ПРОГРАММИРОВАНИЕ, 2008, № 4, с. 1–13

УДК 681.32

DocLine: МЕТОД РАЗРАБОТКИ ДОКУМЕНТАЦИИ СЕМЕЙСТВ ПРОГРАММНЫХ ПРОДУКТОВ*

© **2008 г. Д. В. Кознов, К. Ю. Романовский**

Механико-математический факультет

Санкт-Петербургский государственный университет

198504 Санкт-Петербург, Старый Петергоф, Университетский пр., 28

E-mail: *dim@dk12687.spb.edu*

Поступила в редакцию

В работе представляется метод DocLine, предназначенный для разработки документации семейств программных продуктов. Метод обеспечивает повторное использование фрагментов документов с адаптаций под контекст использования. Метод состоит из языка DRL (Documentation Reuse Language), имеющего графическую (для проектирования структуры пакета документов) и текстовую (для реализации документации) части, включает процесс разработки документации и архитектуру программных средств на базе DSM-подхода и технологии Eclipse GMF.

ВВЕДЕНИЕ

Разработка электронной документации – это обширная область практики, охватывающая принципы разработки, сопровождения, поддержания целостности больших пакетов документов (например, технических заданий к большим программным системам, нормативной базы банка или крупной корпорации, свода законов), специализированные языки и средства, особенности перевода текстов на разные языки и т.д.

Одно из наиболее активных сообществ в области разработки технической документации – ACM SIGDOC[1], которое проводит многочисленные конференции, посвященные этой тематике. Существует много средств и методов разработки электронной документации. Простая документация разрабатывается в редакторах общего назначения, таких, как Microsoft Word. Для более сложной используются специализированные методы и средства, такие, как FrameMaker [1] (издательская система компании Adobe), DocBook [2] (open-source стандарт в области разработки документации для Unix/Linux-приложений), DITA [3] (метод разработки сложной модульной документации компании IBM) и другие.

Распространенным классом технических документов является пользовательская документация ПО. Ее особенности вытекают из свойств ПО – структурная сложность, изменчивость, многоверсионность, многоязычность, многоформатность (руководство пользователя, сайт поддержки, справочная система). В больших пакетах пользовательской документации много фрагментов текста, которые используются почти без изменений в различных контекстах и документах.

Повторное использование различных активов при создании ПО является бурно развивающейся областью. Усилия практиков и теоретиков сегодня сконцентрированы вокруг семейств программных продуктов (product lines, далее – СПП). Эта парадигма предлагает для набора продуктов компании, обладающих общими свойствами, единый процесс разработки на основе повторно используемых активов в рамках хорошо определенных процедур, а также общую стратегию продвижения на рынке.

*Исследование выполнено при поддержке РФФИ (грант 08-01-00716-а), а также корпорации Intel.

[1] Association for Computer Machinery's Special Interest Group on the Design of Communication.

К сожалению, в рамках СПП разработка пользовательской документации ПО не выделена в отдельную задачу, отсутствуют подходящие методы и инструментальные средства. Такие технологии, как FrameMaker, DocBook, DITA, поддерживают повторное использование, но не обеспечивает адаптивности, то есть повторно-используемые фрагменты должны быть идентичны во всех контекстах использования. Отсутствие поддержки адаптивности сильно ограничивает возможности повторного использования. Метод Бассета позволяет повторно использовать произвольную электронную информацию [5] и обеспечивает адаптивность. Однако он не применялся для управления повторным использованием документации. Наконец, существует очень перспективный подход к управлению повторным использованием с помощью визуального моделирования – это диаграммы возможностей (Feature Diagrams) [6]. Однако данный метод также не применялся для управления повторным использованием документации.

В [7] очерчены основные идеи метода DocLine, интегрирующего различные техники повторного использования в единую технологию разработки пользовательской документации СПП. В [8] подробно описана составная часть DocLine – язык DRL, который обеспечивает адаптивное повторное использование документации и состоит из двух частей – графического представления (DRL/GR) и текстового представления (DRL/PR). Графическое представление служит для проектирования пакета документации с учетом повторного использования, текстовое представление обеспечивает XML-реализацию повторного использования, а также служит для задания полиграфической разметки текста документов. В работе [9] представлена апробация метода DocLine в разработке документации для семейства систем управления телевещанием.

В данной работе мы даем систематическое описание метода DocLine, четко выделяя три его составляющих: язык, процесс разработки и архитектуру инструментальных средств. В язык вводятся новые черты – семантические связи модулей документов. Процесс разбивается на две схемы: идеальную (сверху вниз) и реалистичную (снизу вверх), а также обсуждается идея рефакторинга документации. Предлагается архитектура программных средств поддержки DocLine в среде Eclipse с применением DSM-подхода и технологии GMF. Заложены средства интеграции с DocBook, а также многоуровневая диагностика корректности документации.

Мы хотим выразить признательность руководству лаборатории СПРИНТ (системного программирования и информатики), созданной при математико-механическом факультете СПбГУ при поддержке корпорации Intel, за помощь и поддержку в организации учебно-исследовательских проектов, результаты которых были использованы в данной статье.

1. ОБЗОР

В этом разделе мы рассмотрим контекст нашей работы – СПП-парадигму, а также технологии разработки технической документации, в которых есть средства повторного использования – DITA, DocBook, FrameMaker. Кроме того, мы кратко опишем ряд методов и технологий, которые мы использовали в нашем подходе:

- метод фреймов Бассета [5], предназначенный для адаптивного повторного использования электронной информации произвольной природы;
- подход к визуальному моделированию вариативных свойств семейств программных продуктов – Feature Diagrams [10];
- DSM-подход, допускающий быстрое проектирование и реализацию предметно-ориентированных средств визуального моделирования [11], а также конкретную DSM-технологию Eclipse GMF.

