Кознов Дмитрий Владимирович,
Шутак Артем Витальевич,
Смирнов Михаил Николаевич,
Смажевский Михаил Александрович

УДК 681.3

ПОИСК КЛОНОВ ПРИ РЕФАКТОРИНГЕ ТЕХНИЧЕСКОЙ ДОКУМЕНТАЦИИ[1]

Аннотация

Один из ключевых аспектов управления повторным использованием документации является поиск повторяющихся фрагментов текста в уже существующих документах. Дело в том, что документация редко начинает разрабатываться как повторно используемая. Кроме того, ее в принципе удобно создавать как набор обычных текст, переходя к выделению одинаковых фрагментов с определенного момента процесса разработки. В данная статье предлагается использовать для поиска повторов в XML-документации (представление технической документации в XML-формате сегодня является основным направлением в области средств поддержки технической документации) известную технику поиска клонов в ПО (software clone detection). В работе использован алгоритм и программное средство Clone Miner, поиск клонов организован как поиск повторов в «плоском» тексте с XML-фильтрацией. Предложенное решение реализовано в среде Eclipse с технологией DocLine. Представлены также результаты апробации решения.

Ключевые слова: поиск клонов, семейства программных продуктов, повторное использование документация, управление вариативностью.

ВВЕДЕНИЕ

Техническая документация является важной составляющей современного промышленного программного обеспечения (ПО) и бывает очень разнообразной: справочные системы, руководства пользователя, руководства по быстрому старту и т. д. При этом документация может представляться в различных форматах – в виде встроенных справочных систем (WinHelp, HTML), печатная (PDF, MS Word) и др. Так же как и само ПО, его документация изменчива и структурно сложна.

Известно, что большие документы, а также пакеты документов, содержат многочисленные повторы: одна и та же информация представляется в разных частях документа на разном уровне абстракции, с разным количеством деталей и т. д. Существуют также семейства программных продуктов (СПП) [12], в рамках которых создаётся ряд сходных программных приложений, и они, соответственно, имеют похожую документацию. Таким образом, при создании документации часто используются операции copy/paste. Однако многократные повторы затрудняют дальнейшее сопровождение документации – при внесении изменений необходимо находить все такие повторы и изменять их согласовано друг с другом. На практике очень часто это трудно выполнить из-за больших объёмов документации и нехватки времени, и поэтому в документации накапливаются ошибки и противоречия.

Для облегчения сопровождения документации целесообразно организовывать повторное использование. Решению этой задачи посвящена технология DocLine, разрабатываемая на кафедре системного программирования СПбГУ [2, 3, 5, 6, 7, 19, 22].

[1] Работа выполнена при поддержке гранта РФФИ 11-01-00622.

Эта технология включает в себя средства рефакторинга документации [2, 22], которые применяются для улучшения структуры повторного использования документации в процессе её сопровождения. Однако оставался открытым вопрос автоматизированного поиска повторяющихся фрагментов текста.

В данной работе для решения этой задачи предлагается использовать методы поиска клонов в программном обеспечении (software clone detection). В качестве средства базового поиска клонов в XML-документации мы использовали пакет Clone Miner [10]. Выход Clone Miner анализируется нашим алгоритмом, который, в частности, выполняет фильтрацию полученных клонов и выдаёт корректные, с точки зрения XML, результаты. При этом, несмотря на то, что мы ориентировались на специализированные XML-языки разработки документации DRL и DocBook, поддерживаемые в технологии DocLine, предлагаемый алгоритм может использоваться для документации в произвольном XML-формате. В статье также представлен инструмент, реализующий данный алгоритм в среде Java/Eclipse. Инструмент интегрирован со средой DocLine, имеет средства визуализации и навигации по найденным группам клонов, а также позволяет выполнить операции рефакторинга, то есть автоматически (по команде пользователя) выделить ту или иную найденную группу клонов в повторно используемый фрагмент, вынося его описание в отдельное место документа и заменяя каждое его вхождение ссылкой на это описание. Наконец, в статье представлены результаты апробации предложенного программного средства, доказывающие эффективность предложенного подхода и выявляющие направление дальнейших исследований.

1. ТЕХНОЛОГИЯ DOCLINE

1.1. ОБЗОР DOCLINE

Данная технология предназначена для разработки документации семейств программных продуктов и разрабатывается на кафедре системного программирования математико-механического факультета СПбГУ. DocLine состоит из следующих частей:

– процесс разработки документации [3];

– язык разработки документации DRL (Document Reuse Language), включающий текстовую и графическую нотации [5];

– пакет инструментальных средств, включающий в том числе и автоматизированные средства рефакторинга [2, 22].

