Программная среда WebMLDoc для автоматизированного отслеживания изменений в пользовательской документации Web-приложений*

М. Н. Смирнов
smnsmn1979@gmail.com

Д. В. Кознов
dkoznov@yandex.ru

В. А. Дорохов
vadim.dorokhov@gmail.com

К. Ю. Романовский
kromanovsky@yandex.ru

В данной работе представлена программная среда WebMLDoc для автоматизированного отслеживания изменений в пользовательской документации Web-приложений, создаваемых на основе модельно-ориентированного подхода. Среда включает в себя уже существующие продукты — WebRatio, предназначенный для моделирования Web-приложений с последующей автоматической генерацией программного кода, и DocLine, предназначенный для разработки документации на основе идеи адаптивного повторного использования фрагментов текста. Пакет WebMLDoc интегрирует два вышеупомянутых продукта, позволяя связывать гипертекствую модель Web-приложений в WebRatio с разделами документации в DocLine и в дальнейшем использовать эту связь в автоматизированном режиме для ведения документации при сопровождении и эволюции Web-приложения.

*Исследование выполнено при частичной финансовой поддержке РФФИ (грант №08-01-00716).

Введение

Задача связывания кода программных приложений и документации является известной и востребованной на практике [7, 12, 13, 16]. Очевидна трудность ее решения в общем виде, поскольку документация ПО и его программный код используют разные абстракции, которые плохо отображаются друг на друга. Поэтому требуются дополнительные ограничения, позволяющие упростить задачу. В работе [2] предложен подход WebMLDoc для автоматизированного отслеживания изменений в пользовательской документации Web-приложений, создаваемых на основе модельно-ориентированного подхода. В нем используются следующие ограничения, упрощающие исходную задачу.

1. Рассматривается пользовательская документация к ПО.
2. Выбраны Web-приложения, которые удобно создавать на основе модельно-ориентированного подхода с последующей автоматической генерацией программного кода с помощью подхода WebML [14] и продукта WebRatio [6]. Это позволяет:
   - связывать документацию не с программным кодом, а с его моделью, что проще и удобнее;
   - использовать гипертекстовую модель для построения такой связи, которая описывает пользовательский интерфейс приложения и поэтому близка к пользовательской документации (известно, что последняя является «интерфейсоподобной»).
3. Выбран XML-язык разработки документации DRL [4] и пакет DocLine [3], позволяющие удобно связывать разделы документации с внешними средствами.

Таким образом, предлагаемое в статье программное решение интегрирует пакеты WebRatio и DocLine, что позволяет «раскрашивать» гипертекстовую модель WebRatio «привязкой» к разделам документации. В статье представлены мотивации по созданию отдельного редактора, требования к нему, архитектура, алгоритмы работы, решение задачи целостности информации в WebRatio и WebMLDoc. Представлен также пример работы всей среды.

1. Обзор используемых технологий

1.1. Язык WebML и технология WebRatio

Язык WebML [14] позволяет разрабатывать Web-приложения с применением модельно-ориентированного подхода. Для создания отдельного приложения необходимо спроектировать несколько его моделей, по которым будет автоматически сгенерирован исходный код приложения, и каждая из этих моделей описывает характеристики приложения с определенной точки зрения.

Модель данных (data model) позволяет описывать структуру базы данных приложения и является вариантом классической модели «сущность-связь», используемой во многих подходах и программных средствах.

Гипертекстовая модель (hypertext model) служит для описания схемы интерфейса Web-приложения. Самым верхним элементом является *профиль сайта (siteviews*) например, профиль администратора, неавторизованного пользователя, авторизованного пользователя. Приложение может состоять из нескольких профилей, а те, в свою очередь, состоят из областей и страниц. *Область (area)* — это группа страниц, объединенных общим назначением (например, работа с товарами, техническая поддержка и т. д.). Также как и страницы, области могут быть ориентировочными (landmark) — ссылки на такие области будут присутствовать на всех страницах профиля сайта. Область включает в себя набор *страниц (pages)*, которые соответствуют обычным страницам Web-приложения, а также транзакций и операций. Страницы, в свою очередь, состоят из *контент-модулей (content units)*, которые являются атомарными элементами гипертекстовой модели и используются для визуального представления информации, описанной в модели данных. В WebML существуют разные типы контент-модулей (DataUnit, IndexUnit, ScrollUnit, EntryUnit и многие другие), соответствующие различным функциональным блокам пользовательского интерфейса. Стоит отметить, что атрибуты внешнего вида интерфейса (т. е. цвета, конкретный вид отображаемых элементов управления, их размеры и пр.) задаются с помощью стилей и различных свойств и непосредственно в гипертекстовую модель не попадают. Навигация

между контент-модулями и страницами описывается с помощью *связей (links)*. Связи могут передавать данные и/или поток управления в приложении. В отдельный вид модельных сущностей выделяют транспортные связи, которые используются для передачи только информации, по таким связям нельзя осуществить переход (передачу управления).

