткам

CloneMiner не является пакетом, который находит всю повторно используемую информацию. Для обеспечения скорости выполнения, а так же в связи с особенностями алгоритма, некоторые клоны и группы клонов могли выпасть из результата работы CloneMiner. Так же в результате удаления пересечений могли быть отброшены различные фрагменты текста, которые при другом рассмотрении могли бы содержать клоны. В связи с этим, возникла необходимость реализации механизма повторного запуска CloneMiner.

Для того, чтобы избежать пересечений с уже найденными клонами, было принято решение удалять все найденные клоны из текста и запустить пакет на остаточных фрагментах . При этом для того, чтобы избежать ситуации, когда новые клоны пересекались с уже найденными, старые клоны не просто удаляются, а заменяются на специальным образом сформированные уникальные токены.

Таким образом, текст разбивается на фрагменты. Токены, используемые для разделения, формируются таким образом, чтобы вероятность того, что они встречаются в самом тексте, была близка к 0. В результате работы CloneMiner не найдет клоны, начало и конец которых содержатся в разных фрагментах: это значило бы, что токен-разделитель встретился в тексте как минимум 2 раза.

Затем координаты полученных результатов сдвигаются, чтобы соответствовать первоначальному тексту. Далее происходит XML-фильтрация, поиск гиперссылок, удаление дубликатов и пересечений, но только для результатов, найденных при повторном запуске. Полученные группы клонов добавляются в итоговый результат, возвращаемый пользователю. Полную схему работы можно видеть на рис 5.

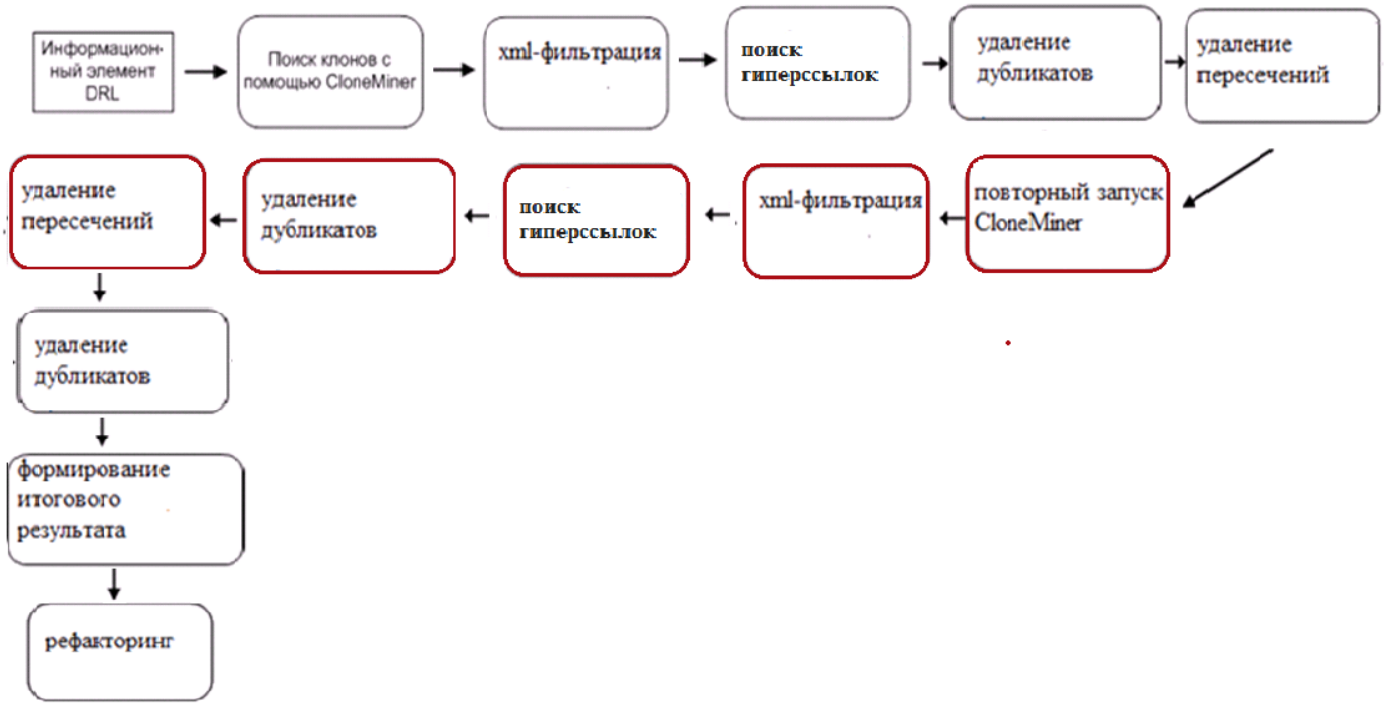

Рис. 5. Алгоритм работы с поиском по остаткам

3.Улучшение пользовательского интерфейса

3.1 Пользовательский функционал

Одной из задач операции рефакторинга “поиск клонов” является предоставление пользователю удобного механизма для поиска элементов, пригодных для повторного использования. При этом возникает необходимость управлять процессом поиска: задавать настройки программы CloneMiner, отключать XML-фильтрацию и поиск по остаткам. Для этого был добавлен диалог с настройками, который можно выбрать щелчком по кнопке ClonePreferences (рис. 6).