Для полиграфической разметки документов DocLine использует известную технологию DocBook [24]. Последняя широко распространена и является стандартом дефакто для разработки документации на свободно распространяемое программное обеспечение, широко применяясь в мире Linux. DocLine успешно применена для организации повторного использования документации ПО семейства телефонных станций [4].

Основной единицей языка DRL, используемого в DocLine, является *информационный элемент* – выделенный фрагмент документации, который может включаться в различные контексты нескольких документов (путём ссылки на него). Также важной конструкцией является *словарь* – структура, содержащая набор пар «ключ – значение». С помощью этой конструкции в тексте документации можно использовать короткий ключ для обозначения некоторого (небольшого) участка текста.

1.2. АВТОМАТИЗИРОВАННЫЙ РЕФАКТОРИНГ В DOCLINE

Рефакторинг – это процесс изменения программного кода с сохранением его исполняемой семантики, направленный на улучшение архитектуры системы, либо на повышение читаемости [8]. Рефакторинг широко используется в гибких методах разработки ПО как средство поддержания кода и архитектуры приложения в хорошем состоянии. Также рефакторинг применяется при разработке СПП [11, 16], в частности для извлечения общих активов и настройки их использования. В работах [2, 22] было предложено использовать рефакторинг для сопровождения XML-документации. В

этом случае рефакторинг означает изменение структуры документации, имеющей специальное внутреннее представление (в настоящее время это, как правило, различные XML-форматы, такие как OpenXML [20], DocBook [24], DITA [17], DRL [5] и др.), с сохранением внешнего вида документов. В рамках DocLine были спроектированы и реализованы операции рефакторинга, обеспечивающие выделение и настройку повторно используемых фрагментов документации. Созданные операции можно разделить на группы, представленные ниже.

1. *Переход к DRL* – в этой группе всего одна операция и она осуществляет перевод документации, разработанной в DocBook, в DocLine.

2. *Операции выделения крупных фрагментов текста* затрагивают общую структуру документации и позволяют создавать крупные компоненты повторного использования. Обеспечивают работу с информационными элементами.

3. *Операции выделения небольших фрагментов текста* обеспечивают работу со словарём и каталогом.

4. *Операции настройки повторного использования* используются, если нужно настроить возможности адаптации общих активов под контекст использования. Обеспечивают работу с точками расширения.

Однако эти операции не обеспечивают автоматизированный поиск повторяющихся фрагментов текста.

2.ОБЗОР ЗАДАЧИ ПОИСКА КЛОНОВ И ОБОСНОВАНИЕ ВЫБРАННОГО ПОДХОДА

2.1. О ЗАДАЧЕ ПОИСКА КЛОНОВ

Задача поиска клонов в программном обеспечении (software clone detection) посвящена поиску дублированного кода, который в дальнейшем может быть устранён при помощи рефакторинга [1]. Обзор существующих методов поиска клонов можно найти в работах [1, 9], а обзор инструментов – в [15].

Среди основных подходов к поиску клонов можно выделить текстовый, синтаксический и графовый. Первый основывается на поиске частых подпоследовательностей символов, второй – на поиске повторяющихся конструкций созданных с помощью некоторого языка (программирования). Третий подход применим к структурам, представимым в виде графа, и в таком случае задача поиска клонов сводится к поиску изоморфных подграфов. Существуют алгоритмы, ищущие точные клоны, а также алгоритмы, способные найти нечёткие совпадения.

2.2. ОБЗОР ИНСТРУМЕНТОВ

Мы просмотрели около тридцати различных инструментов по поиску клонов, следуя списку, представленному в [15]. В итоге мы выбрали пять наиболее типичных инструментов и рассмотрели их более детально.

- CloneDR [14] находит не только точные, но и близкие по значению клоны (near-missduplicates), также может использоваться для разных языков программирования (Ada, COBOL, C, C++, Java и др.).
- Duplo [18] предназначен для поиска дублированных блоков кода в больших приложениях, написанных на языках C, C + +, Java, C # и VB.Net .
- CloneDifferentiator [13] проводит семантическое сравнение исходных файлов Java-программ и является расширением среды Eclipse.
- SDD [23] позволяет искать неточные клоны, не имеет привязки к конкретному языку программирования. Инструмент основан на алгоритме поиска ближайших соседей (N-neighbor search). Часто выдаёт непредсказуемые результаты поиска – клоны в одной группе могут сильно различаться. Является расширением платформы Eclipse.
- Clone Miner [10] – является инструментом для поиска клонов высокого уровня (Higher-Level Clones in Software). Имеет низкоуровневый программный интерфейс (запуск через командную строку, файл с входными данными и файл с результата-

ми), что легко позволяет интегрировать его в целевой проект. Обладает набором настроек, которые позволяют существенно изменить качество выдаваемых результатов поиска (минимальное количество термов в клоне, точность поиска, учёт специальных конструкций и др.). Может применяться к «плоскому» тексту, имеет поддержку русского языка.