1.1. Семейства программных продуктов

Повторное использование ad hoc – когда-нибудь и кем-нибудь – не прижилось в индустрии. Не получили широкого использования компоненты, которые бы покупались и использовались в разработке в виде готовых блоков. Объем уникального кода в разработках ПО остается очень большим, что не позволяет увеличить размах производства ПО и понизить его трудоемкость и риски. Стало понятно, что повторное использование надо тщательно готовить.

Парадигма СПП, во-первых, сужает контекст повторного использования, ограничивая его группой сходных проектов одной компании.

Во-вторых, повторное использование тщательно готовится и планируется. Процесс разработки разбивается на два подпроцесса – разработку и сопровождение повторно используемых активов и изготовление на их основе целевых систем[2].

Существует несколько моделей процессов разработки СПП. Классический тяжеловесный процесс "сверху вниз" [15] предполагает, что на начальном этапе разработки семейства проводится всесторонний анализ возможностей повторного использования, строится архитектура семейства и продуктов, разрабатываются общие активы. И только после этого начинается разработка конкретных продуктов. В идеале разработка конкретного продукта представляет собой просто выбор и конфигурирование общих активов [16].

Легковесный процесс "снизу вверх" [17] рекомендует начинать с разработки конкретного продукта. Затем, когда наступит очередь создавать второй продукт, необходимо выделить общие активы из первого продукта и на их основе выполнить разработку второго и рефакторинг первого. Основное отличие от тяжеловесного подхода – общие активы создаются только тогда, когда они действительно нужны более чем для одного существующего продукта.

1.2. Средства разработки документации

DocBook – технология разработки документации, предложенная в 1991 г. компаниями HaL Computer Systems и O'Reilly&Associates. Ее основная идея – разделение содержания документа и его форматирования, что позволяет создать единое исходное представление документа (single source) и на этой основе автоматически генерировать документацию в различных форматах, например, печатную документацию (PDF), справочные системы (HTMLHelp/WinHelp), электронную документацию (HTML) [18].

Технология включает в себя язык, который позволяет выполнить полиграфическую разметку и форматирование текстов документов. Современная версия является XML-языком, описание схемы является открытым, стандартизовано консорциумом OASIS и доступно в нескольких форматах – DTD, XML Schema, Relax NG[3].

Целый ряд инструментальных средств поддерживает разработку документов DocBook. В первую очередь, это пакет XSLT-трансформаций, позволяющий по документам DocBook получить документы в виде HTML, HTMLHelp, FO, PDF, RTF и в некоторых других форматах [19]. Также многие коммерческие XML-редакторы (например, XML Spy и Oxygen) предлагают поддержку редактирования DocBook-документов.

В настоящее время DocBook широко используется для разработки документации операционных систем семейства UNIX (FreeBSD, Linux), а также в мире разработки открытого ПО.

Технология DITA предложена компанией IBM в 2001 г. для разработки модульной технической документации. Документация в DITA представляется в виде набора независимых топиков (topic), которые могут иметь типы. Таким образом поддерживается повторное использование крупных и логически завершенных фрагментов текста в разных контекстах. В дальнейшем будем называть такое повторное использование *крупноблочным*.

Язык форматирования документов DITA, также как и DocBook, основан на XML и позволяет не только описывать топики, но полностью задать форматирование текста. DITA позволяет также организовать повторное использова-

[2]Идея совместной разработки группы программных систем была предложена еще в 1976 г. Парнасом [12]. Сегодня в мире существует два научных центра, вокруг которых сосредоточены исследования в области методов разработки линеек программных продуктов. Это Институт Программной Инженерии (SEI) университета Карнеги-Мелон в США [13] и Европейский Институт Программного Обеспечения [14]. Данной тематике посвящены сотни публикаций, многочисленные научные симпозиумы и конференции.

[3]Ранее в DocBook использовался язык SGML (Standard General Markup Language), разработанный в 1980-х гг. при участии Американского Национального Института Стандартизации (ANSI). Сегодня SGML уступает место принятому в DocBook XML-формату.

ние небольших фрагментов текста – отдельных слов, фраз, терминов, фрагментов предложений и т.д. (будем называть такое повторное использование *мелкозернистым*).

Инструментальные средства DITA содержат пакет преобразований документов DITA в различные выходные форматы. Формат DITA стандартизован международным комитетом OASIS и поддерживается многими XML-редакторами – XML Spy, Oxygen, Frame Maker и др. Стандартные инструменты DITA скорее являются экспериментальными и уступают DocBook в части полиграфического оформления текста. Тем не менее, в ряде крупных компаний, например, в IBM, DITA вполне успешно применяется для разработки больших пакетов документов, содержащих много однообразной информации.

Продукт FrameMaker производится компанией Adobe [20] и является универсальным текстовым WYSIWIG[4]-редактором. Благодаря стабильности при работе с большими пакетами документов, а также развитой поддержке полиграфии, FrameMaker стал стандартом де-факто в разработке технической документации. FrameMaker поддерживает структурное редактирование документов на основе аналога XML-схем с помощью своего встроенного языка. Имеется также возможность использовать DocBook, DITA и другие языки XML-разметки. FrameMaker поддерживает также механизм расширений, существует много коммерческих расширений, например, Adobe Robohelp [21], который позволяет представлять документацию в форматах WinHelp/HTMLHelp. FrameMaker предоставляет достаточно мощные средства повторного использования, но не поддерживает адаптивного повторного использования.

