УДК 04.004

П. П. Олейник, канд. техн. наук, системный архитектор программного обеспечения, ОАО "Астон", доцент, Шахтинский институт (филиал) Южно-Российского государственного политехнического университета им. М. И. Платова, Ростов-на-Дону, xsl@list.ru,
В. И. Гурьянов, канд. техн. наук, доц.,
Филиал Санкт-Петербургского государственного экономического университета в г. Чебоксары, Чебоксары, vg2007sns@rambler.ru

UML-профиль для метамодельно-ориентированного проектирования программных приложений баз данных

В настоящее время часть вновь разрабатываемых приложений являются программными приложениями баз данных, для реализации функциональных возможностей которых чаще всего используют объектно-ориентированные (ОО) языки программирования, позволяющие создавать повторно-используемые фрагменты кода, что достигается применением инкапсуляции, наследования и полиморфизма. Дальнейшим развитием ОО-подхода является распространение его принципов на разработку всех уровней приложения от структуры БД и до графического интерфейса пользователя. В итоге разработчику предоставляется единая среда разработки, позволяющая создать конечное приложение. При этом на разработчика возлагается задача корректного проектирования и создания модели предметной области в понятиях выбранного инструмента.

В данной статье представлен UML-профиль, который многократно использован авторами при разработке приложений баз данных в собственной среде разработки. Этот профиль позволяет выполнить метамодельно-ориентированное проектирование приложения и описывает модель предметной области в понятиях развитой объектной метамодели. В конце статьи представлены примеры применения профиля и выдвинуты идеи о путях развития работы.

Ключевые слова: *UML, пользовательский профиль UML, объектно-ориентированные базы данных, MDA, программные приложения баз данных, объектно-ориентированное проектирование, предметно-ориентированное проектирование, проектирование баз данных*

Введение

В настоящее время часть вновь разрабатываемых приложений являются программными приложениями баз данных, для реализации функциональных возможностей которых чаще всего используют объектно-ориентированные (ОО) языки программирования, позволяющие создавать повторно-используемые фрагменты кода, что достигается применением инкапсуляции, наследования и полиморфизма. Дальнейшим развитием ОО-подхода является распространение его принципов на разработку всех уровней приложения от структуры БД и до графического интерфейса пользователя. В итоге разработчику предоставляется единая среда разработки, позволяющая создать конечное приложение. При этом на разработчика возлагается задача корректного проектирования и создания модели предметной области (ПрО) в понятиях выбранного инструмента.

Однако моделирование ПрО в понятиях среды разработки требует кардинального изменения мышления разработчика, что заставляет искать подходы к преодолению этого препятствия. Таковым подходом может стать технология Model-driven architecture (MDA), смысл которой сводится к тому, что разработчик работает с UML-моделью, а код для языка программирования или платформы генерируется автоматически. Технически это можно реализовать посредством UML-профиля; элементы профиля поставляют данные, необходимые для генерации корректного кода. Профиль UML — это совокупность стереотипов, ограничений и помеченных значений, предназначенных для адаптации языка UML под специфические задачи проектирования, свойственные конкретным ПрО. Профиль создается посредством механизмов расширения UML, определенных в самом языке UML.

Ряд современных UML-редакторов поддерживают режим создания и работы с пользовательскими UML-профилями.

В данной работе представлен UML-профиль, который многократно использован авторами при разработке приложений баз данных в собственной среде разработки. Этот профиль позволяет выполнить метамодельно-ориентированное проектирование приложения и описывает модель предметной

области в понятиях развитой объектной метамодели. В конце статьи представлены примеры применения профиля и выдвинуты идеи о путях развития работы.

1. Обзор имеющихся работ

Благодаря своей гибкости и наличию множества спецификаций UML позволяет расширять существующую нотацию и добавлять новую семантику к существующим элементам (графическим объектам) диаграмм и тем самым использовать полученную нотацию для упрощения моделирования приложений различного назначения и для разных областей знаний [1, 2]. Этот подход активно применяют многие авторы, которые используют собственные UML-профили для своих потребностей (для адаптации UML под специфические задачи проектирования). Так, в работе [3] предложен подход концептуального проектирования реляционных баз данных с помощью построения диаграмм использования прецедентов (Use Case).