2.4. ОБСУЖДЕНИЕ И ВЫВОДЫ

Итак, мы решили использовать в нашем исследовании алгоритм поиска клонов в «плоском» тексте, так как приняли гипотезу о том, что такой подход «в лоб» может дать хорошие результаты, а эксперименты, выполненные с использование такого инструмента, укажут нам дальнейший путь развития нашего исследования. Тем более, что фактически, нам требуется смешанный подход – даже при применении к текстам, имеющим XML-разметку, кроме распознавания XML-конструкций, нам необходима работа с «плоским» текстом, который находится либо внутри тегов, либо между тегами. При этом мы не просто берём или не берём весь такой текст целиком, но нас могут интересовать отдельные его фрагменты. Для обеспечения корректности выдаваемых результатов с точки зрения XML (в нашем случае – DRL и DocBook) было решено проводить фильтрацию результатов поиска в XML-файлах. Кроме того, поиск «плоских» клонов в контексте DocLine необходим сам по себе, так как существует задача оценки применимости DocLine к имеющейся документации – целесообразно проверить, имеются ли в документации повторы, и если их много, то тогда применение DocLine оправдано (ведь данная технология предназначена именно для организации повторного использования).

Мы выделили следующие характеристики инструментов по поиску клонов, важные с точки зрения нашего исследования.

1. *Способ использования*: автономные средства (CloneDR, Duplo, Clone Miner) и встраиваемые в среды разработки, такие как Eclipse и VisualStudio (SDD, CloneDifferentiator). Нас интересовали автономные средства (нам достаточно того, чтобы инструмент запускался из командной строки, имел бы набор настроек-параметров, а также, чтобы нам был известен формат файла, который выдаёт результаты его работы).

2. *Возможность поиска клонов в «плоском» тексте.* Почти все рассмотренные инструменты осуществляют поиск клонов в программах, то есть в текстах на языках программирования (CloneDR, Duplo, CloneDifferentiator), и либо вообще не поддерживают поиск в «плоском» тексте, либо показывают в этой ситуации неудовлетворительные результаты (SDD).

3. *Качество найденных клонов:* «чёткие» клоны – результаты в одной группе клонов полностью совпадают (Duplo, Clone Miner), «нечёткие» клоны – результаты, находящиеся в одной группе могут совпадать не полностью, но близки по какой-то эвристике (SDD, CloneDR, CloneDifferentiator, Clone Miner). Мы решили использовать чёткие клоны, потому что эвристики для поиска нечётких клонов в документации могут оказаться существенно иными, чем в программах на языках программирования.

4. *Лицензия на продукт*. Нас интересовали продукты, имеющие свободно распространяемую лицензию.

5. *Поддержка русского языка* – возможность поиска клонов в текстах на русском языке. Нас, конечно же, интересовали именно такие средства. Из всех рассмотренных продуктов русский язык поддерживал только Clone Miner.

В итоге мы выбрали пакет Clone Miner, который максимально соответствовал нашим требованиям.

2.5. CLONE MINER

Различные методы и подходы позволяют находить так называемые простые клоны. Авторы инструмента Clone Miner [10] предлагают анализировать группы простых клонов (с помощью методов интеллектуаль-

ного анализа данных) и за счёт этого получать более сложные и интересные для пользователя результаты – клоны высокого уровня (структурные клоны Higher-Level Clones in Software). Новизной данного подхода является концепция структурных клонов и применение для их поиска интеллектуального анализа данных (DataMining).

Инструмент позволяет работать из командной строки и имеет следующие параметры: файл (или файлы) с входными данными (указывается путь), файл с результатами поиска, минимальный размер клона. Например, если последний параметр будет равен трём, то в результатах поиска не будет клонов, состоящих из одного или двух слов (термов). Результаты поиска выводятся в специальный файл, который содержит список групп клонов. Первая строка каждой группы клонов содержит уникальный идентификатор группы, длину клона в термах и количество клонов в группе (представителей клона). Следующие далее *n* строк (где *n* – количество клонов в группе) содержат информацию о начальной и конечной позициях (позиция – это номер строки и номер символа в этой строке) каждого конкретного клона в исходном тексте.

В инструменте реализована поддержка формата Unicode, что позволяет применять инструмент к русскоязычной документации.

