Санкт-Петербургский Государственный Университет
Математико-механический факультет
Кафедра информатики

Создание среды разработки документации для семейств программных продуктов

Дипломная работа студента 542 группы
Яковлева Константина Сергеевича

Научный руководитель старший преподаватель	…….	К. Ю. Романовский /подпись/
Рецензент к.ф.-м.н., доцент	…….	Д. В. Кознов /подпись/
«Допустить к защите» Зав. кафедрой, д.ф.-м.н., профессор	…….	Н. К. Косовский /подпись/

Санкт-Петербург
2007 год

Оглавление

Введение

В любом коммерческом проекте неизбежно возникает необходимость написания значительного объема документации. При этом зачастую перед техническим писателем встают задачи, для эффективного выполнения которых недостаточно имеющихся средств. Спектр таких задач может включать, например, повторное использование готовых частей документации при одновременной разработке схожих программных продуктов или семейств программных продуктов. В такой ситуации, обычно, приходится использовать банальное копирование текста, что может привести к значительным трудностям при исправлении ошибок и дополнении документации. Ведь, как правило, практически любое изменение в коде требует корректировки документации.

Вопрос повторного использования фрагментов документации особенно актуален при разработке семейств программных продуктов. Семейство программных продуктов – это набор программных систем, имеющих общий набор функциональности, удовлетворяющих нуждам конкретного сегмента рынка или конкретной цели и разрабатываемых установленным образом из общего набора повторно используемых активов [1]. В роли повторно используемых активов, как правило, выступают программные компоненты, архитектура, тестовые сценарии и т.п. При этом за рамками рассмотрения, обычно, оказывается документация, разработка которой является достаточно трудоемким процессом. Специфика задачи в данном случае состоит в том, что для каждого продукта из семейства имеется свой пакет документации, и все эти пакеты довольно схожи между собой. В такой ситуации, очевидно, использование лишь стандартного текстового редактора становится неэффективным. Возникает потребность в более мощных средствах разработки документации.

На данный момент популярно несколько подходов создания сложной технической документации. Среди них стоит выделить подход, предложенный компанией IBM, под названием DITA (Darwing Information Typing Architecture) [2]. Этот подход позволяет достичь повторного использования отдельных блоков текста документации, так называемых «топиков» (от англ. topic). Другой распространенный подход – DocBook [3]. Среди его достоинств стоит выделить поддержку принципа единого исходного представления (Single Sourcing) [4], т.е. возможность автоматически создавать документацию в различных форматах из единого набора исходных текстов.

Эти подходы, однако, имеют ряд недостатков. В частности, в контексте разработки документации семейств программных продуктов, они не предоставляют достаточной гибкости. Это связано с тем, что, как отмечалось в [5,6], единственным средством реализации вариативности в них является условное включение блока текста. Более тонкая, «мелкозернистая» адаптация фрагментов текста не предусматривается.

В работе [6] описан подход к разработке документации DocLine, решающий указанные проблемы. Данный подход включает в себя процесс разработки документации, язык разметки документации DRL (Document Reuse Language), имеющий текстовую (DRL/PR – DRL Phrase Representation) и графическую (DRL/GR – DRL Graphic Representation) нотации, а также архитектуру инструментальных

средств. В отличие от упомянутых выше подходов, подход DocLine обеспечивает как «крупноблочное» повторное использование текстов, так и «мелкозернистую» адаптацию фрагментов текста. Кроме того, в языке DRL/PR для разметки документации используются конструкции DocBook, что в итоге позволяет использовать преимущества принципа единого исходного представления для создания пакетов документации в различных форматах.

Данная работа выполнена в рамках исследовательского проекта по созданию метода разработки документации семейств программных продуктов, выполняемого на кафедре системного программирования СПбГУ [5,6]. В работе представлена реализация инструментария для разработки документации семейств программных продуктов, включающая специализированный текстовый редактор для DRL/PR и систему трансляции DRL-документации в форматы HTML и PDF. Также относится к проекту дипломная работа Алексея Семенова [8] – в ней представлена реализация средств графического моделирования в нотации DRL/GR с поддержкой прямой и обратной трансляции в DRL/PR.

1. Постановка задачи

Цель данной работы состоит в создании среды разработки документации на языке DRL/PR, в соответствии с подходом DocLine [6]. Среда разработки должна поддерживать редактирование текстов в формате DRL/PR, а также предоставлять возможность трансляции документации в форматах PDF и HTML. Таким образом, необходимо выполнить следующие задачи.

1. Создать специализированный текстовый редактор для языка DRL/PR.

2. Создать модуль трансляции документации в целевые форматы – PDF и HTML.

3. Создать систему валидации (обнаружения ошибок).

4. Создать сквозной пример, иллюстрирующего применение разработанного инструментария.

Для реализации было решено использовать язык Java [9] и популярную открытую платформу разработки Eclipse [10]. Eclipse – это интегрированная среда разработки (IDE) с очень мощными возможностями расширения. Поставленные задачи реализуются в виде модулей (plugins), интегрируемых с платформой. Eclipse все больше набирает популярность и, фактически, является единственной в своем роде, предоставляющей настолько гибкие возможности для интеграции.

2. Обзор литературы

2.1 Семейства программных продуктов

Разработка семейств программных продуктов – сравнительно новый подход к разработке программного обеспечения. Этот подход обеспечивает высокую степень повторного использования активов разработки, благодаря тому, что продукты семейства схожи между собой и разделяют много общей функциональности. При таком подходе, имеется набор общих активов, которые разрабатываются в контексте целого семейства, и могут быть использованы при разработке каждого конкретного продукта. Вот некоторые из активов, которые могут быть повторно использованы при разработке семейства:

- требования и анализ требований;
- модель предметной области;
- архитектура программного обеспечения;
- анализ/оптимизация производительности;
- тест-планы, тест-кейсы, тестовые данные;
- человеческие ресурсы: навыки и знания людей;
- процессы, методы, инструменты;
- бюджеты, планы;
- готовые программные компоненты;
- документация.

Ключевая концепция подхода может быть выражена следующим образом: «Использование общей базы активов в разработке набора связанных между собой продуктов». Подход также характеризуется стратегическим, планируемым повторным использованием с предсказуемыми результатами [1].

Сформулируем основные преимущества, ради которых многие компании используют этот подход:

- значительно повышается производительность труда;
- быстрый вывод продуктов на рынок;
- поддержание присутствия на рынке;
- повышение качества продуктов;

- снижение стоимости разработки;
- снижение необходимого количества разработчиков в штате.

Согласно [1] разработка семейства программных продуктов состоит из трех основных процессов:

1. Разработка основных активов (англ. Core Assets Development).
2. Разработка конкретного проекта (англ. Product Development).
3. Менеджмент (англ. Product Management).

Все три процесса тесно связаны между собой, и, как правило, характеризуются высокой итеративностью.

Существует также несколько подходов к разработке семейств программных продуктов, определяющих последовательность создания тех или иных активов [5]:

Активный: разработка общих активов в начале.

- разработка в первую очередь общей концепции;
- продукты выводятся на рынок быстро с минимальными трудозатратами;
- необходимы крупные начальные инвестиции и точные прогнозы будущих требований;

Реактивный: в начале разрабатываются один или несколько продуктов.

- из готовых продуктов извлекаются общие активы;
- существенно меньшая начальная стоимость инвестиций;
- архитектура и другие общие активы должны быть надежны, расширяемы и приспособлены под дальнейшие нужды;

Итеративный: постепенная эволюция с изначальным планом – создать семейство программных продуктов.

- разрабатывается часть общих активов, включая архитектуру и некоторые компоненты;
- разрабатывается один или несколько продуктов;
- разработка еще части общих активов;
- разрабатываются остальные продукты семейства.

2.2 Метод DocLine и язык DRL

В данном разделе использованы примеры из работы [7].

Метод DocLine [5,6] состоит из следующих частей:

- процесс разработки документации;
- язык разработки документации DRL (Document Reuse Language), включающий текстовую и графическую нотации;
- архитектура пакета инструментальных средств.

2.2.1 Процесс

Процесс разработки документации, как и процесс разработки семейства в целом, состоит из двух частей – разработки документации семейства и разработки документации продуктов. При необходимости допускается вносить изменения в документацию семейства при разработке документации продуктов – это может потребоваться для тонкой настройки повторного использования документации.

2.2.2 Обзор графической нотации DRL

Графическая нотация языка DRL (DRL/GR – DRL/Graphic Representation) предназначена для визуального проектирования документации семейства. В нее входит три типа диаграмм: главная диаграмма, диаграмма вариативности и диаграмма продукта.