Весь процесс концептуального проектирования БД авторы разделяют на четыре этапа, после каждого из которых выполняется построение определенной UML-диаграммы. Результатом выполнения первого этапа является отображение диаграммы прецедентов для всех бизнес-процессов, которые необходимо отобразить сущностями в БД. После выполнения второго этапа авторы получат подробную диаграмму деятельности. На третьем этапе выполняется построение общей диаграммы классов, содержащей только информацию о классах, наследование и некоторые важные отношения ассоциаций. Завершающий четвертый этап позволяет построить подробную диаграмму классов, содержащую все выделенные классы и пригодную после использования правил отображений, описанных авторами, для реализации в РСУБД в виде набора реляционных отношений. Авторы подробно описали разработанный подход на примере проектирования приложений визового центра.

В работе [4] авторы предлагают UML-профиль, позволяющий упростить процесс разработки программных продуктов для промышленных объектов, которые управляют процессом выпуска продукции. Специфика данного ПО заключается в том, что оно работает в режиме реального времени под управлением соответствующей операционной системы с поддержкой параллельности на уровне ядра. При этом само предложение обеспечивает определенный уровень доступности услуг. Все это авторы учли в своем профиле и выделили пользовательские стереотипы для описания входных и выходных портов, контрольных устройств и таймеров, позволяющих их опрашивать.

Решение аналогичной задачи описано в работе [5], где рассматривается процесс обработки информации как набор выделенных авторами фаз и описывается принцип передачи информации от одной фазы на другую. Сделано это в собственной среде разработки, построенной на базе Eclipse, которая управляется UML-диаграммами, описываемыми на языке SysML. Авторы предлагают использовать модельно-управляемую архитектуру и реализуют данный подход в виде набора плагинов. Затем разработчик создает с помощью предложенного профиля программно-независимую модель (PIM) и определяет правила трансформации в программно-зависимую модель (PDM). Сам процесс разработки представляет собой цепочку последовательных детализаций требования к системе, от высокоуровневого описания до построения диаграмм активности.

В работе [6] описан профиль, позволяющий моделировать высокоуровневую архитектуру любого программного продукта. Авторы представили набор введенных ими пользовательских стереотипов и краткие примеры их использования. В заключение дан комплексный пример и описаны основные атрибуты каждой представленной сущности на диаграмме классов.

В работе [7] описан подход, применяемый авторами при проектировании баз данных программных приложений реального времени, и представлен UML-профиль, используемый для решения задачи. Структура базы данных была логически разделена на методы, объекты и атрибуты реального времени. Затем в разработанном UML-профиле были выделены базовые типы данных и пользовательские стереотипы, а также описаны правила установления ассоциаций между объектами. В конце работы представлены правила трансформаций из UML-профиля в пользовательский тип данных, объявленный с помощью объектных расширений языка SQL.

В работе [8] авторы представляют пользовательский профиль, используемый для проектирования приложений, выполняющий загрузку информации в хранилища данных, а также их подготовку, т. е. для ETL-систем (Extract/Transform/Load). При этом авторы выделили набор базовых операций, которые выполняют системы, и на их основе определили пользовательские стереотипы. В конце работы представлен пример использования профиля и графический интерфейс реализованного приложения.

В работе [9] представлен профиль, позволяющий описать паттерны объектно-реляционного отображения (ORM). Это сделано для применения модельно-управляемой архитектуры при разработке приложений баз данных. Авторы ввели множество пользовательских типов данных и описали набор правил трансформации на РСУБД. При этом потребовалось описать такие объекты баз данных, как таблицы, первичные ключи, ограничения и правила соединения таблиц, а также некоторые синтаксические конструкции языка SQL. В заключение представлен пример отображения иерархии классов на таблицы БД.

В работе [10] описан профиль, применяемый авторами при проектировании геоинформационных систем и реализованный в UML-редакторе Enterprise Architect. Этот профиль предназначен для модельно-ориентированной архитектуры и задает набор правил трансформации моделей. В статье представлена тестовая предметная область и продемонстрирована поэтапная ее реализация вплоть до генерации SQL-скриптов для создания БД.

В работе [11] представлен профиль для проектирования сложных веб-ориентированных приложений. Профиль предполагает выделение трех этапов проектирования. На первом этапе выполняется концептуальное проектирование приложения, на втором этапе — навигационное проектирование, позволяющее определить правила перехода между страницами в приложении. А на заключительном этапе происходит проектирование графического интерфейса пользователя. Для реализации этого подхода авторами был создан собственный фреймворк, состав и структура модулей которого описаны в статье.

