САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Лебедкова Татьяна Александровна

Расширение метода рефакторинга документации семейств программных продуктов

Выпускная квалификационная работа

«Допустить к защите»
Зав. кафедрой
д. ф.-м. н., профессор …… А. Н. Терехов

Научный руководитель …… Д.В. Кознов
к. ф.-м. н, доцент

Рецензент …… К.Ю. Романовский
к. ф.-м. н,
старший преподаватель

Санкт-Петербург
2010

St. Petersburg State University

Faculty of Mathematics and Mechanics

Chair of Software Engineering

Improving of product line documentation refactoring techniques

Graduate paper by

Tatiana Aleksandrovna Lebedkova

"Admitted to proof"
Head of the chair, Prof., PhD

_____ A. N. Terekhov

Scientific advisor
Associate Prof., PhD

_____ D. V. Koznov

Reviewer
Assistant Prof., PhD

_____ K. Y. Romanovsky

St. Petersburg, 2010

Содержание

1. Введение

Любой программный продукт обычно сопровождается технической документацией. Такая документация бывает разной: общие описания программных систем, различные руководства (пользователей, администраторов, операторов), справочные системы и др. Чем сложнее программный продукт, тем сложнее его техническая документация, и разработка документации оказывается трудоёмким процессом.

Зачастую программные продукты удобно разрабатывать семействами. Разработка семейств программных продуктов (СПП) является одним из подходов к промышленной разработке программного обеспечения и использует единый процесс разработки на основе повторно используемых активов [5]. Документация к таким продуктам имеет широкие возможности для переиспользования: она может разрабатываться в контексте целого семейства и уточняться для каждого конкретного продукта.

Существуют разные XML-подходы к разработке технической документации, в некоторой степени поддерживающие повторное использование. Среди них выделяются DITA [13] и DocBook [16]. Основной постулат технологии DITA состоит в том, что текст технической документации состоит из типизированных фрагментов (т.н. топиков), которые и переиспользуются. DocBook — это технология разработки документации, позволяющая из единого представления документа автоматически получать разные выходные форматы (наиболее распространенные — HTML, PDF), с разными вариантами компоновки и оформления. Несмотря на широкое распространение, оба подхода не поддерживают независимую настройку общих активов в документации каждого из продуктов семейства: либо общие фрагменты текста используются без адаптации под конкретный продукт, либо они копируются, и изменяются уже их копии.

На кафедре системного программирования математико-механического факультета СПбГУ выполняется исследовательский проект DocLine, посвященный созданию метода разработки документации на основе повторного использования [2]. Он включает в себя процесс разработки документации, язык разметки документации, имеющий текстовую и графическую нотации, а также пакет инструментальных средств. В отличие от упомянутых выше технологий, данный подход ориентирован на СПП.

В рамках проекта DocLine была предложена идея рефакторинга документации [1]. Под рефакторингом понимается изменение внутренней структуры документации с сохранением её конечного представления. Предложенные в работах [1, 5] операции

рефакторинга поддерживают извлечение повторно используемых фрагментов из обычного текста и их настройку.

Тем не менее, не все необходимые операции были реализованы, а, реализованные не лишены недостатков (например, неудобство пользовательских интерфейсов), что осложняет их практическое использование. Первая из задач данной работы состоит в реализации недостающих операций рефакторинга и усовершенствовании существующих.

Текущая версия программного средства DocLine реализована в виде модулей, подключаемых к среде разработки Eclipse, и для работы с технологией необходима установка этой среды. Техническим писателям, не работающим с Eclipse, удобнее будет использовать автономное приложение. Вторая задача данной работы состоит в создании версии программного средства DocLine, распространяемой отдельно от среды разработки Eclipse[1].

[1] Eclipse – среда разработки для создания модульных кроссплатформенных приложений.

2. Постановка задачи

Задача данной работы состоит в улучшении средств рефакторинга документации, реализованных в рамках исследовательского проекта DocLine. Под улучшением подразумевается расширение набора реализованных операций рефакторинга и предложение способов улучшения существующей реализации. Задачу можно разбить на следующие этапы.

1. Улучшение рефакторинга документации.
   - Изучить процесс разработки документации в рамках технологии DocLine.
   - Проанализировать реализованные методы рефакторинга и архитектуру их программной поддержки [3].
   - На основании результатов этого анализа предложить способы усовершенствования реализованных операций.
   - Спроектировать и реализовать дополнительные операции рефакторинга.
   - Встроить их в технологию DocLine, используя существующую архитектуру средства поддержки рефакторинга.
2. Создание версии программного средства DocLine, распространяемой отдельно от платформы Eclipse.

3. Обзор литературы

3.1. Семейства программных продуктов

Повторное использование программных компонент значительно снижает затраты при разработке ПО. Однако без специальной организации процесса разработки, программные компоненты сложно сделать переиспользуемыми. Для решения этой задачи был создан подход разработки программных продуктов семействами.

Семейство программных продуктов – это набор продуктов, которые предназначены для одного сегмента рынка или выполняют единую миссию и при этом создаются на базе общих, повторно используемых активов, с помощью хорошо налаженного процесса повторного использования [5]. В качестве повторно используемых активов, например, могут выступать:

- требования;
- архитектура;
- компоненты;
- различные модели (анализа, проектирования и т.д.);
- средства разработки;
- процессы (например, методы управления проектом);
- квалификация сотрудников.

Каждый продукт семейства разрабатывается на основе общих активов и заложенной в них вариативности, а также новых индивидуальных активов, необходимых именно для этого продукта.

Процесс разработки семейства программных продуктов состоит из следующих тесно связанных между собой этапов [15].

1. Разработка общих активов (Core Assets Development). Основывается на требованиях к продуктам, стратегии разработки, ресурсах и ограничениях. При этом разрабатываются: архитектура семейства программных продуктов, спецификация области разработки семейства продуктов, метод использования общих активов, «привязка» требований к архитектуре и сами общие активы.

2. Разработка продуктов семейства (Product Development). Здесь происходит на основе данных, полученных на предыдущем этапе, и требований к конкретному продукту.

3. Управление разработкой семейства продуктов (Product Management). Подразумевает контроль исполнения метода разработки семейства, управление проектами, анализ рынка, обеспечение ресурсами, управление взаимодействием с внешним миром и осуществление стратегического планирования.

Основные преимущества описанного подхода заключаются в следующем:

- снижение стоимости разработки ПО;
- быстрый вывод продуктов на рынок;
- обеспечение широкой приспособляемости к требованиям заказчиков.

3.2. Технология DocBook

DocBook – это XML-технология, предназначенная для разметки документов [12]. Технология DocBook [16] состоит из языка, основанного на XML, и программных утилит, осуществляющих преобразование DocBook-документа в форматы, доступные для печатного представления (PDF, HTML и др.). Основной идеей технологии является разделение содержания документа и его форматирования, что позволяет поддерживать единое исходное представление документа (single source) [9] и на его основе автоматически генерировать документацию в различных форматах. Язык DocBook является стандартом для разработки технической документации на протяжении многих лет, несмотря на наличие большого количества других форматов и текстовых редакторов (OpenOfficeWriter, MicrosoftWord и т.д.) [12].