Главная диаграмма позволяет определить все продукты семейства, все необходимые шаблоны документов (т.н. *Информационные Продукты*, ИП), а также задать связи между ними. Связь между продуктом и ИП соответствует специализации этого ИП для данного продукта. Каждому ИП соответствует своя диаграмма вариативности. На диаграмме вариативности отображается структура ИП, задаются связи с повторно используемыми фрагментами документации – *Информационными Элементами* (ИЭ). Таким образом, диаграмма вариативности позволяет описать крупноблочное повторное использование. Еще один тип диаграмм – диаграмма продукта. Эта диаграмма представляет собой, фактически, диаграмму вариативности, адаптированную под конкретный продукт. На рисунке 1 показаны примеры этих диаграмм.

Рисунок 1: DRL/GR - примеры диаграмм

2.2.3 Обзор текстовой нотации DRL

Текстовая нотация DRL (DRL/PR – DRL/Phrase Representation) предназначена для непосредственного написания текстов документации и последующей трансляции в различные форматы для публикации. Любая диаграмма в графической нотации может быть преобразована в DRL-текст, без каких-либо потерь информации, но не наоборот – то есть текстовая нотация более выразительна. Помимо «крупноблочного» повторного использования блоков документации, в текстовой нотации имеются средства для реализации «мелкозернистого» повторного использования, а также адаптации блоков текста под нужды конкретного продукта.

Язык DRL/PR основан на XML, его синтаксис задан при помощи Relax NG [11] схемы (см. Приложение 1). Конструкции языка можно разделить на две группы.

1. Средства разметки документации.

2. Управляющие элементы для определения структуры документации и организации повторного использования.

Средства разметки документации позаимствованы из формата DocBook [3], что позволяет воспользоваться преимуществами этого формата. Управляющие элементы позволяют организовывать как «крупноблочное», так и «мелкозернистое» повторное

использование, а также адаптацию фрагментов текста к контексту использования. Поскольку описанию формата DocBook посвящено большое количество литературы [3], не будем на нем останавливаться, а сфокусируемся на управляющих конструкциях DRL/PR.

В основе документации семейства программных продуктов лежит описание самого семейства. Это описание хранится в отдельном XML-файле. В листинге 1 приведен пример такого описания. Как видно из листинга, каждый продукт имеет название (`name`) и идентификатор (`id`) .

```
<ProductLine name="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: пример описания семейства

Компоненты документации могут быть определены в одном из следующих контекстов:

- контекст семейства (повторно используемые компоненты документации);
- контекст продукта (компоненты документации, относящиеся к конкретному продукту).

Некоторые компоненты документации могут быть определены лишь в одном из контекстов. Например, ИП, который является шаблоном документа, очевидно, может быть определен только в контексте семейства. В DRL/PR элементы `DocumentationCore` и `ProductDocumentation` используются для определения, соответственно, контекста семейства и контекста продукта. Эти элементы являются корневыми, то есть каждый файл документации определяет компоненты, относящиеся к одному контексту.

В листинге 2 приведен пример описания `DocumentationCore`.