В работе [12] представлен модельно-ориентированный подход к проектированию мобильных приложений независимо от целевой платформы исполнения и без привязки к предметной области. Авторы предложили собственный UML-профиль, который применяется для построения диаграмм прецедентов, диаграмм классов и диаграмм конечных автоматов. Затем происходит генерация программного кода по созданной модели предметной области.

Консорциум OMG рекомендует использовать UML-профили как для поддержки технологии MDA для различных платформ и языков программирования, так и для хорошо определенных предметных областей [2]. Выше дан обзор разработки и использования профилей для различных платформ. Кратко рассмотрим применение профилей для моделирования предметных областей. Хорошо известными профилями моделирования являются: язык SysML/UML, предложенный OMG в качестве инструмента инженерного проектирования, и многочисленные профили моделирования бизнес-процессов, такие как Ericsson-Penker Profile и RUP Business Modeling. Моделирование предметных областей возможно потому, что UML — это формальный язык и элементам языка можно назначать разную семантику, а не только вычислительную. Из последних работ в этой области можно выделить работы по разработке профилей для моделирования онтологий, таких как OUP (Ontology UML Profile). Один из авторов данной статьи предложил UML-профиль для объектно-ориентированного имитационного моделирования [13, 14], который также можно отнести к этому направлению. Этот UML-профиль позволяет отображать концептуальные модели систем непосредственно на диаграммы классов, что дает возможность создавать объектные имитационные модели. С одной стороны, этот профиль разрешает использовать *унифицированный процесс* для разработки имитационных моделей, с другой стороны, позволяет читать диаграммы классов как фреймовые семантические сети, что делает возможным выполнять валидацию имитационных моделей с самых первых шагов их разработки.

2. Метамодельно-ориентированный подход к разработке приложений баз данных

В настоящее время при проектировании новых программных продуктов на объектно-ориентированном языке программирования используется предметно-ориентированный подход (Domain-Driven Design, DDD) [15]. В результате применения предлагаемых наборов принципов и схем разработчикам удается создавать программные абстракции, которые называются моделями предметных областей и представляются в виде диаграмм классов унифицированного графического языка моделирования UML. Подход DDD полезен в ситуациях, когда разработчик программного продукта не является экспертом в прикладной предметной области разрабатываемого продукта и сталкивается с ней впервые. Он не может знать все области, но с помощью правильного представления структуры, посредством предметно-ориентированного подхода может быть спроектировано работоспособное приложение.

Полученная в ходе DDD диаграмма классов может быть реализована с помощью любого объектно-ориентированного языка программирования в выбранной среде разработки. Современные среды разработки используют модельно-управляемую архитектуру (Model-Driven Architecture, MDA). То есть создается определенная модель (представляющая модификацию модели предметной области), которая реализуется в выбранном инструменте разработки с использованием предоставленных средств.

В этом разделе представлено дальнейшее развитие DDD, которое названо авторами метамодельно-ориентированным подходом, основанным на создании экземпляров метаклассов объектной метамодели. Решение авторы многократно применяли при реализации различных проектов. По сути разработчику предоставляется несколько видов сущностей, которые используются для построения модели предметной области. Виды создаваемых сущностей могут быть следующих типов.

1. **Сохраняемые.** Их используют для представления информации, которая физически хранится в базе данных. Описание подобных сущностей выполняется путем создания экземпляров метакласса DomainClass.

2. **Вспомогательные.** Экземпляры этих сущностей не сохраняются в БД, но их представление в системе необходимо для упрощения как в реализации бизнес-логики, так и для отображения вспомогательной информации в интерфейсе пользователя.

Описание подобных сущностей выполняется путем создания экземпляров метакласса HelperClass.

3. **Классы-параметры методов.** Основная бизнес-логика приложения реализуется в виде методов классов. Этот подход соответствует объектно-ориентированной парадигме. В конечном итоге методы представляются пользователю в виде элементов графического интерфейса, например в виде кнопок. При этом методы могут принимать набор параметров. Описание подобных сущностей выполняется с помощью создания экземпляров метакласса MethodParameterClass.

4. **Классы-перечисления/множества.** Эти классы используют в том случае, когда атрибут может принимать определенное значение из описанного ранее набора и этот набор не может расширяться пользователями. Описание подобных сущностей выполняется с помощью создания экземпляров метакласса Enum.