Модель управления контентом (content management model) расширяет гипертекстовую модель дополнительными конструкциями — операциями и транзакциями. *Операции (operation units)* используются для обозначения выполнения какого-либо процесса или его отдельного шага при переходе по связи. Это может быть обработка данных (добавить, удалить, модифицировать некоторый фрагмент базы данных), вызов внешнего сервиса и т. д. Для реализации многовариантных переходов по результатам выполнения операции используются так называемые OKLink- и KOLink-связи, соответствующие успешному и неуспешному завершению операции. *Транзакция (transaction)* объединяет в себя набор операций, который обладает всеми свойствами обычной транзакций (ACID — Atomicity, Consistency, Isolation, Durability).

Модель запросов (derivation model) расширяет модель данных вычислимыми конструкциями и является аналогом представлений (view) в языке SQL.

Язык WebML реализован в пакете WebRatio [6], который позволяет по моделям WebML генерировать исходный код приложения с использованием стандартов JSP, HTML и XHTML.

Основным достоинством языка WebML для рассматриваемой задачи является то, что весь целевой код приложения генерируется автоматически по моделям и не требует «ручных» доработок. То есть модели полностью отражают функциональность программного продукта, и работа с моделями, а не с исходным кодом, не приводит к потерям информации.

Важность гипертекстовой модели (далее мы будем объединять ее с модель управлением контентом) заключается в том, что она является основной при генерации программного кода Web-приложения.

1.2. Язык DRL и технология DocLine

Для проектирования и разработки документации с акцентом на повторное использование на кафедре системного программирования математико-механического факультета СПбГУ был разработан метод DocLine [3], включающий в себя оригинальный язык разработки документации DRL [4], процесс разработки документации, а также инструментальный пакет. Данный метод охватывает весь жизненный цикл разработки документации от проектирования до публикации итоговых документов и поддерживает плановое адаптивное повторное использование документации.

Язык DRL (Document Reuse Language) имеет две нотации — графическую (DRL/GR) и текстовую (DRL/PR). Графическое представление служит для проектирования структуры повторного использования документации. Текстовое представление позволяет описать в виде XML-представления варианты конфигурирования повторно используемых компонент и конкретные конфигурации для порождения конечных документов.

Внутреннее представление документа на языке DRL реализовано в виде специального XML-формата. Язык DRL поддерживает модульность документов. В итоге оказывается возможным легко разбирать структуру документа и извлекать из нее разделы документа.

1.3. Виды пользовательской документации

Фактически, существуют два следующих основных принципа организации пользовательской документации ПО [9, 1].

1. Описание сценариев работы с приложением для выполнения определенных функций, например, открыть файл, изменить пароль. Таким образом описываются последовательности выбора пользователем определенных элементов интерфейса, что обеспечивает выполнение системой той функции, которая нужна пользователю.
2. Непосредственное описание пользовательского интерфейса приложения — всех рабочих областей, окон, элементов меню и т. д. Каждый элемент описывается подробно, полностью, включая входящие в него другие элементы.

В обоих случаях документация опирается на элементы пользовательского интерфейса приложений, так что модель ПО, описывающая приложение в терминах интерфейса ПО, максимально точно соответствует структуре пользовательской документации и находится с ней на близком уровне абстракции.

2. Подход WebMLDoc

Пользовательская документация часто создается параллельно с разработкой самого приложения, кроме того, изменения исходного кода приложения (добавление новых функциональных возможностей, изменение пользовательского интерфейса и пр.) требуют внесения соответствующих изменений и в пользовательскую документацию. Например, возможна ситуация, когда изменение имени продукта или его версии может потребовать внесения десятков исправлений в документации продукта. Нашей задачей является разработка подхода, реализующего трассировку изменений программного продукта и его пользовательской документации. При этом мы хотим автоматически определять те места в документации, которые нужно изменить при точечных изменениях ПО. Сами изменения документации предполагается выполнять «вручную», поскольку полностью автоматизировать процесс эволюции документации по изменениям в коде, очевидно, невозможно.