```
<DocumentationCore>
  <InfProduct id="userguide" name="Руководство"> <!--...--> </InfProduct>
  <InfElement id="numberid" name="АОН"> <!--...--> </InfElement>
  <Dictionary id="maindict" name="Словарь"> <!--...--> </Dictionary>
  <Directory id="maindir" name="Каталог"> <!--...--> </Directory>
</DocumentationCore>
```

Листинг 2: пример описания контекста семейства

В данном примере заданы четыре типа основных конструкций для организации повторного использования. Рассмотрим их по порядку. `InfProduct` описывает ИП – адаптируемый шаблон документа. Этот элемент может содержать конструкции DocBook для разметки документации, условные включения блоков текста (возможно, также размеченного при помощи DocBook), и ссылки на ИЭ. `InfElement` описывает ИЭ – адаптируемый фрагмент документации, предназначенный для повторного

использования. Он может содержать специальные точки расширения, которые позволяют адаптировать ИЭ под используемый контекст. В листинге также представлены конструкции `Dictionary` – *Словарь*, и `Directory` – *Каталог*. Эти конструкции используются для достижения «мелкозернистого» повторного использования.

В листинге 3 представлен пример описания документации конкретного продукта.

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

Листинг 3: пример описания контекста продукта

Представленный в листинге DRL-документ имеет привязку к конкретному продукту (атрибут «productid» в корневом элементе). В данном случае соответствующий продукт – «Унител-АОН» (см. Листинг 1). Документация продукта может содержать элементы типа `FinalInfProduct` *(Специализированный Информационный Продукт, СИП)*, а также уже упомянутые выше *Словарь* и *Каталог.*

СИП используется для адаптации конкретного ИП под контекст конкретного продукта. Адаптация происходит следующим образом: ИП содержит ссылки на ИЭ, каждая из этих ссылок имеет уникальный идентификатор. Для каждой ссылки в СИП может иметься конструкция *Адаптер*, позволяющая модифицировать соответствующее вхождение ИЭ. Таким образом, в процессе трансляции DRL-текстов ссылка на ИЭ заменяется содержимым самого ИЭ, модифицированным при помощи соответствующего *Адаптера*. Этот подход основан на концепции фреймов Бассета [12]. Помимо *Адаптеров*, в языке также существуют конструкции для условного включения блоков текста.

Как было сказано выше, в состав документации продукта могут входить *Словари* и *Каталоги.* Их использование в документации продукта позволяет переопределить имеющиеся в составе `DocumentationCore` *Словари* и *Каталоги* с теми же идентификаторами, за счет чего достигается «мелкозернистая» вариативность.

2.3 Технология DocBook

Формат DocBook был создан в 1991 году и, в разное время, развивался и поддерживался различными организациями, среди которых Novell, Sun, OASIS.

Первоначально DocBook разрабатывался как приложение SGML (Standard Generalized Markup Language) и обрабатывался при помощи DSSSL (Document-Style Semantics and

Specification Language), но в дальнейшем DocBook стал эволюционировать в направлении XML, а для обработки стал использоваться XSLT (Extensible Stylesheet Language for Transformations).

Разработка DSSSL и XSLT скриптов, предназначенных для работы с DocBook, ведется в рамках Open Source проекта docbook.sourceforge.net [13].

Уже на протяжении многих лет XML формат DocBook является стандартом де-факто для разработки технической документации, например, в большинстве Linux-проектов применяется именно DocBook.

Основные преимущества формата DocBook:

- **Многовариантное представление документов.** Разделение контента и форматирования в XML формате DocBook позволяет трансформировать один и тот же документ во множество различных форматов (например, PDF, RTF, HTML и Eclipse Help). Стандартная поставка DocBook включает таблицы стилей для трансформации в PDF, HTML и Eclipse Help.

- **Унифицированное форматирование.** Документы Docbook не содержат форматированного контента (явной разметки текста, страниц, таблиц и заголовков). Форматирование задается при помощи унифицированного предопределенного набора тегов, на основе которых в последующем производится трансформирование документа в произвольный формат при помощи преобразования на основе таблиц стилей. Таблицы стилей (stylesheet) представляют собой написанные на специальных языках (XSL — Extended Specification Language или DSSSL — Document Style Semantics and Specification Language) файлы скриптов, которые описывают формат вывода различных элементов документа (используемые шрифты, форматирование, нумерацию страниц и т.д.). Использование фиксированного набора тегов гарантирует, что конечный документ будет корректно сформирован независимо от количества авторов.

- **Работа с системами контроля версий.** Так как документы сохраняются в обычных текстовых XML файлах, то можно легко контролировать всю историю изменения любых частей документов. Жесткая спецификация на XML-формат упрощает процесс слияния документов при совместной работе нескольких авторов.

- **Модульность.** Применяемый формат позволяет легко разделить документ на независимые части для упрощения работы над большими документами. При работе удобно разбивать документ на главы, причем широко практикуется создание документов одновременно на нескольких языках. Выбирается базовый язык (в свободных проектах обычно выбирают английский язык за основу) составляющие модули которого в дальнейшем являются опорными для аналогичных модулей на других языках.

2.4 Платформа Eclipse

Разработка платформы Eclipse [10] началась в 2001 году, когда ряд крупнейших компаний IT-индустрии, среди которых были IBM, Borland, Red Hat, SuSE и др., сформировали организацию Eclipse Foundation, с целью создания и продвижения собственной интегрированной среды разработки (IDE). В 2004 году было принято решение о реорганизации Eclipse в некоммерческое сообщество, и открытии исходных кодов. Так Eclipse вошел в мир Open-Source (ПО с открытым исходным кодом). С этого момента технология Eclipse и исходные коды стали доступны любому желающему, бесплатно, в соответствие с Eclipse Public License. В настоящее время в разработке принимают участие более 115 компаний. Они ведут работу над 9 основными Open-Source проектами, содержащими более 50 подпроектов.

Перечислим основные особенности платформы Eclipse:

- поддержка разработки ПО на множестве языков (основным является Java);
- кросс-платформенность;
- модульность, нацеленность на расширение независимыми разработчиками;
- открытость исходных кодов;
- поддержка Фонда Eclipse, куда входят такие компании-разработчики ПО, как IBM, Oracle и Borland.

Eclipse, в первую очередь, – это полноценная среда разработки Java-приложений, нацеленная на групповую разработку, снабжённая средствами для работы с системами контроля версий (поддержка CVS входит в поставку Eclipse, активно развиваются несколько вариантов модулей для поддержки SVN, существует поддержка VSS и других). В силу бесплатности, во многих организациях Eclipse является корпоративным стандартом для разработки Java-приложений.

Второе назначение Eclipse – служить платформой для разработки новых расширений (отчасти, этим объясняется популярность платформы – любой разработчик может расширить функциональность Eclipse своими модулями). Таковыми стали C/C++ Development Tools (CDT), разрабатываемые инженерами QNX совместно с IBM, COBOL, FORTRAN, PHP-средства от различных производителей. Множество расширений дополняет Eclipse средствами для работы с базами данных, серверами приложений и др.

С технической точки зрения, основой Eclipse является так называемая платформа расширенного клиента (от англ. Rich Client Platform, сокращенно RCP). Она состоит из следующих компонент:

- ядро платформы (загрузка Eclipse, запуск модулей);
- OSGi Framework (модель жизненного цикла приложения и служебного реестра);

- SWT (пользовательский интерфейс);
- JFace (файловые буферы, работа с текстом, текстовые редакторы);
- рабочая среда Eclipse (панели, редакторы, проекции, мастеры).

Пользовательский графический интерфейс (GUI) в Eclipse разработан с использованием инструментария SWT (Standard Widget Toolkit). По сравнению со стандартной библиотекой Swing, входящей в поставку Java, у SWT имеются определенные преимущества. Так, в отличие от Swing, в котором эмулируются отдельные графические элементы платформы на Java, в SWT используются «родные» (native) графические элементы платформы, что позволяет достичь более высокого быстродействия. Пользовательский интерфейс Eclipse также зависит от промежуточного слоя GUI, называемого JFace, который упрощает построение пользовательского интерфейса, базирующегося на SWT.