На рис. 1 представлен фрагмент метамодели, содержащий иерархию метаклассов, которые применяют для описания сущностей предметной области. Эта иерархия была разработана в унифицированной среде быстрой разработки корпоративных приложений SharpArchitect RAD Studio [16], которая была протестирована при реализации различных приложений [17—20].

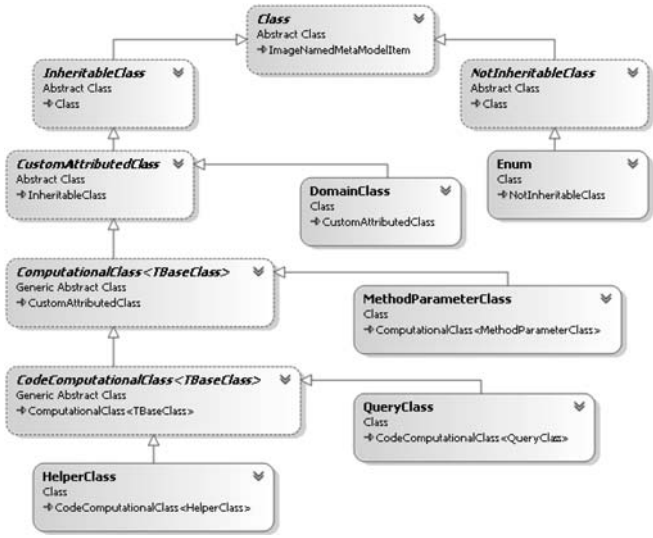

Рис. 1. Фрагмент метамодели, содержащий иерархию метаклассов описания сущностей предметной области

На рис. 2 представлен фрагмент метамодели, содержащий ассоциации между метаклассами.

Из рисунков видно, что представленная метамодель ортогональна по отношению к различным

