Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра информатики

Расширение гипертекстовой модели языка WebML средствами поддержки вариативности

Дипломная работа

Голубев Александр Михайлович
541 группа

Допущен к защите.

Зав. кафедрой:
д.ф.-м.н., профессор Косовский Н. К.

Научный руководитель:
к.ф.-м.н., доцент Кознов Д. В.

Рецензент:
ст. преп. Смирнов М. Н.

Санкт-Петербург
2010

St. Petersburg State University
Faculty of Mathematics and Mechanics

Chair of Computer science

WebML hypertext model extension with variability support tools

Graduate paper

Golubev Alexander
541 group

Admitted to proof.

Head of the chair:
PhD, Professor N.Kosovskyi

Scientific advisor:

PhD, Associate Professor D.Koznov

Reviewer:
Senior Lecturer M.Smirnov

St. Petersburg
2010

Оглавление

1. Введение

Web-приложения являются популярными и чрезвычайно востребованными в современном мире. Своей популярностью они обязаны распространенности Интернета, простоте использования Web-браузеров в качестве клиентского приложения, удобству обновления и поддержки без распространения и установки, а также кросс-платформенности. Многие современные приложения не просто имеют Web-аналоги, а изначально создаются таковыми. Среди Web-приложений сейчас можно найти не только привычные почтовые клиенты, аукционы и Интернет-магазины, но и такие "тяжелые" приложения, как графические редакторы, программы для обработки звука и пр.

Увеличение числа Web-приложений и их активное использование в бизнес-решениях привело к значительному увеличению требований к качеству и надежности таких приложений. Одним из путей повышения этих характеристик традиционно считается использование модельно-ориентированного подхода (Model Based Approach) с автоматической генерацией кода по моделям [1]. Для этих целей был создан ряд языков моделирования, ориентированных на Web-приложения – OOHDM [2, 3], UWA [4], WebML [5, 6], WISDOM [7] и другие.

Наиболее зрелым на сегодняшний день языком моделирования является язык WebML (Web Modeling Language) [6]. WebML предоставляет собой средства для графического описания процесса проектирования сложных Web-приложений, работающих с большим объемом данных. Основными целями языка WebML являются:

- задать структуру Web-приложения с помощью высокоуровнего описания, которое может быть использовано для дальнейшей разработки и поддержки приложения;
- предоставить несколько представлений (view) одной и той же информации;
- разделить описание Web-приложения на несколько частей – страницы, навигацию, представления и базу данных, каждая из которых может разрабатываться независимо;
- хранить метаинформацию, полученную в ходе разработки, в рамках единой модели и использовать ее в дальнейшем для динамического создания Web-страниц;
- явно моделировать поведение системы для конкретных пользователей или групп;

- определить операции с данными, для обновления содержания сайта или взаимодействия с внешними сервисами.

Язык WebML реализован в CASE-системе WebRatio [8], которая предоставляет возможности моделирования и автоматической генерации интерфейсных Web-приложений. В дополнение к этому модели, уже имеющиеся или вновь созданные в WebRatio, могут быть использованы как заготовки при разработке новых приложений – при этом значительно сокращается требуемое время и ресурсы. WebRatio позволяет создавать большие и сложные решения, такие как: корпоративные информационные системы, порталы и системы управления контентом, системы электронной коммерции и другие.

Метод разработки семейств программных продуктов (Software Product Lines) – это популярный метод разработки программного обеспечения, позволяющий создавать наборы похожих приложений, повторно используя общие активы разработки [9]. Одним из способов реализации product lines является создание обобщенного приложения с отмеченными точками вариативности, на базе которого строятся конкретные продукты. Использование этого метода может не только сократить время и расходуемые ресурсы на проектирование, разработку и тестирования приложений, но и повысить качество создаваемых продуктов.

При наличии возможности разрабатывать семейства программных продуктов в CASE-системах можно получать высококачественные и надежные приложения, затратив меньшее количество ресурсов. Кроме того, получившиеся приложения будут просты в поддержке и дальнейшей разработке.

В данной работе представлен подход к разработке семейств Web-приложений – в модельно-ориентированный подход WebML добавляются средства поддержки вариативности. А именно, на основе главной модели WebML – гипертекстовой модели – строится модель вариативности, с помощью которой задается множество продуктов семейства, использующих общие активы и имеющих свои уникальные программные компоненты. В работе так же представлен редактор PLWeb реализующий данный подход.

2. Постановка задачи

Основной целью данной работы является разработка системы, позволяющей на основе продукта WebRatio создавать семейства программных продуктов посредством добавления вариативности в гипертекстовую модель WebML. Для достижения этой цели было необходимо решить следующие задачи.

1. Изучить язык WebML, его гипертекстовую модель, продукт WebRatio, а так же существующие вариативные модели.
2. Разработать подход для работы с семейством программных продуктов и новую модель вариативности на базе языка WebML.
3. Реализовать средства для работы с семейством программных продуктов на базе WebRatio, использующие эту вариативную модель.
4. Опробовать разработанные средства на реальном примере.

3. Обзор

3.1. Семейства программных продуктов

Семейство программных продуктов – 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 по сравнению с традиционными подходами к разработке программного обеспечения. Среди них можно выделить повышение производительности труда, повышения качества выпускаемых продуктов, уменьшение общей стоимости разработки, уменьшение времени вывода на рынок и другие. Многие крупные компании, такие как Nokia, HP, LG и Philips, успешно используют данный подход в повседневной практике.

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.

Среди переиспользуемых ресурсов можно выделить следующие:

- требования и их анализ;
- предметная модель;
- архитектура программного обеспечения;
- работы по улучшению производительности;
- документация;
- планы по тестированию, тест-кейсы и тестовые данные;
- человеческие ресурсы, их знания и навыки;
- процессы, подходы и инструменты;
- бюджеты, расписания и планы работ;
- компоненты и сервисы.

Согласно [10] 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.

В [10] так же выделяется несколько основных подходов к разработке семейства программных продуктов, характеризующий последовательность создания тех или иных активов.

- Упреждающий (proactive) – разработка общих активов в начале:
  - разработка общей концепции в первую очередь;
  - продукты быстро выводятся на рынок, с минимальным количеством написанного кода;
  - требуются большие начальные инвестиции и заранее изученные требования.
- Реагирующий (reactive) – разработка одного или нескольких продуктов в начале:
  - из готовых продуктов извлекаются общие активы и потом новые продукты, общая концепция может сильно измениться;
  - гораздо меньшие начальные инвестиции;
  - архитектура и другие общие активы должны быть надежны, расширяемы и должны подходить для будущих продуктов.

- Накапливающий (incremental) – поэтапная разработка основных общих активов с изначальным планом создать семейство программных продуктов.

3.2. Диаграммы возможностей (feature diagrams)

Одной из основных задач разработки семейств программных продуктов – явное моделирование общих и различающихся частей продуктов. Диаграммы возможностей часто, используются для этого [11]. Они представляют собой визуальное представление возможностей и их зависимостей в виде И/ИЛИ-дерева. Под возможностью понимается явный, отличительный аспект, качество или характеристика программного продукта.

Отношения возможностями и их родителями в дереве могут быть следующих типов:

- обязательные – возможность необходимо включать всегда;
- необязательные – возможность можно не включать;
- "или" – хотя бы одна из возможностей должна быть включена;
- альтернативные – одна из возможностей должна быть включена.

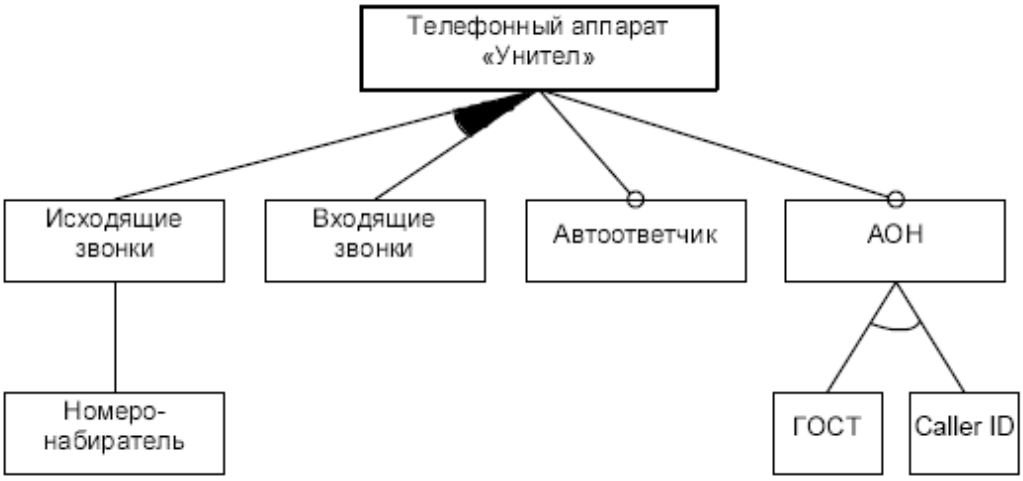


Рисунок 1: пример диаграммы возможностей [12]

3.3. Язык WebML

3.3.1. Общее описание языка

Язык WebML был разработан для моделирования Web-приложений, обладающих сложной логикой и работающих с большим количеством данных. Он предоставляет инструменты для графического описания проектируемых процессов и приложений.