2.1. Проблема определения связей между кодом и документацией

Итак, если мы хотим связывать код приложения не с детальной документацией, описывающей тот же код, а с пользовательской документацией, которая имеет иной уровень абстракции, то мы сталкиваемся с проблемой наличия подходящих для трассировки абстракций в программном коде. Функциональность приложения, описанная в документации как единое целое, может быть не локализована в программном коде (и как правило, так и есть!), а рассосредоточена и при этом не оформлена в виде отдельных методов, классов и прочих. структурных конструкций используемого

языка программирования. Связывать с каким-либо разделом документации такие отдельные фрагменты кода очень неудобно, так как они могут разрастаться, мигрировать (полностью или частично) из одной части программы в другую, распадаться и т. д. Кроме того, один и тот же программный код может участвовать в реализации различной функциональности.

Прежде всего мы связываем документацию не с кодом приложения, а с его моделью. Это оказывается продуктивным, если по модели генерируется исходный код приложения, который не меняется потом «руками»[1]. Таким образом, все изменения, которые нужно вносить в приложение, вносятся в модель, а затем автоматически переносятся в исходный код программы. Однако возникает задача поиска наиболее подходящей для нашего случая модели, которая будет максимально близка к абстракциям пользовательской документации.

2.2. Достоинства гипертекстовой модели WebML

В качестве модели описания ПО WebMLDoc использует гипертекстовую модель языка WebML, которая позволяет задавать схему интерфейса и навигацию между его элементами, т. е. содержит информацию о сценариях. Кроме того, эта модель достаточно широко используется на практике при генерации программного кода для Web-приложений (точнее, для data intensive systems, т. е. различных более-мнеее сложных сайтов), в отличие от других моделей пользовательских интерфейсов, предназначенных для desktop-приложений [8].

2.3. Обзор подхода WebMLDoc

Таким образом, раздел документации может быть связан со Web-страницей, если он описывает эту страницу, либо с последовательностью контент-модулей и переходов, реализующих пользовательский сценарий (например, открытие файла), который описывается в этом разделе документации.

[1]Такой способ применения визуального моделирования классифицируется Мартином Фаулером как использование в виде языка программирования [5].

Ниже представлена схема нашего подхода.

1. Производится «привязка» конструкций гипертекстовой модели приложения на языке WebML к разделам пользовательской документации в модели DocLine.
2. После изменений гипертекстовой модели на WebML определяется набор итоговых изменений, отличающих текущую версию гипертекстовой модели от предыдущей.
3. По выявленным локальным изменениям гипертекстовой модели, а также на основе имеющейся «привязки» гипертекстовой модели и документации определяются разделы в документации, которые нужно изменить.

Дальнейшие детали подхода WebMLDoc можно найти в работе [2].

3. Инструментарий

Созданный нами инструмент называется WebMLDoc и является графическим редактором для «привязки» гипертекстовой модели WebML к разделам документации в DRL, который таким образом связывает два пакета — WebRatio и DocLine. Отдельный редактор понадобился потому, что продукт WebRatio не имеет открытых средств для создания надстроек, чтобы «раскрашивать» гипертекстовую модель прямо в редакторе гипертекстовой модели в WebMLDoc.

3.1. Обзор редактора WebMLDoc

Сформулируем исходные требования к редактору WebMLDoc:

- обеспечение визуальной схожести своей модели с гипертекстовой моделью в графическом редакторе WebRatio;
- поддержка создания, изменения и удаления связей элементов интерфейса приложения с разделами документации в DocLine;
- поддержка циклической разработки, т. е. при добавлении новых элементов гипертекстовой модели не генерировать «с нуля», а «достраивать» модель WebMLDoc.

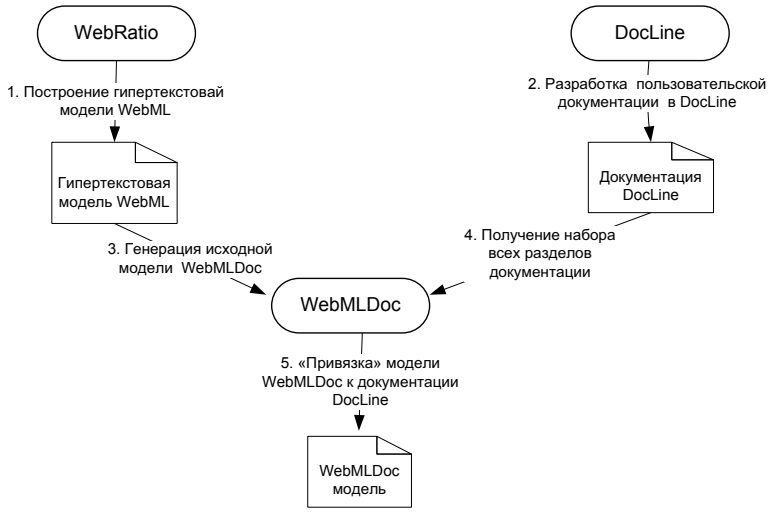

Рис. 1. Построение модели WebMLDoc и ее «привязка» к пользовательской документации

На рис. 1 представлена схема работы с WebMLDoc на этапе создания «привязки» гипертекстовой модели к документации. На рисунке овалами обозначены программные пакеты, прямоугольниками с завернутым правым уголком — результирующие документы, а стрелками — шаги процесса работы. Рассмотрим эти шаги более детально.