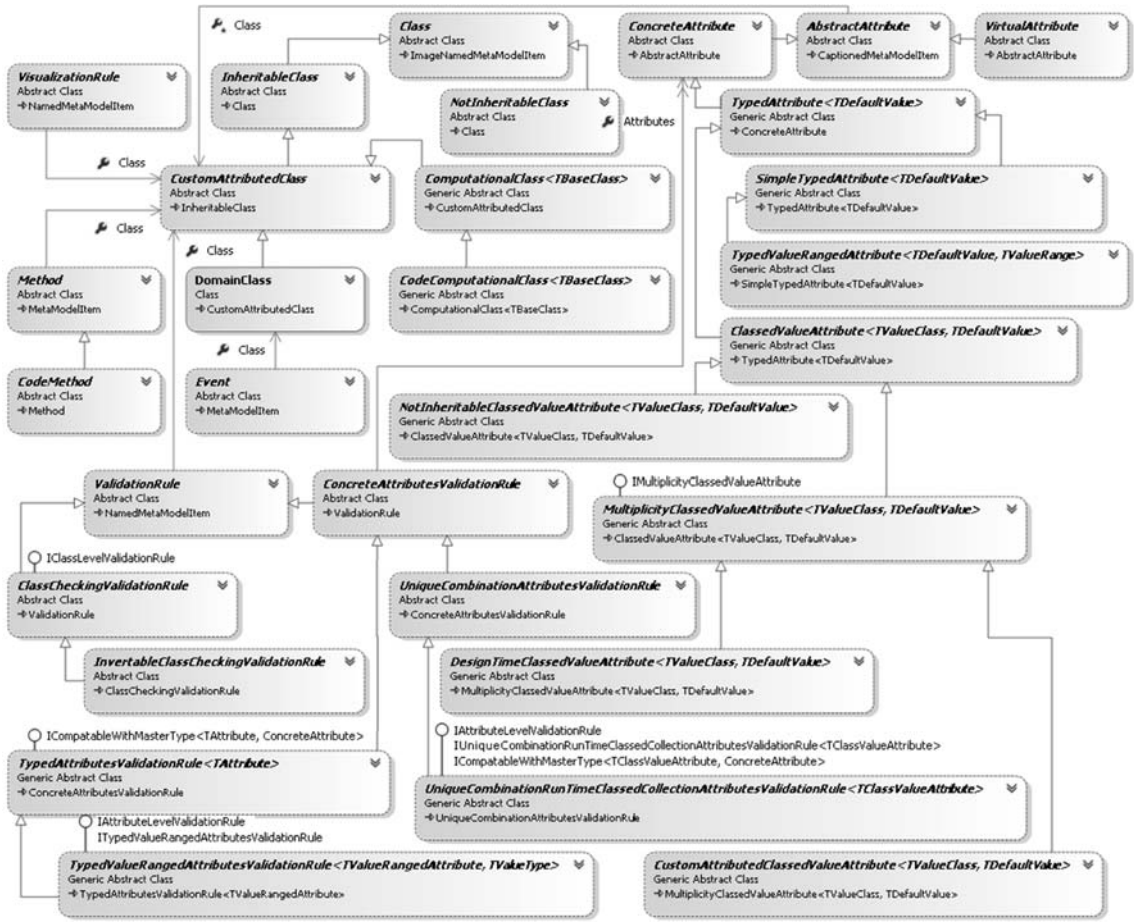

Рис. 2. Фрагмент метамодели, содержащий ассоциации между метаклассами

предметным областям, поэтому ее удалось применить во многих крупных приложениях. При этом данная метамодель используется на всех уровнях приложения, начиная от генерации структуры БД и заканчивая формированием графического интерфейса пользователя. В результате предлагаемый подход метамодельно-ориентированного проектирования имеет следующие преимущества:

- автоматическое создание интерфейса пользователя на основе модели приложения с возможностью настройки конечным пользователем;
- редактор модели приложения позволяет вносить изменения, а именно — описывать классы, связывать их между собой в соответствии с UML-диаграммами и задавать сигнатуру методов в момент выполнения приложения; при этом для вступления изменений достаточно перезапустить приложение одного конкретного пользователя (остальные могут продолжить работу);
- возможность декларативного описания валидационных правил, позволяющих проверять корректность и непротиворечивость данных в момент сохранения их в базе данных;
- возможность декларативного описания визуализационных правил, позволяющих выделить с помощью изменения цвета различные графические элементы на форме, а также управлять видимостью и доступностью элементов;
- множество системных (встроенных) классов, которое позволяет упростить реализацию поведения часто используемых видов сущностей (например, в системе имеется внутренняя поддержка древовидных структур).

3. Необходимость разработки собственного UML-профиля

Разработка собственного профиля — кропотливый и трудоемкий процесс. При проектировании приложений баз данных в рассмотренной среде разработки SharpArchitect RAD Studio один из авторов ранее использовал иной подход, основанный на создании диаграмм объектов, содержащих экземпляры метаклассов, описывающих сущности предметной области (см. рис. 1) [21—22]. На рис. 3 представлен используемый подход.

Несмотря на то что описанное решение было протестировано многократно, за время эксплуатации обнаружены следующие проблемы.

1. Полученная диаграмма объектов не похожа на первоначальную диаграмму классов. Это становится серьезной проблемой при доработке исходной диаграммы. Так как нет прямого соответствия, то не ясно, куда именно вносить изменения в диаграмме объектов.

2. Полученная диаграмма более громоздкая по сравнению с исходной. Эта проблема становится очень серьезной при проектировании программного приложения, в котором имеются сотни различных сущностей.

3. Для моделирования повсеместно используются экземпляры классов, что стирает грань между атрибутами и классами предметной области. Диаграмма классов оперирует такими понятиями, как класс, атрибут, ассоциация. С концептуальной точки зрения это совершенно различные сущности, которые нельзя смешивать. В предложенном подходе они смешаны.

4. Представленный подход требует от проектировщика глубокого знания не только диаграмм классов, но и диаграмм объектов. Это особенно сложно для людей, начинающих изучать UML, например для студентов вузов.

5. Наличие на диаграмме экземпляров вспомогательных классов, которые отсутствуют в пред-