3 Архитектура

Данная работа посвящена реализации инструментария для работы с текстовым представлением документации DRL/PR. Инструментарий включает в себя специализированный текстовый редактор, а также средства публикации, позволяющие из имеющихся DRL-текстов получить документацию в пригодных для публикации форматах – HTML и PDF. Работа является частью исследовательского проекта, включающего также графический редактор. Общая архитектура построена таким образом, чтобы обеспечить пользователя возможностью в любой момент переключаться между графическим и текстовым представлением. На рисунке 2 показан жизненный цикл документа DRL в рамках DocLine-процесса.

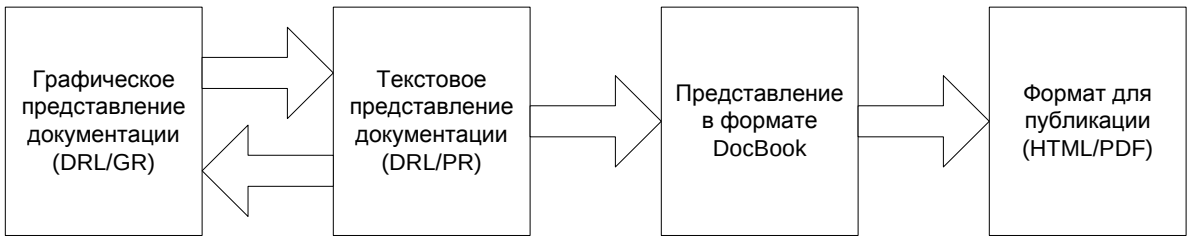

Рисунок 2: Жизненный цикл документа

Как видно из рисунка, документ может быть преобразован как из текстового формата в графический, так и наоборот. Фактически, имеет место сосуществование двух разных представлений одного документа, при этом модификация одного из них автоматически отражается и на другом. Такая концепция носит название round-trip. Это также позволяет использовать графический редактор не только как средство проектирования, но и как средство анализа уже существующей текстовой документации.

В качестве платформы для реализации требуемого инструментария была выбрана платформа Eclipse. Для реализации модулей расширяющих функциональность в Eclipse имеется набор так называемых *точек расширения* (англ. extension point). Точки расширения используются для интеграции функциональности с конкретной подсистемой платформы. Каждая точка расширения имеет уникальный идентификатор, например `org.eclipse.ui.editors` – точка расширения для создания редактора.

В данной работе были использованы следующие точки расширения Eclipse:

- `org.eclipse.ui.editors` – для создания текстового редактора и сопутствующих элементов интерфейса (панели инструментов и меню) для запуска трансляции;
- `org.eclipse.ui.newWizards` – для мастеров по созданию элементов DRL.

3.1 Текстовый редактор

Редактор является основным интерфейсным элементом системы. Его назначение – создание и редактирование исходных текстов на языке DRL/PR. Также редактор включает различные дополнительные интерфейсные элементы, такие как панели инструментов и меню для запуска трансляции.

Хотя для написания текстов DRL/PR можно использовать любой XML-редактор, специализированный редактор обеспечивает ряд преимуществ. Такие преимущества обусловлены тем, что специализированный редактор может учитывать семантику языка, например, при выводе подсказок. Также важно отметить тот факт, что платформа Eclipse позволяет интегрировать в редактор различные интерфейсные элементы, такие как меню и панели инструментов. Хотя их и можно реализовать как отдельные компоненты с помощью специальных точек расширения, в случае разработки редактора, интеграция удобнее, так как платформа автоматически отображает соответствующие панели инструментов и меню при активации соответствующего редактора, и скрывает их, когда пользователь переключается в другой редактор.

Для создания редактора была использована точка расширения `org.eclipse.ui.editors`. В Eclipse имеется инструментарий для создания текстовых редакторов (text editor framework), позволяющий относительно просто реализовать различные функции в текстовом редакторе, такие как подсветка синтаксиса, подсказки и др. Имеется также реализация абстрактного редактора с базовыми функциями, такими как простейшее редактирование текста, копирование/вставка, поиск по тексту.

Инструментарий для создания текстовых редакторов в Eclipse, как и сама платформа, имеет модульную структуру. Для конфигурации редактора используется специальный класс `SourceViewerConfiguration`, который позволяет «подключить» к редактору реализации различных функций. Каждая такая функция определяется

своим интерфейсом. В данной работе были разработаны реализации для следующих интерфейсов, определяющих функции редактора:

- Интерфейс `IContentAssistant` – менеджер интерактивных подсказок. Используется текстовым фреймворком Eclipse для отображения подсказок, по мере того, как пользователь набирает текст, или по нажатию определенной комбинации клавиш. Данная функция была реализована для ссылок по идентификатору. В любом контексте DRL/PR, подразумевающем наличие ссылки по идентификатору автоматически предлагается на выбор набор вариантов. Например, если речь идет о ссылке на ИЭ, то в момент написания значения атрибута `infelemid` будет предложен список всех имеющихся в данном проекте ИЭ. При выборе одного из них, в качестве значения атрибута автоматически подставится соответствующий идентификатор.

- `Интерфейс ITextDoubleClickStrategy` – стратегия обработки двойного клика мышью. Была реализована одна из простейших стратегий – выделение слова.

- `Интерфейс IAutoEditStrategy` – стратегия авто-редактирования. Авто-редактирование – это функция текстового редактора, призванная избавить пользователя от рутинных действий. Простейший пример авто-редактирования – автоматическая подстановка закрывающей скобки. В нашем случае была реализована подстановка закрывающего тега и закрывающей кавычки.

- `Интерфейс IPresentationReconciler` – данный компонент отвечает за отображение редактируемого текста, в частности за подсветку синтаксиса. Для реализации подсветки синтаксиса не требуется реализовывать этот интерфейс, а достаточно использовать стандартный компонент, правильным образом его сконфигурировав. Для этого необходимо указать правила, по которым текст разбивается на сегменты (реализовав интерфейс `IDocumentPartitioner`), и для каждого типа сегмента в свою очередь указать правила по которым «раскрашивать» текст в данном сегменте (реализовав интерфейсы `IPresentationDamager` и `IPresentationRepairer` или воспользовавшись стандартной реализацией). В нашем случае было определено 3 типа сегментов: комментарий, XML-тэг и тип «по умолчанию», включающий все остальное. Для комментариев использовалась стандартный класс `NonRuleBasedDamagerRepairer`, позволяющий задать сплошной цвет на весь сегмент. Для XML-тегов и сегментов «по умолчанию» использовался стандартный класс `DefaultDamagerRepairer`, позволяющий использовать лексический анализатор для разбора соответствующей области текста и указания требуемых цветов.

3.2 Трансляция

Модуль трансляции позволяет создавать на основе текстов DRL файлы документации, пригодные для публикации. Поддерживаются форматы HTML и PDF. Общая схема трансляции представлена на рисунке 3. Итоговый документ получается путем

адаптации ИП (шаблон документа, один для всех продуктов семейства) под нужды конкретного продукта семейства при помощи СИП (для каждого продукта семейства – свой; имеет привязку к конкретному ИП).

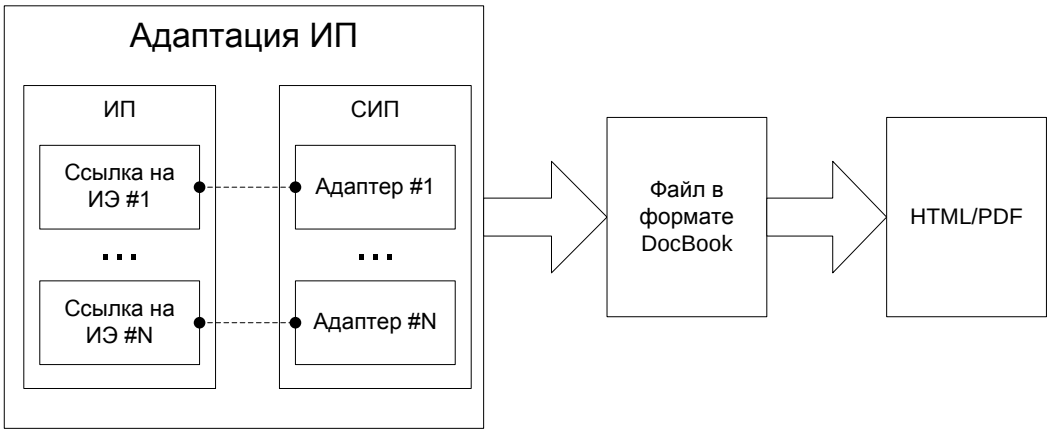

Рисунок 3: Общая схема трансляции

Как можно видеть из рисунка, ИП содержит некоторое количество ссылок на ИЭ. Каждая такая ссылка имеет уникальный идентификатор. Для каждой ссылки из ИП, СИП может иметь адаптер. В процессе трансляции ссылка заменяется содержанием самого ИЭ, при этом происходит адаптация содержимого соответствующим адаптером. В таблице 1 приведен пример такой адаптации.

ИП	`<InfProduct ...>`
	`  <!-- ... -->`
	`  <InfElemRef id="ougoing_calls_ref" infelemid="outgoing_calls"/>`
	`  <!-- ... -->`
	`</InfProduct>`

ИЭ	`<InfElement id="outgoing_calls" name="Документация модуля Исходящие Звонки">`
	`  <!-- ... -->`
	`  <Nest id="number">`
	`  Набор номера в импульсном режиме...`
	`  </Nest>`
	`  <!-- ... -->`
	`</InfElement>`

Адаптер	`<Adapter infelemrefid="outgoing_calls_ref">`
	`  <Insert-After nestid="number">`
	`  Набор номера в тоновом режиме...`
	`  </Insert-After>`
	`</Adapter>`

Результат адаптации	`<!-- ... -->`
	`Набор номера в импульсном режиме...`
	`Набор номера в тоновом режиме...`
	`<!-- ... -->`

Таблица 1: адаптация ИЭ

В приведенном примере демонстрируется конструкция `Nest`, являющаяся точкой вариативности. В Адаптере могут быть использованы конструкции Insert-Before,

Replace-Nest и Insert-After для, соответственно, вставки блока текста до точки вариативности, замены текста в точке вариативности и вставки блока текста после точки вариативности.

