Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Разработка методов рефакторинга документации семейств программных продуктов

Дипломная работа студента 544 группы
Минчина Леонида Аркадьевича

Научный руководитель ст. преподаватель	……	К.Ю.Романовский
Рецензент к. ф.-м.н, доцент	……	Д.В. Кознов
“Допустить к защите” заведующий кафедрой, д. ф.-м.н., профессор	……	А. Н. Терехов

Санкт-Петербург
2008

ОГЛАВЛЕНИЕ

Введение

Техническая документация – важная составляющая современного промышленного программного обеспечения (ПО). Как и ПО, документация изменчива и структурно сложна. Кроме того, документация часто существует на разных уровнях детализации (подробное руководство пользователя, руководство по быстрому старту, справочная система) и в различных форматах: электронная (HTML), печатная (PDF), и др.

Разработка семейств программных продуктов (СПП) – это популярный подход к промышленной разработке ПО, характеризующийся тем, что создаваемые продукты обладают схожей функциональностью, направлены на один сегмент рынка и разрабатываются на основе общих активов с использованием заранее определенного метода [1]. В качестве повторно используемых активов могут выступать программные компоненты, архитектура, процесс разработки, документация, и т.п. Документация СПП имеет широкие возможности для повторного использования, как на уровне одного продукта (различные уровни детализации), так и на уровне всего семейства (документы схожего назначения для различных продуктов).

Типы процессов разработки программной части СПП можно разделить на тяжеловесные и легковесные. В случае тяжеловесного процесса сначала, в основном, происходит разработка общих активов, подготовленных для использования в нескольких продуктах. А потом с их использованием происходит разработка самих продуктов. В случае легковесного процесса сначала происходит разработка 1-го продукта. При этом возможно, что будут созданы какие-нибудь общие активы, но особого акцента на их выявление не делается. Потом, когда возникнет необходимость в разработке новых продуктов, общие активы могут быть выделены или модифицированы.

Процесс разработки документации в рамках СПП схож с описанными выше процессами разработки программной части. При разработке документации может также понадобиться выделять общие фрагменты текста (как новые, так и из существующих документов) и настраивать их использование. В этих случаях эффективно применение методов рефакторинга, что по отношению к документации означает изменение структуры документации с сохранением конечных документов.

Из экономических соображений легковесный процесс чаще применим на практике. Поэтому в данной работе предлагаются методы рефакторинга документации СПП, направленные в первую очередь на поддержку данного процесса.

Моя работа является частью исследовательского проекта DocLine [2], выполняющегося на кафедре системного программирования математико-механического факультета СПбГУ. DocLine – это метод разработки документации СПП на основе повторного использования. Данный подход включает в себя процесс разработки документации, язык разметки документации, имеющий текстовую и графическую нотации, а также пакет инструментальных средств.

В методе DocLine документация представляется в виде шаблонов документов и правил адаптации этих шаблонов для получения конечных документов. Шаблоны, в свою очередь, строятся на основе повторно используемых фрагментов текста. Адаптация шаблона возможна в точках вариативности. В методе DocLine это точки включения повторно используемых фрагментов и точки расширения – места в документе, в которых может добавляться или удаляться текст при адаптации.

В моей работе предлагаются операции рефакторинга, поддерживающие извлечение повторно используемых фрагментов текста и их настройку.

Постановка задачи

Целью данной работы является создание средства рефакторинга документации семейств программных продуктов, разрабатываемой в технологии DocLine.

Для достижения этой цели были поставлены следующие задачи.

- Изучить процесс разработки документации и выявить потребности в средствах рефакторинга.
- Разработать операции рефакторинга, позволяющие выделять и настраивать общие активы.
- Разработать архитектуру средства поддержки рефакторинга.
- Реализовать операции рефакторинга и встроить их в технологию DocLine.
- Выполнить апробацию средства рефакторинга.

1. Обзор литературы

1.1 Семейства программных продуктов

Разработка семейств программных продуктов позволяет эффективно реализовывать повторное использование различных активов. Среди преимуществ данного метода можно отметить повышение производительности труда, повышение качества продуктов, уменьшение времени на разработку конкретного продукта.

Существует два типа процессов разработки продуктов в рамках СПП: тяжеловесный и легковесный. Тяжеловесный («сверху вниз») предполагает, что сначала, в основном, происходит разработка общих активов, а потом с их использованием разработка нескольких продуктов семейства. В случае легковесного процесса сначала происходит разработка одного продукта семейства, а потом, когда возникнет необходимость или возможность, на его основе разрабатываются последующие продукты семейства.

Легковесный процесс используется чаще, чем тяжеловесный из экономических соображений. Во-первых, требуются меньшие начальные капиталовложения, чем в случае тяжеловесного процесса, т.к. сначала нужно разработать только один продукт. Во-вторых, можно раньше начать получать прибыль, т.е. можно быстро разработать один продукт и начать его продавать. И, в-третьих, меньше риски – если первый продукт не будет продаваться, то, возможно, в этом случае не стоит разрабатывать последующие.