1.3. Используемые подходы

Метод Бассета [5] – это универсальный подход к повторному использованию произвольной информации, разработанный в 1980-х годах Полом Бассетом в университете Йорк (г. Торонто, Канада). Пол Бассет проанализировал различные подходы к повторному использованию информации, а также практику их применения, и пришел к выводу, что в различных контекстах переиспользуемые модули должны различаться. Это и является *адаптивным* повторным использованием. Информация представляется в виде архетипа и набора дельт. *Архетип* – это модель или шаблон, общее в разных информационных объектах, *дельта* – это различия объектов.

Метод Бассета позволяет организовывать повторное использование произвольных данных, даже если их формат сам по себе не предусматривает такой возможности. Для этого поверх собственной структуры данных строится специальная разметка, позволяющая выделить архетипы, которые называются фреймами, и описать изменения, требуемые в различных контекстах использования (дельты).

Существует несколько вариантов языка разметки, реализующего метод фреймов. Их объединяет общий набор основных конструкций – фрейм (архетип), точки расширения (предусмотренные позиции, в которых архетип может быть модифицирован), адаптация фрейма (дельта). Последняя реализация языка фреймов – проект XVCL, разрабатываемый группой ученых из Университета Сингапура [22] с начала 2000-х годов. В этой реализации предлагается использовать XML как основу языка разметки. XVCL предназначался для организации платформенно-независимого повторного использования в программных приложениях, наиболее впечатляющие успехи были достигнуты для Java-приложений [23]. Язык XVCL использовался также для повторного использования UML-спецификаций [24].

Известной реализацией метода Бассета является продукт Netron Fusion компании Netron. Этот продукт появился в 1980-х годах, предназначался для реструктуризации программ на

[4] What You See Is What You Get – тип редакторов, которые позволяют в режиме редактирования видеть конечное представление документа.

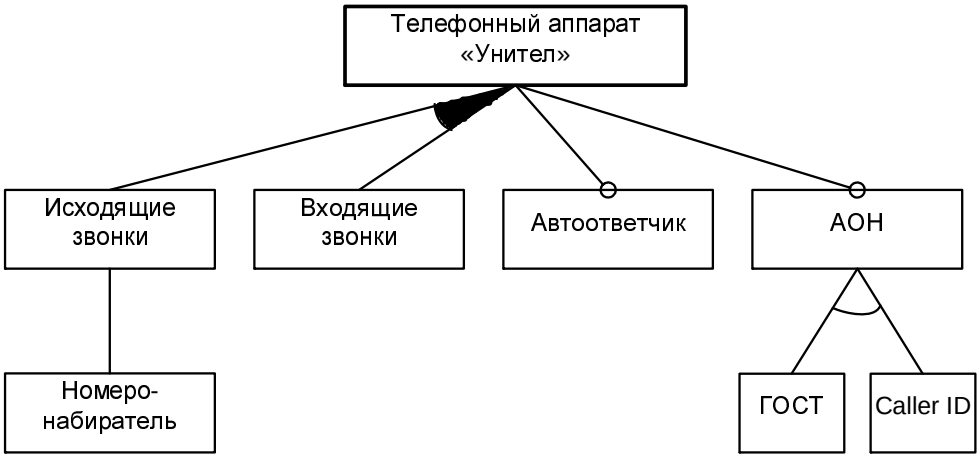

Рис. 1. Пример диаграммы возможностей.

языке COBOL и имел значительный коммерческий успех[5].

Feature Diagrams (диаграммы возможностей) предложены в 1990 г. в составе метода FODA [6] группой исследователей из института программной инженерии США во главе с Кио Кангом (Kyo Kang). Основная цель этих диаграмм – в наглядном виде формализовать архетипы и дельты в СПП. Ключевое понятие этих диаграмм – возможность (feature), то есть обособленное свойство системы, распознаваемое с точки зрения пользователя или разработчика. Вся функциональность представляется в виде набора возможностей, связанных между собой отношениями использования. Отношение использования может быть необязательным (optional), что означает, что не во всех продуктах СПП оно выполняется. Отношения могут объединяться в группы, а на группы могут накладываться дополнительные ограничения – например, что в каждый конечный продукт может входить не более одного отношения из данной группы.

Пример диаграммы возможностей показан на рис. 1. Телефонный аппарат “Унител” обозначает семейство телефонных аппаратов, поддерживающих различные функции. В целом, такие аппараты во многом похожи, однако есть и различия. На данной диаграмме показано, что телефонные аппараты должны поддерживать исходящие звонки или входящие звонки (или и те и другие), могут содержать автоответчик и АОН (автоматический определитель номера), который, в свою очередь, реализуется одним из двух способов – либо в российском стандарте ГОСТ, либо в международном Caller ID.

Существует несколько программных реализаций диаграмм возможностей[6], но все они являются исследовательскими проектами.

DSM-подход и технология Eclipse GMF. Предметно-ориентированное моделирование (DSM, Domain-Specific Modeling) – подход, который появился в конце 1990-х гг. и в настоящее время поддерживается и развивается такими компаниями, как Microsoft, IBM, Borland и др. Его основная идея – ограничение предметной области (вплоть до специфики конкретного проекта разработки ПО) и повышение за счет этого уровня используемых абстракций визуальных языков [27].

[5] Netron [24] – канадская компания, предлагающая инструменты для организации повторного использования в процессе разработки ПО. Успех этого продукта связан с тем, что в языке COBOL отсутствуют встроенные средства структуризации кода. Общепринятой практикой программирования на COBOL было создание модулей по несколько тысяч строк кода, имевших изрядно запутанную структуру.

