Санкт-Петербургский государственный университет
Математико-механический факультет

Кафедра информатики

Поддержка управления документацией в семействах Web-приложений интенсивной обработки данных

Дипломная работа

Лебедева Мария Валентиновна
541 группа

«Допустить к защите»
Зав. кафедрой:

д.ф.-м.н., профессор Косовский Н. К.

Научный руководитель:

к.ф.-м.н., доцент, зав. лаб. CASE-технологий
мат.-мех факультета СПбГУ Кознов Д.В.

Рецензент:

ст. преп. Смирнов М. Н.

Санкт-Петербург
2011

St. Petersburg State University
Faculty of Mathematics and Mechanics

Chair of Computer science

Documentation development for the problem of simultaneous software and related documentation development on the basis of the product lines

Graduate paper

Lebedeva Maria
541 group

"Approved by"
Head of the chair:

PhD, Professor N. Kosovskyi

Scientific advisor:

PhD, Associate Professor, Head of the CASE- technology laboratory
at the faculty of Mathematics and Mechanics SPbGU D. Koznov

Reviewer:

Senior Lecturer M. Smirnov

St. Petersburg
2011

Оглавление

Введение

В современном мире человек имеет дело с множеством программных продуктов, выполняющих самые разные задачи и работающих на различных платформах. В помощь пользователю вместе с программным обеспечением (ПО) разрабатывается и предоставляется пользовательская документация. Как правило, процесс разработки документации идет параллельно с процессом создания ПО. При этом в ПО вносится множество изменений, в связи с чем необходимо редактировать и пользовательскую документацию. Так как пользовательская документация имеет иную структуру, нежели программный код, то незначительные изменения кода могут потребовать большой работы по «точечному» изменению документации [1]. А это означает, что задача сопровождения и корректировки документации в связи с изменением исходного кода приложения весьма сложна.

В настоящее время активно развивается метод разработки семейств программных продуктов (Software Product Lines), позволяющий создавать наборы похожих приложений, повторно используя общие активы разработки [5]. Разработка пользовательской документации для семейств программных продуктов выдвигает дополнительные требования – управление многочисленными повторами. Для решения этой задачи создана технология DocLine [7, 8].

Для облегчения задачи сопровождения пользовательской документации в работах [1, 2, 3] предлагается подход WebMLDoc для автоматического прослеживания изменений в документации по изменениям в исходном коде приложения, автоматически определяя фрагменты документации, требующие корректировки при точечных изменениях в ПО. При этом сами изменения необходимо вносить «вручную». Подход ориентирован на Web-приложения, созданные в среде разработки WebRatio на основе языка моделирования WebML с последующей полной генерацией кода приложений. В качестве средства разработки документации в подходе WebMLDoc используется DocLine, и блоки документации связываются с элементами гипертекстовой модели языка WebML. Таким образом реализуется трассировка изменений гипертекстовой модели языка WebML и пользовательской документации. Однако работы [2] и [3] оказались связанными лишь на уровне начального замысла, представляя разные и не интегрированные технологии: в [2] решалась задача трассировки изменений гипертекстовой модели WebML/WebRatio и

документации в DocLine, в [3] в WebML/WebRatio добавлялась вариативность. Требовался следующий шаг – интегрировать обе технологии.

Постановка задачи

Данная дипломная работа, вместе с [4], нацелена на решение задачи интеграции результатов [2] и [3]. Ее цель состоит в создании «привязки» пользовательской документации к модели вариативности семейства программных продуктов (а не просто к гипертекстовой модели WebML/WebRatio, как в [2]) и на основе этой «привязки» поддержки трассировки пользовательской документации и модели вариативности и формированию документации конечных продуктов семейства. В [4] решается задача создания модели вариативности семейства программных продуктов для WebML/WebRatio гипертекстовых моделей, основываясь на работе [3].

Для достижения цели дипломной работы было необходимо решить следующие задачи.

1. Изучить язык WebML, его гипертекстовую модель, среду разработки WebRatio, технологию Eclipse GMF (Graphical Modeling Framework).
2. Усовершенствовать подход [3] для работы с «привязкой» пользовательской документации к модели вариативности семейства программных продуктов, построенной по *обобщенному проекту WebRatio*.
3. Реализовать *трассировку* изменений в пользовательской документации в формате DocLine по изменениям в *обобщенном проекте WebRatio*.
4. Усовершенствовать реализацию подхода [3] как для работы с описательной документацией, так и для работы со сценарной документацией.
5. На базе технологии GMF интегрировать реализацию пунктов 2, 3, 4 в общую с [4] среду WebMLDoc.
6. Опробовать разработанные средства на примере.

Глава 1. Обзор

1.1 Язык WebML

1.1.1 Общее описание языка

Язык WebML на сегодняшний день является стандартом де-факто в области Web-моделирования. Он предоставляет графические инструменты для проектирования интерфейса приложения и процессов работы интерфейса. Для создания отдельного Web-приложения на языке WebML необходимо спроектировать несколько его моделей. Каждая из этих моделей описывает создаваемое Web-приложение с разных уровней абстракции.

Ниже перечислены модели языка WebML.

- *Модель данных* (data model) описывает структуру базы данных приложения, является вариантом модели сущность-связь.
- *Модель запросов* (derivation model) расширяет модель данных вычислимыми конструкциями и является аналогом (view) в языке SQL. Эта модель является текстовым языком запросов к модели данных.
- *Гипертекстовая модель* (hypertext model) служит для задания схемы клиентского интерфейса Web-приложения, то есть позволяет определить страницы (pages), показываемые пользователю, находящиеся на страницах элементы управления – контент-модули (content units), – их связь с базой данных и навигацию между ними и страницами.
- *Модель контента* (content model) определяет дополнительные конструкции, которые можно использовать в гипертекстовой модели для задания сложных экранных форм – операции (operation units). Они не могут присутствовать на странице создаваемого Web-приложения, но их можно соединять как со страницами, так и с контент-модулями с помощью связей. С помощью модели контента задается бизнес-логика создаваемого Web-приложения. В действительности эта модель и гипертекстовая модель составляют неразрывное целое.

1.1.2 Гипертекстовая модель

Гипертекстовая модель позволяет описать интерфейс Web-приложения и правила навигации по этому интерфейсу.

Элементами интерфейса, предоставляемыми пользователю для просмотра, являются страницы (pages). Страницы состоят из контент-модулей (content units) – атомарных элементов интерфейса. Они представляют информацию, описанную в модели данных. Страницы группируются в области (areas). Страницы в области имеют общее назначение (область работы с продуктами, область редактирования информации о магазинах сети и пр.). Страницы и области можно группировать в профили сайта (site views). Профили сайта – крупнейшие структурные элементы. Они описывают работу Web-приложения для определенного класса пользователей (профиль зарегистрированного/ незарегистрированного пользователя, профиль администратора). Таким образом, гипертекстовая модель имеет иерархическую структуру.

Страница может быть помечена как домашняя (home) – она будет показываться первой при заходе пользователя на конкретный профиль сайта, как ориентировочная (landmark) – ссылка на такую страницу будет доступна на всех остальных страницах профиля сайта или как страница по умолчанию для какой-либо области сайта (default). Области так же могут быть помечены как ориентировочные (landmark) и ссылка на такие области так же будет видна на всех остальных страницах профиля. На страницу по умолчанию (default) пользователь может попасть, например, если при навигации по сайту он выбрал ориентировочную область (ссылка на которую доступна с любой страницы), но не указал конкретную страницу этой области, на которую хочет перейти.

Навигация в приложении описывается с помощью ссылок (links). Ссылки могут быть определены между контент-модулями на одной странице, между контент-модулями на разных страницах или между страницами. Информация, передаваемая вместе со ссылками, называется контекстной (context). Ссылки, которые передают такую информацию, называются контекстными (contextual links). Обычно такая информация необходима для корректного отображения контент-модулей. В отдельный вид выделяют *транспортные связи,* которые используются для передачи только информации, по таким связям нельзя осуществить переход (передачу управления).

Стоит отметить, что детали внешнего вида интерфейса (вид отображения элементов управления, их цвета, геометрические размеры и пр.) задаются с помощью стилей уже на уровне модельного инструмента.

1.2 Среда разработки WebRatio

WebRatio – это многоцелевая среда разработки, предназначенная для построения уникальных Web-приложений в различных областях. В WebRatio реализован язык WebML, и предоставляются возможности создания модели данных, гипертекстовой модели и модели управления контентом. Для модели данных и гипертекстовой модели в пакете реализованы графические редакторы диаграмм. По окончанию процесса моделирования приложения, существует возможность автоматической генерации кода приложения с использованием стандартов JSP, HTML и XHTML. Среда WebRatio построена на основе платформы Eclipse.