1.2 Рефакторинг СПП

Рефакторинг – это процесс изменения программного кода с сохранением исполнимой семантики, направленный на улучшение архитектуры системы, либо на повышение читаемости [12]. Рефакторинг широко используется в гибких методах разработки ПО и позволяет реагировать на изменение требований к программному продукту без необходимости переписывать весь код.

Также рефакторинг применим в разработке СПП [10, 11], в частности для извлечения общих активов и настройки их использования. Аналогично рефакторинг применим к документации СПП. В случае документации рефакторинг означает изменение структуры документации с сохранением конечных документов.

1.3 Технология DocLine

Метод разработки документации семейств программных продуктов DocLine предложен на кафедре системного программирования математико-механического факультета СПбГУ.

Метод DocLine [2, 5] состоит из следующих частей:

- процесс разработки документации;
- язык разработки документации DRL (Document Reuse Language) [5], включающий текстовую и графическую нотации;
- пакет инструментальных средств.

Технология DocLine опирается на известную технологию DocBook [6].

Технология DocBook широко распространена и является стандартом де-факто для разработки документации на свободно распространяемое программное обеспечение.

Эта технология позволяет осуществлять публикацию документов в различных форматах (PDF, HTML и др.) на основе единого исходного представления. Достигается это путём разделения содержимого документа и его форматирования

1.3.1 Процесс разработки документации

Процесс разработки документации, предлагаемый в DocLine, как и процесс разработки СПП в целом, состоит из двух частей – разработки повторно используемых

компонент и разработки документации продуктов. При необходимости допускается вносить изменения в повторно используемые компоненты при разработке документации продуктов. Это может потребоваться для тонкой настройки повторного использования документации.

1.3.2 Язык DRL

Язык DRL [5] состоит из текстовой и графической нотаций. Графическая нотация предназначена для крупноблочного проектирования документации, текстовая – для написания непосредственно текста документации. При разработке документации процесс допускает одновременную работу с этими двумя представлениями – технический писатель может переходить от написания текста к редактированию диаграмм и обратно без нарушения структуры текста.

1.3.3 Графическая нотация DRL

Графическая нотация языка DRL (DRL/GR – DRL/Graphic Representation) предназначена для визуального моделирования структуры документации. В неё входит три типа диаграмм: главная, вариативности и продукта.

Для облегчения описания диаграмм нужно дать объяснение некоторым понятиям. Информационный продукт (ИП) – это шаблон документа. Обычно есть один шаблон на каждый тип документов. К примеру, в проекте может быть создан один ИП для справочной системы, один ИП для руководства пользователя и т.д. При создании документов для конкретного программного продукта (ПП) эти ИП могут быть адаптированы под контекст использования – т.е. специализированы. Информационный элемент (ИЭ) – это фрагмент документации, который может включаться в различные контексты нескольких документов (путем организации ссылки на него).

Главная диаграмма предназначена для описания того, какие ИП входят в какие программные продукты семейства. ИП будут специализироваться для использования с программным продуктом. Диаграмма вариативности предназначена для описания ИП. На ней отображаются связи между ИП и ИЭ. Диаграмма продукта предназначена для описания документа, специализированного для конкретного продукта. На рис. 1 приведены примеры этих диаграмм, представленные в работе [5].

Рисунок 1. DRL/GR - примеры диаграмм, приведённые в работе [5]

1.3.4 Текстовая нотация DRL

Текстовая нотация DRL (DRL/PR – DRL/Phrase Representation) предназначена для непосредственного написания текстов документации. DRL/PR является XML[1] языком, включающим в себя конструкции языка DocBook[6], что позволяет проводить публикацию документа в различных форматах.

DRL/PR позволяет описывать всё, что можно описать с помощью графической нотации. Кроме конструкций, поддерживаемых графической нотацией, текстовая нотация содержит конструкции для тонкой настройки ИП под контекст использования и конструкции для организации т.н. мелкозернистого повторного использования – повторного использования небольших фрагментов текста.

Файлы документации в формате DRL делятся на три вида, каждому из которых соответствует свой корневой элемент. Для дальнейшего описания рассмотрим три листинга, все они взяты из работы [5] и немного модифицированы.

Файл с корневым элементом ProductLine содержит информацию о том, какие программные продукты входят в семейство. В листинге 1 приведён пример такого файла.

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

[1] XML (*eXtensible Markup Language)* — расширяемый язык разметки

```
</ProductLine>
```

Листинг 1. Схема семейства продуктов

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

Описание самой документации можно разделить на две компоненты: DocumentationCore, предназначенной для описания повторно используемых компонент, и ProductDocumentation, предназначенной для описания конечной документации, адаптированной для конкретного ПП.