Механизм XML-фильтрации приводит к потере данных. У пользователя может возникнуть желание отключить фильтрацию, чтобы самому решать, какой фрагмент он считает корректным и подходящим для выделения в общий актив. Так же процесс фильтрации занимает значительную часть времени поиска клонов (до 30 %), что так же приводит к необходимости предоставить пользователю возможность управления данной функцией.

CloneMiner позволяет выбрать минимальное количество токенов в искомых элементах (клонах), но не позволяет задавать это количество непосредственно из редактора. Пользователь же ведущий документацию, не заинтересован в постоянном изменении файла `.bat`, в котором и указывался данный параметр.

Аналогично, повторный запуск CloneMiner для поиска оставшихся клонов может занимать до 40 % времени поиска. При этом эффективность данной опции зависит от структуры текста и не всегда полученный результат будет соответствовать затратам на время выполнения.

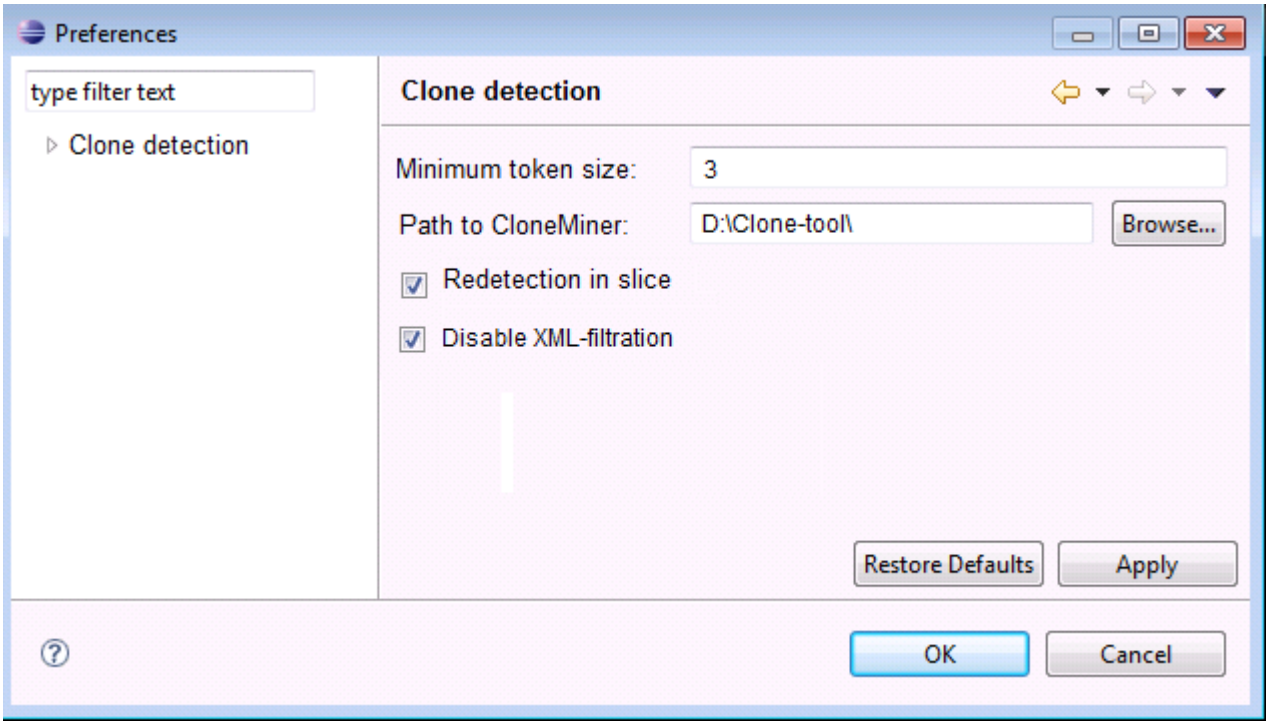

Рис. 6. Окно настроек поиска клонов

3.2 Доработка механизма отображения результатов

После окончания поиска клонов, дальнейший процесс выделения повторно используемых элементов находится полностью под управлением пользователя. Он может просмотреть, какие элементы программа ему предлагает выделить в общие активы, и выбрать те, что он посчитает полезными. В связи с тем, что эта стадия уже выходит за рамки автоматического выполнения, важным элементом становится удобство просмотра пользователем результатов.

По двойному клику на элементе дерева результатов, выбранный клон выделялся в экране редактора. При этом границы выделенного элемента не всегда соответствовали границам, которые были указаны в дереве, а так же не совпадали с реальным расположением клона. Данный дефект происходил в результате неудачной интеграции с CloneMiner – на вход подавался измененный, по сравнению с содержанием экрана, текст.

Так, например

```
<section xml:id="architecture-overview">
```

заменялась на

```
<section xml:id = "architecture-overview">
```
(отступы перед знаком ‘=’).

Другим частым случаем было преобразование последовательности < в соответствующий символ – знак меньше (<). Так же при правильном значении границ клонов, сдвиг порождался наличием символов табуляции в предшествующих началу клона. Данные проблемы были решены с помощью учета различных пробельных символов, а так же изменением интеграции с CloneMiner (изменен способ передачи входного текста и разбор результатов работы).