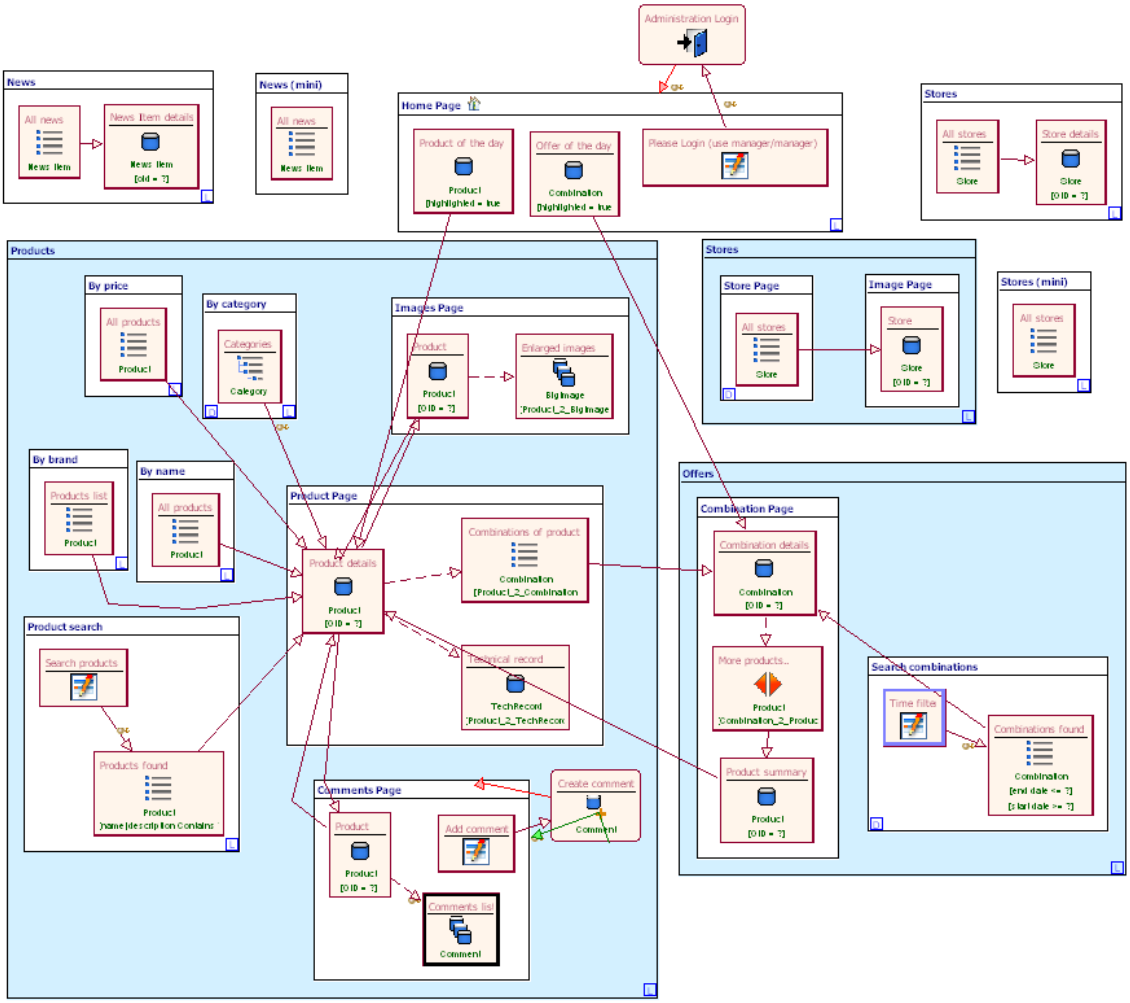

Рисунок 1: Пример гипертекстовой модели в редакторе WebRatio

1.3 Семейства программных продуктов

Семейство программных продуктов (software product lines) – это набор программных систем, обладающих отличительными характеристиками, удовлетворяющими потребностям определенного сегмента рынка, основывающихся на определенном наборе общих ресурсов, а так же разрабатываемых определенным способом [10]. Разработка таких семейств предоставляет много преимуществ (повышения качества выпускаемых продуктов, уменьшение общей стоимости разработки, уменьшение времени вывода на рынок и другие) по сравнению с традиционными подходами к разработке ПО.

Основная идея подхода заключается в использовании общей базы ресурсов для разработки связанных между собой продуктов. Часто для достижения этой цели строится обобщенный продукт с отмеченными точками вариативности. Благодаря таким точкам появляется возможность отобразить все возможные вариации продукта.

Согласно [6] процесс разработки семейства программных продуктов состоит из трех основных видов деятельности:

1) разработка общих активов (core asset development);
2) разработка целевых продуктов (product development);
3) управление разработкой (management).

При разработке общих активов на основе требований к продуктам, стратегий разработки, производственных ограничений и наличия ресурсов, активов и опыта строятся общие активы, методы их использования, архитектура семейства продуктов и спецификации области охвата. В процессе разработки целевых продуктов используются данные об общих активах, а также добавляются требования, выдвигаемые к конкретным продуктам. Для каждого из требований принимается решение о его реализации – отдельная, как новый общий ресурс или нужная функциональность уже реализована в общих активах. Управление разработкой подразумевает под собой контроль за исполнением метода разработки, определенного для семейства, управление проектами, конфигурационное управление, а так же анализ рынка, обеспечение ресурсами, управление взаимодействием с внешним миром, разрешение конфликтов, стратегическое планирование и управление людьми.

Все три вида деятельности имеют выраженный итеративный характер и тесно связаны между собой.

1.4 Диаграммы возможностей (Feature Diagrams)

Диаграммы возможностей (feature diagrams) были предложены в 1990 году группой исследователей из США для формализации вариативности в процессе разработки семейства программных продуктов. Диаграмма возможностей – это набор возможностей (features) и иерархических связей между ними с явно выделенным корнем иерархии, который называется концептом (concept). Под возможностью (feature) понимается явный отличительный аспект или характеристика программного продукта. Таким образом, диаграмма возможностей это дерево, корнем которого, является концепт, а его потомками возможности. В свою очередь возможности могут быть листьями дерева или иметь в качестве потомков другие возможности. Иерархические связи являются отношениями включения.

Отношение между одной возможностью и её родителем (который является либо возможностью, либо концептом) может быть следующим:

- обязательное включение – возможность всегда включается;
- необязательное включение – возможность можно не включать.

Отношения между группой возможностей и их родителем так же может быть двух типов:

- нестрогое альтернативное включение (“OR”) - хотя бы одна из возможностей должна быть включена;
- строгое альтернативное включение (“XOR”) - только одна из возможностей может быть включена.

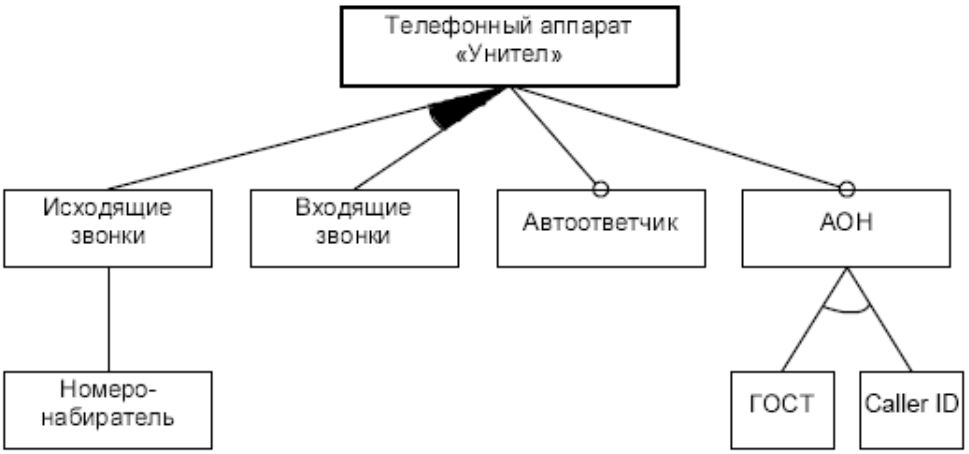

Рисунок 2: Пример диаграммы возможностей

1.5 Технология DocLine и язык DRL

Метод DocLine был разработан для проектирования и разработки документации на основе повторного использования. Он включает в себя:

- процесс разработки документации;
- оригинальный язык разработки DRL, имеющий графическую (DRL/GR – DRL Graphic Representation) и текстовую (DRL/PR – DRL Phrase Representation) нотации;
- инструментальный пакет.

1.5.1 Процесс разработки документации

Процесс разработки документации состоит из двух частей – разработки документации семейства и разработки документации продуктов. При необходимости допускается вносить изменения в документацию семейства при разработке документации продуктов.

Графическая нотация предназначена для «крупноблочного» проектирования и представления документации, а текстовая – соответственно для написания текста. При работе с документацией процесс допускает одновременную работу с этими двумя представлениями и существует возможность перехода от написания текста к редактированию диаграмм и обратно, без нарушения структуры текста.

1.5.2 Обзор текстовой нотации DRL

Текстовая нотация DRL (DRL Phrase Representation) предназначена для непосредственного написания текста документации с последующей генерацией пакета документов в различные форматы.

DRL является XML-языком. Каждая его конструкция представляется XML-тэгом.

Основной конструкцией DRL является конструкция, описывающая схему семейства программных продуктов. Листинг 1 иллюстрирует текстовое представление продуктов семейства:

```
<d:ProductLine name="Семейство телефонных аппаратов «Унител»">
  <d:Product name="Унител-таксофон" id="tax"/>
  <d:Product name="Унител-автоответчик" id="answer"/>
  <d:Product name="Унител-АОН" id="callerid"/>
<d:/ProductLine>
```

Листинг 1: Схема семейства продуктов

Конструкция `ProductLine` обозначает семейство продуктов, а конструкция `Product` – отдельный продукт семейства. Каждый продукт имеет имя (атрибут `name`) и идентификатор (атрибут `id`) для ссылок.

Информационный продукт (ИП) – документ или набор документов, который является самостоятельным результатом процесса разработки документации. Документация продукта из семейства может состоять из нескольких ИП. Структура ИП задается с помощью диаграммы вариативности. В DocLine диаграмма вариативности основана концепции Feature Diagrams, описанной выше. Концептом диаграммы является ИП, возможностями ИЭ (информационный элемент – фрагмент документации, предназначенный для использования в одном или нескольких ИП).

Текстовая DRL-спецификация документации семейства представлена в Листинге 2. Документация семейства указывает на то, какие информационные продукты могут входить в семейство и из каких информационных элементов состоят, где и какая настройка повторного использования должна быть сделана. Фактически, ИП можно понимать как шаблон документа, который после указания всех параметров и настроек будет готов к включению в документацию конечного продукта.

```
<d:DocumentationCore>
  <d:InfProduct id="userguide" name="Руководство пользователя">
    <book>
      <bookinfo>
        <title><d:DictRef dict=”main” Entry=”productname”/>
        </title>
      </bookinfo>
      <d:InfElemRefGroup id=”in_out” modifier=”OR”
name=”in_out”/>
      <d:InfElemRef id=”out_ref” infelemid=”Исходящие”
groupid=”in_out”/>
      <d:InfElemRef id=”in_ref” infelemid=”Входящие”
groupid=”in_out”/>
      <d:InfElemRef id=”aon_ref” infelemid=”AOH”
optional=”true”/>
    </book>
  </d:InfProduct>
  <d:InfProduct id="help" name="Справочная система"/>
  <d:InfElement id="AOH" name="AOH"/>
</d:DocumentationCore>
```

Листинг 2: Документация семейства

`DocumentationCore` – это ядро документации, т.е. та ее часть, которая содержит набор повторно используемых компонент. Конструкция `InfProduct` указывает на ИП, конструкция `InfElement` – на ИЭ. `InfElemRefGroup` указывает на группу ИЭ, связанных OR или XOR отношением, описанным в параметре modifier. А параметр optional в ссылке на ИЭ указывает на альтернативное включение этого ИЭ в конечный продукт документации.