[6] Например, Feature Modeling Plug-in [25] и XFeature [26].

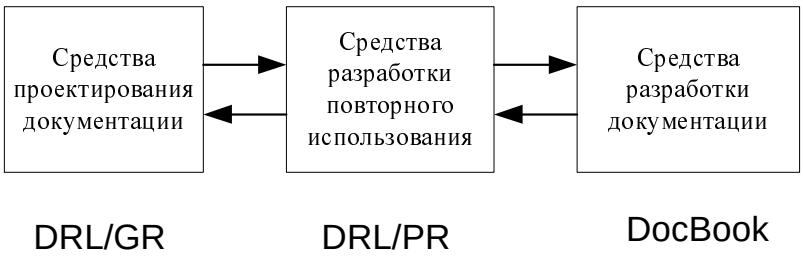

Рис. 2. Схема языка DRL.

В настоящее время бурно развиваются различные технологии поддержки DSM-подхода. Одна из них, GMF (Graphical Modeling Framework), входит в состав платформы Eclipse и разрабатывается при участии таких компаний, как IBM, Borland, HP, BEA и Red Hat. Eclipse GMF позволяет по метамодели языка и набору описаний сгенерировать репозиторий, графические редакторы, процедуры сохранения/восстановления моделей и диаграмм. Сгенерированные инструментальные средства интегрируются в платформу Eclipse, обеспечивая привычный для многих разработчиков пользовательский интерфейс и возможность взаимодействия с большим количеством бесплатных и коммерческих модулей расширения.

1.4. Выводы

Существует ряд зрелых подходов к разработке документации, часть из которых поддерживает повторное использование (DITA, DocBook, FrameMaker). Имеется также общий метод адаптивного повторного использования (метод Бассета), но он не приспособлен для разработки документации. Имеются и методы моделирования общих и различных свойств систем, которые могут быть применены к задаче разработки документации (диаграммы возможностей). Однако нет единого метода разработки документации, поддерживающего проектирование сложной документации и адаптивность повторного использования. Для документации СПП эти аспекты являются ключевыми.

Для реализации DocLine мы выбрали готовые открытые технологии – DocBook (полиграфическая разметка текстов и публикация документов в целевых форматах) и Eclipse GMF (для реализации графических средств проектирования структуры сложных пакетов документации). То есть технологию не надо создавать “с чистого листа”, и есть возможность сосредоточить основные усилия на главном – на поддержке адаптивного повторного использования документации.

2. ЯЗЫК DRL

Язык DRL, как и язык SDL [28], содержит две нотации – графическую и текстовую. По аналогии с SDL они названы GR (Graphic Representation) и PR (Phrase Representation) соответственно. Схема языка DRL представлена на рис. 2.

Средства проектирования документации (DRL/GR) предназначены для задания структуры документации: описания состава СПП, видов документов СПП и их связей с конкретными продуктами, а также для проектирования крупноблочного повторного использования.

Средства разработки повторного использования (DRL/PR) позволяют детально специфицировать различные аспекты повторного использования фрагментов.

Средства разработки документации позволяют задать полиграфическую разметку и форматирование документов.

2.1. Средства проектирования документации

Для проектирования документации в DRL/GR предлагаются следующие виды диаграмм:

- главная диаграмма – отображает список продуктов семейства и состав документации типового продукта;

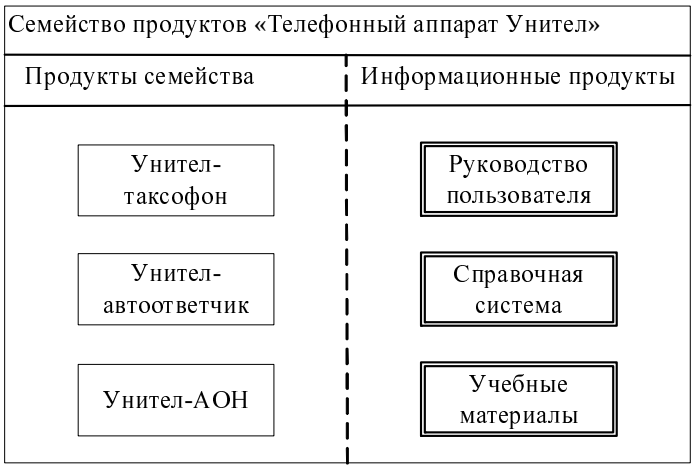

Рис. 3. Главная диаграмма.

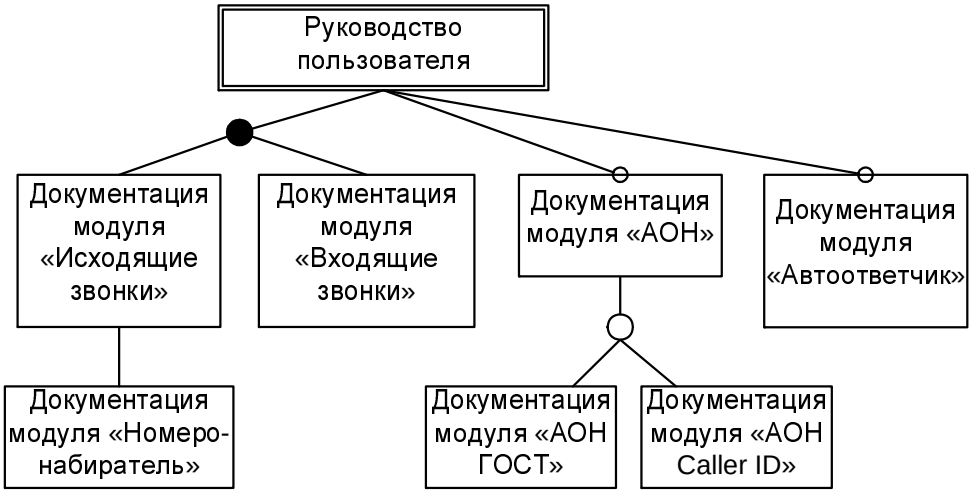

Рис. 4. Диаграмма вариативности.