Построение гипертекстовой модели Web-приложения происходит в продукте WebRatio и является первым шагом нашего процесса. После этого (или параллельно с этим) разрабатывается пользовательская документация в продукте DocLine, что обозначено как шаг 2 на рис. 1. Необходимо отметить, что перед началом работы над пользовательской документацией гипертекстовая модель должна быть в общих чертах готова, так как документация описывает во многом именно эту модель. Далее (шаг 3) выполняется генерация исходной модели для WebMLDoc по гипертекстовой модели We-

bRatio (то, что будет раскрашиваться в соответствии с функциями-сценариями), и (шаг 4) определяются все разделы существующей пользовательской документации, созданной в DocLine. При построении модели WebMLDoc учитывается как структура гипертекстовой модели WebML (профили, страницы, контент-модули, связи), так и расположение элементов модели на диаграмме в WebRatio (их размеры и координаты). При следующем изменении модели в редакторе WebRatio добавленные новые элементы будут появляться в модели WebMLDoc на тех же местах, на которых они расположены в редакторе WebRatio. Это позволяет добиться визуальной схожести модели WebMLDoc с гипертекстовой моделью WebML, но пользователь WebMLDoc может изменять расположение и размеры элементов. Далее проводится «привязка» модели WebMLDoc к документации DocLine (шаг 5). Сначала определяется, как была изменена гипертекстовая модель WebML, и с учетом существующей раскраски модели WebMLDoc пользователю предлагается список возможных изменений в документации. Шаг 5 автоматически вызывается при открытии редактора WebMLDoc, однако помимо этого разработчик может запустить эту проверку в любое время, чтобы получить актуальный список изменений. Все изменения, полученные на данном этапе, протоколируются и в дальнейшем могут быть исправлены разработчиком.

3.2. Поддержка трассировки изменений гипертекстовой модели

Поскольку пользовательская документация напрямую связана с пользовательским интерфейсом приложения, изменение последнего может потребовать внесения большого количества исправлений в документацию. Но не наоборот — мы не поддерживаем трассировку изменений из документации в WebRatio. Это понятно — нелепо изменять модель приложения на основе изменения документации. В редакторе WebMLDoc реализован процесс определения необходимых изменений в документации, показанный на рис. 2.

Кратко прокомментируем шаги данного процесса.

1. Пользователь меняет модель приложения в WebRatio в связи с доработкой и внесением новой функциональности, исправлением ошибок и т. д.

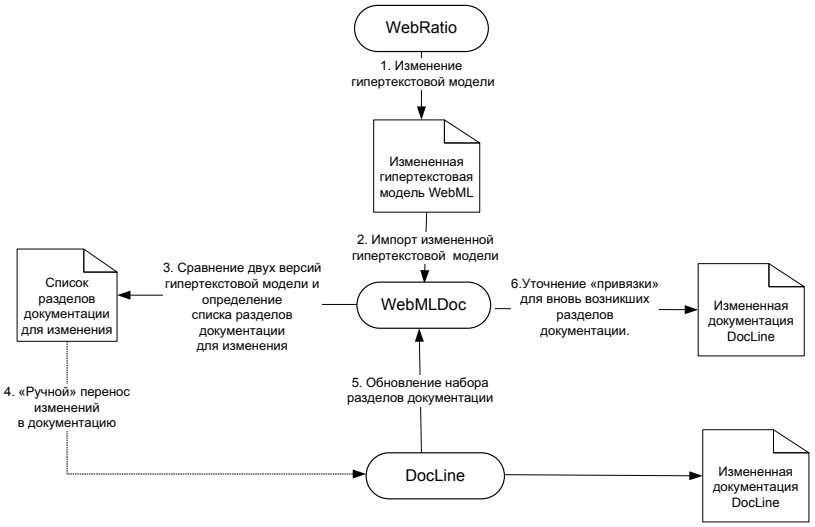

Рис. 2. Определение необходимых изменений в документации

2. Новая гипертекстовая модель импортируется в WebMLDoc.
3. Сравнение новой и предыдущей версий гипертекстовой модели, выявление разницы, генерация списка разделов документации, которые должны быть в связи с этим изменены.
4. Пользователь «вручную» изменяет документацию в связи с рекомендациями WebMLDoc.
5. Строится обновленный список разделов обновленной документации в WebMLDoc.
6. Уточняется «привязка» для вновь созданных в DocLine разделов документации, т. е. поддерживается трассировка из DocLine в WebMLDoc.