Документация конечного продукта состоит из набора специализированных информационных продуктов (СИП) и описывает, каким образом нужно модифицировать документацию семейства, чтобы получить пакет документов для конкретного продукта семейства. Т.е. каждый СИП соответствует ИП в документации семейства и описывает, какие ИЭ должны войти в документацию конкретного продукта и как их настроить. Также он содержит ссылку на продукт из схемы семейства, к которому полученный документ будет относиться. Пример документации продукта представлен ниже в Листинге 3.

```
<d:ProductDocumentation productid="callerid">
  <d:FinalInfProduct id=”UserGuide1” infproductid="userguide">
    <d:Adapter infelemrefid="CallerID_aon_ref">
      <d:Insert-After nestid="Способы индикации номера">
        Также предусмотрена возможность
        звуковой индикации номера абонента.
      </d:Insert-After>
    </d:Adapter>
  </d:FinalInfProduct>
</d:ProductDocumentation>
```

Листинг 3: Пример документации продукта

Конструкция `ProductDocumentation` задает описание документации одного продукта семейства. Атрибут `productid` задаёт «привязку» к документу из схемы семейства продуктов. `FinalInfProduct` определяет СИП, идентификатор соответствующего ему ИП задан атрибутом `infproductid`. Теги `Adapter` служат для настройки ИЭ в контексте заданного продукта семейства. 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.

Таким образом, DRL-файлы документации являются XML-файлами, каждому из которых соответствует свой корневой элемент: `ProductLine`, `DocumentationCore` или `ProductDocumentation`.

1.6 Обзор задач трассировки

Существуют разные предметные области, где актуальна задача трассировки. Например, анализ и разработка бизнес-процессов, лабораторные исследования, медицина, промышленное производство, разработка ПО и прочее. В каждой предметной области трассировка имеет свое точное определение. Например, при анализе и разработке бизнес-процессов под трассировкой понимают обеспечение контроля всевозможных данных в этих процессах с целью определения увеличения/уменьшения эффективности, потерь, простоев и т.д.

В области ПО самым известным вариантом задачи трассировки является отслеживание связи требований с различными артефактами проекта (requirement traceability) с целью обеспечить в любой момент процесса разработки переход от описания требования к его реализации и тестам. Существуют так же и другие варианты задачи трассировки в области ПО: отслеживание связей моделей ПО (например, UML-моделей) с кодом, требованием и т.д., отслеживание изменений документации при эволюции ПО. В случае отслеживания связей между кодом приложения и его документацией можно выделить следующие более конкретные задачи трассировки: связь программного кода и документа с требованиями, связь кода с документом, описывающим архитектуру приложения.

Более подробный обзор задач трассировки можно найти в статье [1].

1.7 Управление вариативностью в среде WebMLDoc

Подход к управлению вариативностью семейства программных продуктов, создаваемого на базе *обобщенного проекта WebRatio* (исходный проект в среде разработки WebRatio, содержащий общие активы для разрабатываемого семейства программных продуктов) описан в работе [4].

Точки вариативности добавляются в гипертекстовую модель обобщенного проекта WebRatio. Модели данных, запросов и контента являются общими активами для разрабатываемого семейства.

Профили сайта, области, страницы и контент-модули гипертекстовой модели языка WebML содержатся друг в друге. В подходе [4] на основе гипертекстовой модели строится дерево, отображающее включение элементов интерфейса друг в друга. В этом дереве профили сайта являются потомками корня дерева, который не имеет аналога в гипертекстовой модели. Детьми профилей сайта являются области и страницы, содержащиеся в этих профилях в гипертекстовой модели. Детьми областей являются страницы, содержащиеся в этих областях. В [4] в дереве, построенном по гипертекстовой модели, не отображаются контент-модули.

Построение модели вариативности в [4] основано на концепции Feature Diagrams (диаграмм вариативности). Добавление точек вариативности в гипертекстовую модель состоит в определении типа связей между элементами дерева, принцип построения

которого описан выше. Так же как и в нотации диаграмм возможностей в диаграмме вариативности [4] связь может быть определена как:

- обязательная для включения,
- необязательная для включения.

И также как в нотации диаграмм возможностей, элементы гипертекстовой модели, имеющие одного родителя, могут группироваться в:

- отношения строго (XOR) альтернативного включения,
- нестрогого (OR) альтернативного включения.

Кроме того, в [4] существует еще один тип отношений: потомков одного родителя можно объединять в узлы (Nodes). При таком отношении включается или не включается в конечный продукт все поддерево.

В подходе [4] существует еще одно различие по сравнению с классической нотацией диаграмм возможностей. При создании связей типа OR, XOR или Node, сами эти точки вариативности становятся возможностями, которые могут включаться в отношения типа OR, XOR, Node или обязательных к включению, но такие возможности нельзя соединять отношением необязательного включения.

В [4] точки вариативности разрешено добавлять на трех уровнях гипертекстовой модели: профилей сайтов, областей и страниц. От добавления точек вариативности на уровне контент-модулей было принято решение отказаться, так как контент-модули связаны между собой операциями, и создать корректное добавление точек вариативности только на основе нотации диаграмм вариативности уже невозможно.

Процесс создания конкретного продукта по модели вариативности в [4] состоит из 2 действий:

- разрешение модели вариативности, построенной по обобщенному проекту WebRatio;
- генерация нового проекта WebRatio на основе разрешения модели вариативности (такой проект будем называть *конечным проектом WebRatio*).

Глава 2. Задача «привязки» документации и трассировки изменений

Данная работа основана на работе [3], цель которой состояла в разработке подхода, позволяющего по изменениям в пользовательском интерфейсе Web-приложения, создаваемого в среде WebRatio, автоматически определить список разделов пользовательской документации в DocLine, требующих корректировки или проверки, и предоставить этот список техническому писателю для внесения «ручных» правок.

Данная работа рассматривает процесс разработки семейства программных продуктов, а не отдельного продукта как в [3]. В этом контексте меняются задачи создания «привязки» документации к коду и трассировки изменений документации по изменениям кода ПО. Подход, разрабатываемый в данной работе, создает «привязку» документации не непосредственно к гипертекстовой модели приложения, а к модели вариативности семейства, создаваемой по гипертекстовой модели. Трассировка пользовательской документации также основана не на отслеживании изменений гипертекстовой модели, а отслеживании изменений модели вариативности.

2.1 Создание «привязки» документации к модели вариативности

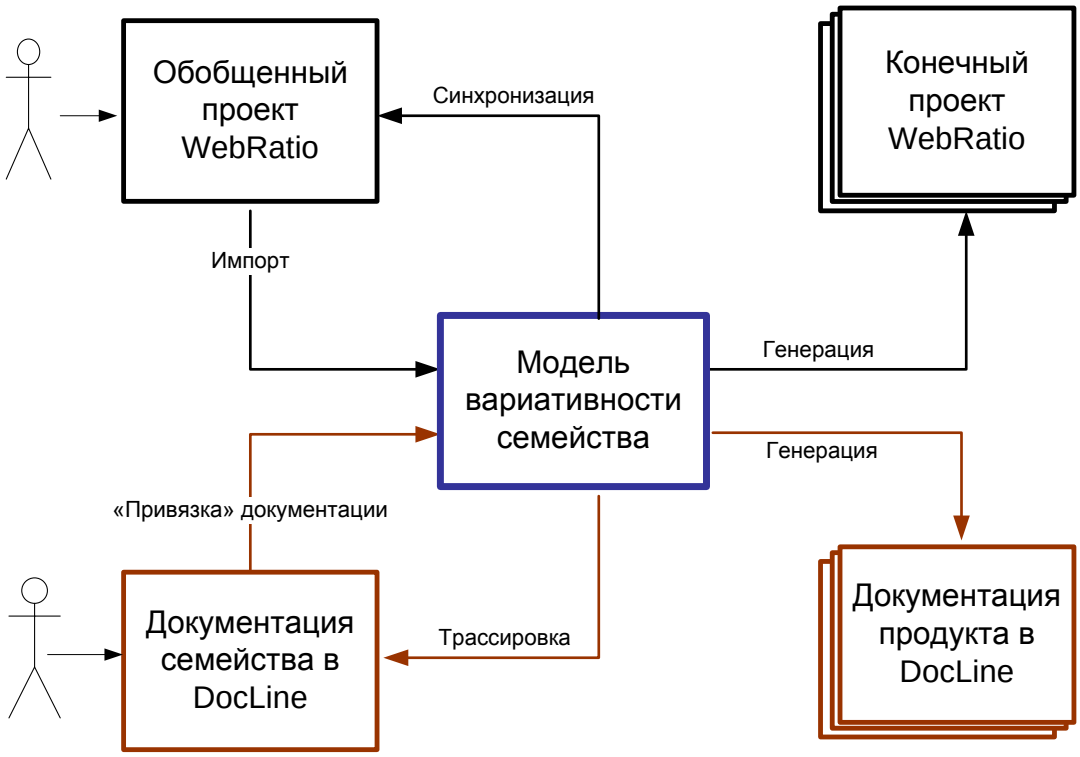

Рисунок 3: Схема общего подхода

На Рисунке 3 изображена общая схема работы подхода, разрабатываемого совместно с [4]. Сценарий использования подхода предполагает работу с обобщенным проектом WebRatio и его документацией на языке DRL. Данный подход не определяет, как создавать документацию обобщенного проекта WebRatio, ее существование рассматривается как данность. Однако подход накладывает некоторые ограничения на нее, о которых будет сказано ниже.

По гипертекстовой модели исходного проекта WebRatio строится модель вариативности. Способ построения модели вариативности в общих чертах описан в обзоре.

Как и в работе [3] в данном подходе предлагается связывать элементы пользовательского интерфейса и информационные элементы документации, определяемые в конструкции InfElement языка DRL. В качестве элементов интерфейса выбраны профили сайта, области и страницы. Контент-модули не принимаются в

рассмотрение, так как на уровне контент-модулей вариативность не задается, и в модели вариативности семейства эти структурные единицы не отображаются (подробнее в [4]).