Использование DocBook дает наибольший эффект в следующих ситуациях [12]:

- когда в документацию постоянно приходится вносить изменения;
- когда над документацией работает коллектив;
- когда документация включает в себя четко структурированные составляющие;
- когда необходимо строго выдерживать определенный стиль оформления для множества документов;
- когда документацию требуется выпускать в различных форматах;
- когда требуется выпускать разные версии документации с отличающимся содержимым.

3.3. Технология DITA

DITA — это основанная на XML технология, охватывающая весь цикл разработки технической документации. Она включает в себя набор принципов, описывающих как создание модулей с «типизированной информацией» на уровне отдельных тем, так и использование этой информации, например, в онлайн-справке или на веб-портале технической поддержки [8].

Технология DITA была разработана в корпорации IBM в 1999–2000 годах, официально объявлено о ней было в 2001 году. В настоящее время технология развивается силами OASIS DITA Technical Committee и IBM. Технология DITA применяется для

разработки технической документации в ряде крупных корпораций, в том числе, IBM, Adobe, Nokia. Во многих средствах разработки технической документации и XML-редакторах имеется встроенная поддержка языка разметки DITA.

Текст технической документации DITA состоит из модулей с типизированной информацией, которые называются топиками. При этом технология DITA позволяет автору декларировать новые типы топиков для нужд конкретного проекта.

Такой подход противопоставляется написанию длинных книгоподобных документов, таких как XML-документы в формате DocBook.

3.4. Технология DocLine и язык DRL

Метод DocLine предназначен для разработки документации семейств программных продуктов. Основная идея данного подхода заключается в возможности повторного использования фрагментов документов с адаптацией под контекст использования.

Метод DocLine [2,3] включает в себя:

- процесс разработки документации;
- язык DRL (имеет текстовую и графическую нотации);
- пакет инструментальных средств.

3.4.1. Процесс разработки документации

Процесс разработки документации СПП схож с разработкой самих СПП. Существуют две схемы разработки СПП: проактивная (снизу вверх) [6] и реактивная (сверху вниз) [7]. Обе модели поддерживаются методом DocLine, применительно к разработке документации.

Проактивная схема разработки документации СПП предписывает, что создание документации начинается с всестороннего анализа возможностей повторного использования, проектирования структуры документации и описания переиспользуемых фрагментов документации. Затем на основе созданных общих активов порождается документация конкретных продуктов. Эта модель применима в том случае, если заранее известно, какие продукты будут в семействе. Однако на практике часто используется реактивная модель, когда разработка документации начинается с создания документации для первого продукта семейства. При появлении необходимость в разработке документации других продуктов, проводится анализ возможностей повторного использования, проектирование пакета документов, выделение общих фрагментов и разработка документации очередного продукта на основе полученных общих активов.

3.4.2. Графическая нотация DRL

Графическая нотация DRL предназначена для визуального проектирования документации. Она дает понимание того, какова структура семейства, структура информационных продуктов (ИП), какие существуют информационные элементы (ИЭ), и как ИП[2] и ИЭ[3] связаны между собой.

Графическая нотация поддерживает три вида диаграмм:

- главная;
- вариативности;
- продукта.

Главная диаграмма, изображенная на рисунке 1, дает представление обо всех продуктах семейства и информационных продуктах и позволяет их связать.

Рисунок 1: Главная диаграмма [4]

Структура ИП представляется с помощью диаграммы вариативности, представляющей собой деревянную конструкцию. Диаграмма изображена на рисунке 2.

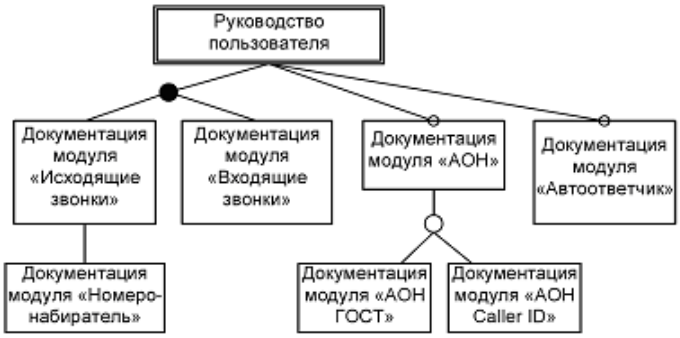

Рисунок 2: Диаграмма вариативности [4]

[2]Информационный продукт (ИП) – шаблон целевого документа [2].

[3] Информационный элемент (ИЭ) – часть ИП, является обособленным фрагментом документации, подготовленным к переиспользованию, и может быть как синтаксически обособленным модулем (главой, секцией и т.п.), так и плоским фрагментом текста [2].

ИП, отображаемый прямоугольником в двойной рамке, является корнем этого дерева, остальные узлы – ИЭ. Связи на этой диаграмме обозначают отношения включения.

Отношения, отображаемые кругами[4], могут быть следующих типов:

- строгое альтернативное включение (XOR): необходимо включение ровно одной ссылки из группы;
- нестрогое альтернативное включение (OR): допустимо включение непустого множества ссылок из группы.

Также включения могут быть необязательными (например, связь между ИП «Руководство пользователя» и ИЭ «Документация модуля «АОН») и обязательными (связь между ИЭ «Документация модуля «Исходящие звонки» и ИЭ «Документация модуля «Номеронабиратель»).

Диаграмма продукта, представленная на рисунке 3, предназначена для описания документации конкретного продукта.

Рисунок 3: Диаграмма продукта [4]

Диаграмма документации продукта получается из диаграммы вариативности путем выбора нужных ссылок в группах ссылок и удаления ненужных ссылок из числа необязательных.

3.4.3. Текстовая нотация DRL

Текстовая нотация DRL предназначена для создания текста документации с последующей трансляцией в форматы PDF или HTML.

Язык DRL основан на XML, каждая его конструкция является XML-тегом. Элементы DRL бывают двух видов: относящиеся тексту и его разметке и управляющие структурой документации. Элементы первого вида заимствованы из DocBook – XML-формата разметки.

[4] Круг в данном случае – группа ссылок на ИЭ.

Элементы управления структурой позволяют организовать повторное использование частей документации – как «крупноблочное», так и «мелкозернистое», а также адаптировать фрагменты текста к контексту использования.

Корневая конструкция DRL – семейство продуктов, с её помощью описывается всё, что касается семейства программных продуктов. Семейство продуктов в DRL содержит схему семейства и документацию.

Схема семейства описывает продукты, из которых состоит семейство. Документация в DRL делится на документацию семейства и документацию продукта.

Документация семейства указывает, какие документы ИП могут входить в семейство, и как они устроены: из каких ИЭ состоят, где допустима настройка повторного использования и какая.