В листинге 2 приведён пример файла, содержащий описание компонент повторного использования.

```
<DocumentationCore>
  <InfProduct id="userguide" name="Руководство пользователя">
     <InfElemRef id="numberIdRef" infelemid="numberId"/>
  </InfProduct>
  <InfElement id="numberid" name="АОН">
     ……
     <Nest id="number_indication_methods">…</Nest>
  </InfElement>
  <Dictionary id="maindict" name="Основной словарь"/>
  <Directory id="maindir" name="Основной каталог"/>
</DocumentationCore>
```

Листинг 2. Документация семейства

Мы видим описание ИП – InfProduct. Он содержит ссылку на ИЭ (InfElement), определенный позже. Также внутри ИЭ мы видим элемент Nest (точка расширения), предназначенный для описания точек документации, в которых при специализации могут произойти изменения. Элементы Dictionary и Directory предназначены для описания компонент мелкозернистого повторного использования. Dictionary (словарь) содержит набор пар «ключ - значение». С помощью этого ключа можно в тексте документации ссылаться на элемент словаря. Directory (каталог) предназначен для описания наборов «ключ - несколько значений». Таким образом хранится информация о некоторых понятиях. Для использования каталога необходимо создать описания различных способов представления понятий из каталога. В документации можно ссылаться на элемент каталога с указанием того, какое представление использовать.

В листинге 3 представлен пример описания документации конкретного продукта.

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

Листинг 3. Документация продукта

Видно, что документ, приведённый в листинге 3, имеет привязку к конкретному ПП (атрибут productid тега ProductDocumentation). Далее идёт описание различных специализированных документов, в DRL они носят название финальных информационных продуктов (ФИП), которые должны входить в состав данного ПП.

Каждый ФИП (тег FinalInfProduct) имеет привязку к какому-либо ИП (атрибут infproductid). ФИП описывает, как будет адаптироваться ИП. К примеру, в листинге 3 для этого используются конструкция Insert-After (которая позволяет производить вставку фрагмента текста после указанной точки расширения) и определение словаря с таким же именем, как и в документации семейства (это позволяет переопределить значения некоторых элементов словаря). Существуют также следующие возможности адаптации: включение или нет всего ИЭ (для каждой ссылки на ИЭ в отдельности), переопределение каталога, условное включение фрагмента текста.

1.3.5 Программные средства

В рамках дипломных работ 2007 года [7, 8] в виде встраиваемых компонент для платформы Eclipse [9] были реализованы редакторы графической и текстовой нотаций DRL. Редакторы поддерживают весь процесс разработки документации от проектирования до публикации. Редактор графической нотации позволяет создавать главную диаграмму и диаграмму вариативности. Текстовый редактор позволяет создавать тексты на языке DRL/PR, проводить их валидацию и экспорт в документы форматов pdf и html. Редакторы интегрированы друг с другом, изменения в диаграммах отражаются в тексте документации после сохранения и, наоборот, изменения в тексте отражаются в диаграммах после сохранения.

1.4 Платформа Eclipse

Eclipse [9] – свободно распространяемая платформа для разработки модульных кросс-платформенных приложений. Eclipse представляет собой основанную на Java расширяемую платформу разработки с открытым исходным кодом. По сути это набор сервисов для создания встраиваемых компонент (плагинов). Среди распространённых встраиваемых компонент можно отметить Java Development Tools (JDT) – инструментарий для создания приложений на языке Java, и Plug-in Development Environment (PDE) - среду разработки встраиваемых компонент. Существуют также встраиваемые компоненты для разработки на многих языках программирования: C/C++, COBOL, FORTRAN, PHP и др.

В последние годы Eclipse завоёвывает всё большую популярность среди разработчиков и, в силу бесплатности, во многих организациях стал корпоративным стандартом для разработки приложений.

1.5 Выводы

Документация СПП имеет широкие возможности для организации повторного использования. Проблеме разработки документации семейства посвящен метод разработки документации DocLine. Основой метода является предложенный его авторами язык разметки документации DRL, позволяющий организовать высокую степень повторного использования блоков текста.

В то же время из-за высокой степени повторного использования требуется проводить более тщательный анализ документации, чтобы сохранить конечные документы при внесении изменений. Поэтому важную роль в такой ситуации будет играть средство автоматизированного проведения рефакторинга.

2. Рефакторинг документации

Рефакторинг по отношению к документации означает изменение структуры документации с сохранением конечных документов. Автор работы предлагает ряд операций рефакторинга, обеспечивающих выделение и настройку повторно используемых фрагментов документации. В первую очередь такие операции имеет смысл использовать при разработке «снизу вверх», тем не менее, разработанные операции также можно использовать и в других ситуациях.

Рассмотрим описание случаев использования рефакторинга при разработке документации в контексте легковесного процесса.