- диаграмма вариативности – отображает набор повторно используемых фрагментов документации и правила их объединения в документы;

- диаграмма продукта – адаптирует диаграмму вариативности к конкретному продукту СПП.

Пример главной диаграммы представлен на рис. 3.

На ней, в левой секции, представлено семейство “Телефонный аппарат “Унител” (левая секция), которое содержит три вида различных телефонных аппаратов: “Унител-таксофон” (уличный таксофон, поддерживающий только исходящую связь), “Унител-автоответчик” (домашний аппарат с функцией автоответчика), “Унител-АОН” (домашний аппарат с поддержкой функции АОН). Типовой пакет документации этого СПП определен в правой секции диаграммы и включает “Руководство пользователя”, “Справочную систему” и “Учебные материалы”. Части этого пакета называются *информационными продуктами* (ИП) и фактически являются шаблонами целевых документов. Каждый конкретный продукт семейства может включать какие-то определенные ИП, что обозначается линией между продуктом и ИП.

Пример диаграммы вариативности представлен на рис. 4.

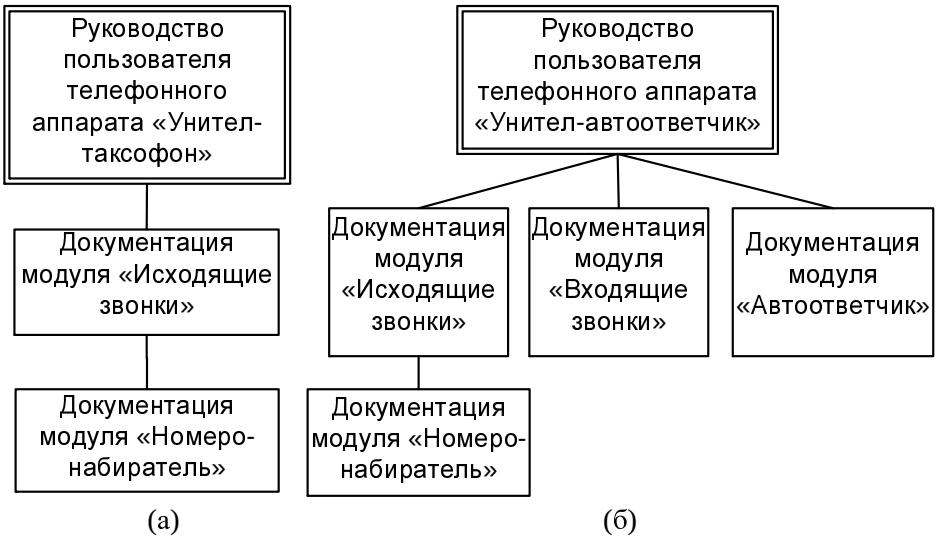

Рис. 5. Пример диаграмм продукта.

На этом рисунке показана структура ИП под названием “Руководство пользователя”. Видно, что этот ИП состоит из следующих частей, называемых *информационными элементами* (ИЭ): “Документация модуля “Исходящие звонки”, “Документация модуля “Входящие звонки”, “Документация модуля “Автоответчик” и “Документация модуля “АОН”, которые далее декомпозируются. ИЭ является обособленным фрагментом документации, подготовленным к повторному использованию, и может быть как синтаксически обособленным модулем (главой, секцией и т.п.), так и плоским фрагментом текста. ИЭ является основной единицей крупноблочного повторного использования.

С помощью связей на диаграммах вариативности показывается иерархия каталожного агрегирования информационных элементов. Здесь использована несколько модифицированная нотация Feature Diagrams [10].

Помимо иерархических связей, на диаграмме вариативности допустимы также семантические связи между произвольными ИЭ. Семантическая связь не имеет точной семантики и предназначена для указания произвольной взаимозависимости двух ИЭ. Ее конкретная интерпретация зависит от особенностей ситуации. Такие связи удобны для сопровождения документов – с их помощью показывается, что два ИЭ содержат одну и ту же информацию, которая, однако, не нашла более формального описания. Поэтому, когда меняется один ИЭ, то нужно изменить и второй.

Пример диаграмм продуктов представлен на рис. 5: ИП “Руководство пользователя” адаптируется для продукта “Унител-таксофон” (рис. 5, а) и для продукта “Унител-автоответчик” (рис. 5, б).

На этих диаграммах “разрешены” все неопределенности исходной диаграммы вариативности. Например, для “Унител-таксофон” (рис. 5, а) в группе произвольного включения ИЭ “Документация модуля “Входящие звонки”” и “Документация модуля “Исходящие звонки”” выбран только последний ИЭ.