3. ОПИСАНИЕ ПОДХОДА

Алгоритм работы нашего подхода представлен на рис. 1. Сначала пользователь выбирает в DRL-документе информационный элемент, в котором он хочет искать клоны. Если имеется обычный текстовый документ и стоит задача оценки применимости технологии DocLine, или имеется DocBook-текст, в котором ещё не выделены повторно используемые компоненты и отсутствует DRL-разметка, то ко всему такому тексту применяется операция рефакторинга «Переход к DRL». В результате выполнения этой операции появляется информационный элемент, в который входит весь текст, и в нём уже можно искать клоны. Далее выбранный информационный элемент записывается в отдельный файл, который подаётся на вход Clone Miner. Последний выдаёт результаты, которые фильтруются, чтобы стать корректными с точки зрения DRL и DocBook – поиск «плоских» клонов может нарушить XML-структуру документа и выдать некорректные XML-конструкции.

Далее пользователю предлагается выполнить одну из следующих операций рефакторинга – «Выделить в информационный элемент» (группа 2 операций рефакторинга DocLine) или и «Выделить в элемент словаря» (группа 3 операций рефакторин-

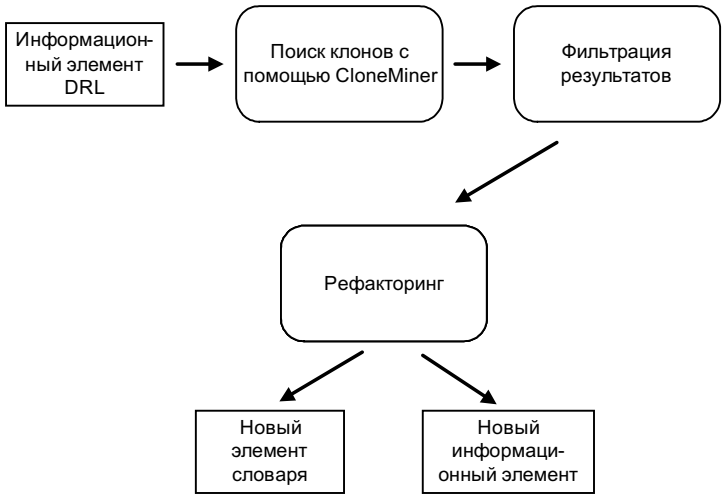

Рис. 1. Схема подхода

га DocLine). Применение первой операции целесообразно в том случае, когда в клоны попал большой фрагмент текста, применение второй – если найден многократно повторяющийся небольшой фрагмент текста (как правило, 1–3 слова). В результате выполнения этих операций все вхождения клонов в данной группе будут заменены ссылками на выделенный элемент, а сам выделенный элемент будет вынесен в отдельное описание.

Автоматическая фильтрация результатов, полученных после работы инструмента Clone Miner, необходима для предоставления пользователю только корректных DRL-клонов. Если клон является некорректным с точки зрения DRL, то наш алгоритм пытается выделить в нем максимальные корректные элементы (учитывая древовидную структуру XML). В некоторых случаях это может быть один DRL-элемент (или несколько подряд идущих элементов, являющихся «братьями» с точки зрения XML-структуры), от которого мы отбрасываем некорректное начало и/или некорректный хвост. В более сложных случаях, когда отбрасываемые начало и/или хвост также содержат существенные конструкции (например, элементы находятся на одном уровне вложенности с точки зрения XML, но имеют разных родителей), и выделение только первого валидного фрагмента текста слишком расточительно, приходиться разбивать данный элемент на несколько валидных DRL-фрагментов. В редких случаях клон игнорируется как незначащий.

4. ОПИСАНИЕ РЕАЛИЗАЦИИ

Предложенный подход был реализован средствами Java/Eclipse и интегрирован со средой DocLine. На рис. 2 представлено окно редактора DocLine, позволяющее запустить операцию поиска клонов для вы-