Алгоритм, обозначенный на шаге 3 и выполняющий определение разделов документации, подлежащих проверке при изменениях в гипертекстовой модели в WebRatio, работает следующим образом.

1. Алгоритм запускается автоматически при открытии редактора WebMLDoc или при вызове соответствующей команды из редактора WebMLDoc.

2. Гипертекстовая модель, хранящаяся в иерархичной структуре файлов и папок, собирается в один файл.
3. На основе обновленной пользователем гипертекстовой модели и ее предыдущей копии, сохраненной при предыдущем запуске алгоритма, вычисляются изменения, внесенные в модель разработчиком с момента последнего запуска этого алгоритма. Это делается с помощью функции XML *Diff*, которая позволяет вычислять разницу двух XML-файлов и предоставляется свободно распространяемой библиотекой JExamXML [11]. Она запускается через интерфейс командной строки или из Java-приложения с использованием специального программного интерфейса.
4. Для каждого измененного элемента в модели WebMLDoc выбираются разделы документации, с которыми он был связан, и добавляются в список разделов, требующих проверки.
5. Полученный список протоколируется, отображается пользователю и заносится в специальный журнал, который может быть открыт пользователем позже в любой момент.
6. Сохраняется новая резервная копия гипертекстовой модели.

Потери информации могут возникнуть, если разработчик добавит новую сущность в модель WebRatio, свяжет ее с разделом документа и затем поменяет ее имя. В этом случае будет потеряна информация о переименованной сущности, так как в резервной копии страницы еще нет и при сравнении будет только ее последнее имя. Данная проблема решается с помощью промежуточных дополнительных вызовов алгоритма из среды WebMLDoc после добавления новой сущности в гипертекстовую модель, но перед ее переименованием.

3.3. Решение проблемы целостности информации в WebMLDoc и WebRatio

Создание отдельного редактора WebMLDoc вместо непосредственной надстройки пакета WebRatio привело к необходимости решать дополнительную проблему — поддерживать целостность информации в WebRatio и в WebMLDoc. Эта задача возникает по следующим причинам.

1. Обе модели создаются не разово — наш подход предназначен для сопровождения приложений, т. е. предполагается, что в гипертекстовую модель будут вноситься изменения, которые должны будут отражаться в модели WebMLDoc. В обратную сторону поток изменений не предполагается, т. е. гипертекстовая модель не может меняться из-за изменений в модели WebMLDoc — в последнюю мы запрещаем добавлять контент-модули, новые связи и пр., что выглядит вполне естественным.
2. Обе модели имеют большое семантическое пересечение (это прежде всего структура гипертекстовой модели), а также различия — в гипертекстовой модели есть большое количество атрибутов у сущностей и связей, в частности, для связи с моделью данных; в модели WebMLDoc есть информация о связях с документацией (сценарии и их «привязка» к DRL). Таким образом, при изменениях в гипертекстовой модели модель WebMLDoc нельзя просто перегенировать — будет потеряна информация о связи с DRL!

Данная задача решается нами следующим образом. При каждом открытии графического редактора WebMLDoc его модель генерируется заново по текущей гипертекстовой модели WebRatio. Понятно, что в получившейся модели отсутствуют связи с документацией. Эти связи импортируются из предыдущей версии резервной копии модели WebMLDoc, создаваемой при каждом открытии редактора. Связи копируются по идентификаторам элементов, к которым они относятся. Это позволяет добиться следующих результатов:

- если элемент был удален в графическом редакторе WebRatio, то относящаяся к нему связь с документацией также будет удалена в редакторе WebMLDoc;
- если элемент был изменен в графическом редакторе WebRatio (изменено его имя или любой другой атрибут), то связь с документацией у этого элемента все равно останется.

Добавление новых, а также удаление или редактирование старых связей осуществляются пользователем с помощью стандартных средств графического редактора WebMLDoc.

3.4. Особенности реализации

Мы разработали WebMLDopc на основе платформы Eclipse с использованием технологии GMF (Graphical Modeling Framework) [10], предназначенной для создания графических редакторов диаграмм. Этот выбор обусловлен тем, что пакеты WebRatio и DocLine разработаны с помощью этих же средств.