Важно заметить, что адаптер ссылается не на сам ИЭ, который он адаптирует, а на его *ссылку*. В данном примере видно, что адаптируемый ИП содержит ссылку на ИЭ ”outgoing_calls”, и идентификатор *этой ссылки* равен ”outgoing_calls_ref”. Такой подход очень важен для организации эффективного повторного использования, так как позволяет ссылаться на один и тот же ИЭ несколько раз в разных местах ИП, и каждый раз адаптировать его по-разному.

ИП также может содержать конструкции условного ветвления, которые можно сравнить с конструкцией `if`, встречающейся во многих современных императивных языках.

3.3 Валидация

Для обеспечения эффективной разработки документации, необходимо ясно и точно сообщать пользователю о любых ошибках, имеющихся в его DRL-текстах. Валидацией называется процесс проверки текстов документации на корректность с точки зрения формата DRL/PR, с явным указанием мест обнаруженных ошибок.

Из-за особенностей формата и процесса трансляции, валидация DRL-текстов представляет собой нетривиальный процесс, состоящий из нескольких шагов. Первый, наиболее простой и очевидный шаг, это использование XML-схемы для проверки корректности синтаксиса документа. Этот процесс также включает в себя проверку документа на корректность форматирования (well-formedness) [14].

В качестве нотации для XML-схемы был выбран популярный формат Relax NG [11]. В этом формате была разработана схема для формата DRL/PR (см. приложение 1). В качестве java-библиотеки для валидации в работе была использована популярная библиотека с открытым исходным кодом Jing [15].

Стоит заметить, что схема DRL/PR-документа выражает в основном ограничения на синтаксис управляющих конструкций, в то время как корректность фрагментов DocBook фактически не проверяется. Это связано с тем, что, в действительности, такую проверку осуществить на данном этапе невозможно, т.к. фрагменты DocBook в процессе трансляции могут быть «перетасованы», например, при включении ссылки на ИЭ, и его адаптации. Таким образом, критерий корректности фрагментов DocBook можно выразить следующим образом: в результате трансляции должен получиться корректный документ в формате DocBook. Корректность документа DocBook, в свою очередь, проверяется при помощи стандартизованной XML-схемы, также в формате Relax NG [16].

В такой ситуации возникает проблема поиска места ошибки в исходном документе, ведь корректность фрагментов DocBook можно проверить лишь после трансляции на полученном DocBook-документе.

Для решения этой задачи используется механизм привязок. В процессе трансляции генерируется DocBook-файл, содержащий дополнительные атрибуты в элементах, указывающие на то, из какого файла они были скопированы, а также номер строки в исходном файле. Текстовые блоки дополнительно обрамляются специальными тегами, также содержащими привязку к исходному файлу.

После того как документ DocBook с привязками получен, он особым образом валидируется. Процесс выглядит следующим образом: содержимое документа последовательно считывается при помощи SAX-парсера [17], но прежде чем попасть в валидатор, каждый элемент проходит через *обработку привязок* – во-первых, удаляются атрибуты привязок, а также обрамляющие текст теги, а во-вторых, информация о привязке последнего прочитанного элемента сохраняется в памяти. Таким образом, если после передачи элемента валидатору происходит ошибка, имеются данные о первоначальном положении вызвавшего ошибку элемента, что и позволяет промаркировать *исходный* файл ошибкой, полученной при валидации сгенерированного документа.

Остался не разобранным еще один тип ошибок – ошибки трансляции . В процессе трансляции могут быть выявлены различные проблемы, например, нарушения ссылочной целостности. Примером такой ситуации может служить ссылка на несуществующий ИЭ, или использование несуществующего ключа в словаре. Также сюда входят ошибки, связанные с некорректным синтаксисом выражений в условных включениях блоков текста.

Таким образом, валидация DRL-документа состоит из 3-х шагов: валидация по схеме DRL/PR, валидация в процессе трансляции, и валидация DocBook. Весь процесс схематично изображен на рисунке 4.

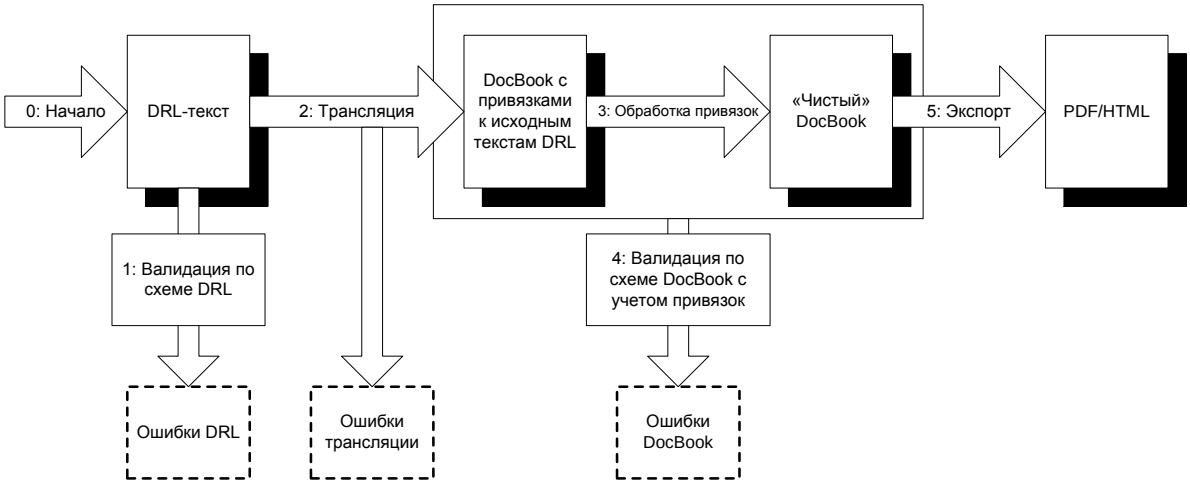

Рисунок 4: Схема валидации DRL-документации

Валидация по схеме DRL/PR происходит автоматически при сохранении документа, так как она не требует много ресурсов. Ошибки трансляции и ошибки DocBook доступны только после попытки оттранслировать DRL-документ в какой-либо из форматов для публикации.

4 Особенности реализации

1. Агрегация XML-документов в процессе трансляции.
2. Механизм привязок, используемый для валидации.
3. Реестр ресурсов.
4. Кеширование трансформаторов и валидаторов.

4.1 Агрегация XML-документов в процессе трансляции.

Одна из проблем, с которой пришлось столкнуться при реализации модуля трансляции – проблема агрегации XML документов, связанная с тем, что в трансляции принимают участие несколько файлов DRL/PR. Рассматривались различные подходы к решению данной проблемы, например слияние необходимых файлов в единый XML-документ перед трансляцией. Однако такое решение не было признано удовлетворительным. В результате было найдено более адекватное решение, заключающееся в использовании XSLT-функции `document()`, позволяющей в процессе трансформации загружать и обрабатывать дополнительные файлы. Для корректной работы данного подхода потребовалось разработать специальный формат URI (Unified Resource Identifier – Унифицированный Идентификатор Ресурса), позволяющий идентифицировать нужные компоненты документации а также обработчик этих URI, позволяющий сопоставить запрашиваемому URI нужный файл. Этот обработчик использует специальный компонент – реестр ресурсов – для запроса информации о расположении конкретных компонент документации и поддержании актуальности этой информации.

4.2 Механизм привязок, используемый для валидации

Разработка системы валидации также вызвала некоторые трудности. Основные сложности были связаны с валидацией корректности сгенерированного DocBook-текста, при которой ошибки бы указывались в исходных DRL-файлах. Ситуация усугублялась еще и тем, что исходных файлов всегда несколько а это значит что привязки по одному лишь номеру строки недостаточно. Найденное в итоге решение можно считать удовлетворительным, так как, хотя оно и увеличивает объем получаемого на промежуточном этапе DocBook-текста, его корректность не вызывает сомнений.

4.3 Реестр ресурсов

Поскольку в DRL/PR используются ссылки по идентификаторам, при этом различные компоненты документации (такие как СИП, ИЭ, Словарь и др.) могут находиться в разных файлах, возникла необходимость в реализации специального реестра, который бы хранил информацию о компонентах документации, их идентификаторах и файлах, в которых они определены. Такой подход также позволяет ускорить трансляцию (отпадает необходимость каждый раз искать нужный компонент по всему проекту).

Явно сформулируем случаи использования реестра ресурсов:

- обработка ссылок во время трансляции;
- вывод подсказок в редакторе;
- выбор СИП при инициации трансляции, в случае если текущий файл содержит несколько СИП;
- реализация в графическом редакторе функции «перехода в текст» [8].

Поскольку в Eclipse возможна работа сразу с несколькими проектами, необходимо было создать систему, позволяющую отдельно регистрировать компоненты документации для разных проектов. Для этого был разработан специальный менеджер реестров, который позволяет управлять реестрами для разных проектов. Также в его задачи входит слежение за состоянием ресурсов и, в случае их модификации, обновление соответствующих данных в реестрах. Для слежения за состоянием ресурсов был использован встроенный в Eclipse механизм ResourceChangeListener, позволяющий отслеживать любые изменения в файлах, относящихся к какому-либо из открытых проектов.

На уровне проекта реестр ресурсов предоставляет ряд операций ориентированных на применение в перечисленных выше случаях использования. Для обработки ссылок во время трансляции наиболее удобным механизмом является поиск по унифицированному идентификатору ресурса (Unified Resource Identifier, URI) [18]. При таком подходе каждому компоненту документации сопоставляется URI, однозначно идентифицирующий этот компонент, и этому URI ставится в соответствие объект, содержащий информацию о месторасположении компонента. Приведем пример такого URI:

```
drlresolve://Core/InfProduct/userguide
^протокол         ^тип       ^идентификатор компонента
             ^контекст
```

Префикс `drlresolve://` определяет протокол URI, в нашем случае он используется для того, чтобы отличить те URI, для которых актуально их применение в реестре ресурсов. Далее следует фрагмент `Core`, его назначение – определить контекст компонента. Значение `Core` обозначает принадлежность компонента к повторно используемой документации (`DocumentationCore`), также может использоваться значение `Product<id продукта>`, например `Productcallerid`, что обозначает принадлежность компонента к документации продукта `callerid`. Далее следует тип компонента документации, в нашем случае это ИП. Последняя часть URI – идентификатор компонента.

Нетрудно видеть, что такого рода URI однозначно идентифицирует компонент документации, а стало быть, может использоваться для реестра ресурсов. Для каждого зарегистрированного компонента в реестре хранится и доступен по соответствующему URI специальный объект, `RegisteredLocation`, содержащий следующую информацию:

- контекст, к которому принадлежит данный компонент;
- тип компонента;
- идентификатор компонента;
- имя компонента;
- файл, содержащий компонент;
- номер строки в файле, с которой начинается описание данного компонента.

Реестр ресурсов также содержит методы для поиска зарегистрированных компонентов по следующим критериям:

- поиск по идентификатору;
- поиск по типу;
- поиск по файлу;
- запрос на список всех зарегистрированных компонентов.

Перечисленных методов вполне достаточно для эффективной реализации остальных случаев использования реестра ресурсов.

4.4 Кеширование транформаторов и валидаторов

В процессе разработки стало понятно, что для достижения приемлемой производительности необходимо оптимизировать наиболее ресурсоемкие операции – создание трансформаторов для применения XSLT трансформаций и Relax NG валидаторов. Поскольку данные компоненты поддерживают многократное использование, возможно создание механизмов кэширования, с тем, чтобы использовать единожды созданные экземпляры соответствующих объектов. Для этих целей были созданы классы `SchemaCache` и `ControllerCache`, для кеширования валидаторов и трансформаторов, соответственно («Controller» в терминах Saxon – аналог трансформатора).

Использование кэширования позволило существенно увеличить скорость трансляции, а также улучшить «отзывчивость» интерфейса в ряде случаев.

5 Пример

В рамках работы был создан пример, иллюстрирующий применение разработанного инструментария. Пример представляет собой документацию вымышленного семейства средств защиты от сетевых угроз «ИнтерБез». Полный листинг примера приведен в приложении 2.

Семейство, описанное в примере, содержит три продукта: домашняя, профессиональная и корпоративная версии. Эти версии имеют различный набор компонент, например, в домашней версии отсутствует функция «Анти-спам». Кроме того, сами компоненты в системах имеют некоторые отличия.

В примере определен ИП – “Руководство пользователя”, а также набор ИЭ. Для каждого из продуктов семейства определен СИП для данного ИП.

Документация была оттранслирована в форматы PDF и HTML:

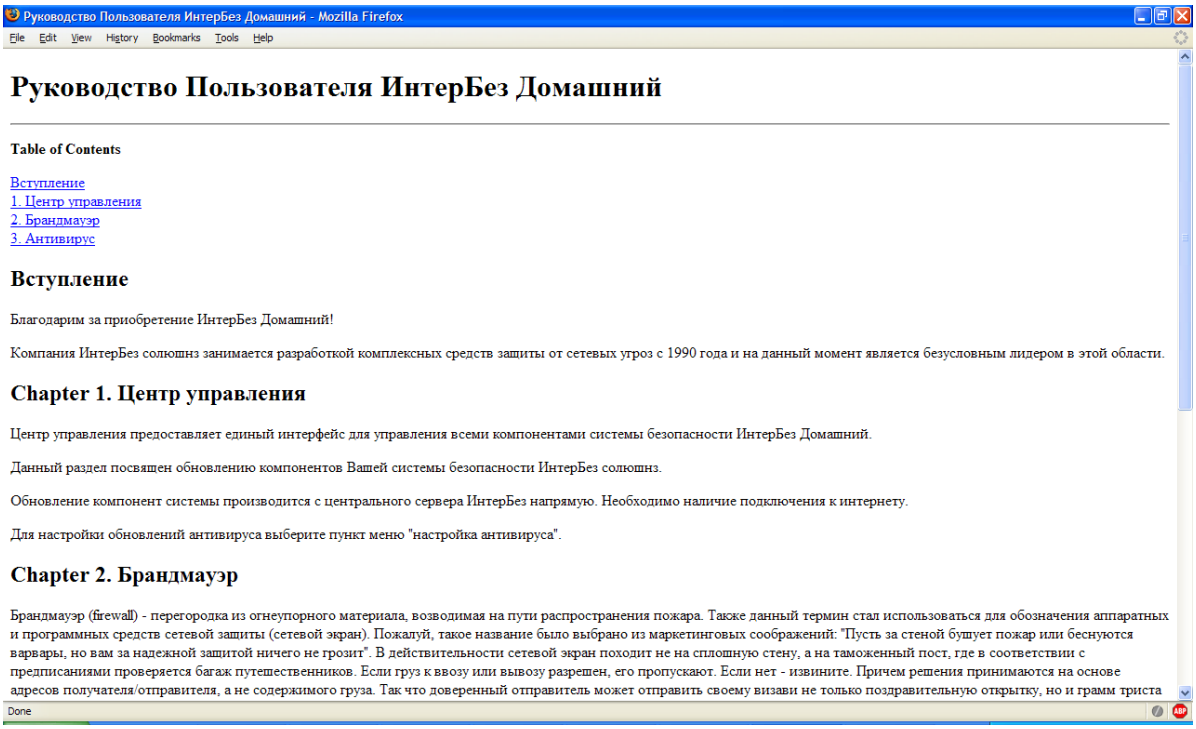

Рисунок 5: "ИнтерБез" домашняя версия, HTML

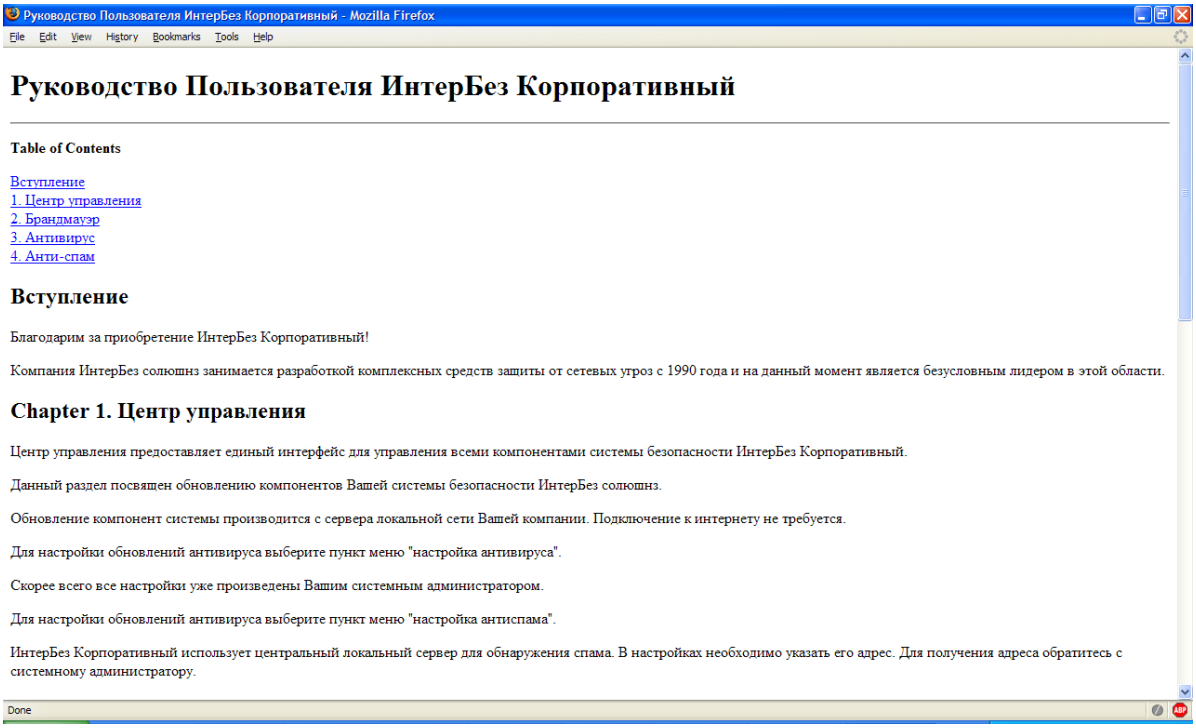

Рисунок 6: "ИнтерБез" корпоративная версия, HTML

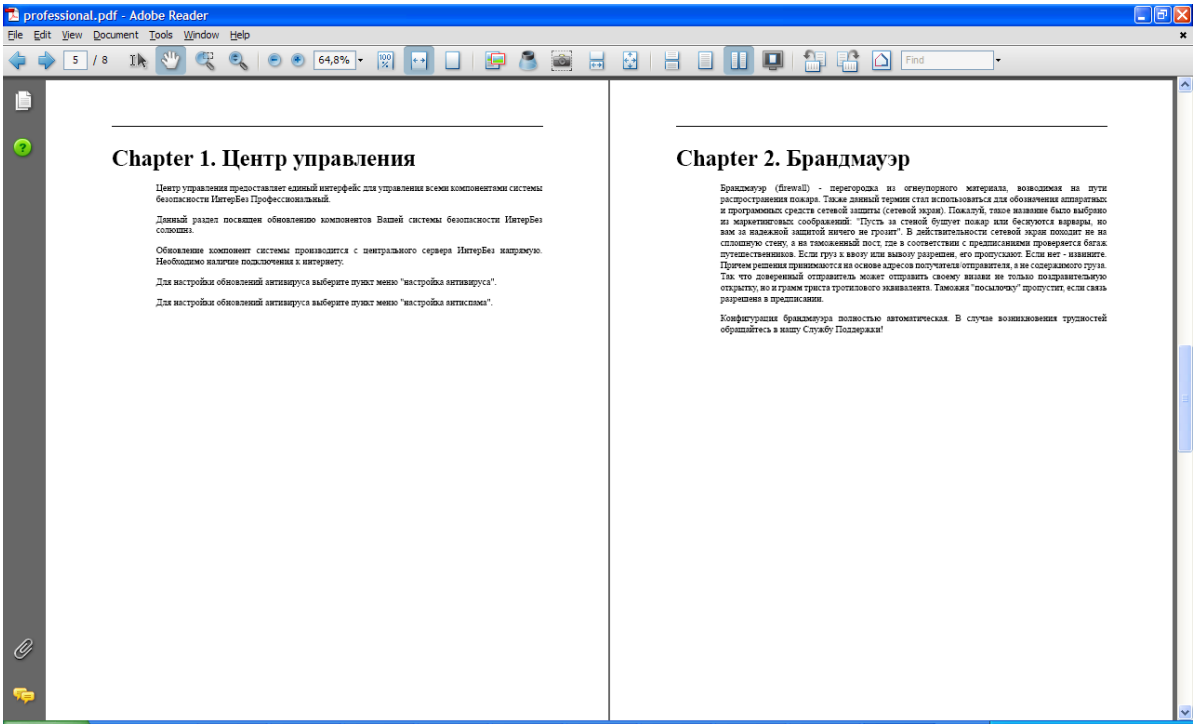

Рисунок 7: "ИнтерБез" профессиональная версия, PDF

Заключение

В рамках данной работы достигнуты следующие результаты:

- разработан текстовый редактор DRL/PR;
- разработан модуль трансляции DRL/PR в форматы HTML и PDF;
- разработана система валидации документов DRL/PR;
- создана схема DRL/PR в формате Relax NG;
- произведена интеграция с платформой Eclipse (реализованы мастера создания различных типов файлов DRL/PR, панели инструментов и др.);
- произведена интеграция с редактором DRL/GR (графическая нотация);
- разработан сквозной пример.

Возможно дальнейшее развитие в следующих направлениях:

- дальнейшее развитие DRL/PR, добавление новых конструкций языка;
- добавление новых возможностей в редактор;
- реализация трансляции в другие форматы: Eclipse Documentation, Microsoft Help, PostScript, и др.

Ссылки

[1] Clements P. & Northrop L.M. (2003). Software Product Lines. // http://www.sei.cmu.edu/programs/pls/sw-product-lines_05_03.pdf

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

[3] DocBook official site // http://www.docbook.org/

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

[5] Jan Bosch, Design and use of software architectures: adopting and evolving a product-line approach // ACM Press/Addison-Wesley Publishing Co., New York, NY, 2000

[6] 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.

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

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

[9] Java Technology // http://java.sun.com/