4. Апробация

4.1 Формирование набора тестовых данных

Для проверки эффективности работы предложенных улучшений, а так же с целью обнаружения недостатков был сформирован набор тестовых данных. В связи с тем, что DocBook широко применяется при разработке документации в Unix/Linux сообществе, удалось найти большой объем документации, которая создана для серьезных продуктов, активно используется, а так же подходит для проведения эксперимента:

Было сформировано 2 тестовых набора:

1) набор для проверки непосредственных улучшений;
2) набор для формирования статистики.

4.1.1. Набор для проверки непосредственных улучшений

Этот набор тестов состоит из текстовых фрагментов, достаточно больших, чтобы содержать информацию, пригодную для повторного использования. При этом фрагменты не очень велики, чтобы можно было проследить прямое соответствие найденных результатов в различных версиях алгоритма. Это необходимо, чтобы выяснить, какие непосредственные улучшения были достигнуты, какие клоны стали распознаваться, а раньше не находились. Так же интерес представлял следующий факт - насколько удобнее стали сгруппированы клоны, укрупнились ли фрагменты повторного использования. Для этих целей и было сформирован данный тестовый набор.

Для его формирования было отобрано 7 фрагментов.

1) Фрагмент из документации по richfaces [26] (библиотеки компонентов для JavaServerFaces, созданной компанией jboss, ныне подразделением Red Hat). Этот пример содержит много XML- конструкций при малом количестве текстового содержания:

```
            </member>
        </simplelist>
    </listitem>
</varlistentry>
```

<varlistentry>

<term>Affected <classname>rich</classname> components</term>

2) Фрагмент взят из документации по Spring Roo [27] (фреймворк для быстрого создания бизнес-приложений на Java, разработчик компания SpringSource) . Содержит “плоский” текст с небольшим количеством XML- конструкций:

hanks to Roo, Java developers can now enjoy this higher productivity <emphasis>plus</emphasis>

3) Фрагмент документации для Zend Framework [28]: популярного фреймворка для разработки web-приложений на php. Представляет собой большой отрывок текста
(400 строк), содержащий как целые абзацы, состоящие только из “плоского” текста, так и фрагменты с большой плотностью XML. Так же содержит много конструкций, представляющих интерес для проверки XML-фильтра. Например:

```
<![CDATA[

    namespace Zend\Controller\Request;

    abstract class RequestAbstract

    {

    // ...

    }
```

4) Пример из документации по Jetty[29] (контейнер сервлетов, полностью написанный на Java): небольшой пример, но содержащий много повторяющихся фрагментов. Так же содержит ряд XML конструкций, которые отсутствуют в других тестах. Например:

<informalexample>

<programlisting language="fetch">

<filename>@GITURL@/jetty-distribution/src/main/resources/webapps/javadoc.xml</filename>

</programlisting>

</informalexample>

5) Библиография из документации по phing [30] (сборщик php проектов наподобии ant). Имеет структуру, не специфичную для документации, содержит большое количество ссылок:

```
<biblioentry id="bibliography.osi-model">
<abbrev>osi-model</abbrev>
<title>OSI (Open System Interconnect) Model</title>
<citetitle><ulink url="http://www.iso.org/">http://www.iso.org/</ulink></citetitle>
```

6) Отрывок из документации по Hibernate [31] - фреймворку для работы с базами данных. Содержит много фрагментов, пригодных для вынесения в общие активы, при этом эти фрагменты имеют большую семантическую значимость.
7) Фрагмент документации по DocBook [32]. Был выбран большой фрагмент текста (5707 строк текста) , с целью нахождения только крупных клонов(не менее 80 токенов).

4.1.2. Набор для формирования статистики

Данный набор тестов был сформирован для того, чтобы получить количественные оценки эффективности улучшений: стал ли алгоритм находить больше повторно используемой информации и насколько, увеличилось ли количество групп и размер клонов. Так как это не требует непосредственного сравнения клонов, найденных различными версиями алгоритма, появляется возможность взять большие фрагменты текста, соответствующие реальным файлам с документацией. Было отобрано 15 тестов для формирования сводной таблицы. Размер каждого текста не менее 800 строк . Подробная информация о данном сформированном наборе находится в приложении.

4.2 Схема эксперимента

Было сформировано 3 варианта алгоритма поиска клонов:

- версия 1 – старая версия алгоритма. Для того, чтобы результаты можно было сравнивать, к старой версии была добавлена фаза удаления пересечений;
- версия 2 – это алгоритм с новым механизмом XML-фильтрации, удалением дубликатов и пересечений, поиском гиперссылок, но без поиска по остаткам;
- версия 3 – это алгоритм с новым механизмом XML-фильтрации, удалением дубликатов и пересечений, поиском гиперссылок, а так же поиском по остаткам.

Тестирование было решено провести в 2 этапа.

На первом этапе все три версии сравнивались на наборе тестов для непосредственной проверки улучшений. Так же на этом этапе проводилось тестирование работы алгоритмов с плоским текстом, полученным путем отбрасывания Xml-конструкций из текста тестов данной группы.
Для проверки работы с плоским текстом использовалась 3 версия алгоритма.