Язык WebML состоит из нескольких частей, каждая из которых является независимой моделью. Эти модели описывают проектируемое приложение с разных уровней абстракции.

- Модель данных (data model) – описывает абстракцию организации данных приложения. Совместима с моделью сущность-связь, используемой при проектировании баз данных.
- Модель запросов (derivation model) – расширение модели данных вычислимыми конструкциями. Является аналогом представлений (view) в проектировании баз данных. При написании запросов используется язык WebML-OQL (WebML-Object Query Language).
- Гипертекстовая модель (hypertext model) – описывает построение и навигацию в разрабатываемом Web-приложении. Более подробно данная модель будет рассмотрена ниже.
- Модель представления (presentation model) – определяет конкретный внешний вид приложения.

3.3.2. Гипертекстовая модель

С помощью гипертекстовой модели описывается клиентский интерфейс Web-приложения: наборы страниц, доступные пользователям, элементы, находящиеся на этих страницах, навигация между ними и их связи с базой данных.

Структура Web-приложения описывается в терминах страниц (pages). Они являются непосредственными элементами интерфейса, показываемыми пользователю, который заходит на сайт. Страницы обычно состоят из нескольких контент-модулей – атомарных элементов, используемых для представления информации, описанной в модели данных. 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).

Профили сайта (site views) и области (area) позволяют организовать страницы в иерархическую структуру. Профили сайта являются крупнейшими структурными элементами. Они описывают часть Web-приложения, разработанную для определенного класса пользователей (например, клиентов, простых посетителей, менеджеров, администраторов и др.) Профили могут быть открытыми (public) или закрытыми (private), во втором случае для доступа к ним необходима аутентификация.

Области представляют собой группы страниц, объединенных общим назначением (например, область работы с продуктам, область поддержки, область решений, и т. д.). Области могут наследоваться. Так же как и страницы, области могут быть помечены ориентировочными (landmark) – ссылки на такие области будут присутствовать на всех страницах профиля сайта.

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.

3.4. Пакет WebRatio

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

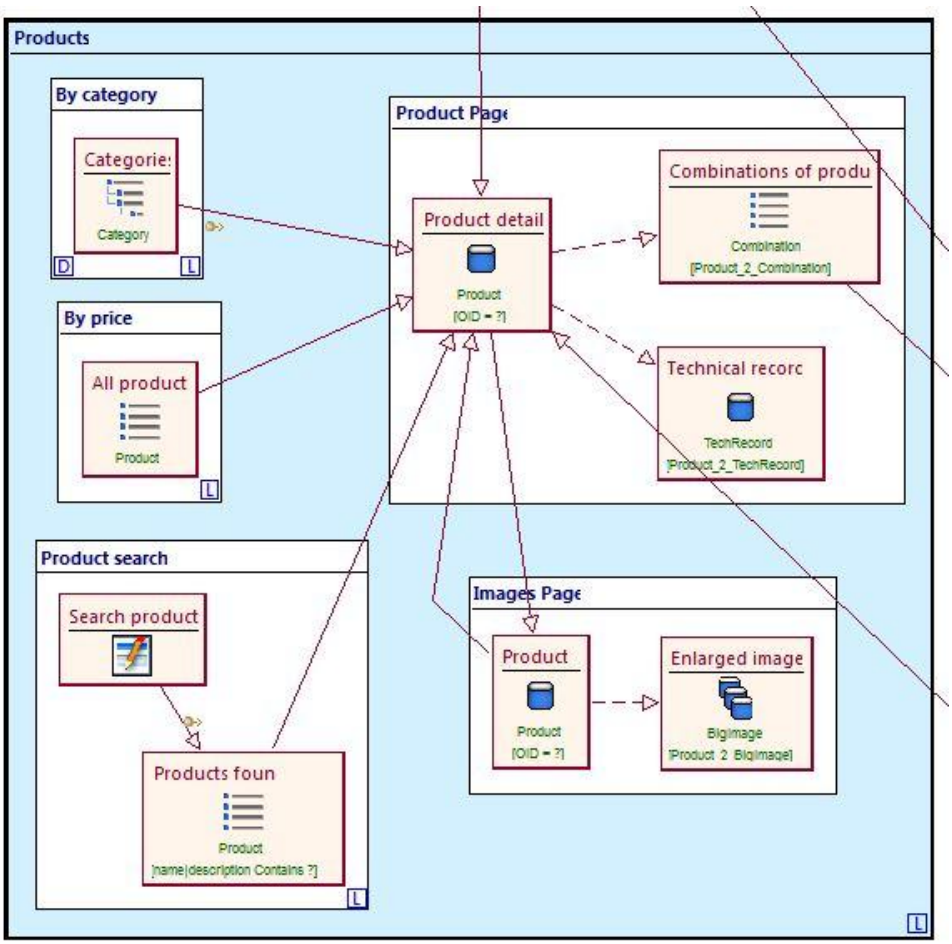


Рисунок 2: пример гипертекстовой диаграммы в редакторе WebRatio.

3.5. Платформа Eclipse и технология GMF

Eclipse – это популярная, постоянно развивающаяся платформа для создания инструментов разработки программного обеспечения [13]. Наиболее известные приложения, построенные на основе этой платформы - различные Eclipse IDE, предназначенные для разработки ПО на множестве языков программирования (Java, C++, PHP, Python и другие).

Разработка платформы началась в 2001 году фирмой IBM в качестве корпоративного стандарта разработки на разных языках для платформы IBM. В дальнейшем исходный код был полностью открыт и сделан общедоступным после того, как Eclipse была передана для дальнейшего развития независимому от IBM сообществу.

Начиная с версии 3.0 в основе архитектуры Eclipse лежит реализация технологии OSGi (Open Services Gateway initiative) – спецификация динамической модульной структуры для Java. Это позволяет разделить функциональность системы между

плагинами (plug-in). Таким образом, для добавления дополнительной функциональности достаточно реализовать необходимый плагин или набор плагинов с определенным интерфейсом.

Graphical Modeling Framework (GMF) – это фреймворк для разработки графических редакторов на основе платформы Eclipse [14]. С технической точки зрения он объединяет технологии Eclipse Modeling Framework (EMF) [15] – средство генерации Java-кода по моделям) – и Graphical Editing Framework (GEF) [16] – фреймворк предоставляющий средства для создания графических редакторов. GMF был впервые выпущен для версии Eclipse 3.2 Callisto в июне 2006.

В основе разработки графического редактора на платформе Eclipse GMF лежит модельно-ориентированный подход. Пользователь создает несколько моделей, по которым автоматически генерируется код редактора.

4. Управление вариативностью

Как было описано выше, язык WebML состоит из нескольких моделей, описывающих проектируемое Web-приложение. Добавление вариативности в гипертекстовую модель позволит создавать семейства программных продуктов, различающихся интерфейсом и логикой. Модели данных и представления предлагается использовать общие для всех продуктов, поскольку добавление вариативности в них только усложнит систему, не предоставив практически никаких преимуществ. Кроме того, данные модели обычно существенно меньше по размерам и более компактные, чем гипертекстовая модель. Последняя же сама по себе достаточно сложна и нагружена различными чертами и свойствами, поэтому ее непосредственное расширение средствами вариативности сделает работу с ней практически невозможной. В связи с этим было принято решение создать отдельную вариативную модель, связав ее с гипертекстовой моделью механизмом циклической разработки.

4.1. Общая схем разработки продуктов семейства

Рисунок 3: общая схема разработки продуктов семейства.

На рисунке 3 представлена общая схема разработки продуктов семейства. Рассмотрим ее подробно по шагам.