Потребность в рефакторинге возникает, когда первый документ уже готов, и появилась необходимость разработать второй документ. Для того чтобы документ, разработанный на чистом DocBook, можно было сопровождать в DocLine, требуется добавление определенной разметки. Для этого автор предлагает операцию рефакторинга «Первичный переход к документации с повторным использованием». Входом этой операции является документ в формате DocBook, а выходом – компонент повторного использования и специализированный документ. Компонент повторного использования (ИП) содержит исходный документа, а специализированный документ (ФИП) использует эту компоненту без каких-либо модификаций для получения конечного документа. Если документация изначально создавалась с использованием метода DocLine, то эту операцию нужно пропустить.

Теперь рассмотрим действия, которые совершает технический писатель при необходимости разработать любой последующий документ. Следует отметить, что приводимый порядок действий не является обязательным.

Технический писатель анализирует существующие документы и определяет, какие фрагменты текста могут быть использованы в новых. Без специализированной поддержки повторного использования технический писатель скопирует интересующие его фрагменты и на их основе продолжит создание новых документов. В DocLine есть возможность явно выделить такие фрагменты и использовать их в контексте различных документов. Для поддержки этого процесса я предлагаю операцию рефакторинга «Выделить информационный элемент». Итак, в отдельные ИЭ имеет смысл выделять фрагменты текста, которые могут быть использованы в нескольких документах, фрагменты, которые будут разрабатываться разными техническими писателями, и крупные фрагменты текста, оказывающиеся лишними при разработке данного документа.

Также для выделения компонент повторного использования можно использовать операцию рефакторинга «Разделить информационный элемент». Эта операция позволяет разделить одну компоненту повторного использования на несколько.

Далее приводится описание операций рефакторинга, которые можно использовать для настройки общих активов.

Возможно, что в новых документах отдельные компоненты повторного использования (ИЭ), которые были нужны во всех предшествующих документах, окажутся лишними. В таком случае можно использовать операцию «Объявить ссылку на ИЭ необязательной». Эта операция позволяет ссылку на ИЭ из обязательной сделать необязательной.

Если в новых документах оказываются лишними отдельные фрагменты текста, можно использовать операции «Выделить в точку расширения» и «Выделить в Insert-After/Before». Эти операции позволяют создавать точки расширения и переносить фрагмент из общих активов в специализированные документы.

Также для тонкой настройки повторного использования возможно применение операций выделения небольших фрагментов текста. Разработанные операции поддержки мелкозернистого повторного использования помогают выделять небольшие фрагменты (например, слова и словосочетания) и производить поиск идентичных фрагментов.

Далее приводится описание в терминах языка DRL и примеры упоминавшихся выше операций. В примерах приводятся лишь необходимые для понимания сути конструкции и атрибуты языка DRL, тогда как несущественные опущены. Операции рефакторинга разделены на 4 вида согласно их назначению: вспомогательные операции; операции, позволяющие выделять крупныt фрагменты текста; операции, позволяющие выделять небольшие фрагменты текста и операции настройки повторного использования. Операции, для которых может быть не понятно, в чем заключается роль разработанного средства рефакторинга, снабжены описанием того, как инструментальное средство (ИС) поддерживает их.

2.1 Вспомогательные операции

2.1.1 Первичный переход к документации с повторным использованием

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

- создаётся ИП и ФИП для этого ИП
- создается ИЭ и внутрь него помещается содержимое исходного документа
- в ИП создаётся ссылка на созданный ИЭ.

Рассмотрим пример применения этой операции.
Возьмём небольшой файл в формате DocBook. Вот его содержимое:

```
<book>
    <chapter>
        <para>Небольшой пример</para>
    </chapter>
</book>
```

После применения операции получим два файла.
Содержимое первого файла (общие активы):

```
<DocumentationCore>
    <InfProduct id = "example_id">
        <d:InfElemRef infelemid = "contents_id">
    </InfProduct>
    <InfElement id = "contents_id">
        <book>
            <chapter>
                <para>Небольшой пример</para>
            </chapter>
        </book>
    </InfElement>
</DocumentationCore>
```

Содержимое второго файла (специализированный документ):

```
<FinalInfProduct infproductid = "example_id"/>
```

2.2 Операции выделения крупных фрагментов текста

Эти операции затрагивают общую структуру документации. Они позволяют создавать крупные компоненты повторного использования.

2.2.1 Выделить информационный элемент

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

- Создаётся ИЭ, в него помещается предназначенный для переноса фрагмент текста.
- Вместо выделенного фрагмента создаётся ссылка на новый ИЭ.

В каждом ФИП, содержащем адаптер (адаптер 1) для ссылки на ИЭ, в котором находился выделенный фрагмент, происходит следующее.

- Создаётся адаптер (адаптер 2) для ссылки на новый ИЭ.
- Вся информация, относящаяся к точкам расширения из выделенного фрагмента (конструкции Insert-After/Before, Replace-Nest), удаляется из адаптера 1 и добавляется к адаптеру 2.

Операция поддерживает возможность сделать ссылку на новый ИЭ необязательной. В этом случае в дополнение к предыдущим действиям происходят следующие: в каждый ФИП добавляется адаптер для ссылки на новый ИЭ, если такого адаптера ещё нет и, если фрагмент, выделенный в отдельный ИЭ, попадал в конечный документ при адаптации с помощью данного ФИП.