2.2. Средства реализации повторного использования

Итак, крупноблочное повторное использование в DRL/GR лишь “набрасывается”, точно же задается в DRL/PR. Например, для ИЭ в терминах DRL/GR задается лишь то, куда он включается и какие ИЭ включаются в него, а в DRL/PR определяются параметры и точки расширения. При этом используется механизм фреймов Бассета – фактически, ИЭ выступает в роли архетипа. Для адаптации ИЭ используется конструкция *адаптер*. Она отличается

от аналогичной конструкции фреймов Бассета, в первую очередь, тем, что позволяет разделить проектирование и специализацию – на этапе проектирования лишь указывается, что один ИЭ включается в другой, а на этапе специализации с помощью адаптера задаются значения параметров и манипуляции с точками расширения в контексте конкретного включения (для разных включений одного и того же ИЭ могут быть заданы различные адаптеры).

Мелкозернистое повторное использование реализуется в DRL с помощью словарей и каталогов. *Словарь* – это набор пар (имя, значение). В произвольной точке документа можно подставить значение элемента словаря, указав его имя. В словаре могут быть такие элементы, как название продукта, версия, набор поддерживаемых операционных систем и т.п. *Каталог* – это набор кортежей (имя, атрибут 1, атрибут 2, ...). С каталогом ассоциируется ряд шаблонов отображения элемента, описывающих, как атрибуты комбинируются при включении элемента каталога в текст. Типичный пример каталога – набор команд пользовательского интерфейса. Каждая команда характеризуется названием, комментарием, пиктограммой, горячей клавишей, строкой меню и т.п. Адаптивность мелкозернистого повторного использования достигается с помощью явного перечисления вариантов использования (с возможностью дополнения списка вариантов в случае необходимости). Типичный сценарий использования вариативности мелкозернистого повторного использования – применение различных словоформ (род, падеж, число и т.п.) одного и того же слова или фразы. Реализуется это с помощью каталога и перечисления вариантов в виде атрибутов элемента каталога. Далее, для каждой словоформы создается набор необходимых шаблонов отображения.

Еще один механизм адаптивности для мелкозернистого повторного использования – продукто-зависимые словари и каталоги. В документации конкретного продукта СПП определяются словари и каталоги с теми же именами, что и в контексте документации всего семейства. При разрешении ссылок на элементы словарей и каталогов в первую очередь выполняется поиск по словарям и каталогам данного продукта, а если подходящий элемент не найден, область поиска расширяется на документацию семейства (документация остальных продуктов не затрагивается).

2.3. Средства реализации документации

Утилитарная функциональность, такая, как полиграфическое оформление текста (главы, разделы, параграфы, рисунки и т.п.) в DocLine реализуется путем интеграции с технологией DocBook. В любую конструкцию DRL может быть вложен текст на DocBook. Можно сказать и по-другому – исходный плоский текст документации размечается сначала средствами DRL, а потом получившиеся фрагменты размечаются с помощью DocBook. При публикации документации набор DRL-описаний транслируется в чистый текст на DocBook, а далее средствами DocBook переводится в различные целевые форматы, такие, как HTML, PDF и т.п.

3. ПРОЦЕСС

Идеальный процесс разработки пользовательской документации СПП предписывает, что создание документации начинается со всестороннего анализа возможностей повторного использования, проектирования структуры документации и описания переиспользуемых фрагментов документации. Затем, когда общие активы документации созданы, на их основе порождается документация конкретных продуктов. Первый практический результат требует довольно большой подготовительной работы, поэтому такой процесс нередко называют “тяжеловесным”. Он сходен с “тяжеловесным” процессом разработки СПП [15] и применим в ситуации, когда заранее известно, какие точно продукты будут в семействе, а также имеется достаточно ресурсов (людей, времени, денег и т.п.) для всестороннего анализа предметной области.

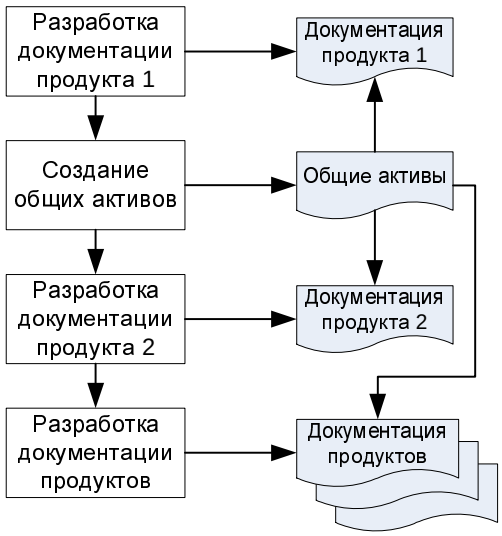

Рис. 6. "Легковесный" процесс разработки документации.

Реалистичный процесс. Однако, на практике многие компании, разрабатывающие ПО, выделяют довольно скромные ресурсы на создание документации. В таких условиях, а также когда разработка семейства начинается с одного продукта (например, в рамках "легковесного" процесса [17]), более применима схема, представленная на рис. 6.

Разработка документации начинается с создания документации для первого представителя СПП. Затем, когда появляется необходимость разработать документацию для других продуктов, выполняется анализ возможностей повторного использования, проектирование пакета документов, выделение общих фрагментов и разработка документации очередного продукта на основе полученных общих активов.

Рефакторинг документации. Создание повторно используемых фрагментов документации требует ее рефакторинга. В случае "легковесного" процесса без этого просто не обойтись, и в случае "тяжеловесного" это также бывает нужно. В DocLine предлагаются следующие виды рефакторинга: извлечение ИП, ИЭ, условного блока, элемента словаря, элемента каталога, шаблона отображения элемента каталога, параметра, точки расширения, адаптера; переименование ИП, ИЭ, элемента словаря, элемента каталога, шаблона отображения элемента каталога.

4. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА

Предлагаемая архитектура средств инструментальной поддержки представлена на рис. 7.