1. Первым шагом работы с системой является *создание нового или доработка существующего проекта WebRatio* – разрабатывается и/или уточняется гипертекстовая модель общих активов семейства всех будущих продуктов. В связи с этим она может быть не полностью корректной с точки зрения генерируемого результата. Изменение данной модели возможно на более поздних этапах работы с системой.
2. Далее следует шаг *генерации заготовки модели вариативности*. Эта процедура является автоматической и не требует участия пользователя. Получившаяся модель становиться связанной с исходной гипертекстовой. Изменения в исходной модели автоматически попадают в модель вариативности и наоборот.
3. Следующим шагом является *задание/доработка модели вариативности*: добавляются точки вариативности, которые далее будут использоваться для задания и выбора альтернативных элементов. Данные точки вариативности никак не влияют на исходную гипертекстовую модель, по сути, они являются только её "разметкой". Таким образом, часть существующих элементов гипертекстовой модели будут объявлены альтернативными. Могут также добавиться и новые альтернативные элементы, они будучи созданными в модели вариативности, автоматически переносятся в гипертекстовую модель.
4. *Построение конкретного продукта* на основе модели вариативности является следующим шагом в работе системы. В системе создается новый продукт посредством разрешения всех точек вариативности в модели. По получившейся явной разметке и исходной проекта в WebRatio создается новый проект, а в нем – новая гипертекстовая модель. Остальные модели (модель данных, модель представлений) копируются в новый проект из исходного неизменными.
5. Наконец, заключительным шагом является *доработка новой гипертекстовой модели конкретного продукта* – ее дополнение уникальными элементами и непосредственная генерация конечных Web-приложений.

Описанный процесс является итеративным – на каждом шаге есть возможность вернуться к предыдущему шагу.

4.2. Модель вариативности

Описанный процесс не ограничивает использование различных моделей вариативности. Предлагаемая в данной работе модель вариативности является лишь одной из возможных. Опишем ее более подробно.

Гипертекстовая модель включает в себя три основных структурных элемента - профили сайта (site views), области (areas) и страницы (pages). Они всегда находятся в иерархической зависимости друг от друга. Именно на уровне каждого из этих элементов добавляется возможность создания вариаций продукта. Для отображения этих вариаций предлагается использовать особый тип диаграмм возможностей – деревья, с наложенными ограничениями на элементы, находящихся в узлах. В корневых узлах таких деревьев находится название структурного элемента, для которого строятся вариации. В листовых узлах находятся варьирующиеся структурные элементы, вложенные в него, согласно иерархической структуре модели. Во всех остальных узлах находятся термины (классификационные группы), описывающие определенное свойство всех своих потомков. В остальном, диаграммы являются обычными диаграммами возможностей. Для каждого узла указывается, обязательно ли его включение или нет. Элементы могут объединяться в группы двух типов - ИЛИ (OR) и ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR). В первом случае только один из потомков может быть включен при разрешении вариативности, во втором - хотя бы один.

Рассмотрим каждый из уровней вариативности более подробно на примерах.

4.2.1. Уровень проекта

Пример диаграммы возможностей для проекта приведен на рисунке 4.

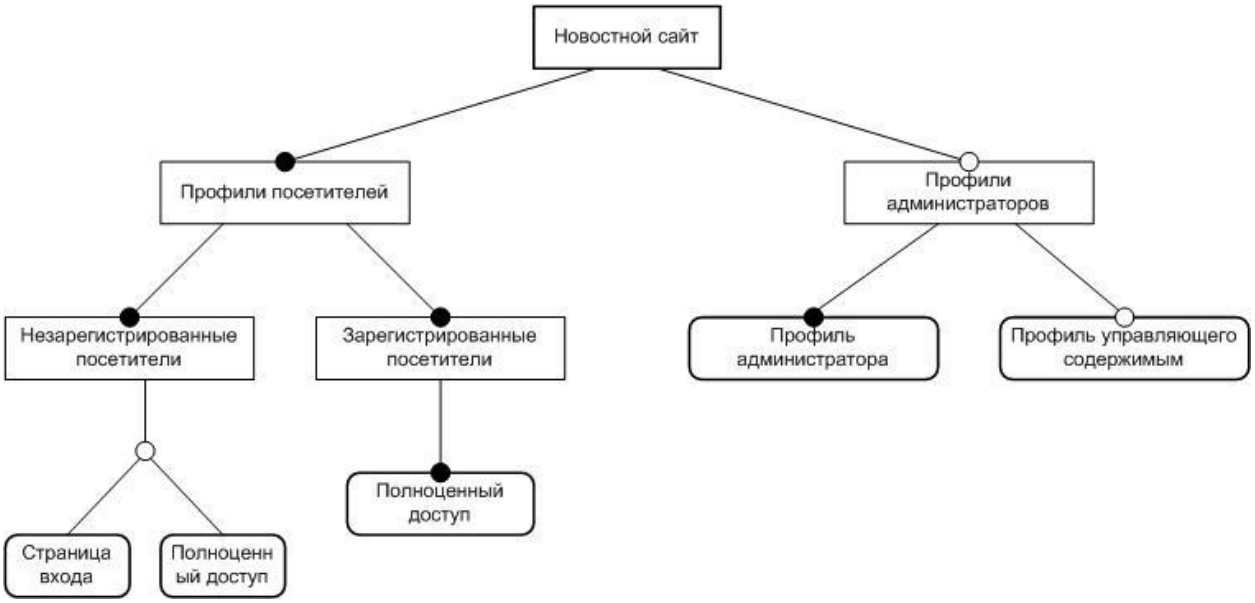


Рисунок 4: пример диаграммы возможностей для уровня проекта.

"Новостной сайт" – это название семейства продуктов. Все профили сайта разбиты на две группы – профили посетителей и профили администраторов, причем наличие профилей посетителей сайта в продукте данного семейства является обязательным, в то время как административных – нет. В частности, обязательным является наличие профиля для незарегистрированного пользователя, который может быть представлен одним из двух вариантов – профиля с единственной страницей входа в систему или профилем с полноценным доступом к содержимому сайта. Для зарегистрированного пользователя представлена только одна альтернатива. Из административных профилей обязателен профиль администратора, а профиль управляющего содержимым – нет. В приведенном примере листья дерева представляют собой конкретные профили сайта из гипертекстовой модели WebML. После разрешения вариативности на этом уровне для конкретного продукта, мы получим набор профилей, которые войдут в этот продукт.

4.2.2. Уровень профиля сайта

Пример диаграммы возможностей для профиля сайта приведен на рисунке 5.

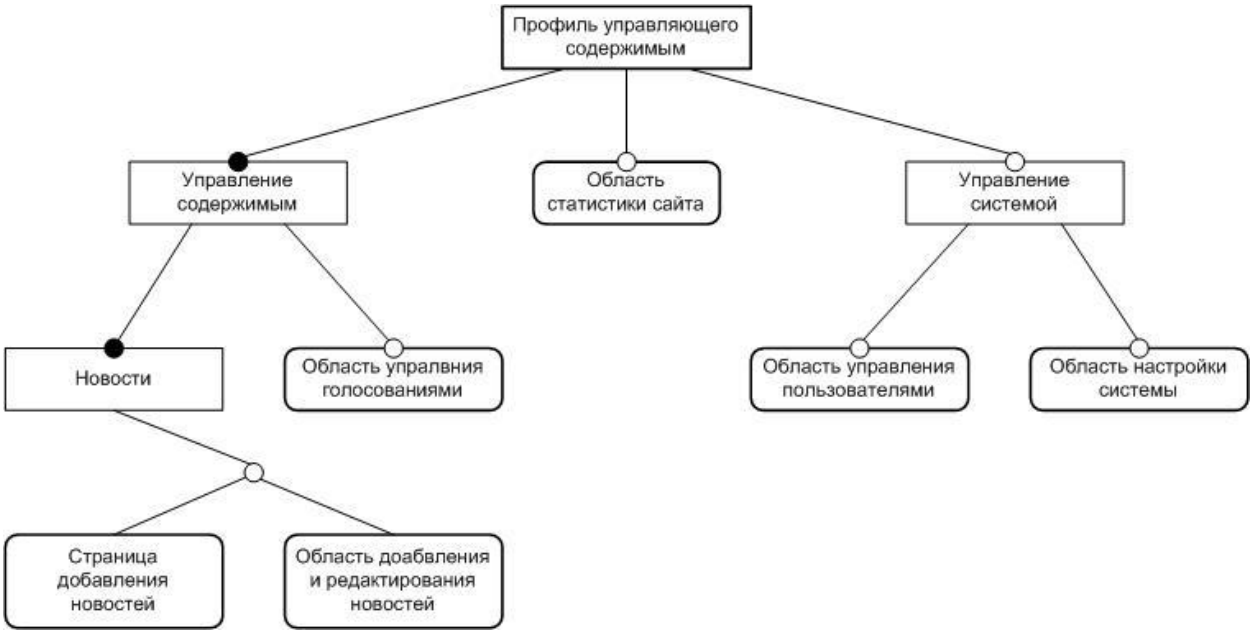


Рисунок 5: пример диаграммы возможностей для уровня профиля сайта.