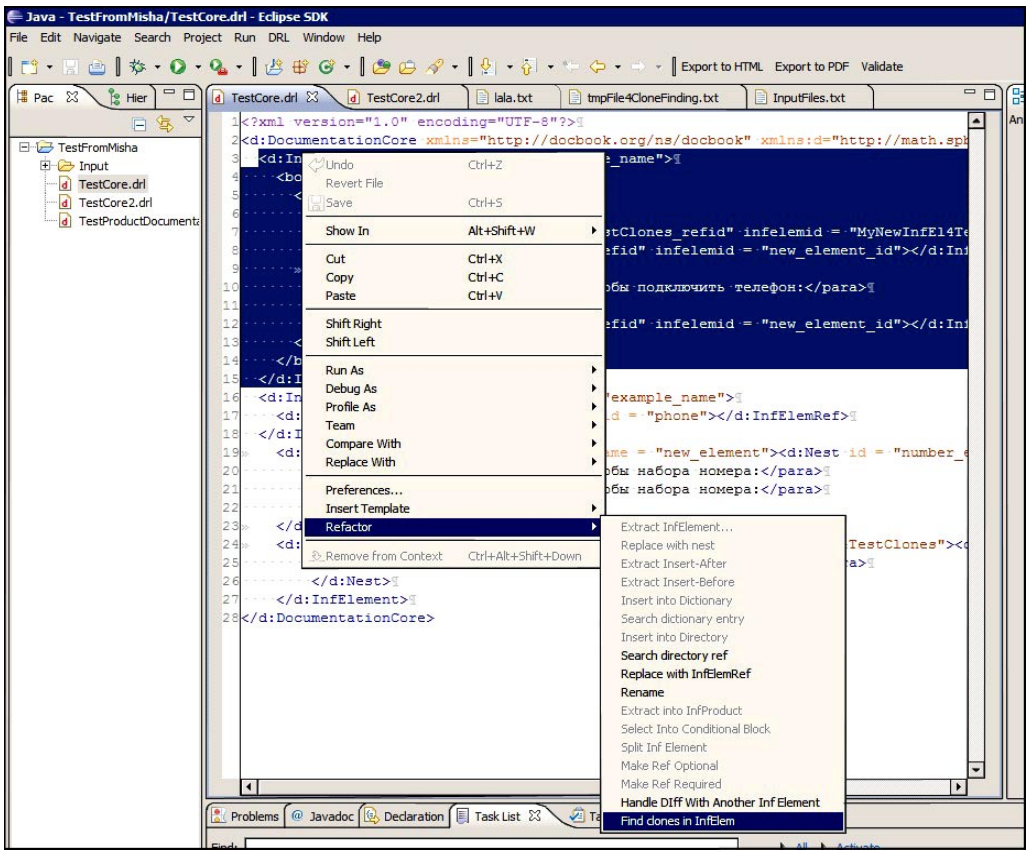

Рис. 2. Вызов операции по поиску клонов

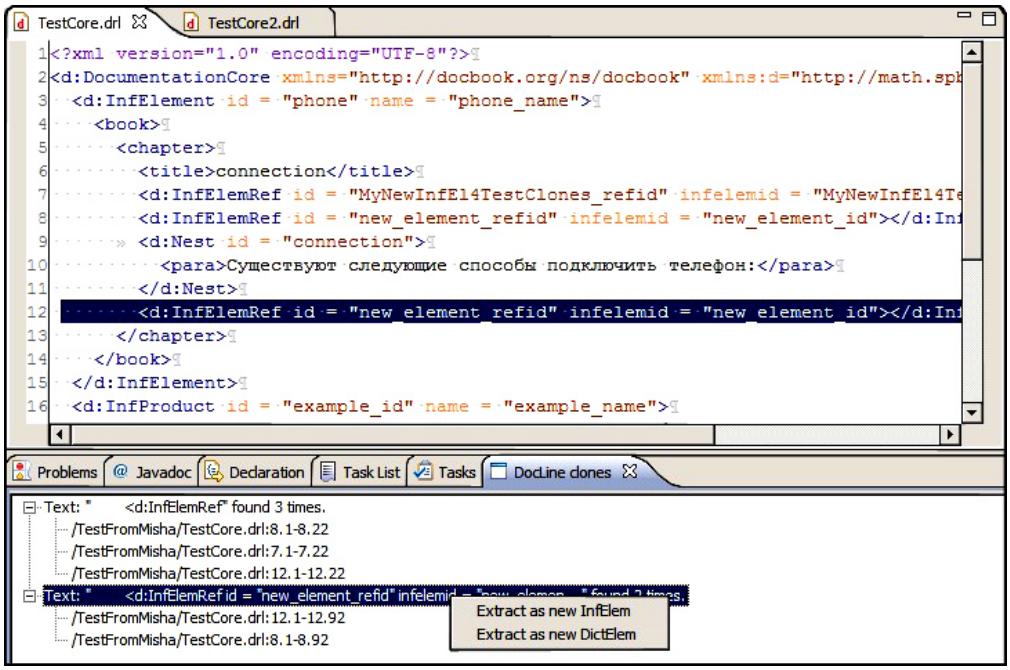

Рис. 3. Работа с выделенными клонами

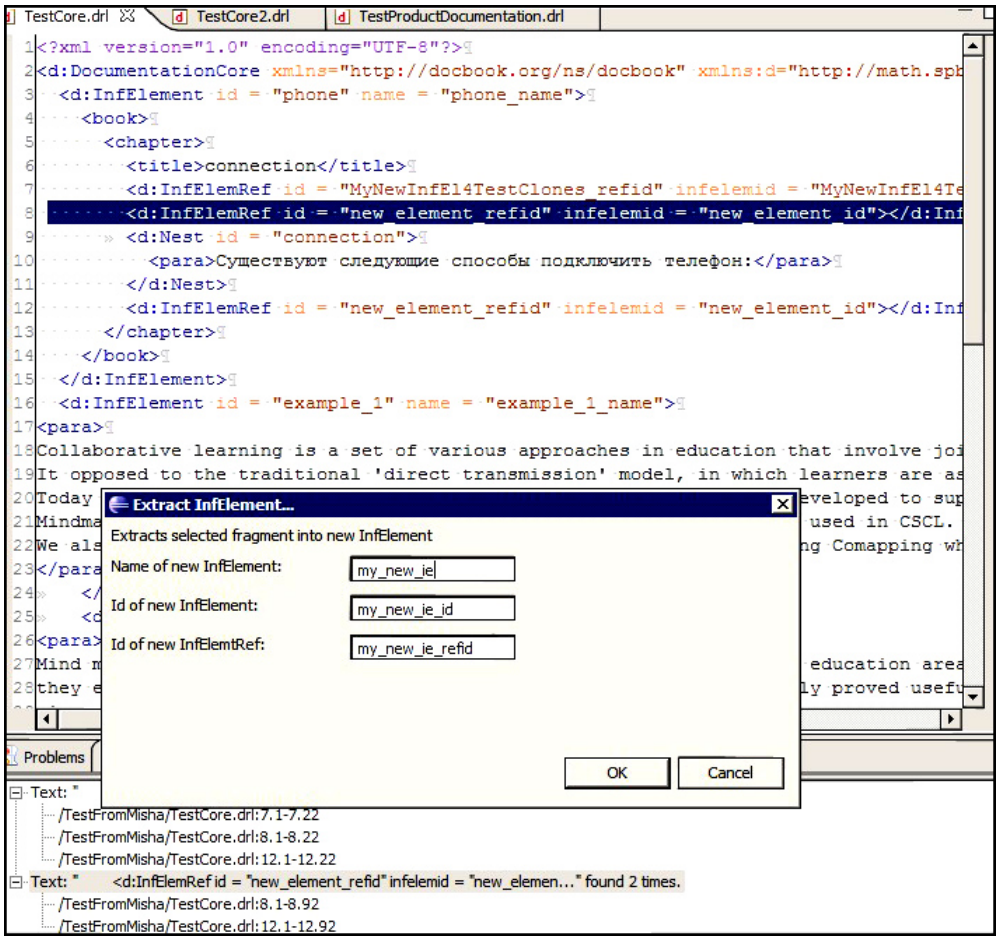

Рис. 4. Создание на основе группы клонов нового информационного элемента

деленного информационного элемента. Если в DRL-тексте выделен не информационный элемент, или информационный элемент выделен не полностью, то данная операция будет неактивной.

После вызова операции поиска клонов в нижней части экрана появляется новая вкладка «DocLine clones» (см. рис. 3), которая содержит двухуровневое дерево с результатами поиска клонов. Первый уровень содержит краткое описание каждой найденной группы клонов (текст клона, количество термов в клоне, количество представителей данного клона), второй уровень содержит полный список представителей данной группы клонов вместе с информацией об их местонахождении в тексте.

При обращении к конкретному представителю данной группы клонов (двойной щелчок мышьи на элемент дерева второго уровня) его содержание подсвечивается в DRL-редакторе (подсвечивается именно данный представитель группы). Если пользователь решит, что он хочет выделить всю группу клонов в общий актив, то он может вызвать всплывающее меню (правый щелчок мышина элемент дерева первого уровня) для данной группы клонов. Всплывающее меню содержит два вида действий: «Выделить в новый информационный элемент» (пункт меню «Extract as new Inf Elem») и «Выделить в новый элемент словаря» (пункт меню «Extract as new Dict Elem») .

Данный выбор предоставлен пользователю, для того чтобы он мог решить, какого рода общий актив должен появиться в системе – новый информационный элемент или новый элемент словаря.

После вызова любой из этих операций открывается соответствующее меню, предлагающее ввести идентификаторы создаваемого элемента (см. рис. 4). Далее автоматически выполняется работа по созданию нового элемента и производится замена всех клонов ссылками на этот новый элемент, а также обновляется список клонов в окне «DocLine clones».