Документация продукта состоит из набора специализированных информационных продуктов и описывает, как именно нужно модифицировать документацию семейства, чтобы получить пакет документов для конкретного продукта семейства. Каждый специализированный информационный продукт соответствует одному информационному продукту и содержит спецификации и настройки, необходимые, чтобы на основе этого информационного продукта получить документ. Также он содержит ссылку на продукт из схемы семейства, к которому полученный документ будет относиться.

Файлы документации в формате DRL являются XML-файлами и делятся на три вида, каждому из которых соответствует свой корневой элемент.

Файл, содержащий информацию о схеме семейства продуктов, в качестве корневого элемента имеет тег ProductLine (Листинг 1).

```
<ProductLinename="Семейство телефонных аппаратов Унител">
<Product name="Унител-таксофон" id="payphone"/>
<Product name="Унител-автоответчик" id="answermachine"/>
<Product name="Унител-АОН" id="callerid"/>
</ProductLine>
```

Листинг 1: Схема семейства продуктов [4]

Каждый из продуктов имеет имя (атрибут name) для восприятия человеком и идентификатор (атрибут id) для ссылок.

Документация семейства хранится в файлах с корневым элементом DocumentationCore (Листинг 2).

```
<DocumentationCore>
<InfProduct id="userguide" name="Руководствопользователя"/>
<InfElement id="numberid" name="АОН"/>
<Dictionary id="maindict" name="Основнойсловарь"/>
```

```
<Directory id="maindir" name="Основнойкаталог"/>
</DocumentationCore>
```

Листинг 2: Документация семейства [4]

Тег InfProduct здесь соответствует объявлению информационного продукта, InfElement – информационного элемента. Теги Dictionary и Directory содержат спецификации соответственно словарей и каталогов и используются для организации мелкозернистого переиспользования. Все элементы имеют идентификатор (атрибут id) и имя (атрибут name), они используются тем же образом, что и в предыдущем примере.

Документация продукта хранится в файле с корневым элементом ProductDocumentation (Листинг 3).

```
<ProductDocumentation productid="callerid">
  <FinalInfProduct infproductid="userguide">
    <Adapter infelemrefid="CallerID_aon_ref">
      <Insert-After nestid="number_indication_methods">
        Также предусмотрена возможность
        звуковой индикации номера абонента.
      </Insert-After>
    </Adapter>
  </FinalInfProduct>
  <Dictionary id="dict_userguide">
    <!-- ... -->
  </Dictionary>
</ProductDocumentation>
```

Листинг 3: Документация продукта [4]

Атрибут productid задаёт привязку к документу из схемы семейства. FinalInfProduct определяет специализированный информационный продукт, идентификатор соответствующего ему информационного продукта задан атрибутом infproductid. Теги Adapter служат для адаптации информационных элементов. В случае, рассмотренном в примере, в информационном элементе с идентификатором CallerID_aon_ref вставляется текст перед точкой расширения с id, равным dict_userguide.

3.4.4. Инструментальные средства

Инструментальный пакет DocLine интегрирован в среду разработки Eclipse [14]. Пакет включает следующие компоненты:

- графический редактор схем повторного использования, позволяющий создавать главную диаграмму (семейства), диаграмму конкретного продукта и диаграмму вариативности;

- текстовый (XML) редактор, позволяющий разрабатывать тексты документации, настраивать повторное использование и проводить публикацию целевых документов в форматах PDF и HTML.

Редакторы интегрированы друг с другом: изменения в тексте после сохранения отображаются на диаграммах и, наоборот, изменения в диаграммах после сохранения отображаются на тексте.

3.4.5. Рефакторинг в DocLine

3.4.5.1. Понятие рефакторинга

Рефакторинг — это процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы [11]. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение) преобразований. Поскольку каждое преобразование маленькое, программисту легче проследить за его правильностью, и в то же время вся последовательность может привести к существенной перестройке программы и улучшению её согласованности и четкости.

Рефакторинг широко используется в «гибких» методах разработки ПО и позволяет реагировать на изменение требований к программному продукту без необходимости переписывать весь код.

Рефакторинг в контексте СПП представляет собой извлечение и уточнение повторно используемых активов, а также улучшение архитектуры всего СПП таким образом, чтобы поведение существующих продуктов оставалось неизменным [1].

3.4.5.2. Рефакторинг документации

Наряду с СПП, документация к ним тоже изменяется с течением времени: уточняется, дополняется. При добавлении нового продукта в семейство может возникнуть задача по извлечению и подготовке новых повторно используемых фрагментов текста для включения их в новый пакет документов. При этом целевое представление существующих документов не должно поменяться. Поэтому на документацию СПП также можно распространить идею рефакторинга.

3.4.5.3. Архитектура существующей реализации

При разработке архитектуры автоматизированного средства рефакторинга [3] учитывались следующие требования: легкость добавления новых операций, работа операций с несколькими файлами документации на DRL. Чтобы достичь этих результатов были спроектированы:

- модуль синтаксического разбора документации в формате DRL;

- модуль генерации текстовых файлов DRL из представления, используемого при проведении рефакторинга;
- набор базовых функций, полезных при проведении рефакторинга;
- структура типовой операции рефакторинга.

Архитектура средств рефакторинга схематично изображена на рисунке 4.