В корне этого дерева вариативности находится профиль сайта, для которого строятся вариации. В листьях дерева находятся области или страницы гипертекстовой модели. В остальном диаграмма этого уровню по структуре повторяет диаграмму для уровня проекта. После разрешения вариативности на этом уровне для конкретного продукта, мы получим набор областей и страниц, которые войдут в выбранный профиль сайта. Таким образом, каждый из выбранных на предыдущем уровне профилей может быть более детально настроен для каждого продукта.

4.2.3. Уровень области

Добавление вариаций на уровне областей сайта полностью повторяет уровень профилей, за исключением того, что в корне дерева диаграммы возможностей находится конкретная область из гипертекстовой модели. После разрешения вариативности на этом уровне мы так же получаем набор страниц и областей, которые войдут в выбранную область.

4.2.4. Более глубокие уровни вариативности

Внутреннее содержание страницы представляет собой сложную структуру из контент-модулей и операций, а также связей между ними. Диаграмма вариативности для каждой страницы будет сложнее представленных выше описаний вариативности проекта,

профиля и области, и уже не сможет основываться на диаграммах возможностей (feature diagrams). Кроме того, что более существенно, проверка корректности получающихся после разрешения вариативности страниц необходимо проводить с учетом всех уровней модели. Так же возникают сложности при формировании уникальных идентификаторов контент-модулей. В исходной гипертекстовой модели не может существовать два элемента с одинаковыми идентификаторами, а при разрешении вариативности, конечный элемент должен иметь определенный идентификатор, не зависимо от выбранной альтернативы. В противном случае, ссылки между элементами потеряются. В связи с этим, на более глубоких уровнях было решено отказаться от возможности добавления точек вариативности.

4.2.5. Проблемы предложенной модели

В работе с предложенной моделью существует ряд следующих проблем

- Отсутствие вариативности на уровне страниц может привести к значительному увеличению их количества. Даже минимально отличающиеся страницы необходимо будет включать в модель как две разные.
- Наличие домашних страниц (home pages) и страниц по умолчанию (default pages) для областей и профилей сайта, зачастую является необходимым, для корректной работы сайта. В предложенной модели расстановка таких отметок невозможна, поэтому проекты конкретных продуктов могут быть не всегда корректными. Например, при наличии альтернатив страницы по умолчанию для области только одна из них может быть помечена таковой из-за ограничений исходной гипертекстовой модели и ее редактора в пакете WebRatio.
- Наличие связей между страницами и областями накладывает дополнительные ограничения на включения элементов в конечные продукты. Так как подобные связи никак не отображаются в предложенной модели вариативности, необходимо дополнительно проверять корректность процесса разрешения точек вариативности уже после того, как выбор сделан.
- Отсутствие повторного использования элементов в пределах одного проекта. Велика вероятность, что среди страниц и областей двух профилей сайта встретятся два одинаковых элемента. В рамках предложенной модели нет возможности такие элементы заменить на один и использовать в двух местах.

4.3. Процесс создания конкретных продуктов

Процесс создания конкретных продуктов можно разбить на два независимых действия:

1) разрешение вариативности в модели, построенной по проекту общих активов;
2) генерация нового проекта WebRatio на основе полученной спецификации.

Второе действие, по сути, является копированием проекта общих активов и заменой в нем гипертекстовой модели на новую. Новая гипертекстовая модель является суженной версией исходной - элементы включаются в неё с учетом выбранных альтернатив.

Для предложенной модели вариативности при создании новых продуктов появляется несколько дополнительных действий, поскольку при разрешении точек вариативности необходимо учитывать связи между элементами в гипертекстовой модели. Связь обычно имеет направление и соединяет страницы с другими страницами, областями или профилями сайтов. В случае, когда элемент, из которого выходит связь, не включен в конечный продукт, дополнительно делать ничего не надо. Однако в том случае, когда такой элемент включен, а элемент, в который связь приходит, не включается в конечный набор, должна возникать ошибка, приводящая к неудачному завершению процесса. Помимо этого, необходимо дополнительно контролировать невключение в конечный продукт элементов, если хотя бы один из их родительских элементов не был выбран для него.

5. Архитектура программных средств

Помимо разработанного подхода, в рамках данной работы была выполнена его пилотная реализация. На основе платформы Eclipse GMF был создан редактора диаграмм вариативности PLWeb.

5.1. Общая схема работы редактора PLWeb

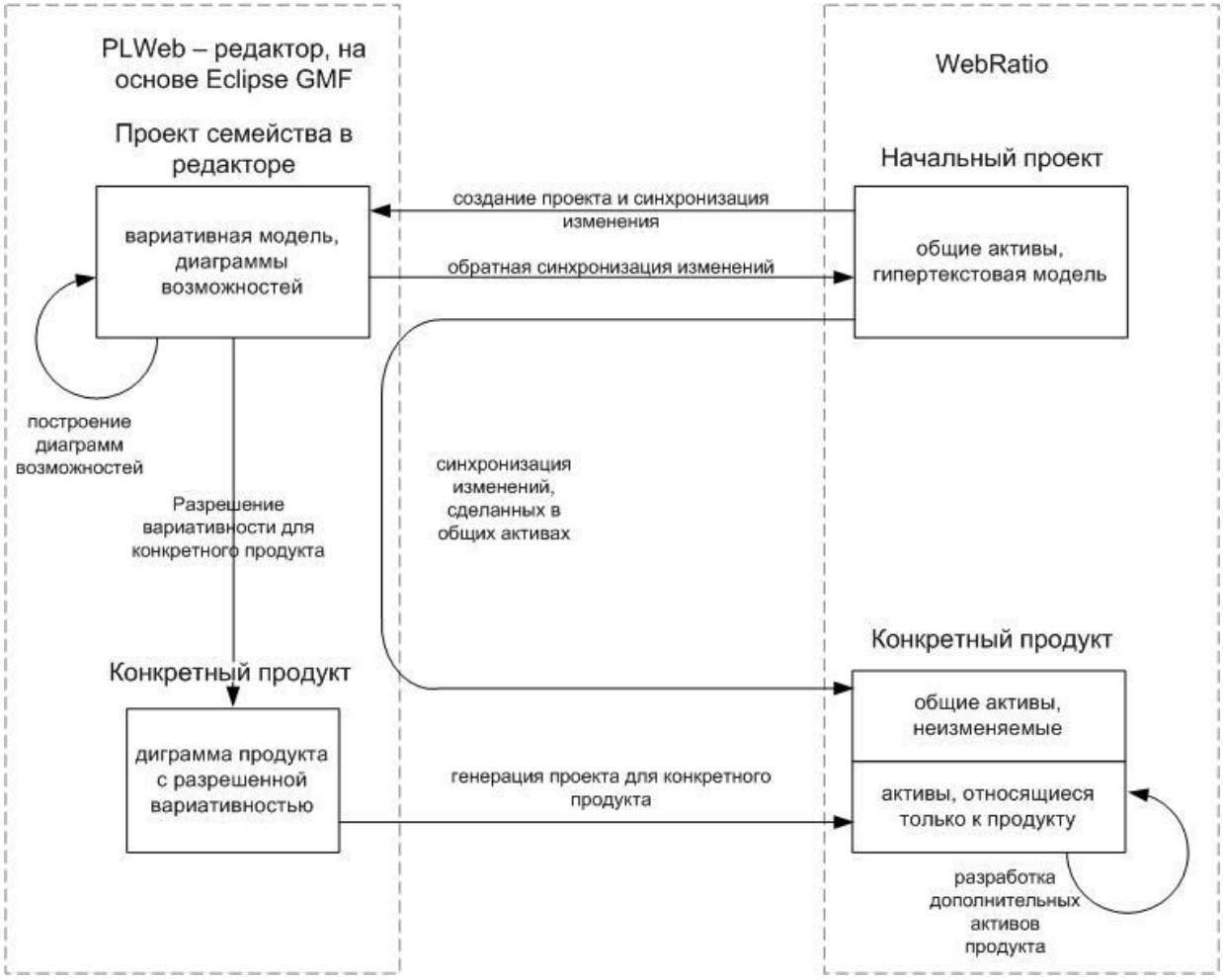


Рисунок 6: общая схема работы редактора PLWeb.

Процесс работы с редактором PLWeb в целом повторяет процесс работы подхода, описанный ранее. На рисунке 6 представлена общая схема работы редактора. Рассмотрим основные этапы работы с ним.

1. Первым этапом работы с редактором является разработка начального проекта в WebRatio. Необходимо создать модели данных и представления, удовлетворяющие всем будущим продуктам. Кроме того, необходимо разработать гипертекстовую

модель, содержащую все общие элементы семейства продуктов. В дальнейшем, на основе активов этого проекта будет происходить построение проектов конкретных продуктов.