5. ЭКСПЕРИМЕНТЫ ПО ОЦЕНКЕ ЭФФЕКТИВНОСТИ ПОДХОДА

Для оценки эффективности предложенного подхода был выбран англоязычный документ, относящийся к проекту с открытым исходным кодом VaadinTrac, который посвящён созданию инструментов для разработки высококачественных Web-приложений [21]. Данный документ является описанием выпусков (release notes) набора последних версий продукта (добавленных новых возможностях и изменённых старых). Данный документ представлен как в виде «плоского» текста, так и в формате DocBook, и является достаточно большим (pdf-версия, сгенерированная на основе DocBook-представления, составляет 53 страницы). Наличие содержательных DocBook-примеров позволяет проверить эффективность предложенного нами подхода на XML-текстах.

Документ содержит большое количество повторяющихся фрагментов: наш инструмент, применённый к «плоской» версии, обнаружил 73 группы клонов.

Далее мы исследовали 10 фрагментов этого документа объёмом по 2–3 страницы. Каждый фрагмент проверялся с помощью инструмента как в виде «плоского» текста, так и в формате DocBook. Полученные результаты сравнивались, во-первых, попарно (для «плоской» и DocBook-версий фрагмента), а также с обычным, «ручным» поиском клонов. Поскольку фрагменты были небольшими, то нетрудно было «вручную» найти все значимые клоны (в случае, когда документы большие, это становится серьёзной проблемой).

В «плоских» фрагментах наш инструмент нашёл все значимые клоны, однако, результаты его работы на DocBook-тестах были неоднозначны.

1. Было найдено много групп клонов, состоящих из пары тегов и слова между ними, например, <title>params</title>. Очевидно, что такие клоны малосодержательны с точки зрения задачи поиска повторов в документации. Кроме этого, различные группы иногда содержали в точности оди-

наковые клоны, что произошло из-за того, что исходно клоны различались, но после фильтрации стали одинаковыми. В целом, при наличии 73 содержательных групп клонов в данном тексте (напомним, его объем составляет 53 страницы), применение нашего инструмента выдало около 120 несущественных групп конов.

2. Ряд существующих в документации повторов, найденных «вручную», а также при анализе «плоского» текста, был не найден в соответствующих DocBook-тестах: недоставало иногда целых групп, а также количество клонов в некоторых группах было меньшим, чем в действительности. В целом было потеряно около 1/3 повторяющихся фрагментов. Это происходило, когда граница повторяющегося фрагмента не совпадала с границей объемлющей его XML-конструкции. В этом случае алгоритм фильтрации обрезает фрагмент слева или справа (или и слева, и справа) до максимальной по длине последовательности корректных XML-конструкций, полностью входящих в него.

6. ЗАКЛЮЧЕНИЕ

В данной работе представлен алгоритм для поиска клонов в XML-документации, основанный на инструменте поиска клонов в «плоском» тексте Clone Miner и идее рефакторинга XML-документации. Алгоритм реализован в среде Java/Eclipse и интегрирован со средой разработки документации DocLine. Также выполнена апробация разработанного инструмента.

Следует отметить, что результаты, изложенные в данной работе, являются, во многом, лишь первым шагом в применении методов поиска клонов в программном обеспечении для поиска повторяющихся фрагментов в технической документации. Поскольку существует большое количество вариантов решения данной задачи, многие из которых весьма трудоёмки, в рамках данного исследования мы поставили себе задачу получить информацию для уточнения дальнейших исследований.

Предложенный подход показал хорошие результаты при поиске повторяющихся фрагментов в обычном, «плоском» тексте. Однако его использование для XML-текстов приводит пока к потере значимых результатов, а также к большому количеству ложных срабатываний. Это происходит, главным образом, из-за того что фильтрация результатов представленным в работе способом оказалось не вполне удачной: поддерживать XML-корректность полученных после поиска клонов результатов нужно более сложным способом, не теряя при этом «текстовой похожести» фрагментов текста.

Кроме того, стало понятно, что необходимо предоставить пользователю более разнообразный спектр автоматизированных операций над полученными группами клонов:

– удаление группы клонов из списка;

– уменьшение/увеличение клона;

– разделение одного клона на несколько (бывает полезно с семантической точки зрения).

Требует также отдельного исследования задача поиска нечётких клонов, поскольку, как показали наши исследования, таких клонов в промышленной документации достаточно много. Тут возможны различные подходы – от простых методик, использующих результаты поиска строгих клонов, до создания различных эвристик «похожести» фрагментов текста, расширяющих результаты поиска (сейчас возможны лишь два значения – совпадает/не совпадает). Например, возможен вариант поиска нечётких клонов по шаблону, который создаётся путём анализа документов с помощью инструмента поиска строгих клонов. Вот пример такого шаблона, созданный при анализе документа [21]: «The following changes have been made to the * code since the 1.*.* release.» Места подстановок обозначены с помощью символа «*».