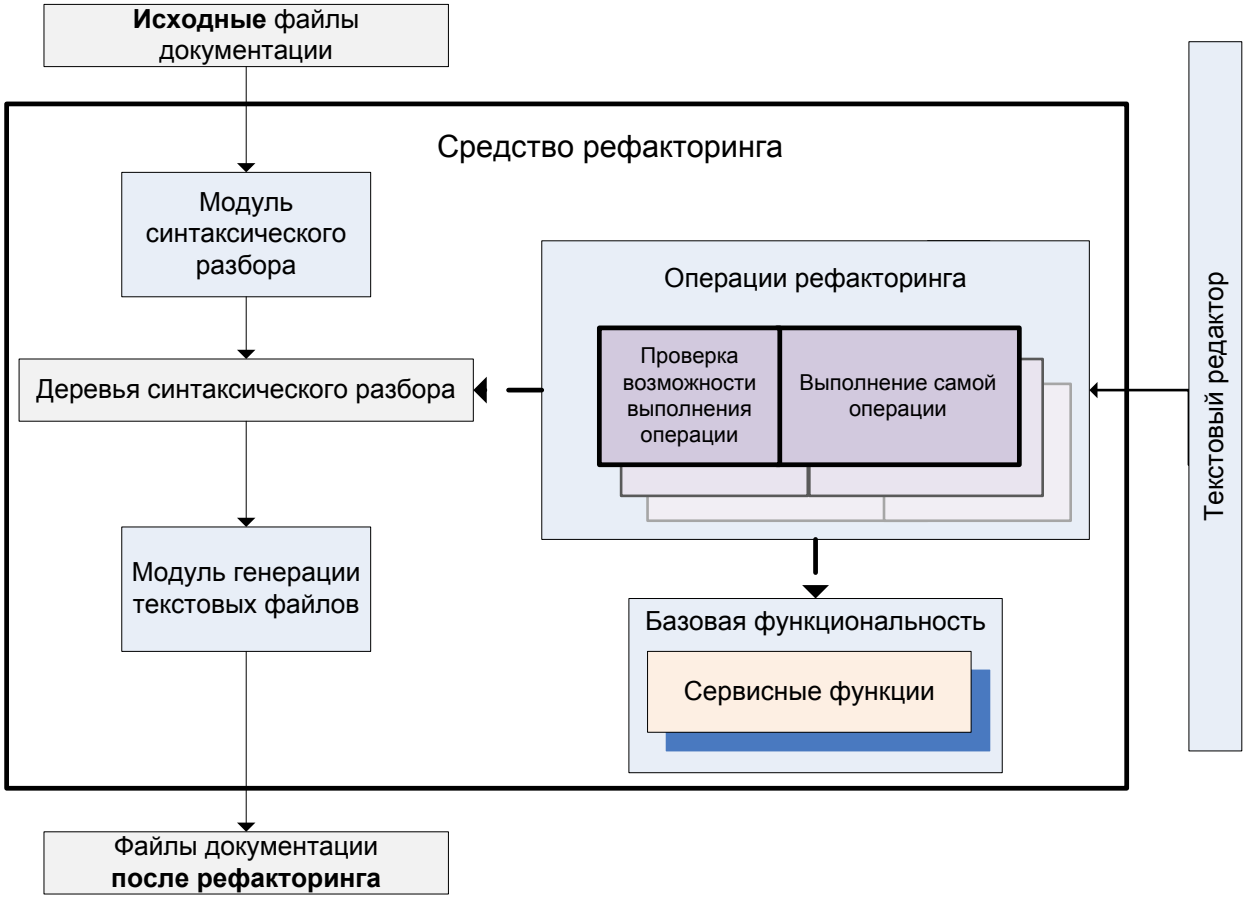

Рисунок 4: Схема средств поддержки рефакторинга в DocLine [3]

Модуль синтаксического разбора. Позволяет создать дерево синтаксического разбора файла DRL (с позициями элементов в тексте документа) и обновить специальный реестр (реестр ресурсов), куда сохраняются деревья синтаксического разбора и некоторые элементы DRL, например, ИЭ (информационный элемент, ссылки на ИЭ и др.). Реестр ресурсов позволяет разделять информацию между различными проектами (рабочее пространство в среде Eclipse делится на проекты), также реестр ресурсов обеспечивает доступ к деревьям синтаксического разбора, быстрый и удобный поиск различных элементов (ИЭ, ссылки на ИЭ и др.).

Модуль генерации текстовых файлов. Используется после проведения рефакторинга. Он позволяет из представления, используемого при рефакторинге, построить текстовые файлы DRL, похожие на исходные, которые были до проведения

рефакторинга (а если рефакторинг не изменял дерево синтаксического разбора, то конечные файлы должны остаться идентичными исходным).

Набор базовых функций. Используется во многих операциях и включает в себя: поиск элемента в дереве синтаксического разбора по позиции в документе, различные итераторы по дереву DRL и др.

Структура типовой операции рефакторинга. Каждая операция рефакторинга состоит из двух частей. В первой происходит проверка того, можно ли провести операцию над объектом, выбранным пользователем, а вторая выполняет саму операцию. Первая часть позволяет динамически отображать в консольном меню доступные для выбранного объекта операции. Для каждой операции доступна информация обо всех документах, находящихся в одном проекте вместе с инициировавшим операцию документом. Операции работают с реестром ресурсов. После того, как операция закончила работу, она должна сохранить произведённые изменения обратно в текстовый файл. Каждая операция должна сама следить за тем, какие файлы следует обновить после проведения рефакторинга (или обновить всю документацию проекта целиком).

3.4.5.4. Описание существующих операций

3.4.5.4.1. Вспомогательные операции

Первичный переход к документации с повторным использованием**.**

Эта операция позволяет переводить документ в формате DocBook в DRL. Для этого производятся следующие действия:

1) создаётся ИП и ФИП для этого ИП;
2) создается ИЭ и внутрь него помещается содержимое исходного документа;
3) в ИП создаётся ссылка на созданный ИЭ.

3.4.5.4.2. Операции выделения крупных фрагментов текста

Выделить информационный элемент.

Эта операция предназначена для переноса фрагмента текста в отдельный ИЭ. Для выполнения операции производятся следующие действия:

1) создается ИЭ, в него помещается предназначенный для переноса фрагмент текста;
2) вместо выделенного фрагмента создаётся ссылка на новый ИЭ.

3.4.5.4.3. Операции выделения небольших фрагментов текста

Вставить фрагмент текста в словарь.

Эта операция позволяет выделенный фрагмент текста вставить в словарь, при этом вместо выделенного фрагмента создаётся ссылка на новый элемент словаря. Операцию удобно использовать вместе с операцией поиска элемента словаря.

Вставить фрагмент текста в каталог.

Эта операция позволяет выделенный фрагмент текста заменить ссылкой на элемент каталога. Эту операцию удобно использовать вместе с операцией поиска элемента каталога.

Поиск элемента словаря.

Позволяет произвести поиск в тексте документации фрагментов, идентичных выбранному элементу словаря, и заменять их ссылкой на этот элемент. В разработанном ИС есть возможность выбора контекста поиска и выбор, производить ли поиск слова целиком (т.е. обязан ли найденный фрагмент в тексте быть ограничен с обеих сторон знаками препинания, пробелом или символом перевода строки) или нет.

Поиск элемента каталога.

Позволяет произвести поиск возможных ссылок на элемент каталога и произвести замену на эти ссылки. В разработанном ИС есть возможность поиска с использованием всех шаблонов, созданных для выбранного каталога и/или всех записей этого каталога.

3.4.5.4.4. Операции настройки повторного использования

Выделить в точку расширения.

Позволяет выделенный фрагмент текста заменить точкой расширения. Вместо выделенного фрагмента создается точка расширения. Далее есть две модификации этой операции. Одна просто помещает выделенный фрагмент в точку расширения. Другая позволяет выделенный фрагмент перенести в описание документации конкретного продукта.

Выделить в Insert-After/Before.

Позволяет выбранный фрагмент текста из общего актива перенести в документацию продуктов. Действия, проводимые для этой операции, отличаются от

действий, проводимых для второй модификации рефакторинга, лишь тем, что в адаптере создается не Replace-Nest, а Insert-After/Before. Эту операцию можно использовать только в том случае, если фрагмент для переноса находится непосредственно после/до точки расширения.

3.5. Платформа Eclipse

Eclipse представляет собой расширяемую платформу для разработки модульных приложений, основанную на Java, а также широкий набор взаимодействующих между собой инструментов, построенных на данной платформе. Наиболее распространенными приложениями, основанными на данной платформе, являются интегрированные среды разработки (IDE), и многие пользователи применяют Eclipse в качестве Java IDE. Кроме того, Eclipse включает в себя среду разработки подключаемых модулей (плагинов), которая используется для создания собственных встроенных инструментов. Создав собственное приложение, встроенное в Eclipse, разработчик может предоставить пользователю инструмент с мощной и цельной средой разработки. Примером такого инструмента может служить программное средство DocLine, реализованное в виде набора подключаемых модулей.