Второй набор данных тестировался на каждой из трех версий. Целью было выявление количество найденной повторно используемой информации, выражаемое количеством символов, из которых состоит результат.

4.3 Анализ результатов

Выделим следующие **прагматические** улучшения нового алгоритма.

1. Найдена и исправлена ошибка с самопересечениями клонов. Ошибка связана с особенностями реализации CloneMiner, влечет некорректность операций рефакторинга.
2. Большие куски стали объемнее и лучше сгруппированы – нас ведь интересуют с одной стороны, семантически значимые повторы (описания какой-то функциональности, модуля и т.д.).
3. Увеличилось качество поиска небольших семантически осмысленных кусков для добавления в словарь – номер версии, перечень поддерживаемых платформ и т.д. .
4. Появился поиск гиперссылок (Интернет-адресов).

Далее приводятся более подробные результаты.

4.3.1 Исправление ошибки с самопересечениями

В результате удаления пересекающихся клонов часть информации теряется, некоторые группы исчезают (у группы с n элементами n-1 имеют пересечения с другими клонами и принято решение удалить элементы именно из данной группы). При этом могут потеряться клоны, более пригодные для повторного использования, чем оставшиеся. Например, у группы состоящей из двух больших представителей, один из них удаляется при пересечении с другой группой, содержащей больше элементов, но с меньшей семантической значимостью.

Для предотвращения этого, приоритетными было принято считать клоны большей длины (функция полезности клона считается как n^2*m, где n размер клона, а m размер группы, в которой он состоит). Хотя следует признать несовершенность алгоритма удаляющего пересечения, стоит отметить, что с основной своей задачей он успешно справляется (удаление пересечений для корректной работы при малом времени выполнения).

4.3.2 Обработка больших семантически значимых фрагментов

С новым алгоритмом XML-фильтрации найденные элементы стали крупнее.

В старой версии:

```
<title>Comprehensive architecture test</title>
Найден 2 раза
    Начало 33.13 Конец 33.59
    Начало 69.13 Конец 69.59

            <para>
                The "comprehensive" architecture abstracts the application
away from the underlying JDBC/JTA APIs and
                allows Hibernate to manage the details.
            </para>

Найден 2 раза
    Начало 34.13 Конец 37.20
    Начало 70.13 Конец 73.20
```

В новой же версии (за счет объединения клонов с соседними границами) этот же фрагмент выглядит следующим образом:

```
<title>Comprehensive architecture test</title>
<para>
                The "comprehensive" architecture abstracts the application
away from the underlying JDBC/JTA APIs and
                allows Hibernate to manage the details.
</para>
```

```
Найден 2 раза
    Начало 33.13 Конец 37.20
    Начало 69.13 Конец 73.20
```

Так же в новой реализации находятся фрагменты текста, которые ранее отбрасывались XML-фильтром. Что совместно с объединением соседних клонов приводит к значительному результату и позволяет выделить большие семантически значимые фрагменты. Например, если раньше клоны вида

```
for these <classname>ProjectHandler</classname> invokes the
<classname>TaskHandler</classname> class. If a tag is presented that
doesn't match any expected tags, then
<classname>ProjectHandler</classname> assumes it is a datatype and
invokes the <classname>DataTypeHandler</classname>.</para>
```

разбивались на отдельные XML конструкции

```
<classname>TaskHandler</classname>
```

или

```
classname>DataTypeHandler</classname>
```

, то теперь они находятся целиком (в указанном виде).

4.3.3 Мелкозернистое повторное использование

Были найдены клоны:

```
java \
    -Dencoding.windows-1252=com.nwalsh.saxon.Windows1252 \
```

находящиеся до фильтрации в составе клонов вида

```
<screen>  java \
    -Dencoding.windows-1252=com.nwalsh.saxon.Windows1252 \
```

Старый вариант алгоритма такие элементы не находил.

Повторный запуск CloneMiner для поиска по остаткам так же позволил найти элементы, пригодные для мелкозернистого повторного использования. Эти фрагменты были утеряны в результате удаления пересечений и XML фильтрации, а так же не найденные в силу особенностей работы CloneMiner. Например, были найдены клоны вида:

```
<member>
     <sgmltag>&lt;rich:treeNode&gt;</sgmltag>
</member> .
```

4.3.4 Поиск гиперссылок

Алгоритм стал корректно работать с гиперссылками.

У некоторых из них были расширены границы: клон *//www.gnu.org/licenses/fdl* перешел в *http://www.gnu.org/licenses/fdl.html*. Так же были найдены новые клоны вида: *http://www.w3c.org/People/Raggett/tidy/*, которые не раньше не находились из-за особенностей CloneMiner.

4.3.5 Численные оценки

Набор для проверки непосредственных изменений

Тесты 1-6 проводились для минимального числа токенов – 3,тест 7 для минимального числа 80. В некоторых тестах отсутствует сравнение с плоским текстом.