Также имеет смыл разработать универсальный инструмент для поиска клонов в текстах, разметка которых выполнена популярными XML-форматами, такими как DocBook [24], DITA [17] и некоторыми другими.

Однако необходимо отметить, что, несмотря на указанные недостатки, инструмент, представленный в данной работе, даже в текущем состоянии способен оказать существенную помощь при анализе промышленной документации на наличие повторно используемых фрагментов.

Литература

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, 2009. Т. 4, № 1. С. 128–150.

3. *Романовский К.Ю.* Метод разработки документации семейств программных продуктов // Системноепрограммирование, 2006. Т. 2, № 1. С. 191–218.

4. *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.

5. *Романовский К.Ю., Кознов Д.В.* Язык DRL для проектирования и разработки документации семейств программных продуктов // Вестник С.-Петербург. ун-та. Сер. 10, 2007. № 4. С. 110–122.

6. *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, 2010. Т. 5, № 1. С. 32–51.

7. *Смирнов М.Н., Соколов Н.Е., Романовский К.Ю.* DocLineFM: среда разработки повторно используемой документации семейств программных продуктов на базе пакета AdobeFrameMaker // Системное программирование, 2011. Т. 6, №1. С. 77–94.

8. *Фаулер М.* Рефакторинг: улучшение существующего кода / Пер. с англ. СПб: Символ-Плюс, 2003.

9. *Akhin M., Itsykson V.* Clone detection: why, what and how? 2010 6th Central and Eastern European Software Engineering Conference, CEE-SECR 2010 Moscow, 2010. С. 36–42.

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

11. *Calheiros F., Nepomuceno F.* Product Line Variability Refactoring Tool / http://twiki.cin.ufpe.br/twiki/pub/SPG/GenteAreaPublications/WRT07_calheiros.pdf (дата обращения: 29.12.2012).

12. *Clements, P., Northrop, L.* Software Product Lines: Practices and Patterns. Boston. MA: Addison-Wesley, 2002.

13. CloneDifferentiator tool / http://www.comp.nus.edu.sg/~yinxing/clonedifferentiator/ (дата обращения:29.12.2012).

14. CloneDR tool насайте Semantic Design / http://students.cis.uab.edu/tairasr/clones/literature/ (дата обращения:29.12.2012).

15. Code clone Literature on site of The University of Alabama at Birmingham // http://students.cis.uab.edu/tairasr/clones/literature/ (дата обращения:29.12.2012).

16. *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.

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

18. Duplo tool на сайте Sourceforge // http://sourceforge.net/projects/duplo/.

19. *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.

20. OpenXML, http://www.philo.de/xml/index.shtml (дата обращения: 29.12.2012).

21. Release Notes document for the VaadinTrac version 5.2 // http://dev.vaadin.com/svn/versions/5.2/build/docbook/xsl/ (дата обращения:29.12.2012).

22. *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.

23. SDD tool на сайте Eclipse / http://wiki.eclipse.org/index.php/Duplicated_code_detection_tool_%28SDD%29 (дата обращения: 29.12.2012).

24. *Walsh N., Muellner L.* DocBook: The Definitive Guide. O'Reilly, 2003.

Abstract

One of he key issues of reuse management of software documentation is a finding duplicated document fragments. It is important when we apply reusable approaches to existing documentation. In turn it can be done under documentation revision: it appeared, the volume of the documentation was big, and there were a lot of problems in maintenance. We consider XML documentation because this is mainstream in technical documentation development now. We offer to use software clone detection technique for finding document duplicated fragments. As a basis clone detection tool we used Clone Miner. We present also a tool that implements of the approach we offered.

Keywords: software clone detection, software product lines, documentation reuse, variability management.

Кознов Дмитрий Владимирович, ***кандидат физико-математических наук, доцент кафедры системного программирования математико-механического факультета СПбГУ,*** ***dkoznov@yandex.ru,***

Шутак Артем Витальевич, ***аспирант кафедры системного программирования математико-механического факультета СПбГУ,*** ***artem.shutak@gmail.com,***

Смирнов Михаил Николаевич, ***старший преподаватель кафедры системного программирования математико-механического факультета СПбГУ,*** ***mihail.smazhevsky@gmail.com***

Смажевский Михаил Александрович, ***аспирант кафедры системного программирования математико-механического факультета СПбГУ,*** ***smnsmn1979@gmail.com.***