3.5.1. Eclipse RCP

Поскольку платформа Eclipse создана для того, чтобы служить открытой платформой разработки, ее архитектура позволяет создать почти любое приложение из ее компонентов. Минимальный набор вспомогательных программ, необходимых для создания расширенных клиентских приложений известен под общим названием Rich Client Platform (RCP) [10].

Eclipse RCP состоит из следующих компонентов.

- Eclipse Runtime с OSGi (Open Services Gateway Initiative) — спецификация динамической плагинной (модульной) шины для создания Java-приложений [19].
- SWT (Standard Widget Toolkit) — библиотека с открытым исходным кодом для разработки графических интерфейсов пользователя на языке Java [17].
- JFace – набор Java-классов, реализующий наиболее общие задачи построения GUI [18].
- Workbench UI – основное окно интегрированной среды разработки Eclipse

Выводы

Документацию СПП можно разрабатывать, используя разные технологии, самыми известными из которых являются DocBook и DITA. Однако, ни одна из них не создавалась для разработки документации СПП специально, поэтому в них отсутствуют механизмы тонкой настройки повторно используемых активов. Технология DocLine предоставляет эти механизмы посредством языка разметки документации DRL.

При работе с активами, используемыми в документации разных продуктов, изменения, вносимые техническими писателями «вручную», распространяются на все эти продукты, вследствие чего возрастает вероятность появления ошибок. Это является главным аргументом в пользу автоматизированного средства рефакторинга документации, предоставляемого технологией DocLine. В работе [3] предложен ряд операций рефакторинга, однако есть возможность сделать процесс настройки и извлечения переиспользуемых активов более продуктивным.

Описанная технология реализована в программном средстве DocLine в виде плагинов к Eclipse IDE, и использование этой технологии невозможно без установки самой IDE, в то время как с помощью Eclipse RCP можно сделать автономный вариант поставки DocLine.

4. Реализация

4.1. Проектирование новых операций рефакторинга

В рамках данной дипломной работы были реализованы следующие операции рефакторинга:

- извлечение документа или фрагмента документа в информационный продукт;
- преобразование фрагмента документации в условный блок;
- переименование конструкций языка DRL;
- разделение одного информационного элемента на два информационных элемента;
- объявление ссылки необязательной;

4.1.1. Извлечение фрагмента документации в информационный продукт

Эта операция предназначена для переноса фрагмента документации в отдельный информационный продукт. Операцию используют при импорте документации из формата DocBook в формат DRL, если есть намерение извлечь из нее несколько новых информационных продуктов.

Данная операция работает по следующему сценарию.

- Создается информационный элемент, куда помещается предназначенный для переноса в информационный продукт фрагмент документации.
- Вместо выделенного фрагмента создается ссылка на новый информационный элемент.
- Новый информационный элемент помещается в конец документа, содержащего эту ссылку.
- Создается информационный продукт, куда помещается ссылка на созданный информационный элемент.
- Создается финальный информационный продукт, соответствующий новому информационному продукту.

Рассмотрим пример.

Пусть после импорта документации из формата DocBook в формат DRL получились следующие файлы.

Общие активы:

```
<d:InfProduct id = "prod">
<d:InfElemRef id = "book_ref" infelemid = "book_elem"/>
</d:InfProduct>
<d:InfElement id = "book_elem"
 <book id=" book">
   <title>Very simple book</title>
```

```
   <chapter id="chapter_1">
     <title>Chapter 1</title>
     <para>Hello world!</para>
   </chapter>
   <chapter id="chapter_2">
     <title>Chapter 2</title>
     <para>Hello again, world!</para>
   </chapter>
 </book>
</d:InfElement>
```

Специализированный документ:

```
<d:FinalInfProduct id = "final_prod" infproductid = "prod">
    <d:Adapter infelemrefid = "book_ref"></d:Adapter>
</d:FinalInfProduct>
```

После выделения в информационный продукт фрагмента **chapter_2** документ приобретает следующий вид:

```
<d:InfElement id = "prod_elem">
 <book id = "book">
   <title>Very simple book</title>
   <chapter id = "chapter_1">
     <title>Chapter 1</title>
     <para>Hello world!</para>
   </chapter>
   <d:InfElemRef id = "chapter2_ref" infelemid = "chapter2"/>
 </book>
</d:InfElement>
<d:InfElement id = "chapter2">
<chapter id = "chapter_2">
     <title>Chapter 2</title>
     <para>Hello again, world!</para>
   </chapter>
</d:InfElement>
```

Вместо выделенного фрагмента появилась ссылка на новый информационный элемент, который находится в конце данного документа.

Созданные информационный и финальный продукты выглядят следующим образом:

```
<d:InfProduct id = "chapter2_prod">
   <d:InfElemRef id = "chapter2_elem_ref" infelemid = "chap
ter2"/>
</d:InfProduct>
```

```
<d:FinalInfProduct id = "chapter2_finalProd" infproductid = "chap-
ter2_prod">
    <d:Adapter infelemrefid = " chapter2_elem_ref "/>
</d:FinalInfProduct>
```

4.1.2. Преобразование фрагмента документации в условный блок

Эта операция предназначена для переноса фрагмента документации в условный блок с логическим условием, заданным техническим писателем.

Условие формулируется в виде задания имени (varname) и необходимого значения переменной (varvalue):

condition = "varname = varvalue"

Для выполнения этой операции производятся следующие действия.

- Выделенный фрагмент документации помещается в условный блок. Технический писатель задает условие с помощью специального диалога.
- Во всех финальных информационных продуктах, содержащих указанный фрагмент, условие устанавливается истинным. Это нужно, чтобы гарантировать неизменность существующих конечных документов.

Определение рефакторинга документации подразумевает неизменность представления конечных документов. Тем не менее, может случиться так, что техническому писателю понадобится создать несколько условных блоков с разными значениями одной и той же переменной. В таком случае, если придерживаться определения, технический писатель сможет с помощью описываемой в этом разделе операции рефакторинга создать только один блок, а для создания следующих блоков операцию нельзя будет применить из-за наличия переменной во всех финальных информационных продуктах. Для решения этой проблемы в данной реализации операции «Преобразование фрагмента документации в условный блок» техническому писателю предоставляется возможность указать значения переменной, которую он хочет видеть в каждом финальном информационном продукте. Таким образом, конечные документы могут измениться, но, во-первых, сам писатель об этом оповещается, во-вторых, он включает выделяемый блок в те документы, где это необходимо.

Ниже приведен пример конфликта.

Пусть есть несколько информационных продуктов и два фрагмента текста – *frag1, frag2,* – которые мы хотим выделить в условные блоки с условиями *OS=Linux* и *OS=MacOS*.