Приведем пример этой операции без использования необязательной ссылки. Пример дополнительных действий, производимых в случае необязательной ссылки, приводится в описании операции «Изменение опциональности ссылки на ИЭ».

Итак, допустим, у нас есть следующие ИЭ, ИП и ФИП.

```
<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>
```

Теперь посмотрим, что получится после выделения точки расширение `<Nest id = "number_enter">` в отдельный ИЭ. Видно, что был создан новый ИЭ, ссылка на него, и в ФИП появился новый адаптер, куда переместилась часть содержимого старого адаптера.

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

```
    </Nest>
    <InfElemRef id="numberref" infelemid="num_enter_elem">
</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>
    </Adapter>
    <Adapter infelemrefid = "numberref">
        <Insert-After nestid = "number_enter">
            Набрать с помощью встроенной в телефон клавиатуры
        </Insert-After>
    </Adapter>
</FinalInfProduct>
```

2.2.2 Разделить информационный элемент

Эта операция позволяет разделить ИЭ на несколько. Для выполнения операции производятся следующие действия.

- Проверяется, не существует ли ссылка на выбранный для разделения ИЭ, состоящая в группе ссылок, позволяющей использовать только одну ссылку при адаптации (группа ссылок с `modifier = "XOR"`). Если такая ссылка есть, то рефакторинг проводить нельзя.
- Пользователь разделяет содержимое ИЭ на несколько фрагментов, и каждый фрагмент помещается в отдельный ИЭ.
- Для каждого нового ИЭ создаётся ссылка там, где была ссылка на исходный ИЭ.
- Там где были адаптеры для ссылки на исходный ИЭ, создаются адаптеры для новых ссылок, и между ними делится содержимое старого адаптера.

Как и в случае предыдущей операции есть возможность сделать ссылки на новые ИЭ необязательными.

Рассмотрим пример.

Допустим, у нас есть следующий ИЭ, ИП и ФИП.

```
<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>
```

Теперь посмотрим, что получится после разделния ИЭ на два ИЭ, в каждый из которых попадёт одна точка расширения. Видно, что появились два новых ИЭ, старый при этом исчез. Также вместо ссылки на старый ИЭ появились ссылки на новые ИЭ. 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>
```

2.3 Операции выделения небольших фрагментов текста

2.3.1 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. Эту операцию удобно использовать вместе с операцией поиска элемента словаря.

Рассмотрим пример.

Имеется следующий фрагмент и словарь:

…

```
Компания ООО «Суперсофт»  занимается разработкой программного
обеспечения с 1994 года.
…
<Dictionary id = "company">
    <Entry id = "address">Измайловский пр. д. 10</Entry>
</Dictionary>
```

Посмотрим, что получится после выделения фрагмента «ООО «Суперсофт» в элемент словаря. Видно, что в словарь добавилась новая запись, а в тексте появилась ссылка на эту запись:

```
…
Компания <DictRef entryid="name" dictid="company"> занимается
разработкой программного обеспечения с 1994 года.
…
<Dictionary id = "company">
    <Entry id = "address">Измайловский пр. д. 10</Entry>
    <Entry id = "name">ООО «Суперсофт»</Entry>
</Dictionary>
```

2.3.2 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.

Дадим точное описание действий, которые нужно выполнить для проведения этого рефакторинга.

- Выбрать каталог, в который требуется произвести вставку.
- Выбрать шаблон, который будет использоваться для создаваемой ссылки. Если подходящего шаблона нет, то создать его.
- Если есть элемент в каталоге, который можно использовать с выбранным шаблоном для замены выделенного текста, то создать ссылку вместо выделенного текста.
- Если нет подходящего элемента, то либо создать новый элемент, либо добавить атрибуты к существующему. И после этого произвести замену с использованием созданного или модифицированного элемента.

Инструментальное средство в случае этого рефакторинга помогает пользователю выбирать подходящие шаблоны и каталоги, создать шаблон и элемент каталога, исходя из выделенного текста, добавлять атрибуты к выбранному элементу каталога для использования с выбранным шаблоном.

Рассмотрим пример.

Имеется следующий фрагмент и каталог:

```
…
Существуют следующие операции:
Сохранить – сохраняет текущий файл
Сохранить всё – сохраняет все открытые файлы
…
<Directory id = "operation">
</Directory>
```

Допустим, пользователю хочется фрагмент «Сохранить – сохраняет текущий файл» перенести в каталог. Посмотрим, что получится после выделения этого фрагмента в элемент каталога. Во время выполнения этой операции пользователю понадобилось создать разметку выделяемого фрагмента, указав какие его части являются ссылками на

атрибуты элемента каталога, а какие частью шаблона. На основании этой разметки были автоматически созданы шаблон, элемент каталога и ссылка на него вместо выделяемого фрагмента. В результате получается следующее:

```
…
Существуют следующие операции:
<DirRef templateid = "simpleList" entry = "save">
Сохранить всё – сохраняет все открытые файлы
…
<DirTemplate id = "simpleList" dictid = "operation">
   <AttrRef attrid="name"/> - <AttrRef attrid="description"/>
</DirTemplate>

<Directory id = "operation">
   <Entry id = "save">
       <Attr id = "name">Сохранить</Attr>
       <Attr id = "description">сохраняет текущий файл</Attr>
   </Entry>
</Directory>
```

Созданный во время этой операции шаблон, можно будет использовать, к примеру, при выделении фрагмента «Сохранить всё – сохраняет все открытые файлы». Пользователю в этом случае нужно будет лишь выбрать шаблон, а разработанное ИС автоматически сопоставит выделяемый фрагмент с выбранным шаблоном и на основании этого сопоставления создаст элемент каталога и заменит выделяемый фрагмент ссылкой на новый элемент каталога.

2.3.3 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 того, 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.

2.3.4 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.

2.4 Операции настройки повторного использования

Эти операции используются, если нужно настроить возможности адаптации общих активов под контекст использования.

2.4.1 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.

Опишем более подробно те действия, которые выполняются для проведения второй модификации этого рефакторинга.

- Вместо выделенного фрагмента создаётся точка расширения.

- В каждом ФИП для каждой используемой ссылки на ИЭ создаётся (если такого адаптера ещё нет) адаптер, содержащий выделенный фрагмент. *Ссылку будем называть используемой при адаптации в специализированном документе, если при адаптации вместо неё вставляется ИЭ.*
- Во всех адаптерах для ссылки на ИЭ, содержащий выделенный фрагмент, создаётся элемент Replace-Nest, содержащий выделенный фрагмент.

Рассмотрим пример применения второй модификации этой операции.

```
<InfElement id = "phone">
    Существуют следующие способы набора номера:
    - Набрать с помощью встроенной в телефон клавиатуры
</InfElement>
<InfProduct id = "example_id">
    <InfElemRef id = "phoneref" infelemid = "phone">
</InfProduct>

<FinalInfProduct infproductid = "example_id"/>
```

Теперь посмотрим, что получится после переноса фрагмента «- Набрать с помощью встроенной в телефон клавиатуры» в ФИП. Видно, что вместо выделяемого фрагмента появилась точка расширения, а в ФИП появился адаптер, и в него поместился выделяемый фрагмент с помощью конструкции Replace-Nest.

```
<InfElement id = "phone">
    Существуют следующие способы набора номера:
    <Nest id = "number_enter"/>
</InfElement>
<InfProduct id = "example_id">
    <InfElemRef id = "phoneref" infelemid = "phone">
</InfProduct>

<FinalInfProduct infproductid = "example_id">
    <Adapter infelemrefid = "phoneref">
        <Replace-Nest nestid = "number_enter">
            - Набрать с помощью встроенной в телефон
клавиатуры
        </Replace-Nest>
    </Adapter>
</FinalInfProduct>
```

2.4.2 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 2.4.1, лишь тем, что в адаптере создается не Replace-Nest, а Insert-After/Before. Эту операцию можно использовать только, если фрагмент для переноса находится непосредственно после/до точки расширения.

2.4.3 Объявить ссылку на ИЭ необязательной

Позволяет сделать ссылку на ИЭ из обязательной необязательной.

Для выполнения этой операции во всех ФИП происходит создание адаптера для этой ссылки, если она является используемой в ФИП (и, если такой адаптер не был создан ранее).

Рассмотрим пример применения этой операции.

Имеется ИП, два ИЭ (один ИЭ ссылается на другой) и два ФИП. В одном ФИП используется при адаптации ссылка, которую будем делать 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>
```