	Версия 1	Версия 2	Версия 3	Плоский текст
Тест 1				
Количество групп клонов	21	4	6	6
Среднее число токенов в группе	5	15	16	10
Тест 2				
Количество групп клонов	11	11	11	11
Среднее число токенов в группе	3	3	3	3
Тест 3				
Количество групп клонов	25	28	30	27
Среднее число токенов в группе	5	6	6	7
Тест 4				
Количество групп клонов	16	20	21	21
Среднее число токенов в группе	3	3	3	3
Тест 5				
Количество групп клонов	10	10	10	-
Среднее число токенов в группе	3	4	4	-
Тест 6				
Количество групп клонов	22	33	33	32
Среднее число токенов в группе	3	3	3	3
Тест 7				
Количество групп клонов	16	12	12	-
Среднее число токенов в группе	82	111	111	-

Рис. 7. Результаты работы различных версий алгоритма на наборе для проверки улучшений

Из результатов показанных на рис.7 видно, что наблюдается рост среднего числа токенов найденных клонов. Количество же найденных групп возрастает не всегда. Так в тесте 1 наблюдается уменьшение числа групп, при значительном увеличении числа токенов. В данном тесте в первой версии было найдено много групп, состоящих из XML конструкций вида:

```
<member>
     <sgmltag>&lt;rich:autocomplete&gt;</sgmltag>
</member> .
```

Данные группы были объединены в группы с большим размером клонов.

Набор для формирования статистики

Для того, чтобы численно представить насколько эффективны улучшение, было принято решение провести статистический тест.

Для теста использовались 3 версии алгоритма:

- версия 1 – старая версия алгоритма. К ней был добавлен механизм удаления пересечений.
- версия 2 – это алгоритм с новым механизмом XML-фильтрации, удалением дубликатов и пересечений, поиском гиперссылок, но без поиска по остаткам;
- версия 3- это алгоритм с новым механизмом XML-фильтрации, удалением дубликатов и пересечений, поиском гиперссылок и поиском по остаткам. Данная версия включает в себя все нововведения.

На рис.8 приведены результаты работы трех версий алгоритма на наборе для формирования статистики. Все алгоритмы запускались для минимального числа токенов в клонах равного 3. Алгоритмы сравнивались по трем параметрам:

- количество групп клонов;
- найденный повторно используемый текст в символах (количество символов во всех клонах всех групп, сокращенно п.и.т);
- среднее количество токенов по всем группам (т.е. средняя длина клона в токенах без учета разбиения на группы)

	Количество групп			П.И.Т			Среднее кол-во токенов		
номер	версия 1	версия 2	версия 3	версия 1	версия 2	версия 3	версия 1	версия 2	версия 3
1	124	130	138	6600	7343	7687	3,419	3,787	3,785
2	90	91	97	5779	6032	6186	3,863	4,209	4,157
3	82	87	102	8042	12222	13124	5,549	7,621	7,108
4	32	37	44	4623	6420	6679	5,899	7,211	6,806
5	81	81	87	4123	4454	4776	3,656	3,945	3,875
6	105	112	129	5686	6112	6892	3,886	4,184	4,096
7	103	108	117	8809	11155	11868	4,099	4,933	4,9
8	127	133	147	9079	11421	12464	4,18	4,81	4,795
9	158	150	167	8892	10606	11713	3,663	4,632	4,658
10	36	51	57	5912	10200	10765	5,11	5,319	5,272
11	258	263	297	15987	18463	20076	3,654	4,215	4,122
12	153	154	174	10139	11594	13386	3,955	4,749	4,738
13	179	187	198	10552	11727	12398	3,515	3,901	3,917
14	262	272	340	18608	19829	25169	4,194	4,668	4,699
15	112	120	141	9657	13240	14536	4,58	6,07	5,81

Рис.8.Результаты работы различных версий алгоритма на наборе для формирования статистики

На рис.9 представлен сравнительный анализ работы алгоритмов 2 и 3 по сравнению со старой версией (версия 1). Сравнение проводится по тем же самым трем параметрам: количество групп клонов, п.и.т.,среднее количество клонов. Из рис.9 видно, что рост количества групп осуществляется за счет поиска по остаткам. Версия без поиска по остаткам новые групп как правило не обнаруживает. Среднее количество токенов в группах стало больше в обоих версиях по сравнению со старой. При этом среднее количество токенов в

результате работы алгоритма 2 оказывается большим, чем в результате работы алгоритма 3. Это свидетельствует о том, что поиск по остаткам находит небольшие фрагменты текста.

Количество п.и.т. увеличилось по сравнению с версией 1 в обоих случаях. Большой разброс в результатах обусловлен особенностями структуры соответствующих тестов: они отличаются по количеству XM-конструкций, структуре самого расположения XML. В целом же наблюдается стабильное увеличение найденной информации как от перехода к версии 2, так и от версии 2 к версии 3. Механизм поиска по остаткам следует признать эффективным улучшением алгоритма. Прирост же в версии 2 объясняется уменьшением потери данных при XML-фильтрации. При ее отключении и ручном контроле за корректностью XML результат будет максимально приближен к реальному количеству клонов в тексте. Сейчас же он составляет порядка 85-90 %.