2. На втором этапе работа ведется с моделью вариативности, построенной по гипертекстовой модели. Это работа происходит в разработанном редакторе PLWeb. Изначально связи между элементами на диаграммах отсутствуют. Необходимо построить диаграммы вариативности, добавив описывающие термины и объединив элементы в группы.
3. Следующий этап заключается в добавлении конкретных продуктов семейства и разрешении точек вариативности для них. На основании принятых решений строится диаграмма продукта, запрещенная к редактированию. По этой диаграмме генериться новый проект WebRatio для конкретного продукта.
4. Наконец, последним этапом является работа с проектом конкретного продукта. В целях сохранения возможности обновления и улучшения исходных общих активов, в данном проекте запрещается редактирование сгенеренной части гипертекстовой модели. Возможно лишь добавление новых активов, специфичных для продукта.

Помимо этапов работы, на схеме указаны основные процессы, происходящие в системе. Они включают в себя:

- процесс создания вариативной модели из гипертекстовой;
- двухсторонняя синхронизация этих моделей;
- процесс разрешения вариативности;
- процесс генерации нового проекта WebRatio;
- синхронизация изменений, сделанных в общих активах, со всеми проектами конкретных продуктов.

Более подробно эти процессы и их реализация будут рассмотрены позже.

5.2. Реализация модели вариативности

Предложенная ранее модель вариативности, имеет несколько уровней, по структуре повторяющих друг друга. В рамках данной работы, было принято решение реализовать только нижний уровень – уровень области. Этого достаточно, для того чтобы показать все возможности модели вариативности, исследовать алгоритмы происходящих

процессов и опробовать подход на реальном примере. Таким образом, редактор PLWeb работает только с гипертекстовыми моделями, содержащими один профиль сайта, одну область и набор страниц в этой области.

5.2.1. Редактор PLWeb

Редактор реализован на основе платформы Eclipse GMF. Как было описано выше, эта платформа использует модельно-ориентированный подход. Процесс создания редактора основан на разработке набора моделей и описаний. Одной из таких моделей является метамодель данных. Диаграмма разработанной метамодели для редактора PLWeb приведена на рисунке 7.

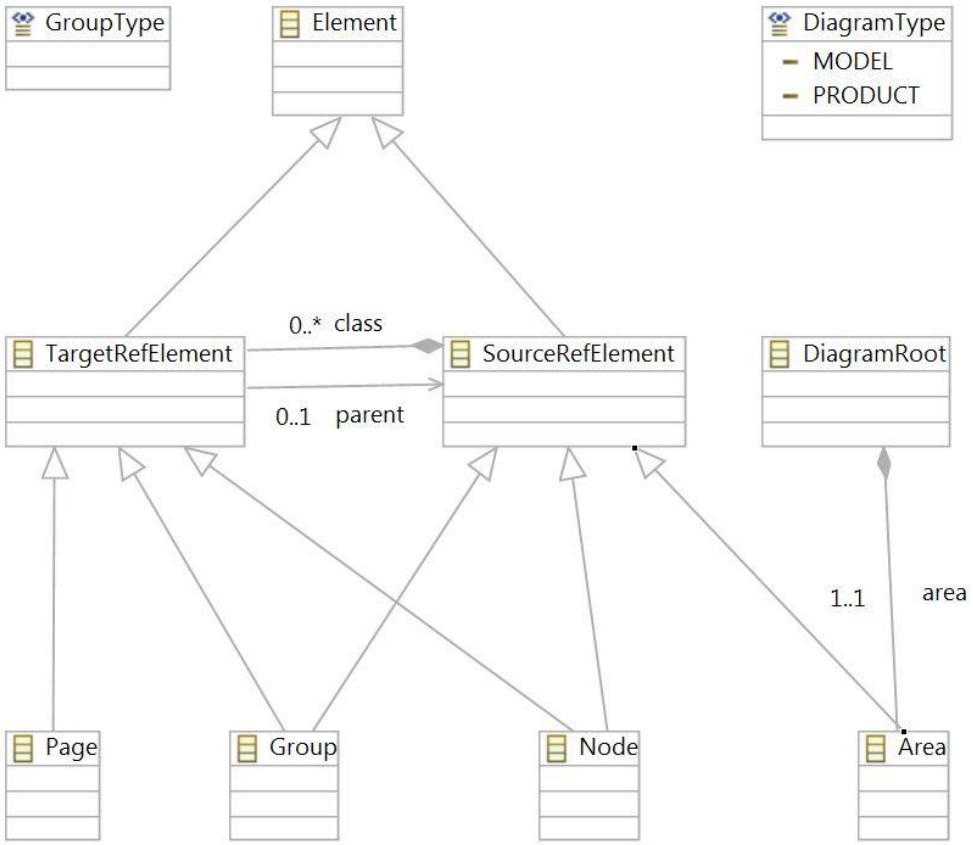


Рисунок 7. метамодель данных редактора PLWeb (plweb.ecore).

Элементы метамодели соответствуют элементам диаграмм возможностей, а именно:

- Area – область, может быть только корневым элементом диаграммы;
- Node – термин, описывающий набор страниц;

- Group – группа страниц;
- Page – страница Web-приложения, листья дерева диаграммы.

У всех перечисленных элементов, кроме Area, присутствует атрибут "optional", определяющий обязательность включения элемента при разрешении вариативности. Остальные элементы являются вспомогательными и используются для хранения дополнительной информации.

Кроме метамодели данных, были так же разработаны модели plweb.gmfgraph (описание всех представлений графических элементов диаграммы), plweb.gmftool (описание дополнительных средств, в частности инструментов создания элементов диаграммы) и plweb.gmfmap (описание связи остальных моделей между собой).

После генерации кода редактора инструментами GMF, в него потребовалось внести дополнительные изменения. Они были связаны, в основном, с расширением возможностей редактора. Была добавлена следующая функциональность.

- Импорта проектов WebRatio и построение начальных диаграмм для них.
- Автоматическая синхронизация гипертекстовой и вариативной модели.
- Процесс разрешения вариативности для конкретного продукта.
- Генерация новых проектов WebRatio для конкретных проектов.
- Синхронизация изменения исходных моделей со сгенерированными проектами продуктов.

Более подробно эти изменения будут рассмотрены ниже.

5.2.2. Работа с редактором

Работа с редактором начинается с создания нового проекта, при этом указывается путь к месторасположению проекта WebRatio, на основе которого создается семейство продуктов. При инициализации нового проекта создается файл <имя проекта>.plweb, описывающий модель вариативности в XMI-формате [17]. На его основе можно создать файл диаграммы модели, работать с которым можно через графический интерфейс. Пример построенной в редакторе диаграммы возможностей можно увидеть на рисунке 8.