Теперь посмотрим, что получится после того, как ссылка <InfElemRef id="numberref"> станет необязательной. Видно, что в ФИП, использующем ссылку, появился адаптер, а в ФИП, не использующем 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>
```

3. Реализация

Данная работа является частью исследовательского проекта, посвященного методу создания документации на основе повторного использования. Этот проект выполняется на кафедре системного программирования математико-механического факультета СПбГУ. В 2007 году в рамках дипломных работ были разработаны текстовый и графический редактор DRL [7, 8]. Редакторы поддерживают весь процесс разработки документации от проектирования до публикации. Также редакторы интегрированы друг с другом: изменения в диаграммах отображаются в тексте документации и наоборот.

В мою задачу входили разработка архитектуры средства рефакторинга, реализация предложенных операций и интеграция с разработанными редакторами. Диаграммы языка DRL отображают только общую структуру документации, в то время как большинство операций рефакторинга требуют большего уровня деталей, который есть в текстовом представлении. Поэтому для создания средства рефакторинга, я решил дорабатывать существующий текстовый редактор.

3.1 Архитектура

При создании архитектуры средства рефакторинга необходимо было учесть следующие требования: легкость добавления новых операций, возможность получения информации о структуре документации из нескольких файлов. Для этого:

- был разработан модуль синтаксического разбора документации в формате DRL
- был разработан модуль генерации текстовых файлов DRL из представления, используемого при проведении рефакторинга
- был разработан набор базовых функций, полезных при проведении рефакторинга
- была разработана структура типовой операции рефакторинга.

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, 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 DRL используется после проведения рефакторинга. Этот модуль позволяет из представления, используемого при проведении рефакторинга, получить текстовые файлы DRL похожие на те, которые были до проведения рефакторинга (а если рефакторинг не изменял дерево синтаксического разбора, то идентичные исходным файлам). Есть возможность сохранить всю документацию целиком или отдельные файлы.

Также был создан базовый набор функций, которые могут понадобиться во многих операциях. Среди этих функций есть следующие: поиск элемента в дереве синтаксического разбора по позиции в документе, различные итераторы по дереву DRL и др.

Каждая типовая операция рефакторинга должна состоять из двух частей. В первой происходит проверка того, можно ли провести операцию над выбранным пользователем объектом, а вторая выполняет саму операцию. Первая часть позволяет динамически отображать в консольном меню доступные для выбранного объекта операции.

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).

Схема разработанного средства приведена на рис. 2.

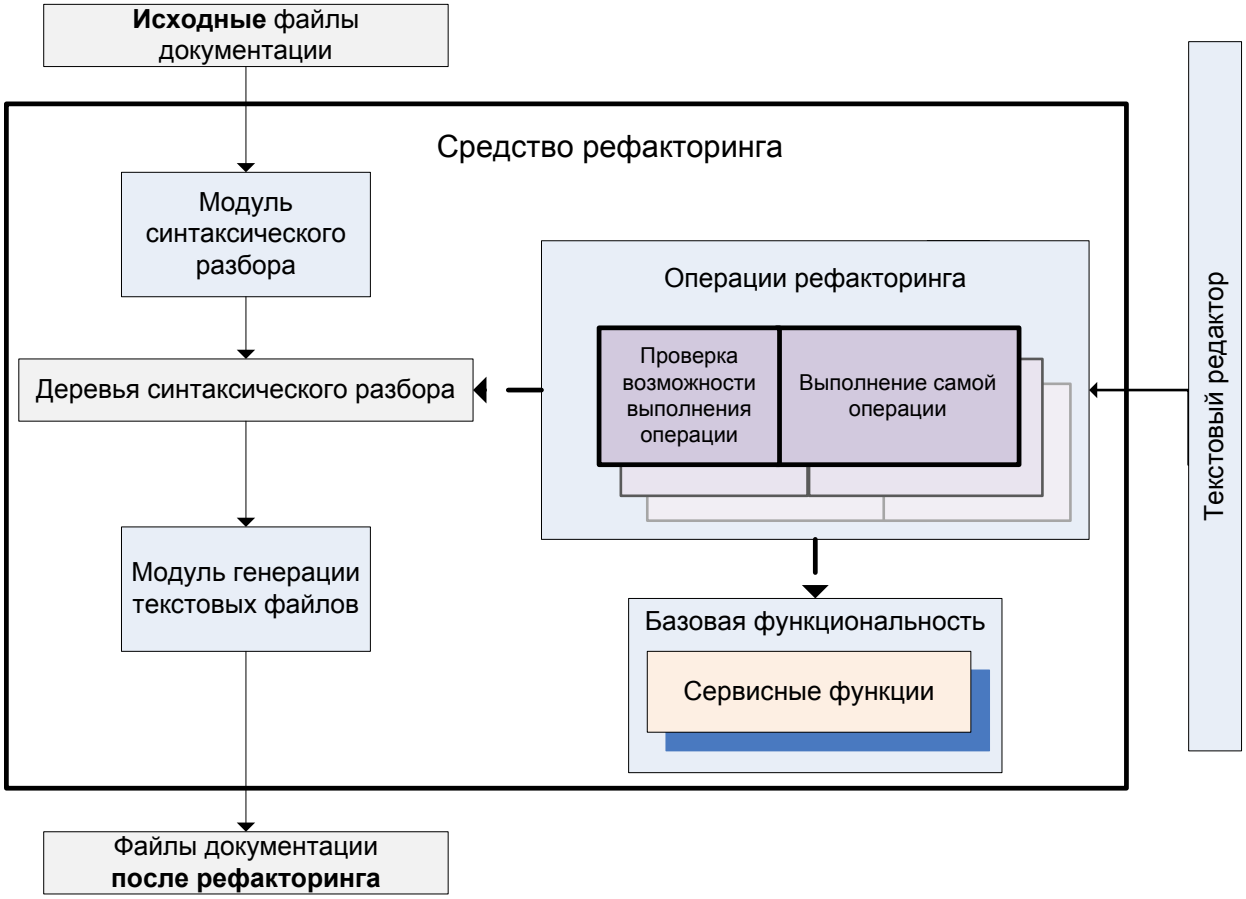

Рисунок 2. Схема средства рефакторинга

Среди реализованных операций лишь одна не создавалась по описанной выше схеме. Эта операция «Первичный переход к документации с повторным использованием». Для её реализации проведена интеграция с мастером создания файлов DRL.

3.2 Трудности реализации

Для создания дерева синтаксического разбора мной использовался SAX[2] парсер. Важной задачей при проведении синтаксического разбора было сохранение максимально точной информации о структуре документа, в том числе получение позиции каждого элемента в тексте документа и получение информации о комментариях.

Для поддержки комментариев понадобилось доработать функциональность используемого в проекте SAX парсера. Для получения этой информации о комментариях был реализован следующий подход: при создании дерева DRL отслеживаются ситуации, когда окончание элемента не совпадает с началом следующего, и из исходного документа извлекается текст комментария по полученным позициям.

[2] SAX - Simple API for XML, механизм чтения данных из XML-документа

4. Описание апробации

4.1 Структура документов