	Номер теста	1	2	3	4	5	6	7	8
	Объем	52кб	41кб	46кб	27кб	40кб	35кб	71кб	72кб
Версия2	кол-во групп	4,84%	1,11%	6,10%	15,63%	0%	6,67%	4,85%	4,72%
	п.и.т	11%	4%	51%	39%	8%	7%	27%	25%
	ср-нее кол-во токенов	10,76%	8,96%	37,34%	22,24%	7,90%	7,67%	20,35%	15,07%
Версия 3	кол-во групп	11,29%	7,78%	24,39%	37,50%	7,41%	22,86 %	13,59%	15,75%
	п.и.т	16%	7%	63%	44%	16%	21%	35%	37%
	ср-нее кол-во токенов	10,70%	7,61%	28,10%	15,38%	5,99%	5,40%	19,54%	14,71%

	Номер теста	9	10	11	12	13	14	15	среднее
	Объем	72кб	68кб	47кб	116кб	66кб	83кб	89кб	
Версия 2	кол-во групп	4,72%	-5,06%	41,67%	1,94%	0,65%	4,47%	3,82%	6,57%
	п.и.т	25%	19%	72%	15%	14%	11%	6%	23,07%
	ср-нее кол-во токенов	15,07%	26,45%	4,09%	15,35%	20,08%	10,,98%	11,30%	16,01%
Версия 3	кол-во групп	15,75%	5,70%	58,33%	15,12%	13,73%	10,61%	29,77	19,98%
	п.и.т	37%	31%	82%	26%	32%	17%	35%	34,13%
	ср-нее кол-во токенов	14,71%	27,16%	3,17%	12,81%	19,80%	11,44%	12,04%	14,71%

Рис. 9. Сравнительный анализ

5. Особенности реализации

В качестве основных требований к коду в порядке убывания значимости были взяты: корректность, эффективность (результат при скорости работы), возможность управлять параметрами алгоритма из внешней среды, а так же возможность дальнейшего улучшения.

Для достижения корректности на каждом этапе написание проводилось регрессионное тестирование и проверка результатов на правильность (клоны не пересекаются, в качестве клонов не находятся неправильные представители, границы клонов определены правильно и в соответствии с ожиданиями и др.).

В целях достижения эффективности было решено не использовать сложные алгоритмы, требующие большого времени выполнения, дополнительных входных данных и большого объема памяти, так как старый алгоритм справлялся со своей задачей (нахождение клонов) достаточно эффективно. Не ожидалось, что новый алгоритм найдет значительное количество новых клонов. При этом результаты по нахождению на 10-15 % уже можно считать значимыми. Поэтому для алгоритмов не использовались какие-нибудь специфичные структуры для хранения информации, сложные алгоритмы на деревьях и графах. В основном, вся реализация построена на использовании списком, массивов, сортировках и простых преобразованиях матриц.

Для того, чтобы пользователь мог отключать какие-то элементы при поиске клонов (например убрать XML-фильтрацию или отключить поиск по остаткам), все этапы были оформлены в отдельные независимые модули. Так же был добавлены элементы управления для Eclipse.

Дальнейшее улучшение алгоритмов для получения новых элементов видится нецелесообразным. Однако внесение каких-то улучшений не представляется трудным. Изменение фильтра XML достаточно просто так, так создан набор прикладных функций, позволяющих разбирать текст клонов как XML. Для создания более “мягкого” фильтра (для поиска нечетких клонов) или же, наоборот, более “жесткого” (чтобы находить только конструкции, отвечающие специфичным требованиям) достаточно просто применить данные функции в нужном порядке. Изменение алгоритма отбрасывания пересечений так же не вызовет трудностей (возможно легко изменить, какой из клонов будет отбрасываться, все информация о клонах и группах легко доступна).

Дальнейшее развитие поиска клонов видится не в улучшении каких-то частей алгоритма, а в отделении написанного функционала от Eclipse и особенностей языка DRL.

Заключение

В рамках данной работы были достигнуты следующие результаты.

- Выполнено знакомство с предметной областью: форматом DocBook, технологией DocLine, средствами поиска клонов в DocLine, технологией CloneMiner.
- Улучшен алгоритма поиска XML-клонов в DocLine: изменена XML-фильтрация клонов, удалены пересечения CloneMiner, реализован поиск по остаткам, выполнено удаление повторяющихся групп.
- Выполнена реализация улучшенного алгоритма (Java/Eclipce/Docline)
- Улучшен пользовательский интерфейс модуля поиска клонов: представление результатов поиска, задание параметров поиска
- Выполнена апробация созданных средств: сформированы наборы тестовых данных, проведено сравнение старой и новой версий алгоритма.

Список литературы

[1] Ахин М.Х., Ицыксон В.М. Обнаружение клонов исходного кода: теория и практика. Системное программирование. 2010. Т. 5, №. 1. С. 145-163.

[2] Кознов Д.В., Романовский К.Ю. Автоматизированный рефакторинг документации семейств программных продуктов // Системное программирование / Вып. 4, под ред. Терехова А.Н. и Булычева Д.Ю. СПб.: Изд. СПбГУ, 2009. С. 128-150.

[3] Кознов Д.В., Шутак А.В., Смирнов М.Н., Смажевский М.А. Поиск клонов при рефакторинге технической документации. Компьютерные инструменты в образовании. 2012. № 4. С. 30-40.

[4] Романовский К.Ю. Метод разработки документации семейств программных продуктов. Системное программирование. 2006. Т. 2. № 1. С. 191-218.

[5] Романовский К.Ю. Разработка повторно-используемой документации семейства телефонных станций средствами технологии DocLine. Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. 2009. № 2. С. 166-180.