Однако, как правило, пользовательская документация оперирует именно контент-модулями. То есть фрагменты документации описывают непосредственно контент-модули. Поскольку контент-модули не представлены в модели вариативности семейства, данный подход предлагает прикреплять элементы документации, описывающие контент-модуль, к странице, на которой расположен контент-модуль. Такое решение не нарушает корректности связи между документацией и элементами интерфейса, так как каждый контент модуль закреплен за определенной страницей. А изменения страницы, в том числе и ее содержания, отслеживается методом трассировки, который будет описан ниже.

Элементы документации, описывающие страницы, области и профили сайта, прикрепляются к соответствующим элементам модели вариативности.

Данный подход позволяет создавать связи элементов документации с корнем модели вариативности и с элементами диаграммы вариативности типа Node. К корню диаграммы вариативности прикрепляются «общие» разделы документации, не относящиеся ни к какому определенному элементу интерфейса (например, введение). К элементам типа Node – разделы документации, относящиеся ко всем элементам, включаемым в Node.

Так как может быть несколько информационных элементов, описывающих работу с одним элементом интерфейса, к элементам интерфейса разрешено прикреплять более одного информационного элемента. Также разрешено не прикреплять ни одного информационного элемента, в случае если нет конкретных разделов документации, описывающих данный элемент интерфейса.

Пользовательская документация, описывающая сценарии работы с интерфейсом, скорее всего, содержит в одном информационном элементе описание нескольких элементов интерфейса. Такие информационные элементы разрешено прикреплять к нескольким элементам интерфейса.

Таким образом, несколько фрагментов документации может быть прикреплено к одному элементу интерфейса, и один фрагмент документации может быть прикреплен к нескольким элементам интерфейса.

2.2 Ограничения на документацию DocLine

В методе DocLine существует своя модель вариативности, отличная от модели вариативности, представленной в [4]. Она задается во время создания документации к обобщенному проекту WebRatio. Непосредственно в тексте вариативность в DocLine определяется в конструкции информационных продуктов InfProduct и так же как и в [4] основана на нотации Feature Diagrams. Вариативность в DocLine определяется на информационных элементах, которые в подходе WebMLDoc прикрепляются к элементам диаграммы вариативности семейства программных продуктов. Так как информационные элементы и элементы диаграммы вариативности [4] имеют тип связи многие ко многим, синхронизация этих двух моделей вариативности для разрешения этих диаграмм одним действием является отдельной и объемной задачей.

В данной работе накладываются ограничения на документацию обобщенного проекта WebRatio для одновременного разрешения вариативности в DocLine и WebMLDoc. Во-первых, документация будет состоять из одного информационного продукта. То есть в публикации документации конечного проекта WebRatio будет только один документ – «Руководство пользователя». Во-вторых, в конструкции InfProduct все информационные элементы, входящие в информационный продукт должны быть помечены, как опциональные, т.е. необязательные к включению в документацию конечного продукта. Элементы интерфейса могут содержать другие элементы интерфейса, которые также должны быть опциональными. Данный подход накладывает следующие условие: при решении о включении в документацию информационного элемента, содержащегося в других, автоматически включаются все элементы его содержащие. Это ограничение и условие дают возможность при порождении конечного продукта семейства, включать в схему документации конечного продукта, задаваемую консрукциями FinalInfProduct, любую выборку из линейного списка всех информационных элементов, описанных в проекте DocLine. В общем же случае выборка из списка информационных элементов, включаемых в документацию конечного продукта, не может быть какой угодно и должна подчиняться правилам, описанным в шаблоне документа (конструкция InfProduct).

2.3 Создание документации конечных продуктов

При порождении конечных продуктов семейства фактически происходит выборка элементов интерфейса из гипертекстовой модели обобщенного проекта WebRatio. В

документацию конечного продукта должны включаться только фрагменты документации, связанные с выбранными элементами интерфейса, корнем диаграммы вариативности и элементами диаграммы вариативности типа Node, в том случае, если он был включен в конечный продукт.

Второе ограничение на документацию позволяет включать любое подмножество информационных элементов, описанных в документации к исходному проекту WebRatio. А схема документа задается в конструкции InfProduct на этапе создания документации семейства, не рассматриваемом в данном подходе.

2.4 Трассировка изменений модели вариативности

В ходе разработки семейства программных продуктов гипертекстовая модель обобщенного проекта WebRatio может меняться: в нее могут добавляться и удаляться элементы интерфейса. Также могут меняться названия элементов интерфейса и их атрибуты. Эти изменения отслеживаются и вносятся в модель вариативности семейства программных продуктов.

Трассировка должна предоставлять техническому писателю список разделов документации, требующих корректировки, после удаления, добавления или модификации элемента интерфейса обобщенного проекта WebRatio.

В рамках данной работы разработан метод трассировки для случаев добавления, удаления элементов интерфейса, а также их модификации. Он основан на синхронизации модели вариативности и гипертекстовой модели обобщенного проекта WebRatio.

При удалении элемента интерфейса в гипертекстовой модели, он удаляется из модели вариативности. Техническому писателю выдается информация о том, что определенный элемент интерфейса был удален и прикрепленные к нему разделы документации требуют корректировки.

При добавлении элемента интерфейса в гипертекстовую модель, он добавляется в модель вариативности. Техническому писателю выдается информация о том, что был добавлен новый элемент интерфейса. Во-первых, возможно нужно прикрепить к этому элементу разделы документации. Во-вторых, возможно, что таких разделов еще нет, и технический писатель должен их создать.

Синхронизация гипертекстовой модели обобщенного проекта WebRatio и модели вариативности отслеживает модификацию элементов интерфейса. В таких случаях техническому писателю выдается сообщение о том, что элемент интерфейса был модифицирован, и список разделов документации, прикрепленных к нему, так как вероятно, их необходимо модифицировать.

Глава 3. Редактор WebMLDoc

Помимо разработанного подхода, в рамках данной дипломной работы и дипломной работы [4] была выполнена его пилотная реализация. На основе технологии Eclipse GMF был создан редактор WebMLDoc, предоставляющий графические средства для работы с диаграммой вариативности и «привязкой» разделов документации к модели вариативности.

Создание отдельного редактора обусловлено тем, что технология WebRatio не имеет открытых средств для создания надстроек.

3.1 Сценарий работы редактора

Редактор WebMLDoc поддерживает процесс разработки семейства программных продуктов, сопровождения документации семейства, формирования интерфейса конечного продукта и пользовательской документации конечного продукта.

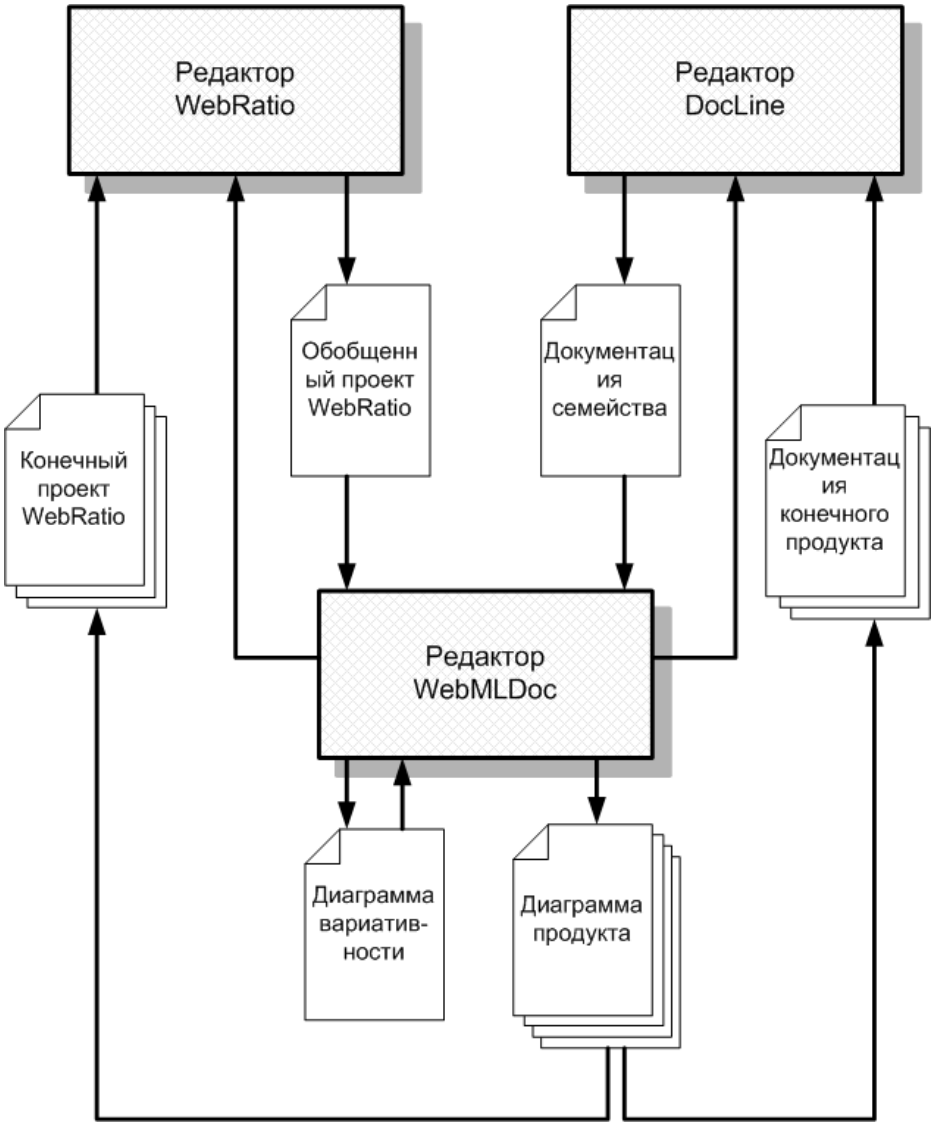

Рисунок 4: Схема работы редактора

На Рисунке 4 представлена схема работы редактора, разрабатываемого совместно с [4].

Ниже представлены основные шаги работы с редактором.

1. Разработка проекта WebRatio, который впоследствии станет обобщенным проектом WebRatio.
2. Разработка соответствующей проекту WebRatio пользовательской документации. На этом этапе создается информационный продукта (InfProduct), который будет схемой документации конечных продуктов, и информационные элементы, содержащиеся в этом информационном продукте.