После выделения в условный блок *frag1* значение переменной *OS* установится равным *Linux* в обоих финальных информационных продуктах. Тогда операцию

выделения в условный блок *frag2* проводить будет нельзя, так как, установив значение а как *Linux* или *MacOS*, из конечных документаций исключится один из фрагментов. При такой реализации операции, как описано выше, во время выделения *frag2* технический писатель выберет сам, в каких финальных информационных продуктах *OS=Linux*, а в каких *OS=MacOS.*

4.1.3. Переименование конструкций языка DRL

Эта операция позволяет переименовывать следующие конструкции языка DRL:

- информационный продукт;
- информационный элемент;
- точку расширения;
- ссылку на информационный элемент;
- словарь;
- каталог;
- шаблон представления элементов каталога.

Большинство конструкций языка имеют атрибуты name и id, однако ссылка на конструкцию всегда использует её id, а не имя, поэтому переименование является сменой именно атрибута name.

После переименования какой-либо конструкции из перечисленных выше все ссылки в документации на переименованные конструкции автоматически обновляются.

4.1.4. Разделение информационного элемента

Эта операция позволяет разделить информационный элемент на два. Операция работает следующим образом.

- Проверяется, не существует ли ссылка на выбранный для разделения информационный элемент, состоящая в группе ссылок, позволяющей использовать только одну ссылку при адаптации (группа с modifier = XOR, описанная в 3.4.2). Если такая ссылка есть, рефакторинг проводить нельзя.
- Пользователь разделяет содержимое информационного элемента на две части, и каждая помещается в отдельный информационный элемент.
- Для каждого нового информационного элемента создается ссылка там, где была ссылка на исходный информационный элемент. Если исходная ссылка была необязательной, то вновь созданные также будут необязательными.
- Вместо адаптеров для ссылок на исходный информационный элемент создаются адаптеры для новых ссылок, и между ними делится содержимое старого адаптера. В случае, если все точки расширения исходного

информационного элемента попали в один новый адаптер, а второй остался пустым, то все равно создаются оба адаптера, так как пустой адаптер может включать необязательную ссылку.

Рассмотрим пример.

Пусть есть информационный элемент, выбранный для разделения, информационный продукт и финальный информационный продукт:

```
<InfElement id = "phone">
    <Nest id = "connection">
        Существуют следующие способы подключить телефон:
    </Nest>
    <Nest id = "number_enter">
        Существуют следующие способы набора номера:
    </Nest>
</InfElement>
<InfProduct id = "example_id">
    <InfElemRef id = "phoneref" infelemid = "phone">
</InfProduct>
<FinalInfProduct infproductid = "example_id">
    <Adapter infelemrefid = "phoneref">
  <Insert-After nestid = "connection">
            Подключить телефон к городской телефонной сети
            Подключить телефон к оператору спутниковой связи
        </Insert-After>
        <Insert-After nestid = "number_enter">
            Набрать с помощью встроенной в телефон клавиатуры
        </Insert-After>
    </Adapter>
</FinalInfProduct>
```

После разделния информационного элемента на два, в каждый из которых попадёт одна точка расширения, видно, что появились два новых информационных элемента, старый при этом исчез. Также вместо ссылки на старый информационный элемент появились ссылки на новые информационные элементы. Вместо старого адаптера появились два новых, между ними разделилось содержимое старого.

```
<InfElement id = "num_enter_elem">
    <Nest id = "number_enter">
        Существуют следующие способы набора номера:
    </Nest>
</InfElement>
<InfElement id = "conn_elem">
    <Nest id = "connection">
```

```
            Существуют следующие способы подключить телефон:
        </Nest>
    </InfElement>
    <InfProduct id = "example_id">
        <InfElemRef id="connectionref" infelemid = "conn_elem">
        <InfElemRef id="numberref" infelemid="num_enter_elem">
    </InfProduct>
    <FinalInfProduct infproductid = "example_id">
        <Adapter infelemrefid = "connectionref">
            <Insert-After nestid = "connection">
                Подключить телефон к городской телефонной сети
                Подключить телефон к оператору спутниковой связи
            </Insert-After>
        </Adapter>
        <Adapter infelemrefid = "numberref">
            <Insert-After nestid = "number_enter">
                Набрать с помощью встроенной в телефон клавиатуры
            </Insert-After>
        </Adapter>
    </FinalInfProduct>
```

4.1.5. Объявление ссылки на информационный элемент необязательной

Эта операция позволяет сделать ссылку на информационный элемент из обязательной необязательной.

Для выполнения этой операции во всех финальных информационных продуктах происходит создание адаптера для этой ссылки, если она в них является используемой и, если такой адаптер не был создан ранее.

Рассмотрим пример применения этой операции.

Пусть есть информационный продукт, два информационных элемента (один ссылается на другой) и два финальных информационных продукта. В одном из них используется при адаптации ссылка, которую будет сделана необязательной, а в другом нет.

```
<InfElement id = "num_enter_elem">
    Существуют следующие способы набора номера:
    - Набрать с помощью встроенной в телефон клавиатуры
</InfElement>
<InfElement id = "phone">
    Существуют следующие способы подключить телефон:
            - Подключить телефон к городской телефонной сети
            - Подключить телефон к оператору спутниковой связи
```

```
    <InfElemRef id="numberref" infelemid="num_enter_elem">
</InfElement>
<InfProduct id = "sample_id">
   <InfElemRef id="phoneref" infelemid="phone" optional="true">
</InfProduct>

<FinalInfProduct id="использующий" infproductid="sample_id">
    <Adapter infelemrefid = "phoneref">
    </Adapter>
</FinalInfProduct>

<FinalInfProduct id="не использующий" infproductid="sample_id">
</FinalInfProduct>
```

После того, как ссылка <InfElemRef id="numberref"> станет необязательной, в финальном информационном продукте, использующем ссылку, появился адаптер, а в том, что не использует ссылку, не появился.

```
<InfElement id = "num_enter_elem">
    Существуют следующие способы набора номера:
    - Набрать с помощью встроенной в телефон клавиатуры
</InfElement>
<InfElement id = "phone">
    Существуют следующие способы подключить телефон:
            - Подключить телефон к городской телефонной сети
            - Подключить телефон к оператору спутниковой связи
  <InfElemRef   id="numberref"   infelemid="num_enter_elem"   optional   =
    "true">
</InfElement>
<InfProduct id = "sample_id">
   <InfElemRef id="phoneref" infelemid="phone" optional="true">
</InfProduct>
<FinalInfProduct id="использующий" infproductid="sample_id">
    <Adapter infelemrefid = "phoneref">
    </Adapter>
    <Adapter infelemrefid = " numberref">
    </Adapter>
</FinalInfProduct>
<FinalInfProduct id="не использующий" infproductid="sample_id">
</FinalInfProduct>
```

4.1.6. Объявление ссылки на информационный элемент обязательной

Данная операция позволяет сделать ссылку из необязательной обязательной. Для того, чтобы выполнить эту операцию, значение атрибута optional у ссылки должно быть равным true. После объявления её обязательной, этот атрибут удаляется, так как по умолчанию его значение равно false.

Для того, чтобы операция стала доступной для некоторой необязательной ссылки, она должна быть включена в каждом финальном информационном продукте, который может её использовать. Это необходимое условие, так как после проведения операции элемент, которому соответствует ссылка, включается в каждый из этих продуктов.