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

[11] RELAX NG Specification // http://relaxng.org/spec-20011203.html

[12] Basset, P. Framing software reuse – lessons from real world. Yourdon Press, Prentice Hall, 1997

[13] The DocBook Project. // http://docbook.sourceforge.net

[14] Extensible Markup Language (XML) 1.0 (Fourth Edition) // http://www.w3.org/TR/xml/

[15] Jing // http://www.thaiopensource.com/relaxng/jing.html

[16] DocBook 4.x // http://www.docbook.org/schemas/4x

[17] The official website for SAX // http://www.saxproject.org/

[18] T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter. Uniform Resource Identifiers (URI): Generic Syntax // http://www.ietf.org/rfc/rfc2396.txt

Приложение 1. Листинг схемы языка DRL

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar
 xmlns="http://relaxng.org/ns/structure/1.0"
 datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
 ns="http://math.spbu.ru/drl">

<start>
  <choice>
   <ref name="ProductLine"/>
   <ref name="DocumentationCore"/>
   <ref name="ProductDocumentation"/>
  </choice>
 </start>

 <define name="ProductLine">
  <element name="ProductLine">
   <attribute name="name"/>
   <zeroOrMore>
    <element name="Product">
     <attribute name="id"/>
     <attribute name="name"/>
    </element>
   </zeroOrMore>
  </element>
 </define>

 <define name="ProductDocumentation">
  <element name="ProductDocumentation">
   <attribute name="productid"/>
   <zeroOrMore>
    <choice>
     <ref name="FinalInfProduct"/>
     <ref name="Dictionary"/>
    </choice>
   </zeroOrMore>
  </element>
 </define>

 <define name="DocumentationCore">
  <element name="DocumentationCore">
   <zeroOrMore>
    <choice>
     <ref name="InfProduct"/>
     <ref name="InfElement"/>
     <ref name="Dictionary"/>
    </choice>
   </zeroOrMore>
  </element>
 </define>

 <define name="InfProduct">
  <element name="InfProduct">
   <attribute name="id"/>
   <attribute name="name"/>
   <ref name="DocbookOrCommonDrl"/>
  </element>
 </define>

 <define name="InfElement">
  <element name="InfElement">
   <attribute name="id"/>
   <attribute name="name"/>
   <ref name="DocbookOrCommonDrl"/>
  </element>
 </define>

 <define name="Dictionary">
  <element name="Dictionary">
   <attribute name="id"/>
   <attribute name="name"/>
   <zeroOrMore>
```

```
   <element name="Entry">
    <attribute name="id"/>
    <text/>
   </element>
  </zeroOrMore>
 </element>
</define>

<define name="FinalInfProduct">
 <element name="FinalInfProduct">
  <attribute name="id"/>
  <attribute name="infproductid"/>
  <zeroOrMore>
   <choice>
    <element name="SetValue">
     <attribute name="id"/>
     <attribute name="value"/>
     <empty/>
    </element>
    <element name="Adapter">
     <attribute name="infelemrefid"/>
     <zeroOrMore>
      <choice>
       <element name="Replace-Nest">
        <attribute name="nestid"/>
        <ref name="JustDocbook"/>
       </element>
       <element name="Insert-Before">
        <attribute name="nestid"/>
        <ref name="JustDocbook"/>
       </element>
       <element name="Insert-After">
        <attribute name="nestid"/>
        <ref name="JustDocbook"/>
       </element>
      </choice>
     </zeroOrMore>
    </element>
   </choice>
  </zeroOrMore>
 </element>
</define>

<define name="DocbookOrCommonDrl">
 <zeroOrMore>
  <choice>
   <text/>
   <element name="InfElemRef">
    <attribute name="id"/>
    <attribute name="infelemid"/>
    <optional>
     <attribute name="groupid"/>
     <attribute name="optional">
      <choice>
       <value>true</value>
       <value>false</value>
      </choice>
     </attribute>
    </optional>
   </element>
   <element name="InfElemRefGroup">
    <attribute name="id"/>
    <attribute name="name"/>
    <attribute name="modifier">
     <choice>
      <value>one</value>
      <value>set</value>
     </choice>
    </attribute>
   </element>
   <element name="DictRef">
    <attribute name="dictid"/>
    <attribute name="entryid"/>
```

```
    </element>
    <element name="Conditional">
     <attribute name="condition"/>
     <ref name="DocbookOrCommonDrl"/>
    </element>
    <element name="Nest">
     <attribute name="id"/>
     <optional>
      <attribute name="descr"/>
     </optional>
     <ref name="JustDocbook"/>
    </element>
    <element>
     <nsName ns="http://docbook.org/ns/docbook"/>
     <zeroOrMore>
      <attribute>
       <anyName/>
      </attribute>
     </zeroOrMore>
     <ref name="DocbookOrCommonDrl"/>
    </element>
   </choice>
  </zeroOrMore>
 </define>

 <define name="JustDocbook">
  <zeroOrMore>
   <choice>
    <text/>
    <element>
     <nsName ns="http://docbook.org/ns/docbook"/>
     <zeroOrMore>
      <attribute>
       <anyName/>
      </attribute>
     </zeroOrMore>
     <ref name="JustDocbook"/>
    </element>
   </choice>
  </zeroOrMore>
 </define>
</grammar>
```