3. Создание нового проекта в редакторе и импорт в систему гипертекстовой модели проекта WebRatio, разрабатываемого на этапе 1, и его пользовательской документации, разрабатываемой на этапе 2. На этом шаге проект WebRatio становится обобщенным проектом WebRatio в рамках нашего подхода. Импорт документации состоит в формировании списка информационных элементов и записи его в систему.
4. Определение точек вариативности.
5. «Привязка» информационных элементов документации к элементам модели вариативности.
6. Синхронизация изменений модели вариативности по изменениям в гипертекстовой модели. Определяется разница между старой гипертекстовой моделью и новой в рамках двух операций: добавление и удавление элемента интерфейса.
7. Сразу после синхронизации изменений модели вариативности, с помощью созданной связи между элементами модели и информационными элементами документации, определяется список измененных элементов интерфейса (добавленных или удаленных). В журнал заносится этот список. В случае удаления элемента интерфейса, в журнал записывается список информационных элементов, связанных с элементом интерфейса. В случае добавления элемента интерфейса, технический писатель должен учесть его появление и, возможно, создать новый информационный элемент в документации или откорректировать существующие информационные элементы, описывающие сценарии работы с интерфейсом.
8. На основе информации, предоставляемой на шаге 7, технический писатель меняет документацию. Затем обновляется список информационных элементов.
9. Создание конкретных продуктов семейства. На основании разрешения точек вариативности в редакторе создается новая диаграмма, являющаяся диаграммой конечного продукта, и формируется список элементов документации. В список включаются только информационные элементы, связанные с элементами созданной диаграммы. И диаграмма и список запрещены к редактированию.
10. По диаграмме, построенной на шаге 9, генерируется конечный проект WebRatio. По конечному проекту WebRatio генерируется конечное приложение.

11. По списку элементов документации, генерируется специализированный информационный продукт на языке DRL (FinalInfProduct) и в схему семейства программных продуктов в DocLine (ProductLine) добавляется информация о новом продукте семейства. По схеме семейства ПО, информационному продукту, информационным элементам и специализированному информационному элементу генерируется документация конечного продукта.

Шаги 1 и 2 могут выполняться последовательно или параллельно. Шаги 4 и 5 всегда идут один за другим, но могут повторяться сколь угодно много. Неопределенно, какой из этапов начинается раньше. Шаги 6 и 7 всегда идут последовательно. После шага 7 есть возможность вернуться на шаг 4 и/или 5. Шаг 8 может производиться параллельно с шагами 4 и 5 или самостоятельно. Шаг 9 производится после завершения всех предыдущих этапов. Шаги 10 и 11 происходят одновременно.

Данная работа реализует шаги 5, 6, 7, и 11, а также частично 3 и 9.

3.2 Работа с документацией в редакторе WebMLDoc

Работа с редактором WebMLDoc начинается с создания нового проекта специального типа. При создании указывается путь к WebRatio проекту и путь к папке с документацией. На основе этой информации происходит генерация «заготовки» диаграммы вариативности и формирование списка информационных элементов документации.

3.2.1 Создание «привязки» в WebMLDoc

Для создания «привязки» информационных элементов документации к элементам диаграммы вариативности пользователь выбирает один или несколько элементов диаграммы, затем в выпадающем меню выбирает опцию Tag documentation topic. После этого открывается мастер, всегда содержащий полный линейный список информационных элементов документации. Пользователь отмечает информационные элементы, которые нужно прикрепить. После завершения работы мастера, выбранные информационные элементы отображаются на диаграмме вариативности, как изображено на Рисунке 5.

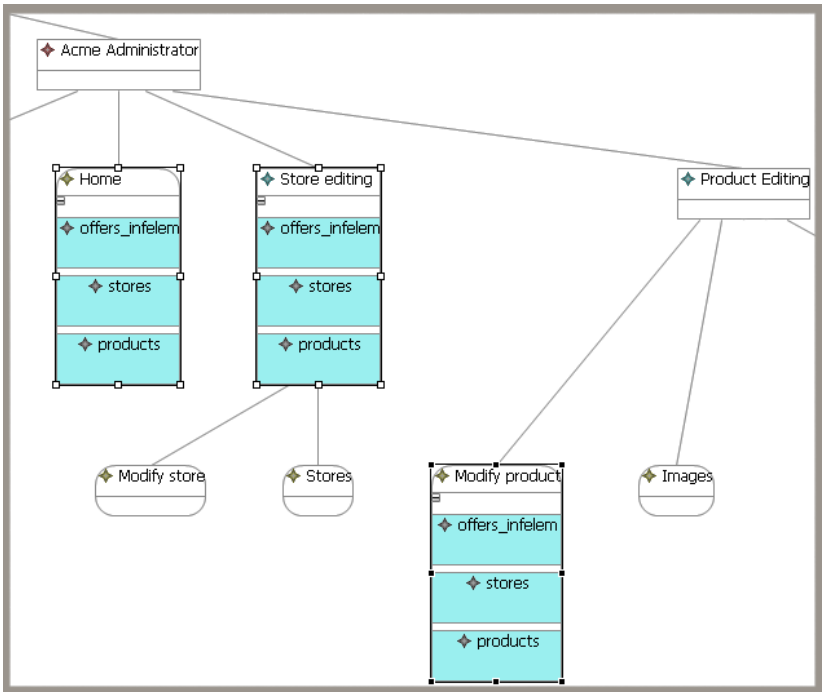

Рисунок 5: «Привязка» элементов документации в редакторе WebMLDoc

Если пользователю необходимо удалить связь раздела документации и элемента модели вариативности, то для этого достаточно в элементе модели выбрать топик документации и удалить.

3.2.2 Генерация документации конечного продукта

В редакторе WebMLDoc создан мастер для разрешения вариативности программных продуктов. Работая с этим мастером, пользователь напрямую не взаимодействует с документацией. По завершении работы мастера, в директории проекта DocLine создается файл с именем конечного продукта семейства (определяется при работе с мастером разрешения вариативности) и расширением drl. Пример такого файла приведен в Листинге 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<d:ProductDocumentation>
  <d:FinalInfProduct id="AcmeLightVersion"
infproductid="AcmeLightVersion">
    <d:Adapter infelemrefid="manager_login"/>
    <d:Adapter infelemrefid="manager_logout"/>
    <d:Adapter infelemrefid="check_news"/>
    …
  </d:FinalInfProduct>
</d:ProductDocumentation>
```

Листинг 4: Пример специализированного информационного элемента, сгенерированного редактором WebMLDoc

Также в DRL-схему продуктов семейства добавляется запись о конечном продукте.