Для апробации был выбран пакет документов «Индивидуальное обучение стажеров ЗАО Ланит-Терком». Этот пакет состоит из 8 документов, имеющих много общих фрагментов текста, созданных при помощи копирования. Каждый такой документ содержит описание тем для обучения стажеров и задания по некоторым темам. Также каждый документ содержит список рекомендуемой литературы.

Моя задача состояла в выделении общих активов из выбранных документов и переводе этих документов на использование общих активов. Если выражаться в терминах DRL, то задача следующая: нужно создать ИП, содержащий общие фрагменты, создать по ФИП на каждый исходный документ и настроить адаптацию общих фрагментов в каждом ФИП. Хотя средства рефакторинга разрабатывались для документации СПП, они могут быть использованы для применения просто к набору документов.

В процессе анализа документов были выявлены следующие общие фрагменты:

- заголовок документа – содержит название документа, название департамента и имена стажеров
- описание темы, посвященной версионному контролю, и задания к ней
- описание темы, посвящённой MS Visual Studio, и задания к ней
- описание темы, посвященной тестированию, и задания к ней
- список рекомендуемой литературы.

На рис. 3 представлена получившаяся диаграмма вариативности. На ней видно, что фрагменты «заголовок» и «список литературы» являются обязательными элементами (на конце связи нет кружочка), т.е. эти элементы должны входить в каждый документ. Остальные элементы могут отсутствовать в некоторых документах.

Рисунок 3. Диаграмма вариативности пакета документов «Индивидуальное обучение стажеров ЗАО Ланит-Терком»

4.2 Использование средства рефакторинга

Работа над набором документов началась с применения операции «Первичный переход к документации в формате DRL» к документу 1. В результате я получил ИЭ содержащий текст документа 1, ИП с ссылкой на созданный ИЭ, ФИП для созданного ИП. Для получения документа, который можно было бы отдать пользователю, нужно провести экспорт ФИП в pdf или html. Далее использовалась операция «Выделение информационного элемента», в результате появились информационные элементы, приведенные на диаграмме вариативности. После этого проводились операции «Вставка в словарь» и поиск элемента словаря.

При переводе последующих документов на использование созданных ИЭ использовались операции «выделение в точку расширения» и «выделение в Insert-After/Before».

Апробация показала удобство использования созданных операций и указала на необходимость создания операций рефакторинга «Объявить ссылку на ИЭ

необязательной» и второй модификации операции «Выделить в точку расширения». Позже эти операции были реализованы и встроены в технологию DocLine.

5. Заключение

В рамках этой работы достигнуты следующие результаты.

- Разработана архитектура средства рефакторинга.
- Предложено 10 операций рефакторинга.
- Все они реализованы на платформе Java/Eclipse и встроены в технологию DocLine.
- Выполнена апробация средства рефакторинга на примере пакета документов «Индивидуальное обучение стажеров ЗАО Ланит-Терком».

6. Список литературы

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

[2] Романовский К.Ю. Метод разработки документации семейств программных продуктов. // Системное программирование. Вып. 2. Сб. статей / Под ред. А.Н.Терехова, Д.Ю.Булычева. СПб.: Изд-во СПбГУ, 2007.

[3] Clark D. Rhetoric of Present Single-Sourcing Methodologies. // SIGDOC'02, October 20-23, 2002, Toronto, Ontario, Canada. - 2002.- P. 20-25

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

[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. Вып. 4. С.

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

[7] Яковлев К.С.: Создание среды разработки документации для семейств программных продуктов, Дипломная работа, Санкт-Петербургский Государственный Университет, Математико-Механический факультет, кафедра системного программирования, 2007.

[8] А. А. Семенов: Система визуального проектирования документации семейств программных продуктов, Дипломная работа, Санкт-Петербургский Государственный Университет, Математико-Механический факультет, кафедра информатики, 2007.

[9] Eclipse Platform // http://www.eclipse.org/

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

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

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