[6] Романовский К.Ю., Кознов Д.В. Язык DRL для проектирования и разработки документации семейств программных продуктов. Вестник С.-Петербург. ун-та. Сер. 10.2007. Вып. 4. С. 110-122.

[7] Смирнов М.Н., Кознов Д.В., Дорохов В.А., Романовский К.Ю. Программная среда WebMLDoc для автоматизированного отслеживания изменений в пользовательской документации Web-приложений. Системное программирование. Том 5, Вып. 1.СПб.: Изд-во СПбГУ, 2010. С. 32-51.

[8] Смирнов М.Н., Соколов Н.Е., Романовский К.Ю. DocLineFM: среда разработки повторно используемой документации семейств программных продуктов на базе пакета Adobe FrameMaker. Системное программирование. Том 6, Вып. 1. СПб.: Изд-во СПбГУ, 2011. С. 77-94.

[9] Фаулер М. Рефакторинг: улучшение существующего кода. - Пер. с англ. СПб: Символ-Плюс, 2003. 432 с.

[10] Akhin M., Itsykson V. Clone detection: why, what and how?

[11] Basit H. A., Jarzabek S. A Data Mining Approach for Detecting Higher-Level Clones in Software. IEEE Transactions on Software engineering, Vol. 35, No. 4, 2009.

[12] Calheiros F., Nepomuceno F. Product Line Variability Refactoring Tool. http://twiki.cin.ufpe.br/twiki/pub/SPG/GenteAreaPublications/WRT07_calheiros.pdf

[13] Clements, P., Northrop, L. Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002. 608 p.

[14] CloneDifferentiator tool. http://www.comp.nus.edu.sg/~yinxing/clonedifferentiator/

[15] CloneDR tool на сайте Semantic Design. http://students.cis.uab.edu/tairasr/clones/literature/

[16] Code clone Literature on site of The University of Alabama at Birmingham // http://students.cis.uab.edu/tairasr/clones/literature/

[17] Critchlow M., Dodd K., Chou J., and A. van der Hoek, Refactoring Product Line Architectures // First International Workshop on Refactoring: Achievements, Challenges, and Effects, November 2003. P.23-26.

[18] DITA, http://dita.xml.org 2010 6th Central and Eastern European Software Engineering Conference, CEE-SECR 2010 Moscow, 2010. С. 36-42.

[19] Duplo tool на сайте Sourceforge // http://sourceforge.net/projects/duplo/

[20] Koznov D.V., Romanovsky K.Yu. DocLine: A method for software product lines documentation development. Programming and Computer Software. 2008. Vol. 34. № 4. P. 216-224.

[21] OpenXML, http://www.philo.de/xml/index.shtml

[22] Release Notes document for the Vaadin Trac version 5.2// http://dev.vaadin.com/svn/versions/5.2/build/docbook/xsl/

[23] Romanovsky K., Koznov D., Minchin L. Refactoring the Documentation of Software Product Lines. Lecture Notes in Computer Science. 2011. Vol. 4980 LNCS. P. 158-170.

[24] SDD tool на сайте Eclipse. http://wiki.eclipse.org/index.php/Duplicated_code_detection_tool_%28SDD%29

[25] Walsh N., Muellner L. DocBook: The Definitive Guide. O'Reilly, 2003.

[26] Документация по richfaces https://github.com/richfaces/docs/blob/develop/Migration_Guide/src/main/docbook/en-US/Changes_and_new_features.xml

[27] Документация по Spring Roo https://github.com/SpringSource/spring-roo/blob/master/deployment-support/src/site/docbook/reference/welcome-intro.xml

[28] Документация по Zend Framework http://framework.zend.com/svn/framework/standard/trunk/documentation/manual/en/ref/coding_standard.xml

[29] Документация по Jetty https://github.com/jetty-project/jetty-documentation/blob/master/src/docbkx/quick-start/configuring/what-to-configure.xml

[30] Документация по phing https://github.com/pmartin/phing-documentation-docbook/blob/master/xml/en/chapters/bibliography.xml

[31] Документация по Hibernate https://github.com/emmanuelbernard/hibernate-core- ogm/blob/master/documentation/src/main/docbook/devguide/en-US/Services.xml

[32] Документация по DocBook http://docbook.sourceforge.net/release/xsl/current/RELEASE-NOTES.html

Приложение

Тесты 1-2. Эти тесты взяты из документации по Spring Roo-фреймворку, предназначенномудля быстрого создания бизнес-приложений на Java (разработчик – компания SpringSource).