```
<d:ProductLine name="Product Line of Acme Web Stores">
  <d:Product name="AcmeBase" id=" AcmeBase"/>
  <d:Product name="AcmeLightVersion" id="AcmeLightVersion"/>
<d:/ProductLine>
```

Листинг 5: Пример добавления записи о продукте в схему продуктов семейства

3.2.3 Реализация трассировки

Как отмечалось выше, трассировка изменений происходит после синхронизации модели вариативности и гипертекстовой модели. В редакторе WebMLDoc синхронизация происходит при каждом открытии проекта и по требованию пользователя.

Синхронизация производится путем считывания и сравнения дерева вариативной модели и дерева гипертекстовой модели обобщенного проекта WebRatio.

При считывании дерева гипертекстовой модели если элемент интерфейса изменился после последнего обновления модели вариативности, то в журнал пишется информация о том, какой элемент интерфейса был изменен и выдается список соответствующих ему разделов документации.

Далее дерево модели вариативности сравнивается с деревом гипертекстовой модели. Если в дереве вариативности, есть элемент, которого нет в дереве гипертекстовой модели, значит, он был удален из гипертекстовой модели, и редактор удаляет его из дерева модели

вариативности. При этом пишется информация в журнал (log) для технического писателя о том, какой элемент интерфейса был удален и выдается список разделов документации, прикрепленный к этому элементу.

На следующем шаге происходит сравнение дерева гипертекстовой модели и дерева модели вариативности. Если какой-то элемент есть в первом дереве, но отсутствует во втором, значит он был добавлен в гипертекстовую модель, и редактор добавляет его в дерево вариативности. При этом в журнале отображается информация о том, что этот элемент интерфейса был добавлен в модель вариативности.

3.3 Трудности реализации

В, совместном с [4], проекте изначально планировалось переиспользовать метамодель, на основе которой был реализован редактор в работе [3].

Напомним, что цель работы [3] состояла в разработке метода, позволяющего по изменениям пользовательского интерфейса проекта WebRatio, определить список разделов пользовательской документации в DocLine, требующих корректировки или проверки. Созданный в [3] редактор импортирует гипертекстовую модель WebML и отображает ее в виде максимально приближенном к отображению модели в редакторе WebRatio.

Однако метамодель [3] оказалась весьма нагруженной, и ее переиспользование представлялось нам весьма трудной задачей. В связи с этим было решено создавать редактор WebMLDoc, основываясь на метамодели, описанной в работе [2]. В работе [2] так же производится импорт гипертекстовой модели WebML, однако в реализованном в [2] редакторе она отображается уже в виде древовидной структуры.

Глава 4. Пример

Разработанный совместно с [4] подход был опробован на Web-приложении AcmeExpanded. AcmeExpanded является расширением приложения Acme, которое входит в стандартную поставку WebRatio.

Acme – это Интернет-магазин, поддерживающий 2 профиля сайта: профиль пользователя (покупателя) и профиль администратора сайта, который может редактировать информацию о товарах, предложениях и магазинах сети.

Фрагмент гипертекстовой модели Web-приложения AcmeExpanded приведен в Рисунке 6.

В AcmeExpanded добавлены дополнительные элементы интерфейса – страницы для просмотра новостей, просмотра каталога товаров, сортированного по производителю и некоторые другие. Ряд добавленных элементов интерфейса является избыточным, то есть они не могут одновременно существовать в одном конечном продукте.

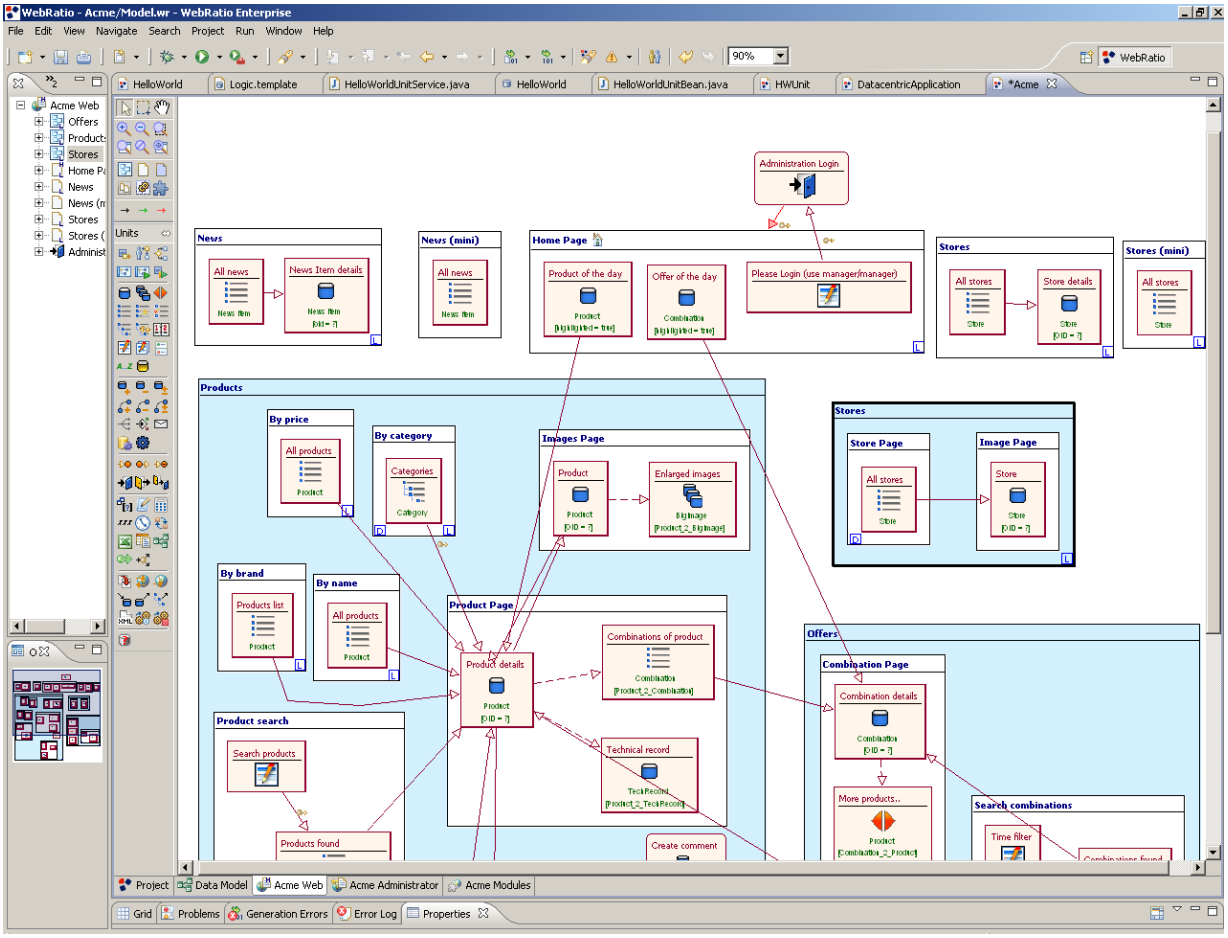

Рисунок 6: Фрагмент гипертекстовой модели AcmeExpanded в редакторе WebRatio

Например, в профиле сайта, изображенном на Рисунке 6, есть две страницы для просмотра новостей.

Работа с редактором WebMLDoc начинается с создания нового проекта. Для этого необходимо указать директории проекта WebRatio и DocLine, как приведено на Рисунке 7.

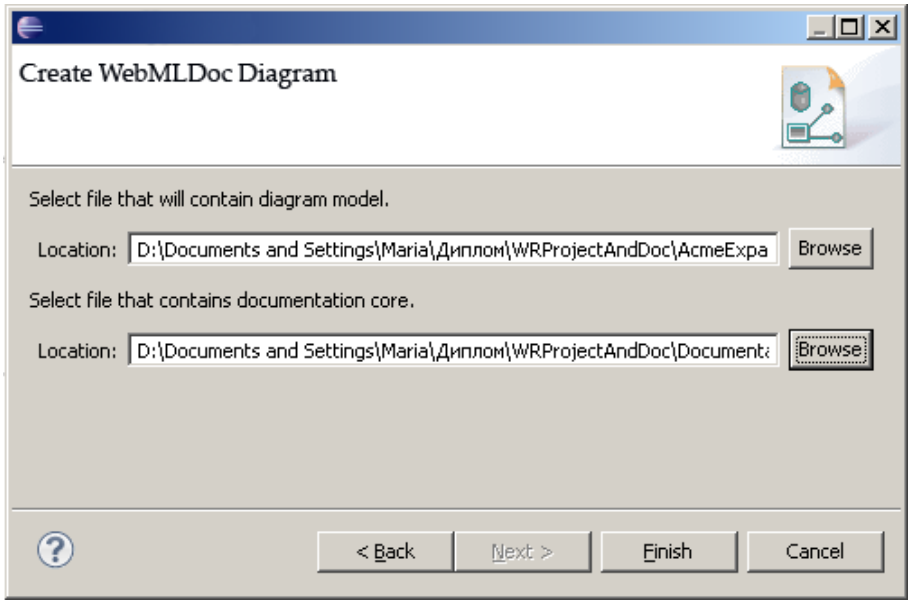

Рисунок 7: Создание нового проекта в WebMLDoc

В редактор импортируется «каркас» модели вариативности (Рисунок 8), автоматически построенный на основе гипертекстовой модели WebRatio проекта AcmeExpanded, и список информационных элементов документации. Графическое представление «каркаса» модели хранится в файле с расширением .webmldoc_diagram.

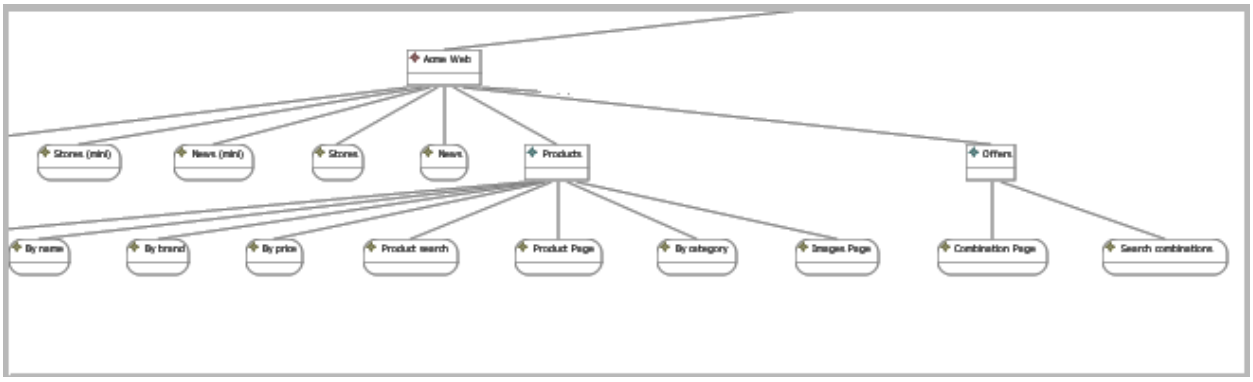

Рисунок 8: Фрагмент «каркаса» модели вариативности в редакторе WebMLDoc

Для создания вариативности в какой-либо точке «каркаса» надо выбрать узел – родителя элементов, для которых задается вариативности. Это делается с помощью пункта SetVariativity контекстного меню, вызываемого на данном узле.

После этого открывается мастер создания точки вариативности. В нем нужно отметить детей выбранного элемента диаграммы, для которых определяется вариативность, название точки вариативности и ее тип.

Мастер создания точки вариативности приведен на Рисунке 9.

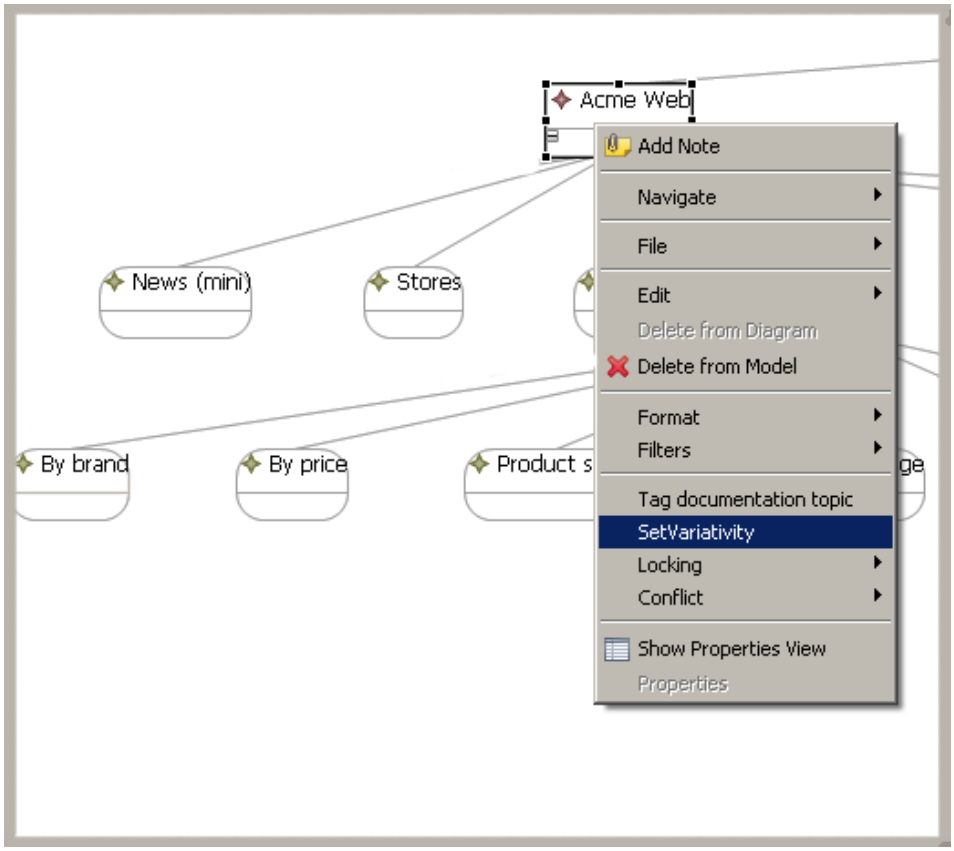

Рисунок 9: Определение параметров точки вариативности

После завершения работы мастера в диаграмме создается точка с заданными параметрами, как отображено на Рисунке 10.

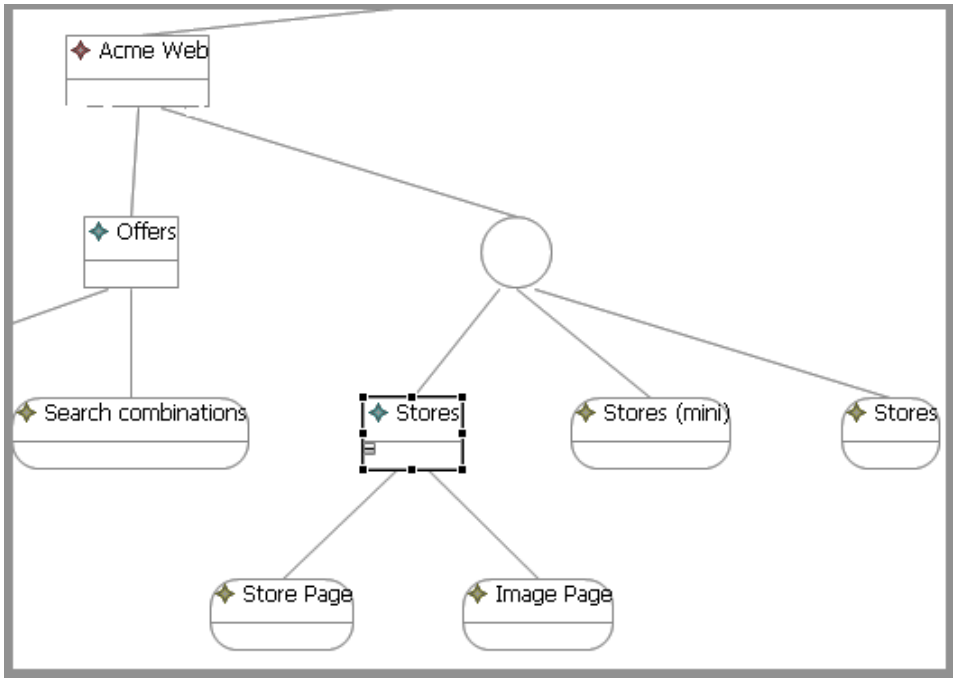

Рисунок 10: Точка вариативности

Для создания связи между элементами диаграммы и разделами документации надо выбрать один или несколько элементов диаграммы (в случае, если к ним добавляется одинаковый набор разделов документации) и выбрать из контекстного меню пункт Tag documentation topic.

После этого открывается мастер создания «привязки» документации, изображенный на Рисунке 11. В нем отображается список разделов документации. Из этого списка выбираются элементы документации, которые надо прикрепить к элементам интерфейса.

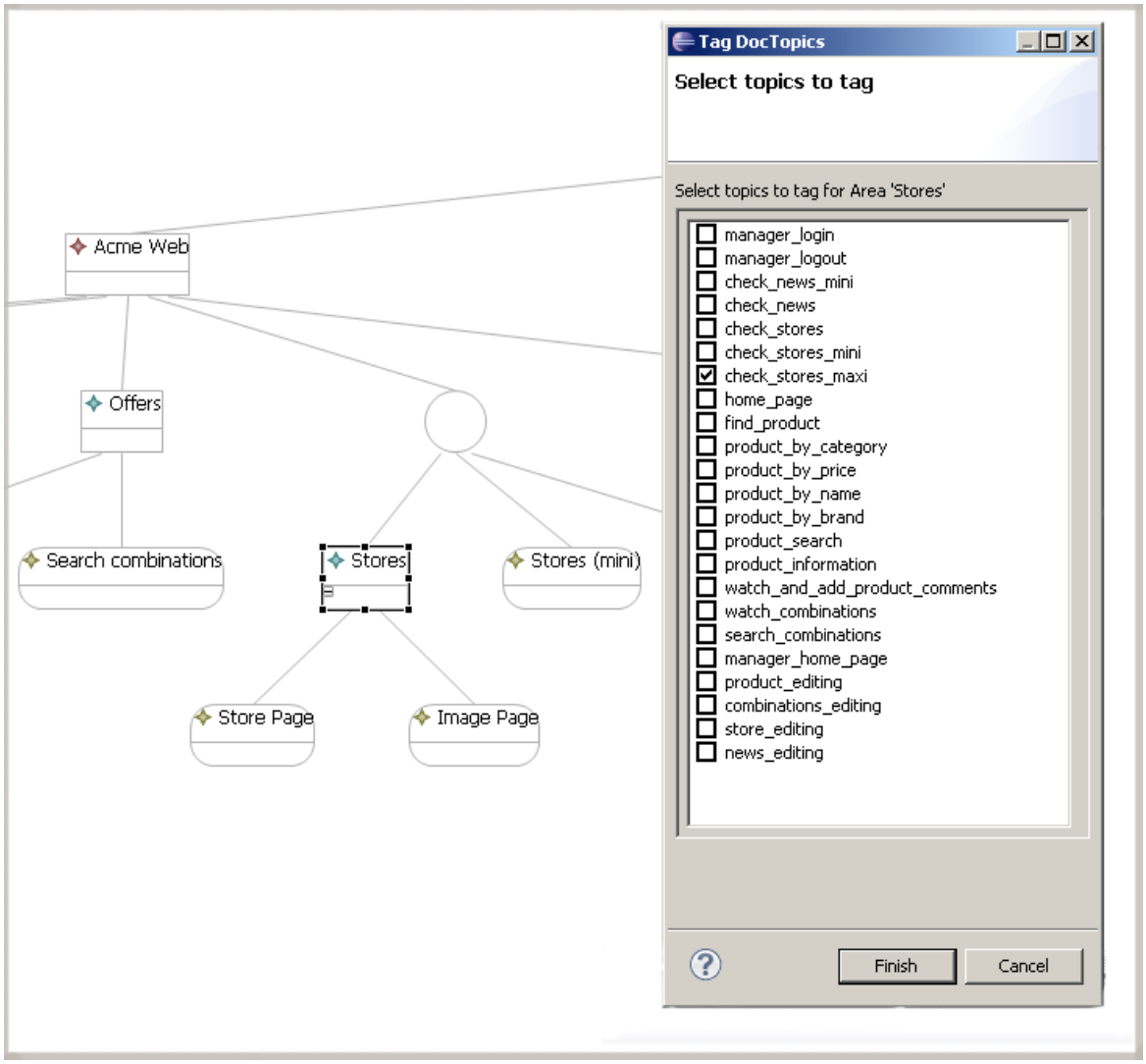

Рисунок 11: Определение набора прикрепляемых разделов документации

После завершения работы мастера в элементе диаграммы создается пометка с именем прикрепленного раздела документации. Пример создания такой поментки изображен на Рисунке 12.

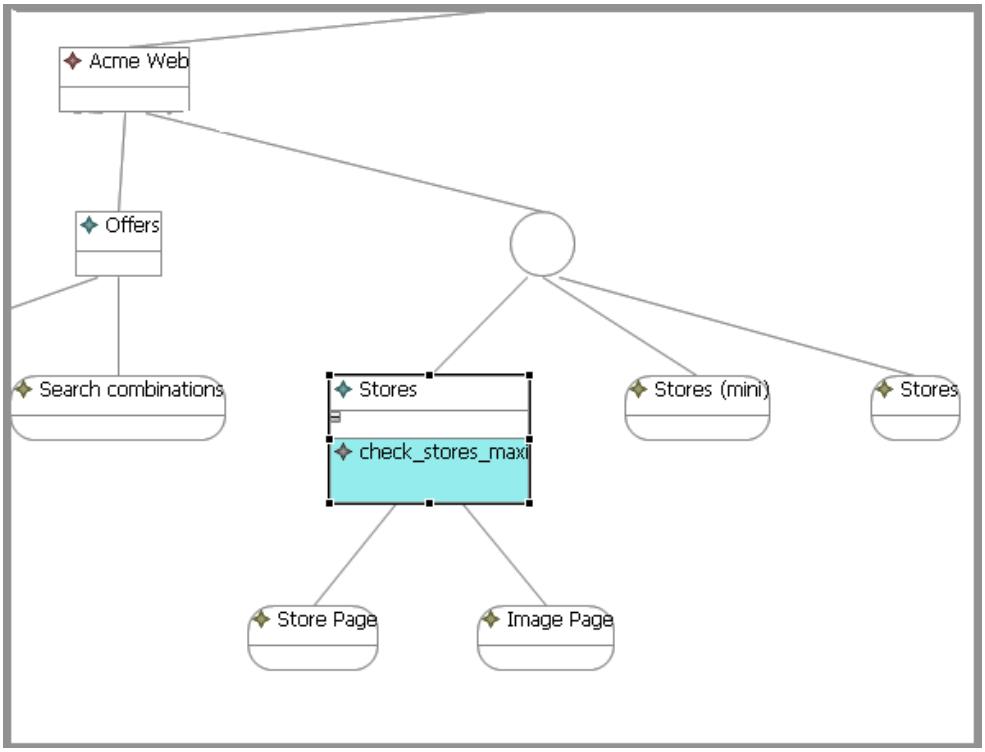

Рисунок 12: Созданная связь между областью Stores и элементом документации check_stores_maxi

На этапе редактирования диаграммы вариативности происходит синхронизация диаграммы вариативности и гипертекстовой модели обобщенного приложения WebRatio. Синхронизация выполняется при каждом открытии редактора WebMLDoc и по запросу пользователя. Сразу же после вызова синхронизации пользователю показывается журнал со списком разделов документации, требующих правки.

На Рисунке 13 изображен фрагмент диаграммы вариативности после разметки всех точек вариативности и прикрепления всех разделов документации.

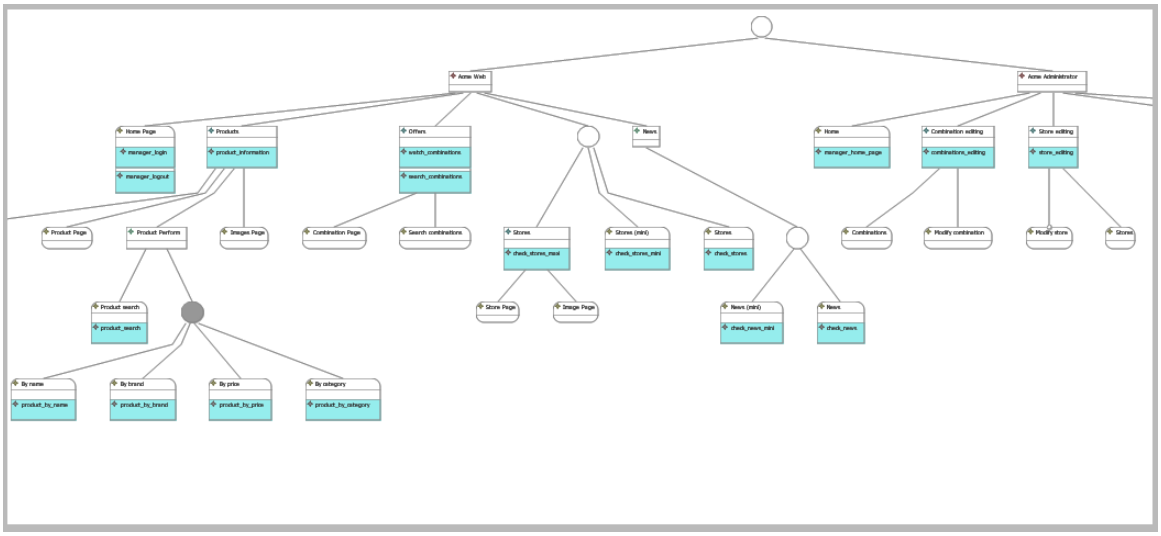

Рисунок 13: Диаграмма вариативности с прикрепленными к ней разделами документации

Для создания конкретного продукта семейства необходимо выбрать файл с расширением .webmldoc и из контекстного меню выбрать команду Add Product. Откроется мастер, содержащий древовидную структуру диаграммы. Пример такого мастера приведен на Рисунке 14. В мастере задается название создаваемого продукта семейства и разрешаются точки вариативности.

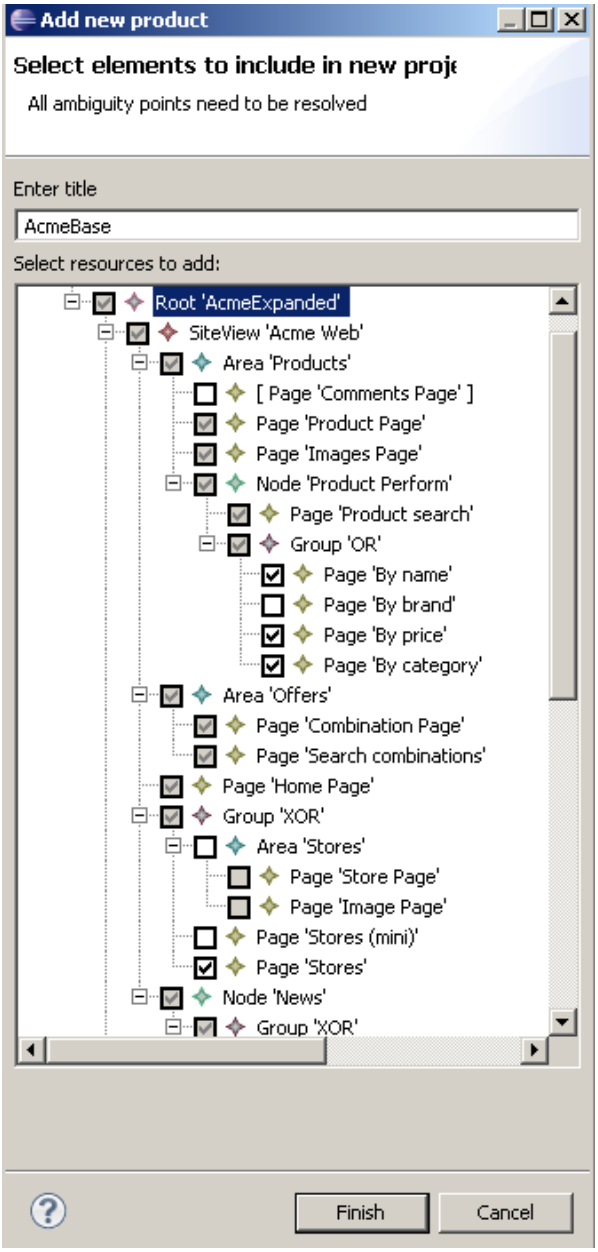

Рисунок 14: Мастер создания продукта семейства

После завершения работы мастера в ту директорию, из которой импортировался обобщенный проект WebRatio, генерируется новый проект WebRatio. В директории исходной документации создается новый .drl файл для FinalInfProduct, определяющий вхождение фрагментов документации в схему готовой документации, и в файл схемы семейства программных продуктов вносится запись о созданном продукте семейства. Примеры таких файлов приведены в Листингах 6 и 7 соответственно. Напомним, что

схема документа, хранится в файле для InfProduct, который входит в документацию семейства.