После проведения операции те адаптеры, которые включают ссылку, можно удалить, если они не выполняют других функций.

4.2. Особенности реализации новых операций рефакторинга

Для реализации каждой из операций рефакторинга, описанных в 4.2, было создано три Java-класса: RefactoringTechniqueAction, RefactoringTechnique и RefactoringTechniqueDialog (они названы так, потому что для каждой операции рефакторинга эти классы выглядят похоже, следовательно, нет необходимости описывать их всех).

Такой способ реализации операций рефакторинга соответствует архитектуре средства рефакторинга, созданного в работе [3].

Далее приводится описание созданных классов и их взаимодействия (рисунок 5).

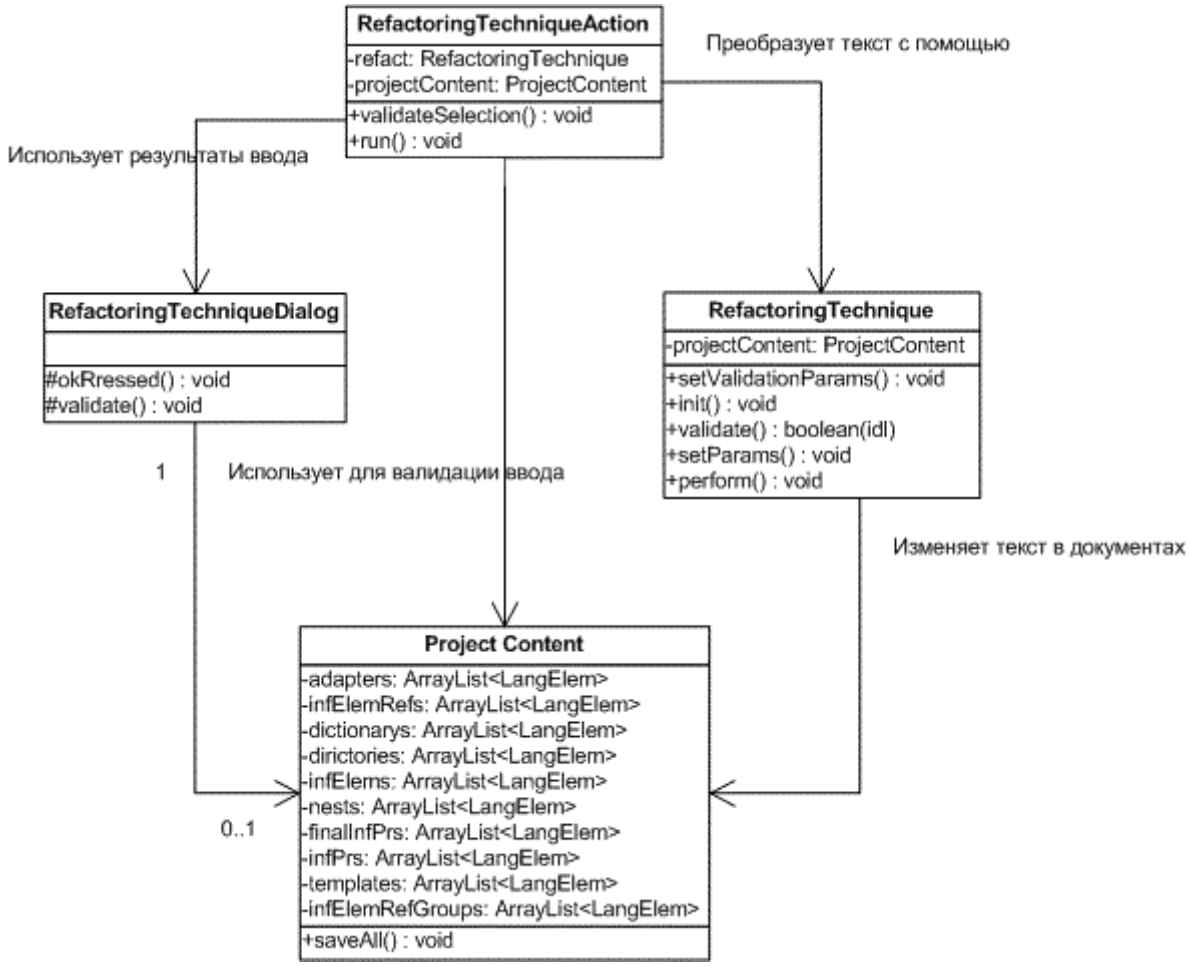

Рисунок 5: Созданные классы и взаимодействие между ними

Класс RefactoringTechnique

Этот класс описывает методы проверки того, допустимо ли выполнение данной операции рефакторинга над выделенным фрагментом текста, а также методы, преобразующие DRL-документы.

Метод *setValidationParams* задает параметры, необходимые для валидации.

Метод *init* инициализирует переменные, необходимые для валидации и преобразования текста.

Метод *validate* определяет, можно ли проводить операцию рефакторинга над выделенным фрагментом текста.

Метод *setParams* задает параметры, необходимые для преобразования текста DRL-документов.

Метод *perform* выполняет преобразование DRL-документов: создает новые элементы языка DRL, удаляет старые, преобразует имеющиеся.

Класс RefactoringTechniqueDialog

Экземпляр этого класса является диалогом, с помощью которого происходит ввод данных для проведения операций рефакторинга.

Метод *okPressed* подготавливает параметры для валидации из введенных пользователем данных.

Метод *validate* проверяет данные, введенные пользователем. Если они не верны, выдает ошибку, описывающую, почему они неверны. Если данные, введенные пользователем, верны, диалог закрывается, после чего проводится операция рефакторинга.

Класс RefactoringTechniqueAction

Такой класс, являющийся наследником класса org.eclipse.jface.action.Action, соответствует каждой операции рефакторинга. Описываемый класс обеспечивает подготовку данных для проведения операции рефакторинга и вызов методов, осуществляющих операцию. Экземпляр такого класса создается при определении, доступна ли операция рефакторинга для выделенного фрагмента текста, а также при выполнении операции рефакторинга.

Поле *refact* – это объект, реализующий функциональность операции рефакторинга. Для передачи параметров полю refact используется метод *setValidationParams.*

Метод *validateSelection* предназначен для подготовки данных для валидации. Для валидации используется метод *validate* поля *refact.* Если валидация закончилась успешно, то соответствующий этой операции пункт в контекстном меню объявляется доступным.

Метод *run* создает диалоговое окно класса RefactoringTechniqueDialog, и, если данные, введенные пользователем, корректны, передает их полю *refact* с помощью метода *setParams*, затем исполняет операцию рефакторинга с помощью метода *perform*, после чего сохраняет сделанные изменения в проекте.

Класс ProjectContent

Этот класс содержит деревья синтаксического разбора файлов на языке DRL, а также списки конструкций языка DRL каждого типа (информационный элемент, информационный продукт и т.д.). Узлы синтаксических деревьев содержат ссылки на элементы этих списков. Меняя эти списки, операции рефакторинга меняют структуру DRL-документа.