Приложение 2. Пример

```
--- ProductLine.drl ---
<?xml version="1.0" encoding="UTF-8"?>
<d:ProductLine name="Линейка продуктов ИнтерБез(tm) (Интернет Безопасность)"
xmlns:d="http://math.spbu.ru/drl">
  <d:Product id="home" name="ИнтерБез Домашний"/>
  <d:Product id="professional" name="ИнтерБез Профессиональный"/>
  <d:Product id="enterprise" name="ИнтерБез Корпоративный"/>
</d:ProductLine>


--- UserManual.drl ---
<?xml version="1.0" encoding="UTF-8"?>
<d:DocumentationCore xmlns:d="http://math.spbu.ru/drl"
xmlns="http://docbook.org/ns/docbook">
  <d:InfProduct id="UserManual" name="Руководство Пользователя">
    <book>
      <bookinfo>
        <title>Руководство Пользователя <d:DictRef dictid="main"
entryid="productname"/></title>
      </bookinfo>
      <d:InfElemRef id="ref_preface" infelemid="preface"/>
      <d:InfElemRef id="ref_control_center" infelemid="control_center"/>
      <d:InfElemRef id="ref_firewall" infelemid="firewall"/>

      <d:InfElemRefGroup id="optional_components" modifier="OR" name="опции"/>
      <d:InfElemRef groupid="optional_components" id="ref_antivirus" infelemid="antivirus"
optional="true"/>
      <d:InfElemRef groupid="optional_components" id="ref_antispam" infelemid="antispam"
optional="true"/>
    </book>
  </d:InfProduct>
</d:DocumentationCore>


--- Elements.drl ---
<?xml version="1.0" encoding="UTF-8"?>
<d:DocumentationCore xmlns:d="http://math.spbu.ru/drl"
xmlns="http://docbook.org/ns/docbook">
  <d:InfElement id="about_company" name="О компании">
    <para>
      Компания <d:DictRef dictid="company" entryid="name"/> занимается разработкой
комплексных
      средств защиты от сетевых угроз с 1990 года и на данный момент является безусловным
лидером
      в этой области.
    </para>
  </d:InfElement>

  <d:InfElement id="preface" name="Вступление">
    <preface>
      <title>Встепление</title>
      <para>
        Благодарим за приобретение <d:DictRef dictid="main" entryid="productname"/>!
      </para>
      <d:InfElemRef id="ref_about_company" infelemid="about_company"/>
    </preface>
  </d:InfElement>

  <d:InfElement id="firewall" name="Брандмауэр">
    <chapter>
      <title>Брандмауэр</title>
      <d:InfElemRef id="ref_firewall_description" infelemid="firewall_description"/>
      <d:InfElemRef id="ref_firewall_config" infelemid="firewall_config" optional="true"/>
    </chapter>
  </d:InfElement>

  <d:InfElement id="firewall_description" name="Что такое брандмауэр">
    <para>
      Брандмауэр (firewall) - перегородка из огнеупорного материала,
      возводимая на пути распространения пожара. Также данный термин
```

```
      стал использоваться для обозначения аппаратных и программных
      средств сетевой защиты (сетевой экран). Пожалуй, такое название
      было выбрано из маркетинговых соображений: "Пусть за стеной
      бушует пожар или беснуются варвары, но вам за надежной защитой
      ничего не грозит". В действительности сетевой экран походит не
      на сплошную стену, а на таможенный пост, где в соответствии с
      предписаниями проверяется багаж путешественников. Если груз к
      ввозу или вывозу разрешен, его пропускают. Если нет - извините.
      Причем решения принимаются на основе адресов получателя/отправителя,
      а не содержимого груза. Так что доверенный отправитель может
      отправить своему визави не только поздравительную открытку, но и
      грамм триста тротилового эквивалента. Таможня "посылочку"
      пропустит, если связь разрешена в предписании.
    </para>
  </d:InfElement>

  <d:InfElement id="firewall_config" name="Конфигурирование брандмауэра">
    <para>
      Конфигурация брандмауэра полностью автоматическая.
      В случае возникновения трудностей обращайтесь в нашу Службу Поддержки!
    </para>
  </d:InfElement>

  <d:InfElement id="antivirus" name="Анти-вирус">
    <chapter>
      <title>Антивирус</title>
      <d:InfElemRef id="ref_antivirus_description" infelemid="antivirus_description"/>
      <d:InfElemRef id="ref_antivirus_config" infelemid="antivirus_config" optional="true"/>
      <para>
        Действия при обнаружении вируса или другой угрозы.
        Доступны следующие виды действий:
        <itemizedlist mark="opencircle">
          <listitem><para>удаление</para></listitem>
          <listitem><para>просмотр журнала событий</para></listitem>
          <d:Nest id="extended_actions"/>
        </itemizedlist>
      </para>
    </chapter>
  </d:InfElement>

  <d:InfElement id="antivirus_description" name="Что такое анти-вирус">
    <para>
      Антивирусная программа (антивирус) — программа для обнаружения и удаления компьютерных
вирусов и других вредоносных программ, предотвращения их распространения, а также лечения
программ зараженных ими.
    </para>
    <para>
      Первые, наиболее простые антивирусные программы появились почти сразу после появления
вирусов. Сейчас разработкой антивирусов занимаются крупные компании. Как и у создателей
вирусов, в этой сфере также сформировались оригинальные приемы - но уже для поиска и борьбы
с вирусами. Современные антивирусные программы могут обнаруживать десятки тысяч вирусов.
      Основные задачи современных антивирусных программ:
      <itemizedlist mark="opencircle">
        <listitem><para>Сканирование файлов и программ в режиме реального
времени.</para></listitem>
        <listitem><para>Сканирование компьютера по требованию.</para></listitem>
        <listitem><para>Сканирование интернет-трафика.</para></listitem>
        <listitem><para>Сканирование электронной почты.</para></listitem>
        <listitem><para>Защита от атак враждебных веб-узлов.</para></listitem>
        <listitem><para>Восстановление поврежденных файлов (лечение).</para></listitem>
      </itemizedlist>
    </para>
  </d:InfElement>

  <d:InfElement id="antivirus_config" name="Конфигурирование антивируса">
    <para>
      Конфигурация антивируса полностью автоматическая. В случае возникновения трудностей
обращайтесь в нашу
      Службу поддержки!
    </para>
  </d:InfElement>

  <d:InfElement id="antispam" name="Анти-спам">
```

```
    <chapter>
      <title>Анти-спам</title>
      <para>
        Компонент Анти-спам встраивается в установленный на Вашем компьютере
        почтовый клиент и контролирует все поступающие почтовые сообщения на предмет спама.
        Все письма, содержащие спам, помечаются специальным заголовком.
        Предусмотрена также возможность настройки Анти-спама на обработку спама
        (автоматическое удаление, помещение в специальную папку и т.д.).
      </para>
    </chapter>
  </d:InfElement>

  <d:InfElement id="control_center" name="Центр управления">
    <chapter>
      <title>Центр управления</title>
      <para>
        Центр управления предоставляет единый интерфейс для управления всеми
        компонентами системы безопасности <d:DictRef dictid="main" entryid="productname"/>.
      </para>
      <d:InfElemRef id="ref_components_update" infelemid="components_update"/>
    </chapter>
  </d:InfElement>

  <d:InfElement id="components_update" name="Обновление компонент">
    <para>
      Данный раздел посвящен обновлению компонентов Вашей системы безопасности <d:DictRef
dictid="company" entryid="name"/>.
    </para>
    <d:InfElemRefGroup id="update_methods" modifier="XOR" name="способы обновления"/>
    <d:InfElemRef groupid="update_methods" id="ref_update_local" infelemid="update_local"
optional="true"/>
    <d:InfElemRef groupid="update_methods" id="ref_update_inet" infelemid="update_inet"
optional="true"/>
    <d:Nest id="av-update">
      <para>
        Для настройки обновлений антивируса выберите пункт меню "настройка антивируса".
      </para>
    </d:Nest>
    <d:Nest id="as-update">
      <para>
        Для настройки обновлений антивируса выберите пункт меню "настройка антиспама".
      </para>
    </d:Nest>
  </d:InfElement>

  <d:InfElement id="update_inet" name="Централизованные обновления">
    <para>
      Обновление компонент системы производится с центрального сервера ИнтерБез напрямую.
      Необходимо наличие подключения к интернету.
    </para>
  </d:InfElement>

  <d:InfElement id="update_local" name="Обновления с сервера компании">
    <para>
      Обновление компонент системы производится с сервера локальной сети Вашей компании.
      Подключение к интернету не требуется.
    </para>
  </d:InfElement>

  <d:Dictionary id="company" name="Глобальный словарь">
    <d:Entry id="name">ИнтерБез солюшнз</d:Entry>
  </d:Dictionary>
</d:DocumentationCore>


--- HomeEdition.drl ---
<?xml version="1.0" encoding="UTF-8"?>
<d:ProductDocumentation productid="home" xmlns:d="http://math.spbu.ru/drl"
xmlns="http://docbook.org/ns/docbook">
  <d:FinalInfProduct id="UserManual_home" infproductid="UserManual">
    <d:Adapter infelemrefid="ref_antivirus"/>
    <d:Adapter infelemrefid="ref_update_inet"/>
    <d:Adapter infelemrefid="ref_components_update">
```

```
      <d:Replace-Nest nestid="as-update"/>
    </d:Adapter>
  </d:FinalInfProduct>

  <d:Dictionary id="main" name="Главный Словарь">
    <d:Entry id="productname">ИнтерБез Домашний</d:Entry>
  </d:Dictionary>
</d:ProductDocumentation>


--- ProfessionalEdition.drl ---
<?xml version="1.0" encoding="UTF-8"?>
<d:ProductDocumentation productid="professional" xmlns:d="http://math.spbu.ru/drl"
xmlns="http://docbook.org/ns/docbook">
  <d:FinalInfProduct id="UserManual_professional" infproductid="UserManual">
    <d:Adapter infelemrefid="ref_antivirus">
      <d:Replace-Nest nestid="extended_actions">
        <listitem><para>помещение в карантин</para></listitem>
        <listitem><para>запрос онлайн-помощи</para></listitem>
      </d:Replace-Nest>
    </d:Adapter>
    <d:Adapter infelemrefid="ref_firewall_config"/>
    <d:Adapter infelemrefid="ref_antivirus_config"/>
    <d:Adapter infelemrefid="ref_update_inet"/>
    <d:Adapter infelemrefid="ref_antispam"/>
  </d:FinalInfProduct>

  <d:Dictionary id="main" name="Главный Словарь">
    <d:Entry id="productname">ИнтерБез Профессиональный</d:Entry>
  </d:Dictionary>
</d:ProductDocumentation>


--- EnterpriseEdition.drl ---
<?xml version="1.0" encoding="UTF-8"?>
<d:ProductDocumentation productid="enterprise" xmlns:d="http://math.spbu.ru/drl"
xmlns="http://docbook.org/ns/docbook">
  <d:FinalInfProduct id="UserManual_enterprise" infproductid="UserManual">
    <d:Adapter infelemrefid="ref_antivirus">
      <d:Replace-Nest nestid="extended_actions">
        <listitem><para>помещение в карантин</para></listitem>
        <listitem><para>запрос о помощи у представителя ИнтерБез в вашей
фирме</para></listitem>
      </d:Replace-Nest>
    </d:Adapter>
    <d:Adapter infelemrefid="ref_firewall_config"/>
    <d:Adapter infelemrefid="ref_antivirus_config"/>
    <d:Adapter infelemrefid="ref_update_local"/>
    <d:Adapter infelemrefid="ref_components_update">
      <d:Insert-After nestid="av-update">
        <para>
          Скорее всего все настройки уже произведены Вашим системным администратором.
        </para>
      </d:Insert-After>
      <d:Insert-After nestid="as-update">
        <para>
          ИнтерБез Корпоративный использует центральный локальный сервер для обнаружения
          спама. В настройках необходимо указать его адрес. Для получения адреса
          обратитесь с системному администратору.
        </para>
      </d:Insert-After>
    </d:Adapter>
    <d:Adapter infelemrefid="ref_antispam"/>
  </d:FinalInfProduct>

  <d:Dictionary id="main" name="Главный Словарь">
    <d:Entry id="productname">ИнтерБез Корпоративный</d:Entry>
  </d:Dictionary>
</d:ProductDocumentation>
```

Приложение 3. Использованные сокращения

ИП – Информационный Продукт

ИЭ – Информационный Элемент

СИП – Специализированный Информационный Продукт

DRL – Document Reuse Language

DRL/GR – DRL Graphical Representation

DRL/PR – DRL Phrase Representation