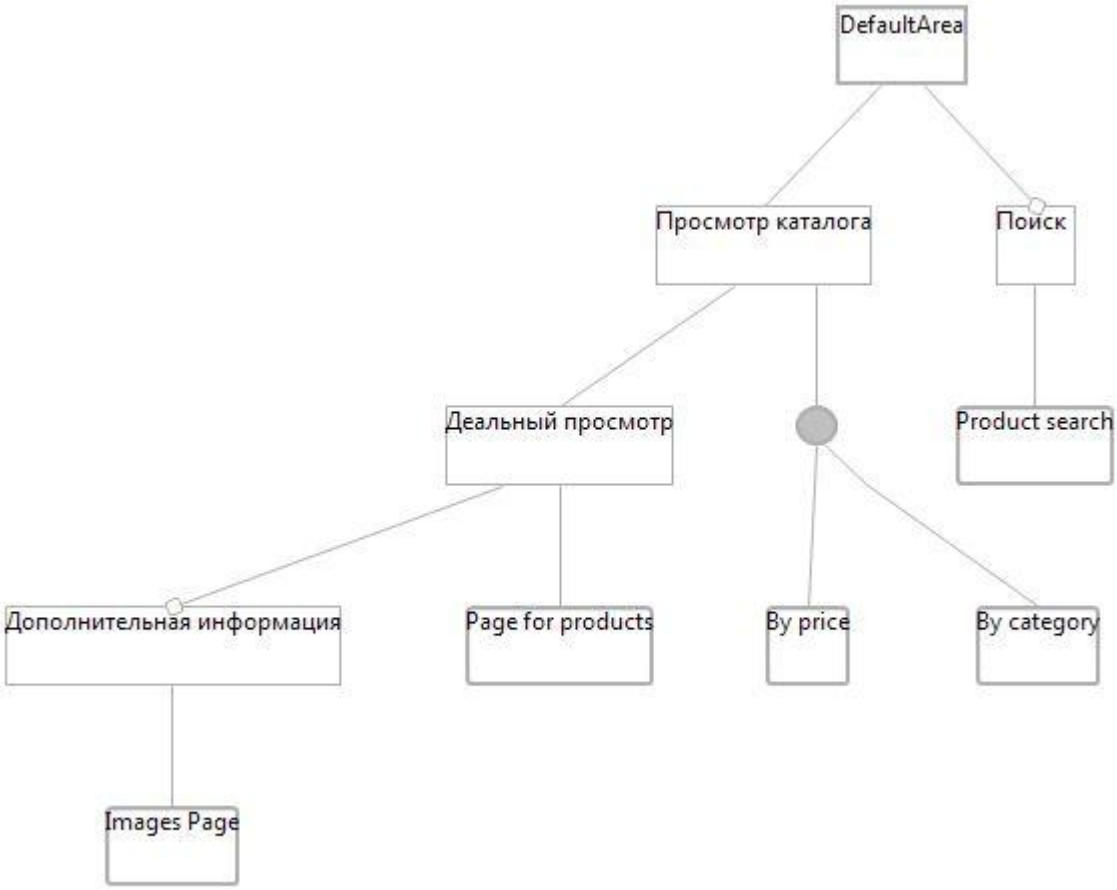


Рисунок 8. пример диаграммы возможностей.

Соответствующее описание модели представлено на листинге 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<plweb:DiagramRoot xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:plweb="http://plweb/1.0"
projectPath="D:\SOURCES\WebRatioWS\Acme">
  <area title="DefaultArea">
    <class xsi:type="plweb:Node" title="Просмотр каталога" parent="//@area">
      <class xsi:type="plweb:Node" title="Деальный просмотр"
          parent="//@area/@class.0">
        <class xsi:type="plweb:Page" title="Page for products"
            parent="//@area/@class.0/@class.0" source="page2.wr"/>
        <class xsi:type="plweb:Node" title="Дополнительная информация"
           parent="//@area/@class.0/@class.0" optional="true">
          <class xsi:type="plweb:Page" title="Images Page"
              parent="//@area/@class.0/@class.0/@class.1" source="page5.wr"/>
        </class>
      </class>
      <class xsi:type="plweb:Group" parent="//@area/@class.0">
        <class xsi:type="plweb:Page" title="By price"
            parent="//@area/@class.0/@class.1" source="page18.wr"/>
        <class xsi:type="plweb:Page" title="By category"
            parent="//@area/@class.0/@class.1" source="page4.wr"/>
      </class>
    </class>
    <class xsi:type="plweb:Node" title="Поиск" parent="//@area" optional="true">
```