Графический редактор DRL/GR – это визуальный редактор, обеспечивающий проектирование структуры документации в терминах DRL/GR и поддерживающий все три вида диаграмм DRL/GR. Основа разработки – технология Eclipse/GMF.

Текстовый редактор DRL/PR – это текстовый XML-редактор, поддерживающий редактирование текстов на DRL/PR, импорт документации из внешних источников, а также рефакторинг документации.

Менеджер циклической разработки выполняет интеграцию текстового и графического редакторов, поддерживает документацию в целостном состоянии. Фактически, реализуется автоматическая roundtrip-процедура [29].

Модуль диагностики ошибок выполняет проверку корректности текстов на DRL с многоступенчатой диагностикой ошибок. Проверка осуществляется в несколько этапов – на соответствие синтаксису языка DRL, на ссылочную целостность и на корректность сгенерированного DocBook-текста.

Модуль трансляции выполняет трансляцию DRL-текстов в DocBook-формат, заодно проверяя корректность сгенерированных DocBook-текстов по запросу модуля диагностики.

Модуль генерации, основываясь на средствах DocBook, выполняет генерацию целевых текстов документации в форматах HTML и PDF.

Текущий статус разработки такой. В настоящее время реализована версия на платформе Java/Eclipse (первая версия была создана в среде Microsoft Visual Studio, и именно она апробировалась на документации семействе ТВ-систем [9]). В нее входят графический редактор, редактор нашего XML (DRL/PR) и транслятор в DocBook. В этом году мы доводим технологию до устойчивого состояния и пилотно внедряем в ЗАО "ЛАНИТ-Терком" для семейства телефонных станций.

Все технологии, которые используются нами, являются Open Source (Eclipse, DockBook и т.д.). Соответственно, пока и мы планируем распространять наш продукт под Open Source лицензией.

5. О ПРИМЕНИМОСТИ МЕТОДА

Насколько все изложенное выше применимо на практике? Основной вопрос здесь в использовании XML-технологий при создании документации. Ведь технические писатели часто являются нетехническими людьми, и непросто приучить их использовать новейшие средства разработки документации.

Однако в общей массе сообщество технических писателей активно осваивает XML-технологии[7]. Преимущества применения XML в средних и крупных компаниях перевешивают затраты на его внедрение (см., например [31]). На практике активно внедряются технологии DocBook и DITA, которые также основаны на XML. Так, DocBook и DITA успешно применяется в десятках крупных компаний и проектов, среди которых IBM, PostgreSQL, Adobe, Sun, Cray Inc. [32], дистрибутивы Unix-подобных систем, графические оболочки (GNOME, KDE) [33] и др.

Метод DocLine является XML-технологией и надстройкой над DocBook, находясь, таким образом, в основном потоке развития новейших технологий разработки электронной документации. При этом DocLine добавляет полезные возможности относительно адаптивного повторного использования, которые в случае использования, например, "чистого" DocBook требуют от технического писателя "продвинутого" знакомства с XML, либо вообще нереализуемы. Визуальный язык проектирования сложных пакетов документов существенно облегчает использование XML, а наличие roundtrip-процедуры для совместной разработки диаграмм и XML-текстов дает возможность пользоваться результатами диаграммного проектирования в дальнейшем.

[7] Обсуждения использования XML в работе технических писателей можно найти, например, на форуме компании PhiloSoft [30].

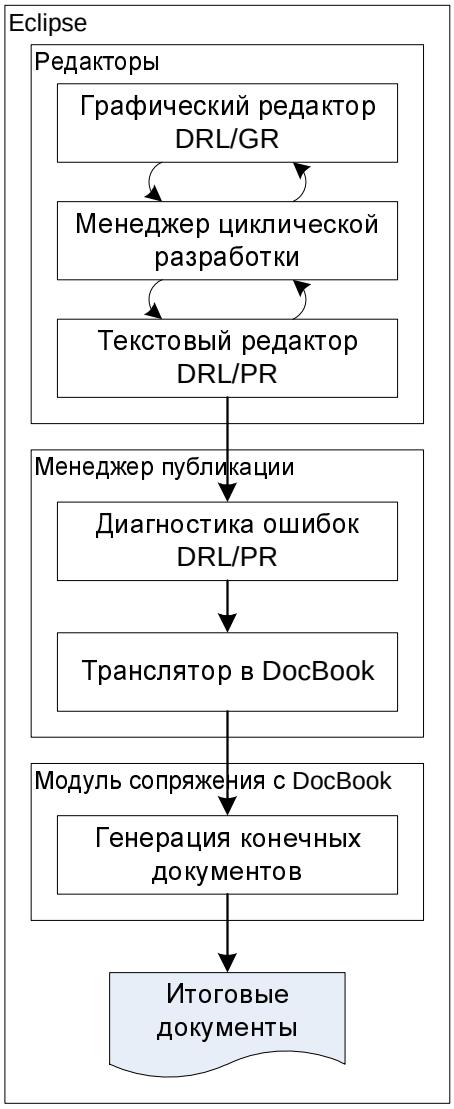

Рис. 7. Схема инструментального пакета DocLine.

Наиболее четко преимущества DocLine проявляются в следующих ситуациях:

- документация различных продуктов имеет много общего, и при этом повторно используемые фрагменты имеют различия;
- документы нуждаются в сопровождении, в частности, периодически выходят новые версии документов, а также создаются новые документы для очередных, только что созданных, продуктов семейства).

Метод DocLine был апробирован в проекте разработки документации семейства систем автоматизации телевещания [9]. Использование

DocLine позволило обеспечить повторное использование текста, однако стало понятно, что для работы технического писателя с XML требуется удобный XML-редактор.