1) Размер 52 кб (1024 строк). Документ является введением в архитектуру продукта, сам текст представляет собой абзацы по 8-10 строк плоского текста, обрамленного xml-тегами (DocBook).

https://github.com/SpringSource/spring-roo/blob/master/deployment-support/src/site/docbook/reference/welcome-architecture.xml

2) Размер 41 кб (915 строк). Документ является описанием дополнений, содержит как большие фрагменты плоского текста, так и списки, состоящие из XML-конструкций с одно-двух строчным содержанием.

https://github.com/SpringSource/spring-roo/blob/master/deployment-support/src/site/docbook/reference/internals-simple-add-ons.xml

Тест 3. Источник – документация по Zend Framework, популярному фреймворку для разработки Web-приложений на php. Источники текста можно скачать по адресам:
http://framework.zend.com/svn/framework/standard/trunk/documentation/manual/en/ref/coding_standard.xml ,

Размер 45 кб (1188 строк). Документ описывает стандарты написания кода для ZendFramework, состоит из множества XML тегов вида *<para>,<section>,* а так же содержит специфичные конструкции вида

```
  <programlisting language="php"><![CDATA[
if ($a != 2) {
    $a = 2;
}
]]></programlisting>
```

Тест 4. Источник – документация по Jetty- контейнеру Java-сервлетов.
Размер 27 кб (884 строки) .Фрагмент с описанием синтаксиса jetty. Состоит в основном из XML тегов.

https://github.com/jetty-project/jetty-documentation/blob/master/src/docbkx/reference/jetty-xml/jetty-xml-syntax.xml

Тесты 5-6 Взяты из документации по пакету Phing, который является сборщиком php-проектов наподобие ant.

5)Размер 40 кб (1034 строк). Содержит как параграфы с плоским текстом, так и таблицы, состоящие из XML.

https://github.com/pmartin/phing-documentation-docbook/blob/master/xml/en/chapters/extending-phing.xml ,

6) Размер 35 кб (831 строк). Первая часть состоит из xml-таблицы, вторая (примерно половина всего текста) представляет собой один кусок плоского текста.

https://github.com/pmartin/phing-documentation-docbook/blob/master/xml/en/appendices/fact-sheet.xml .

Тесты 7-10. Источник – документация по фреймворку Hibernate, предназначенному для работы с базами данных.

7)Размер 72кб (1803 строк). Документ описывает настройку сессии Hibernate, представляет собой XML-таблицы с большим количеством тегов.

https://github.com/emmanuelbernard/hibernate-core-ogm/blob/master/documentation/src/main/docbook/manual/en-US/content/configuration.xml

8)Размер 73кб (1507 строк). Документ содержит таблицы и списки, но содержащие большие фрагменты плоского текста в тегах.

https://github.com/emmanuelbernard/hibernate-core-ogm/blob/master/documentation/src/main/docbook/devguide/en-US/Envers.xml

9) Размер 70кб (1716 строк). Документ содержит списки и конструкции вида

```
<programlisting role="XML">&lt;class name="Document"&gt;
       &lt;id name="id"&gt;
        &lt;generator class="native"/&gt;
    &lt;/id&gt;
    &lt;property name="name" not-null="true" length="50"/&gt;
    &lt;property name="summary" not-null="true" length="200"
lazy="true"/&gt;
```

```
    &lt;property name="text" not-null="true" length="2000" lazy="true"/&gt;
&lt;/class&gt;</programlisting>
```

https://github.com/emmanuelbernard/hibernate-core-ogm/blob/master/documentation/src/main/docbook/manual/en-US/content/performance.xml

10) Размер 48кб (953 строки). Документ состоит из сложных XML-конструкций вида

```
<varlistentry>
                    <term>Initiator</term>
                    <listitem>
                        <para>

<classname>org.hibernate.service.config.internal.ConfigurationServiceInitiator</classname>
                        </para>
                    </listitem>
                </varlistentry>
```

https://github.com/emmanuelbernard/hibernate-core-ogm/blob/master/documentation/src/main/docbook/devguide/en-US/Services.xml

Тесты 11-12. Взяты из документации по ядру Linux.

11)Размер 118 кб (2541 строк). Документ состоит из абзацев плоского текста. Временами встречаются списки, сформированный с помощью небольших XML-конструкций.
https://github.com/torvalds/linux/tree/master/Documentation/DocBook/drm.tmpl

12) Размер 67кб (2147 строк). Документ содержит как конструкции вида

```
<row>
<entry>Timer B</entry>
<entry>SLI</entry>
<entry>SLI</entry>
<entry>SL</entry>
<entry>SL</entry>
```

так и абзацы плоского текста из 3-4 строк.
https://github.com/torvalds/linux/tree/master/Documentation/DocBook/kernel-locking.tmpl

Тест 13. Источник – документация по Spring AMQP-проекту, помогающему облегчить работу с передачей сообщений по протоколу AMQP.

13) Размер 84кб (1741 строк). Документ состоит в основном из абзацев плоского текста.

https://github.com/SpringSource/spring-amqp/tree/master/src/reference/docbook/amqp.xml

Тесты 14-15. Источник – документация по Drools-платформе, предназначенной для реализации бизнес-логики при помощи процессора правил.

14)Размер 90 кб (1724 строк). Документ состоит из секций и абзацев, а те –из плоского текста размером 3-10 строк, обрамлённые DocBook-тегами.

https://github.com/droolsjbpm/jbpm/blob/master/jbpm-docs/src/main/docbook/en-US/Chapter-Processes/Chapter-Processes.xml

15)Размер 47кб (997 строк). Документ состоит из таблицы со строками вида

```
<row>
            <entry><code>processid</code></entry>
            <entry>The id of the process that the process instance is
executing</entry>
          </row>
```

https://github.com/droolsjbpm/jbpm/blob/master/jbpm-docs/src/main/docbook/en-US/Chapter-Persistence/Chapter-Persistence.xml