```
      <class xsi:type="plweb:Page" title="Product search" parent="//@area/@class.1"
          source="page19.wr"/>
    </class>
  </area>
</plweb:DiagramRoot>
```

Листинг 1. Описание модели вариативности.

5.3. Генерация проекта WebRatio конкретного продукта

Процесс добавления нового продукта и генерации его проекта является одним из ключевых в работе всей системы. Для добавления продукта необходимо разрешить все точки вариативности в модели. Для этого строится дерево, отображающее структуру модели вариативности. В нем отображается вся необходимая информация для принятия выбора о конечном списке элементов, входящих в продукт. Помимо ограничений, накладываемых структурой диаграммы возможностей, в построенном дереве учитываются ограничения на включение элементов, возникающие из-за связей между элементами в гипертекстовой модели. Данные ограничения получаются из анализа гипертекстовой модели и заключаются в следующем: если элемент включается в дерево, включаться так же должны и все элементы, на которые он ссылается.

После получения списка элементов создается новый проект продукта. Он получается копированием исходного проекта и заменой в нем гипертекстовой модели на суженную. Проект создается в той же папке, где находится и основной.

5.4. Реализация синхронизации на различных уровнях

5.4.1. Синхронизация гипертекстовой модели с моделью вариативности

Синхронизация гипертекстовой модели с моделью является двухсторонней и происходит в ручном режиме. Для этого существуют специальные элементы управления. Процесс происходит следующим образом:

1) строится список элементов гипертекстовой модели, относящихся к данному уровню диаграммы (в нашем случае список всех страниц области);
2) строится список всех листовых элементов дерева диаграммы вариативности;
3) вычисляется разница между двумя списками;
4) если элемент присутствует в обоих списках, сравниваются его атрибуты, и при необходимости, обновляются;

5) если элемент из гипертекстовой модели отсутствует на диаграмме, он добавляется на нее без связей с остальными элементами;
6) если существуют элементы на диаграмме, отсутствующие в первом списке, они удаляются.

Процесс обратной синхронизации происходит по аналогичному алгоритму.

5.4.2. Обновление проектов продуктов семейства

Обновление проектов продуктов может потребоваться в двух случаях: изменилась гипертекстовая модель в проекте общих активов или изменилась модель вариативности. Во избежание потери данных, выполнение данного действия происходит не автоматически, а по желанию пользователя, в удобный ему момент. Процесс обновления состоит из следующих этапов:

1) строится список, включенных в продукт элементов;
2) элементы из списка находятся в дереве вариативной модели и отмечаются как включенные;
3) для не включенных элементов производится проверка, являются ли они обязательными или нет, в первом случае пользователю выводится предупреждение;
4) после исправлений, внесенных пользователем, строится список новых элементов;
5) элементы, включенные в продукт обновляются в соответствии с построенным списком.

При условии следования ограничениям неизменяемости сгенерированной части гипертекстовой модели, обновленные проекты продуктов либо будут корректными сразу, либо после небольших корректировок.

6. Особенности реализации

6.1. Использование особенностей хранения гипертекстовой модели WebRatio

Гипертекстовая модель в WebRatio храниться на диске в иерархической структуре файлов и папок (рисунок 9). Каждому профилю (site view) соответствует папка с соответствующим названием. Каждой области (area) так же соответствует папка. Каждой странице соответствует XML-файл в котором содержится ее описание. Кроме того в каждой папке находится файл Properties.wr, в котором описаны дополнительные свойства объединений.

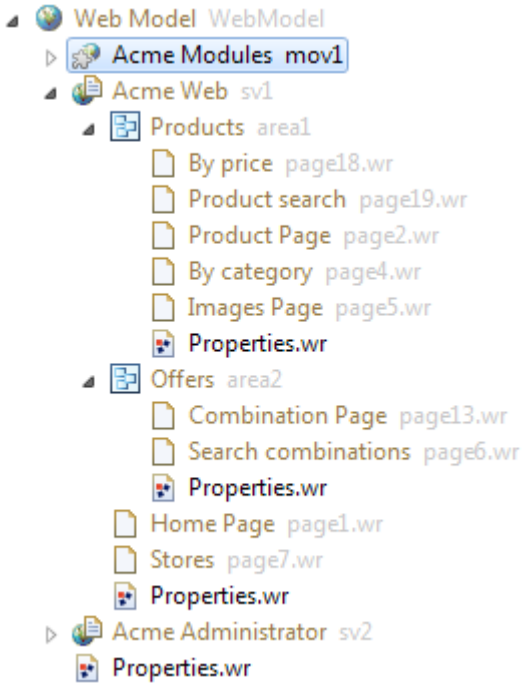


Рисунок 9: Структура файлов в проекте WebRatio.

Особенности этой структуры используются в редакторе PLWeb при генерации и обновлении проектов WebRatio. Так как модель вариативности затрагивает только уровень страниц, добавление или исключение элемента из продукта заключается в копировании или удалении соответствующего файла из проекта. Сами файлы являются копиями файлов проекта обхих активов.

6.2. Работа с XML, использование SAX и DOM парсеров

Описания всех моделей, используемых в работе редактора PLWeb, хранятся в различных форматах, основанных на XML. Для работы с этих форматом существует два основных подхода: использование DOM (Document Object Model) [18] или SAX (Simple API for XML) [19] парсеров. У каждого из них есть свои недостатки и преимущества.

DOM-парсер предоставляет механизм чтения данных из XML документа. Этот подход основывается на представлении данных в виде дерева. При первой загрузке документа, он просматривается полностью и в памяти исполняемой среды на его основе строится дерево. Благодаря этому появляется много возможностей по работе с документом. Таких как: выбор всех детей, всех элементов того же уровня, всех элементов с определенным именем или набором атрибутов и многие другие. Однако, несмотря на удобство обработки данных, существенным недостатком является скорость работы, особенно с большими документами.

SAX-парсер является популярной альтернативой DOM. Этот подход основан на потоковой обработке документа и событийной системе. Обычно при работе с SAX-парсером, пользователь описывает ряд функций, которые будут выполнятся при определенных событиях по мере обработки документа. Такими событиями могут быть:

- текст в XML;
- элементы XML;
- дополнительный инструкции в XML;
- комментарии в XML.

Функции вызываются, когда один из перечисленных элементов встречается в документе и снова вызываются, когда встречается его окончание. XML-атрибуты обрабатываются, как дополнительные данные при работе с XML элементами. SAX-парсер является однонаправленным: однажды обработанные данные не могут прочитаны еще раз без запуска операции парсинга с начала. По сравнению с DOM-парсером, SAX-парсер использует значительно меньшее количество памяти и зачастую работает значительно быстрее.

Гипертекстовая модель описана в большом количестве часто объемных XML-файлов для каждого структурного элемента. Чаще всего информация, используемая редактором содержится в атрибутах корневого элемента – это название и идентификатор

описываемого структурного элемента. В связи с этим для работы с гипертекстовой моделью используется SAX-парсер. Модель вариативности, напротив, описана одним единственным файлом. Алгоритмам, работающим с этой моделью, нужна информация обо всех элементах, их атрибутах и иерархическом положении. Для таких целей DOM-парсер подходит лучше.

7. Заключение

7.1. Результаты

В рамках данной работы были достигнуты следующие основные результаты:

1. Изучены язык WebML и его гипертекстовая модель.
2. Изучен продукт WebRatio, примеры проектов, а так же возможности его расширения и взаимодействия с ним.
3. Разработан подход для работы с семейством программных продуктов на базе WebRatio
4. На основе диаграмм возможностей (Feature Diagrams) разработана вариативная модель, позволяющая работать с гипертекстовой моделью.
5. Реализован редактор на основе платформы Eclipse GMF для работы с семейством программных продуктов.
6. Разработаны и реализованы процессы взаимодействия редакторов и синхронизации их проектов.
7. Разработанный редактор успешно опробован на реальном примере.

7.2. Дальнейшие пути развития

Основные пути развития разработанного подхода заключаются в улучшении модели вариативности, в частности устранении недостатков, описанных ранее.

- Добавление поддержки повторно используемых элементов.
- Учет связей гипертекстовой модели в модели вариативности. Ограничения должны накладываться в момент построения диаграмм вариативности, а не во время их разрешения.
- Хранение дополнительной информации о поведении элементов диаграмм вариативности для более точной настройки проектов конкретных продуктов. Данная информация может включать условия, когда страница должна стать страницей по умолчанию или домашней, условия, для изменения идентификаторов элементов и условия на другие параметры элементов.
- Добавление вариативности на уровне страниц и контент-модулей позволит уменьшить количество элементов, представленных в модели вариативности и

предоставит больше возможностей для еще более детальной настройки проектов продуктов.

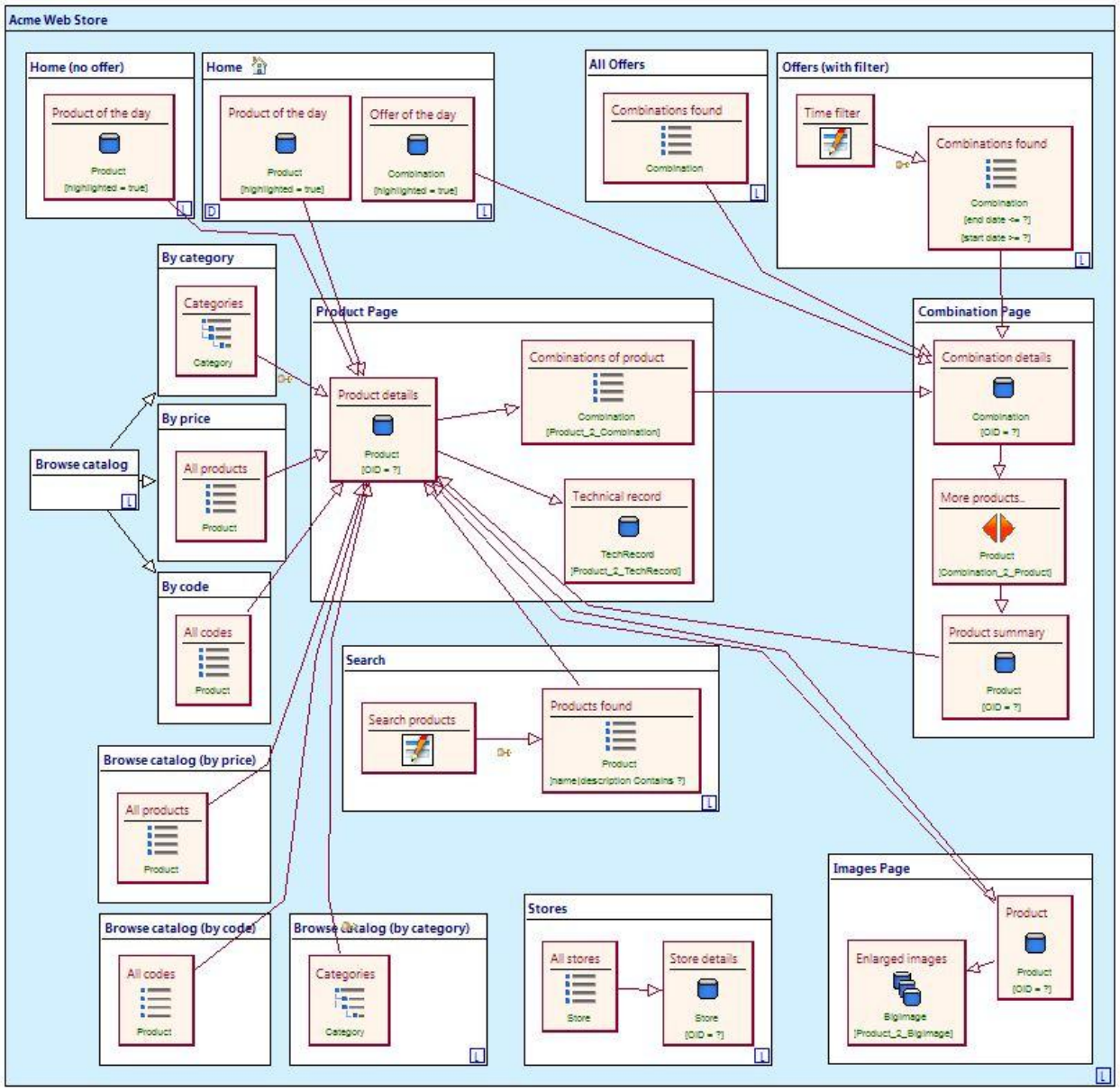
- Добавление вариативность в другие модели языка WebML.

Описанные улучшения могут потребовать изменений, как в конкретных действиях с моделями, так и во всем процессе работы подхода.

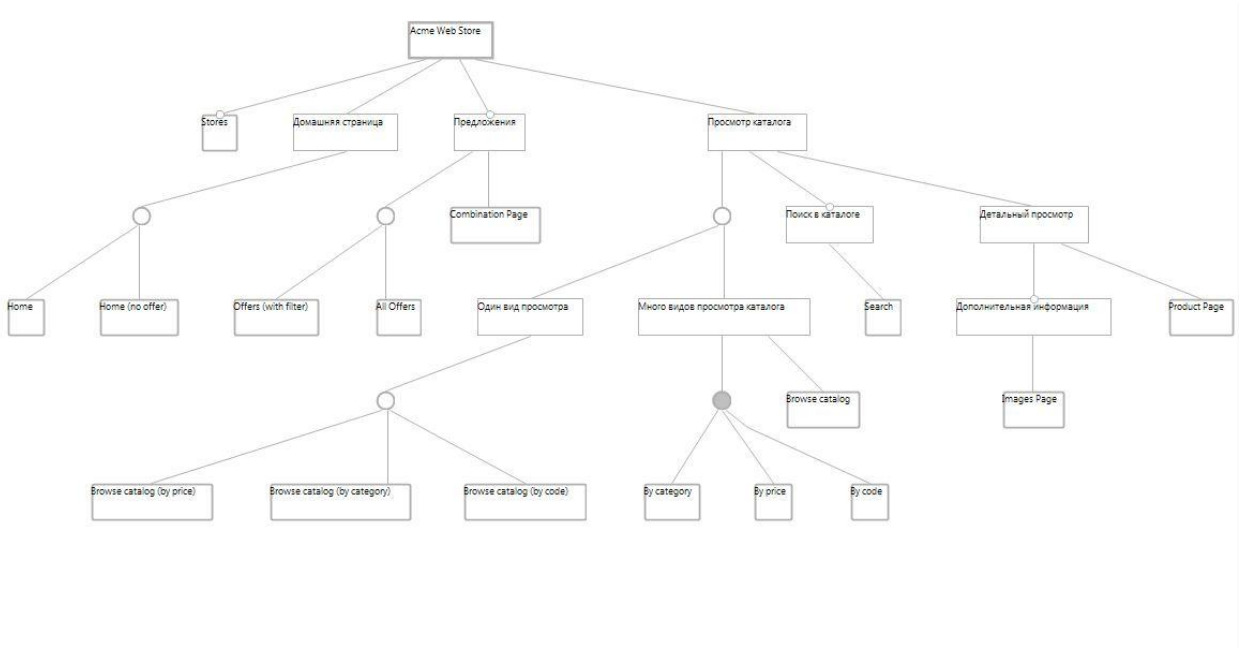
8. Список литературы

1. *David S. Frankel, John Wiley & Sons.* Model Driven Architecture: Applying MDA to Enterprise Computing.
2. *Daniel Schwabe, Gustavo Rossi.* An Object Oriented Approach to Web-Based Application Design.
3. *Daniel Schwabe, Gustavo Rossi.* Developing Hypermedia Applications using OOHDM.
4. *UWA Consortium.* Ubiquitous Web Applications.
5. *Rossi G., Pastor O., Schwabe D., et. al.* Web Engineering: Modeling and Implementing Web Applications. Springer, 2007. 464 p.
6. WebML information site: http://webml.org/.
7. *O.M.F. De Troyer, C.J. Leune.* WSDM: A User Centered Design Method for Web Sites. Proceedings of the 7th International World Wide Web Conference, Elsevier, 1998. P. 85–94.
8. WebRatio product site: http://www.webratio.com/.
9. *Paul C. Clements, Linda M. Northrop.* Software Product Lines: Practices and Patterns. 2001. 576 p.
10. *Linda M. Northrop.* Software Product Line Essentials. http://www.sei.cmu.edu/library/assets/spl-essentials.pdf.
11. *Kang, K.C. and Cohen, S.G. and Hess, J.A. and Novak, W.E. and Peterson, A.S.* Feature-oriented domain analysis (FODA) feasibility study, Technical Report CMU/SEI-90-TR-021, SEI, Carnegie Mellon University, November 1990.
12. *К.Ю.Романовский, Д.В.Кознов.* Язык DRL для проектирования и разработки документации семейства программных продуктов, Вестник Санкт-петербургского университета, Серия 10, Информатика, № 4, 2007. С. 110-122.
13. Eclipse project official site: http://www.eclipse.org/.
14. Graphical Modeling Framework project site: http://www.eclipse.org/gmf.
15. Eclipse Modeling Framework project site: http://www.eclipse.org/modeling/emf/.
16. Graphical Editing Framework project: http://www.eclipse.org/gef/.
17. OMG XMI Specification: http://www.omg.org/technology/documents/formal/xmi.htm.
18. DOM w3 page: http://www.w3.org/DOM/.
19. SAX home page: http://www.saxproject.org/.

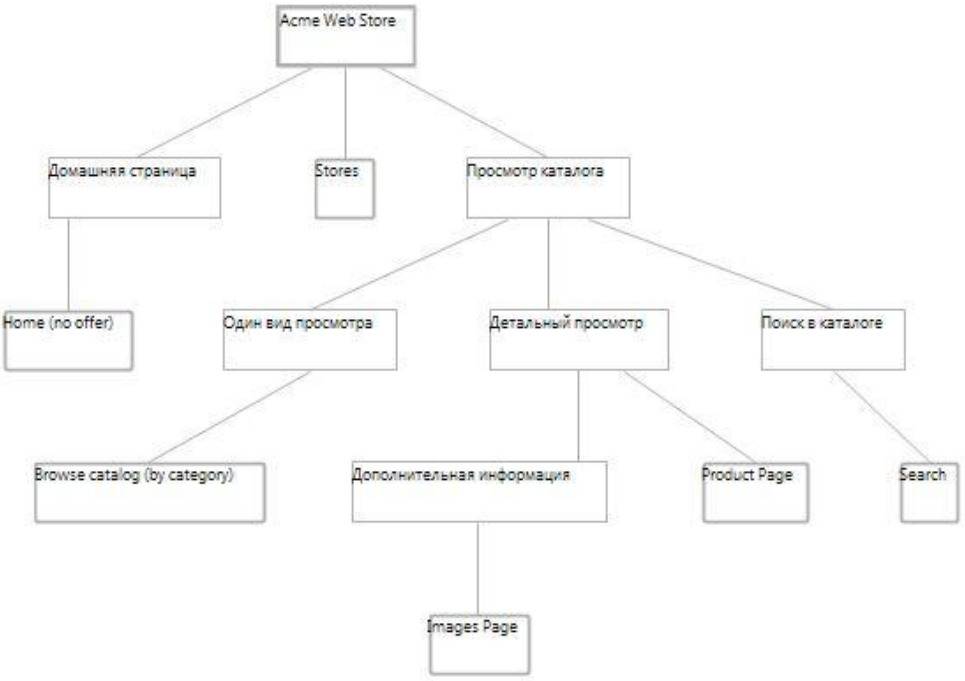
Приложение. Пример



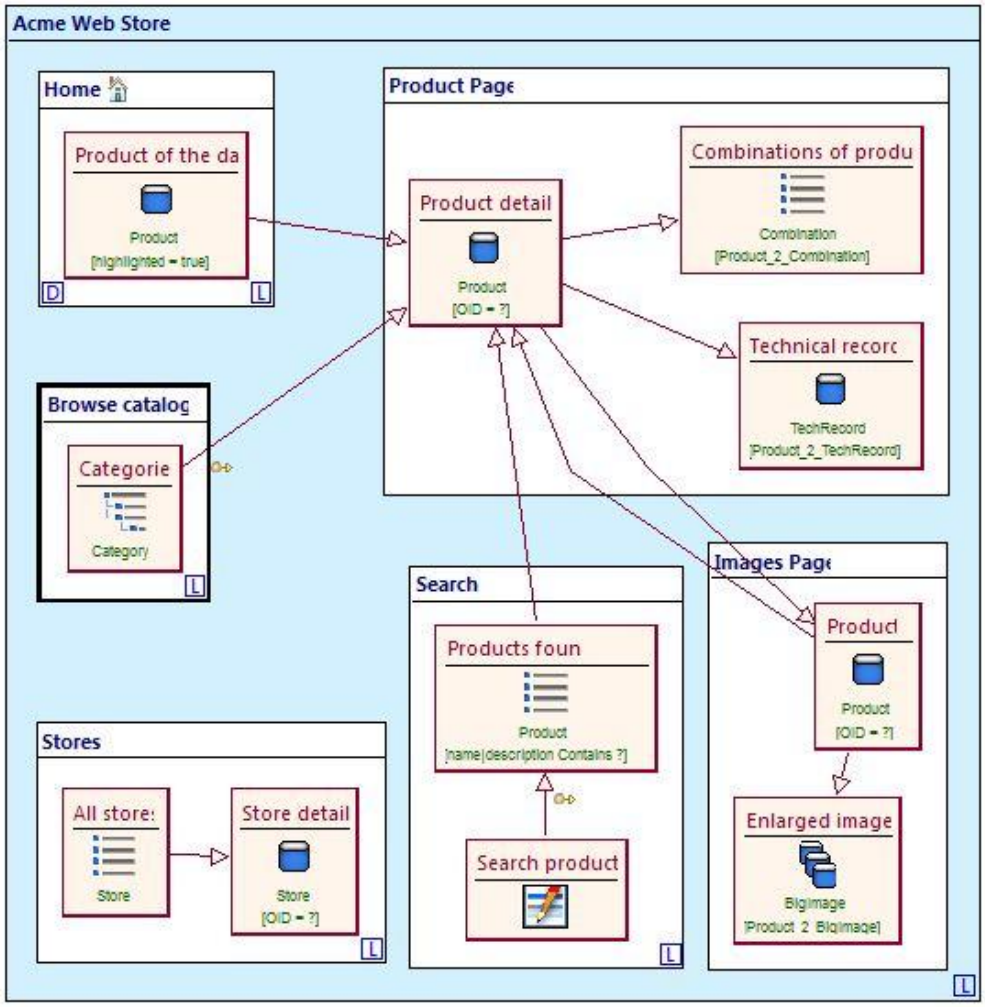
Гипертекстовая модель Интернет-магазина Acme Web Store в редакторе WebRatio.



Вариативная модель Acme Web Store в редакторе PLWeb.



Вариативная модель продукта из семейства Acme Web Store



Гипертекстовая модель продукта из семейства Acme Web Store.