6. ЗАКЛЮЧЕНИЕ

DocLine ориентирован, в первую очередь, на разработку пользовательской документации. Однако с небольшой адаптацией метод можно использовать для документирования процессов (например, в рамках CMM), а также для семейств документов другой природы.

Дальнейшее развитие DocLine предполагается вести в направлении разработки методов рефакторинга документов и более масштабных апробаций на реальных промышленных проектах.

СПИСОК ЛИТЕРАТУРЫ

1. *Marques M.* Single-sourcing with FrameMaker // TECHWR-L Magazine Online. http://www.techwr-l.com/techwhirl/magazine/technical/single-sourcing.html.
2. *Walsh N., Muellner L.* DocBook: The Definitive Guide. O'Reilly, 1999.
3. *Day D., Priestley M., Schell D.A.* Introduction to the Darwin Information Typing Architecture – Toward portable technical information. http://www-106.ibm.com/developerworks/xml/library/x-dita1/.
4. *Clements P., Northrop L.* Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002.
5. *Bassett P.* Framing software reuse – lessons from real world. Prentice Hall, 1996.
6. *Kang K., Cohen S., Hess J., Novak J.,* et al. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21. Software Engineering Institute, Carnegie Mellon University. Pittsburgh, 1990.
7. *Романовский К.Ю.* Метод разработки документации семейств программных продуктов / Терехов А.Н., Булычев Д.Ю. (ред.) Системное программирование. Выпуск 2. СПб.: Издательство Санкт-Петербургского университета, 2007. С. 191–218.
8. *Романовский К.Ю., Кознов Д.В.* Язык DRL для проектирования и разработки документации семейств программных продуктов // Вестник Санкт-Петербургского университета. Серия 10.2007. Выпуск 4. С. 110–122.
9. *Кознов Д.В., Перегудов А.Ф., Романовский К.Ю., Кашин А., Тимофеев А.* Опыт использования UML при создании технической документации / Терехов А.Н., Булычев Д.Ю. (ред.) Системное программирование. Выпуск 1. Сборник статей. СПб.: Издательство СПбГУ, 2005. С. 18–36.
10. *Czarnecki K., Eisenecker U.* Generative Programming: Methods, Tools, and Applications. Reading, Mass.: Addison Wesley Longman, 2000.
11. *Greenfield J., Short K., Cook S., Kent S.* Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Indianapolis, Indiana: Wiley Publishing, Inc., 2004.
12. *Parnas D.* On the Design and Development of Program Families // IEEE Transactions on Software Engineering. March 1976. P. 1–9.
13. Сайт Института Программной Инженерии университета Карнеги-Мелон в США (SEI). http://www.sei.cmu.edu/plp.
14. Сайт Европейского Института Программного Обеспечения (ESI). http://www.esi.es.
15. *Krueger C.* Eliminating the Adoption Barrier // IEEE Software. July/August 2002. P. 29–31.
16. *Krueger C.* New Methods in Software Product Line Practice // Communications of the ACM. December 2006. V. 49. №12. P. 37–40.
17. *Clements P.* Being Proactive Pays Off // IEEE Software. July/August 2002. P. 28–31.
18. *Rockley A., Kostur P., Manning S.* Managing Enterprise Content: A Unified Content Strategy. New Riders, 2002.
19. Сайт проекта DocBook. http://sourceforge.net/projects/docbook.
20. Сайт продукта Adobe FrameMaker. http://www.adobe.com/products/framemaker/.
21. Сайт продукта Adobe RoboHelp. http://www.adobe.com/products/robohelp/.
22. *Jarzabek S., Bassett P., Hongyu Zhang, Weishan Zhang.* XVCL: XML-based variant configuration language. Proceedings of the 25th International Conference on Software Engineering. 2003. 3–10 May 2003. P. 810–811.
23. *Yang J., Jarzabek S.* Applying a Generative Technique for Enhanced Reuse on J2EE Platform. 4th International Conference on Generative Programming and Component Engineering, GPCE'05. September 29–October 1 2005. Tallinn, Estonia. P. 237–255.

24. *Jarzabek S., Zhang H.* XML-based Method and Tool for Handling Variant Requirements in Domain Models. Proc. of the 5th IEEE International Symposium on Requirements Engineering, RE'01. August 2001. Toronto, Canada. IEEE Press, 2001. P. 166–173.

25. Сайт компании Netron. http://www.netron.com/.

26. Сайт продукта Feature Modeling Plug-in. https://sourceforge.net/projects/fmp/.

27. Сайт продукта XFeature. http://www.pnp-software.com/XFeature/.

28. Journal of Visual Languages and Computing. Preface. 2004. V. 15. P. 207–209.

29. ITU-T Recommendation Z.100: Specification and description language (SDL). 1999.

30. *Assmann U.* Automatic Roundtrip Engineering. SC 2003. Workshop on Software Composition. Warsaw, Poland, April 2003 // Electronic Notes in Teoretical Computer Science (ENTCS) 82. 2003. №5. P. 1–9.

31. Форум компании PhiloSoft. http://forum.philosoft.ru/cgi-bin/forum/ikonboard.cgi.

32. *Albing B.* Combining Human-Authored and Machine-Generated Software Product Documentation. Proceedings of the Professional Communication Conference. 2003. IEEE International Volume. 21–24 September 2003. P. 6–11.

33. Список компаний, использующих DITA. http://dita.xml.org/deployments.

34. Список компаний, использующих DocBook. http://wiki.docbook.org/topic/WhoUsesDocBook.