Метод *SaveAll* сохраняет изменения в проекте после проведения операции рефакторинга.

4.3. RCP-версия программного средства DocLine

Платформа Eclipse RCP в данной работе использована для создания приложения DocLineRCP, распространяемого отдельно от Eclipse IDE и обладающего всеми функциями плагинов DocLine. Помимо этих функций, в DocLineRCP добавлены операции, позволяющие работать с технической документацией так же, как это делается в Eclipse IDE. Ниже описаны основные функции, включенные в приложение.

Функции по работе с файлами и текстом

RCP-приложение DocLineRCP предоставляет стандартные операции Eclipse для работы с файлами (создать, открыть, сохранить, закрыть, импорт/экспорт и др.) и текстом (отменить/вернуть, вырезать, копировать, вставить, удалить и др.). Также предусмотрена функция поиска текста по файлу. Эти функции предназначены для работы с DRL-файлами и файлами DocBook-документации в формате XML и редактирования текста, содержащегося в этих файлах.

Функции по работе с файлами реализованы в плагине org.eclipse.ui (папка org.eclipse.ui.file), функции по работе с текстом реализованы в плагине org.eclipse.ui (папка org.eclipse.edit).

Работа с проектами

Для отображения структуры проектов в DocLineRCP включен навигатор, отображающий все проекты в виде иерархической структуры.

Также предусмотрена возможность работы с SVN-репозиториями. Для этого использован проект Subversive. Он состоит из двух частей: плагин Subversive и Subversive SVN Connectors (внешние библиотеки). Основные функции плагина Subversive состоят в следующем:

- просмотр удаленного репозитория;
- возможность загрузить проект в репозиторий или скачать его из репозитория;
- синхронизация проекта для просмотра входящих и исходящих изменений;
- возможность фиксировать и откатывать изменения;
- просмотр истории изменений в проекте.

Отображение ошибок

Для отображения ошибок или предупреждений используется вид Problems, являющийся стандартным видом Eclipse.

Функции плагинов DocLine

Приложение DocLineRCP обладает всеми функциями плагинов DocLine к Eclipse. Пользователю доступен текстовый и графический редакторы DRL. Через текстовый редактор доступен рефакторинг DRL-документов, включая операции, реализованные в рамках данной работы.

4.4. Исправленные недостатки реализации существующих операций рефакторинга

4.4.1. Ошибочное использование XML-директив в результирующем тексте

При импорте документа из формата DocBook в формат DRL в результирующие файлы помещается не только содержательная часть документации, но и специальные теги, содержащие метаданные файла. Как правило, это тег объявления XML (<?xml>). Так происходит, потому что теги никак не фильтруются. Тег объявления может встречаться только в начале XML-документа, поэтому при помещении этого тега в информационный элемент документ становится некорректным с точки зрения формата XML, и при экспорте в PDF или HTML финального информационного продукта, содержащего ссылку на некорректный информационный элемент, возникает ошибка.

4.4.2. Неудобное форматирование конструкций DRL при их создании

При создании новых конструкций языка DRL (например, InfElem) должное внимание не уделяется форматированию текста: все создаваемые конструкции помещаются в одну строку. При этом не видно, сколько элементов создалось, что это за элементы и каково их содержание.

4.4.3. Неудобные диалоговые окна

Для осуществления операций рефакторинга от пользователей требуется ввод данных (например, имя, id информационного продукта). Диалоговые окна, посредством которых выполняется этот ввод, имеют ряд недостатков.

- Размеры диалоговых окон не всегда позволяют получить доступ к текстовым полям.
- Размеры диалоговых окон и кнопок не всегда позволяют прочитать содержащийся на них текст.
- Излишне подробное описание операций рефакторинга в диалоговых окнах.

5. Результаты работы

В рамках этой работы были достигнуты следующие результаты.

1. Спроектировано пять новых операций рефакторинга:
    - извлечение фрагмента документации в информационный продукт;
    - преобразование в условный блок;
    - разделение информационного элемента;
    - объявление ссылки на информационный элемент необязательной;
    - переименование.
2. Спроектированные операции реализованы и встроены в технологию DocLine.
3. Внесены следующие исправления в реализацию операций рефакторинга, созданных в ходе работы [3].
    - Были переработаны диалоговые окна с целью повышения удобства использования операций «Первичный переход к документации с повторным использованием», «Выделить информационный элемент», «Вставка в словарь» и «Создание нового словаря».
    - Исправлена ошибка, из-за которой в некоторых случаях порождались синтаксически некорректные конструкции.
    - Улучшено автоматическое форматирование конструкций создаваемых в процессе рефакторинга.
4. Создана версия программного средства DocLine, распространяемая отдельно от Eclipse IDE.

6. Список литературы

[1] *Кознов Д.В., Романовский К.Ю*. Автоматизированный рефакторинг документации семейств программных продуктов. // Системное программирование. Вып. 4. / Под ред. А.Н.Терехова и Д.Ю.Булычева. СПб: Изд. СПБГУ, 2009. С.128-135.

[2] *Кознов Д.В., Романовский К.Ю.* DocLine: метод разработки документации семейств программных продуктов. // Программирование, 2008, № 4. С.1-13.

[3] *Минчин Л.А.* Разработка методов рефакторинга документации семейств программных продуктов. Дипломная работа, СПБГУ, 2008. С.9-19.

[4] *Романовский К.Ю.* Метод повторного использования документации семейств программных продуктов. Кандидатская диссертация, СПБГУ, 2010. 110 с.

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

[6] *Clements, P*. Being Proactive Pays Off. IEEE Software July/August 2002. P. 28-31.

[7] *Krueger C*. Eliminating the Adoption Barrier. IEEE Software July/August 2002. P. 29-31.

[8] Введение в архитектуру типизированной информации Darwin (DITA). // http://xmlhack.ru/texts/06/dita/dita.html

[9] Принцип единого источника в технологии DocBook. // http://glossterm.org/docbook-basics.zhtml#docbook_singleSource

[10] Разработка Вашего первого RCP-приложения. // http://www.ibm.com/developerworks/ru/edu/os-ecl-rcpapp/index.html

[11] Рефакторинг кода. //http://ru.wikipedia.org/wiki/Рефакторинг

[12] DocBook project official site . // http://docbook.org

[13] *Don Day, Michael Priestley, David Schell*. Introduction to the Darwin Information Typing Architecture. // http://www-106.ibm.com/developerworks/xml/library/x-dita1/

[14] Eclipse project official site. // http://www.eclipse.org

[15] *Northrop L.M*. Software Product Line Essentials. Software Engineering Institute,2008. // http://www.sei.cmu.edu/library/assets/spl-essentials.pdf

[16] *Norman Walsh, Leonard Muellner*. DocBook: The Definitive Guide. O'Reilly, 1999. // http://oreilly.com/catalog/docbook/book2/docbook.html

[17] SWT library specification. // http://eclipse.org/swt

[18] JFace library specification. // http://wiki.eclipse.org/JFace

[19] OSGi Alliance specification. // http://osgi.org