Генерация XML-представления модели WebMLDoc выполняется с помощью специально созданной XSL-трансформации (eXtensible Stylesheet Language Transformations), обрабатываемой процессором Saxon (версия 9, Home Edition) [15]. Структура модели WebMLDoc хранится в файлах с расширением webml. Отдельный файл соответствует одному профилю сайта (siteview). Также в этом файле хранятся связи элементов интерфейса с документацией. При каждом открытии графического редактора WebMLDoc его модель генерируется заново по гипертекстовой модели WebRatio. В сгенерированной с помощью XSL-трансформации модели отсутствуют связи элементов интерфейса с документацией, поэтому актуальным становится вопрос о сохранении существующих связей интерфейса с документацией при каждом открытии редактора WebMLDoc. Для определения внесенных в гипертекстовую модель изменений применяется библиотека JExamXML [11].

4. Пример

Продемонстрируем работу WebMLDoc на примере модификации Acme, который является частью стандартной поставки WebRatio. Acme — это Интернет-магазин с поддержкой двух типов профилей: обычного пользователя (покупателя) и администратора сайта, который может изменять информацию о товарах и предложениях Интернет-магазина. Магазин Acme позволяет просматривать различные товары, сортировать их по цене и по категории, в которой они представлены, а также осуществлять поиск товара по ключевым словам. Кроме этого, в магазине Acme есть специальные предложения (offers), имеющие конечный срок действия. Исходный код магазина Acme был сгенерирован автоматически по моделям WebRatio, в том числе по гипертекстовой модели, представленной на рис. 3.