```
<?xml version="1.0" encoding="UTF-8"?>
<d:ProductDocumentation productid=""
xmlns:d="http://math.spbu.ru/drl"
xmlns="http://docbook.org/ns/docbook">
<d:ProductDocumentation id="UserManual_AcmeBase"
infproductid="UserManual">
  <d:FinalInfProduct id="AcmeBase" infproductid="AcmeBase">
    <d:Adapter infelemrefid="manager_login"/>
    <d:Adapter infelemrefid="manager_logout"/>
    <d:Adapter infelemrefid="check_news_mini"/>
    <d:Adapter infelemrefid="check_stores_maxi"/>
    <d:Adapter infelemrefid="product_by_price"/>
    <d:Adapter infelemrefid="product_search"/>
    <d:Adapter infelemrefid="product_information"/>
    <d:Adapter infelemrefid="watch_and_add_product_comments"/>
    <d:Adapter infelemrefid="watch_combinations"/>
    <d:Adapter infelemrefid="search_combinations"/>
    <d:Adapter infelemrefid="manager_home_page"/>
    <d:Adapter infelemrefid="product_editing"/>
    <d:Adapter infelemrefid="combinations_editing"/>
    <d:Adapter infelemrefid="store_editing"/>
    <d:Adapter infelemrefid="news_editing"/>
  </d:FinalInfProduct>
</d:ProductDocumentation>
```

Листинг 6: Пример файла FinalInfProduct