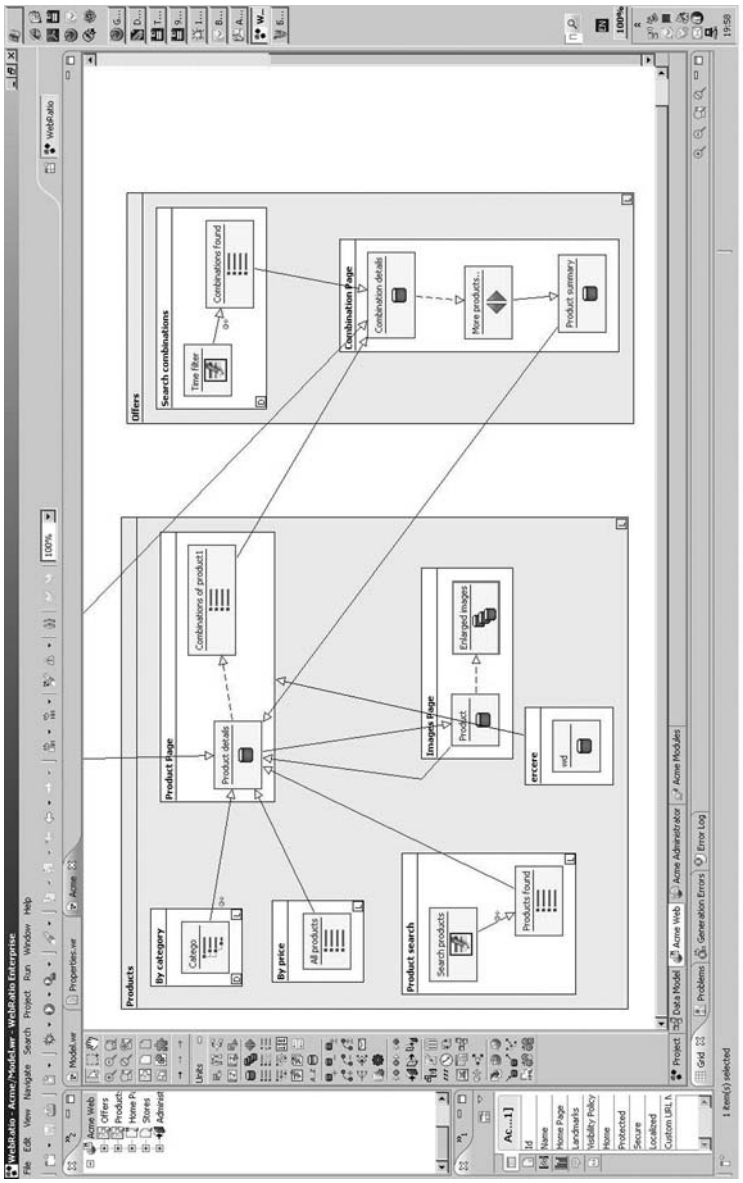

Рис. 3. Гипертекстовая модель Web-приложения Acme

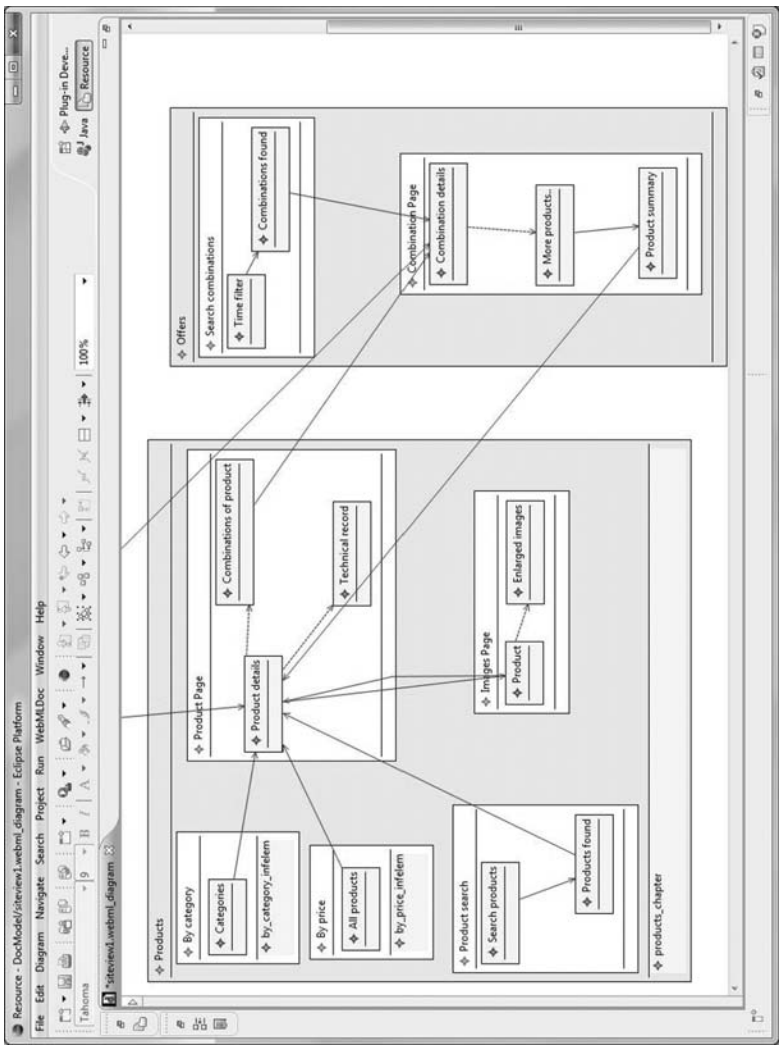

Рис. 4. WebMLDoc-модель приложения Acme

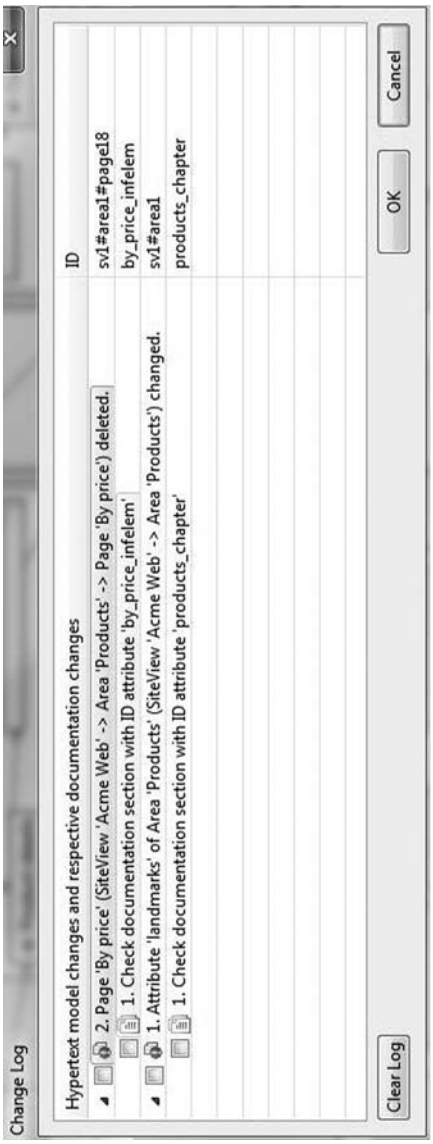

Рис. 5. Журнал изменений

В рамках работы с помощью технологии DocLine была разработана пользовательская документация Web-приложения Acme. Далее в разработанном графическом редакторе WebMLDoc была осуществлена «привязка» элементов гипертекстовой модели к разделам документации — элементам InfElement технологии DocLine. На рис. 4 показан графический редактор WebMLDoc с «привязкой» гипертекстовой модели к документации. Если элемент гипертекстовой модели имеет связь с некоторым InfElement, то имя последнего изображается внизу этого элемента. На рис. 4 элементы гипертекста «Products», «By category» и «By price» имеют такую «привязку» — соответствующие InfElement называются «products_chapter», «by_category_infelem» и «by_price_infelem».

Далее был реализован следующий сценарий модификации приложения Acme. Администратор Интернет-магазина убрал возможность сортировки товара по цене, оставив лишь сортировку по категориям. Для этого он из гипертекстовой модели удалил страницу «By price». При этом изменится пользовательский интерфейс Web-приложения — исчезнет возможность выбора сортировки товара по цене.

В этой ситуации WebMLDoc показывает пользователю все изменения интерфейса, предположительно влекущие за собою изменения документации. Для просмотра этих изменений в графическом редакторе WebMLDoc можно открыть журнал изменений. В рассматриваемом примере в журнале появляется информация, показанная на рис. 5. Как видно из этого рисунка, для исправления документации необходимо проверить корректность раздела «by_price_infelem», поскольку соответствующий ему элемент пользовательского интерфейса был удален, а также исправить раздел «products_chapter». Проверив разделы документации, указанные в журнале изменений, пользователь вносит изменения в документацию.

Заключение

Следующим шагом развития метода WebMLDoc видится слияние парадигмы семейств программных продуктов и метода DocLine в рамках одной технологии и единого инструментария. При этом

будет использоваться одна и та же диаграмма вариативности как для программного продукта, так и для документации, расширенная дополнительными точками вариативности для документации. Такой подход позволит нам в едином месте один раз сконфигурировать проект, состоящий из продукта и его документации. В дальнейшем, также из одной точки, будут отслеживаться изменения конфигурации проекта. В результате данной работы предполагается получить единую программную среду, позволяющую параллельно разрабатывать как семейство программных продуктов, так и пользовательскую документацию.

Список литературы

[1] Единая система программной документации (ЕСПД). ГОСТы серии 19. http://standards.narod.ru/gosts/gost19/gost19.htm

[2] *Кознов Д. В., Смирнов М. Н., Дорохов В. А., Романовский К. Ю.* WebMLDoc: подход к автоматизированному отслеживанию изменений в пользовательской документации Web-приложений. Принята к публикации в Вестнике СПбГУ. Сер. 10: Прикладная математика, информатика, процессы управления. Выйдет в начале 2011 года.

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

[4] *Романовский К. Ю., Кознов Д. В.* Язык DRL для проектирования и разработки документации семейств программных продуктов. Вестник СПбГУ. Сер. 10: Прикладная математика, информатика, процессы управления. 2007. Вып. 4. С. 110–122.

[5] *Фаулер М.* UML. Основы. 3-е издание. СПб.: Символ-Плюс, 2006. 192 с.

[6] *Acerbis R., Bongio A., Brambilla M. et al.* WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications. LNCS. Vol. 4607. Springer Berlin / Heidelberg, 2007. P. 501–505.

[7] *Childs B., Sametinger J.* Literate Programming and Documentation Reuse // Proceedings Fourth IEEE International Conference on Software Reuse, 1996. P. 205–214.

[8] *Da Silva P. P.* User Interface Declarative Models and Development Environments: A Survey. Lecture Notes in Computer Science. Vol. 1946, 2000. P. 207–226.

[9] IEEE Std 1063–2001, IEEE Standard for Software User Documentation.

[10] Eclipse Graphical Modeling Framework, http://www.eclipse.org/modeling/gmf/

[11] JExamXML Java API, http://www.a7soft.com/jexamxml/index.html

[12] *Pierce R., Tilley S.* Automatically Connecting Documentation to Code with Rose // Proceedings of the 20th Annual International Conference on Computer Documentation (SIGDOC'02). P. 157–163.

[13] *Putrycz E., Kark Anatol W.* Connecting Legacy Code, Business Rules and Documentation. N. Bassiliades, G. Governatori, and A. Paschke (Eds.): RuleML 2008, LNCS 5321. Springer-Verlag, Berlin Heidelberg, 2008. P. 17–30.

[14] *Rossi G., Pastor O., Schwabe D., et al.* Web Engineering: Modelling and Implementing Web Applications. Springer, 2007. 464 p.

[15] Saxon 9 Java API, http://www.saxonica.com/documentation/javadoc/index.html

[16] *Wang X., Lai G., Liu C.* Recovering Relationships between Documentation and Source Code based on the Characteristics of Software Engineering. Electronic Notes in Theoretical Computer Science. Vol. 243, Elsevier B. V. 2009. P. 121–137.