```
<d:ProductLine name="Acme Web-application Product Line"
xmlns:d="http://math.spbu.ru/drl">
   <d:Product name="Acme Base" id="Acme_Base"/>
</d:ProductLine>
```

Листинг 7: пример файла схемы семейства программных продуктов

В DocLine редакторе по файлам документации семейства и добавленному файлу конечного продукта генерируется .pdf файл с документацией конечного продукта.

Результаты

1. Изучен язык WebML, его гипертекстовая модель, пакет WebRatio и технология Eclipse GMF.
2. Создан метод «привязки» документации к вариативной модели семейства программных продуктов, построенной по обобщенному проекту WebRatio.
3. Реализована трассировка изменений в пользовательской документации в формате DocLine по изменениям в обобщенном проекте WebRatio.
4. Реализована поддержка работы с документацией, описывающей сценарии работы с пользовательским интерфейсом Web-приложения.
5. Реализованные средства интегрированы в единую среду разработки WebMLDoc.
6. Предложенный подход опробован на примере.

Список литературы

1. Д. В. Кознов, М. Н. Смирнов, В. А. Дорохов, К. Ю. Романовский. WEBMLDoc. Подход к автоматическому отслеживанию изменений в пользовательской документации Web-приложений. Вестник СПбГУ, Сер. 10, 2011. Вып. 3. С. 1-18.
2. А.М. Голубев. Расширение языка WebML средствами поддержки вариативности. Дипломная работа. СПбГУ, мат.-мех. факультет, 2010. 20 с.
3. В. А. Дорохов. Автоматизированная поддержка пользовательской документации Web-приложений, разрабатываемых в среде WebRatio. Дипломная работа. СПбГУ, мат.-мех. факультет, 2010. 32 с.
4. В. Г. Головина. Разработка программного продукта в задаче одновременной разработки продукта и его документации на основе семейств. Дипломная работа. СПбГУ, мат.-мех. факультет, 2011. 43 с.
5. Paul C. Clements, Linda M. Northrop. Software Product Lines: Practices and Patterns. 2001. 576 p.
6. Linda M. Northrop. Software Product Line Essentials. http://www.sei.cmu.edu/library/assets/spl-essentials.pdf.
7. 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, 2006. С. 191-218.
8. DocLine project site http://www.math.spbu.ru/user/kromanovsky/docline/

Глоссарий

Обобщенный проект WebRatio (General WebRatio Project) – исходный проект в среде разработки WebRatio, содержащий общие активы для разрабатываемого семейства Web-приложений. На основе этого проекта строится модель вариативности в WebMLDoc.

Конечный проект WebRatio (Customized WebRatio Project) – проект WebRatio, полученный в результате «разрешения» модели вариативности в WebMLDoc. Модель вариативности строится по обобщенному проекту WebRatio. Результат разрешения модели вариативности экспортируется в среду WebRatio из WebMLDoc, что и является конечным проектом WebRatio.

Генерация (перегенерация) – автоматическое создание «с нуля» одного актива по некоторому исходному; при повторной генерации (перегенерации) прежний сгенерированный актив удалятся и генерируется новый.

Синхронизация – автоматическое внесение точечных изменений в один актив по точечным изменениям в другом активе. Принципиально отличается от перегенерации.

«Привязка» (mapping) – создание связей между элементами двух активов.

Трассировка – автоматическое прослеживание изменений в одном активе по изменениям в другом. В отличие от синхронизации, изменения автоматически не вносятся во второй актив, а только определяется список его элементов, которые нужно изменить. Сами изменения могут вноситься «вручную».