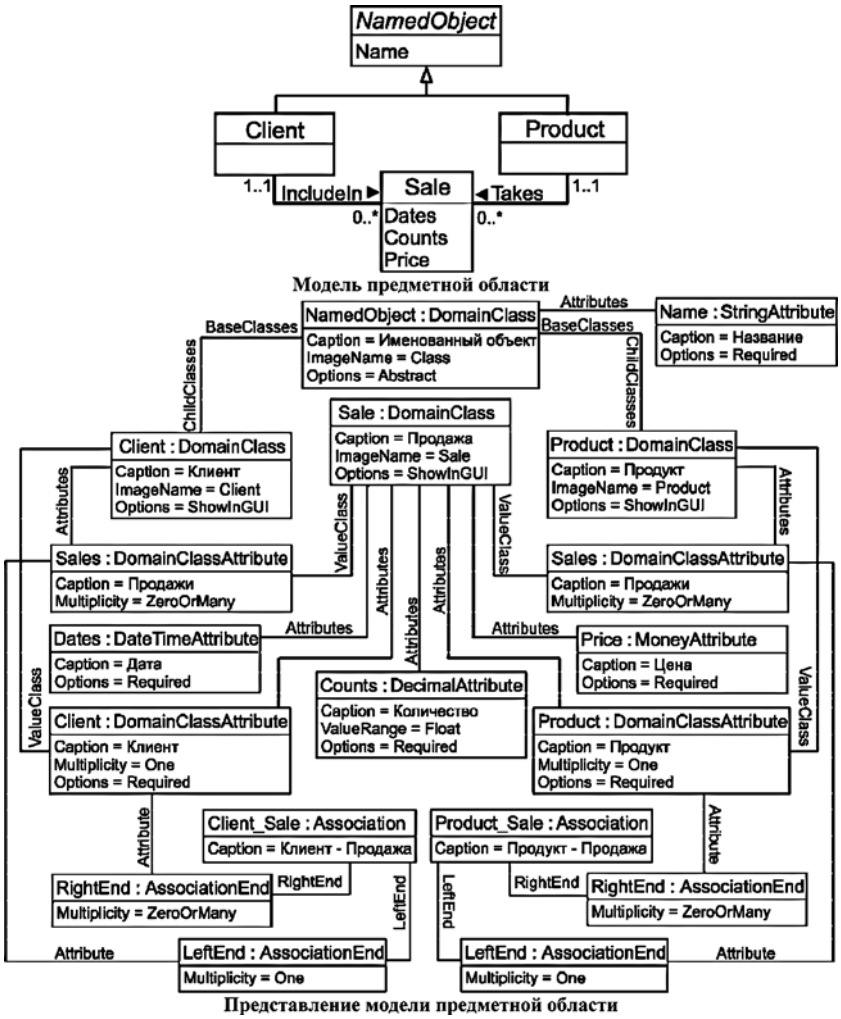

Рис. 3. Проектирование приложения в понятиях метамодели объектной системы с применением диаграммы объектов

метной области. Например, класс AssociationEnd необходим для представления края ассоциации и не присутствует в предметной области. Однако при описании ассоциации экземпляры этого класса загромождают диаграмму объектов.

Анализ выделенных недостатков свидетельствует, что достоинств гораздо меньше. В результате возникла необходимость поиска альтернативного решения. В нашем случае оптимальным решением является разработка собственного профиля.

4. Состав и структура разработанного UML-профиля

UML активно используется при проектировании различных элементов информационных систем, в том числе приложений баз данных. Известны также профили UML, предназначенные для проектирования структуры реляционных БД (РБД). Далее предлагается профиль SharpArhitect UML Profile (SAUP), который предназначен для проектирования ООБД в понятиях метамодели объектной системы [23].

В настоящее время объектно-ориентированный подход используется повсеместно, в том числе и при реализации уровня хранения данных. Использование ОО-парадигмы на уровне БД позволяет применять такие механизмы, как наследование, что отсутствует в других моделях данных, например в РБД. Так, мы может объявить базовый класс и реализовать в нем наиболее общие атрибуты, а затем унаследовать их в производных классах. При использовании РСУБД разработчику необходимо в каждом отношении, реализуемом таблицей, повторно объявлять одинаковые атрибуты. Также ОО-подход позволяет реализовать древовидные (иерархические) структуры с реализацией рекурсивных запросов, а это довольно трудоемкая задача для реляционной модели данных.

Профиль ориентирован на среду разработки SharpArhitect RAD Studio. Профиль SAUP разработан на основе спецификации UML 2.4.1 [2]. Процесс разработки профиля предполагает выполнение следующих шагов: 1) определение метамодели платформы; 2) с использованием элементов метамодели платформы определение множества стереотипов на базе существующих элементов метамодели UML; 3) для каждого стереотипа определение помеченных значений (*tagged values*) и ограничений (*constraints*). Множество стереотипов помещается в пакет со стереотипом "*profile*", и выполняется распределение стереотипов по семантически однородным группам. В качестве метамодели платформы используется метамодель, приведенная на рис. 1. Общая структура профиля показана на рис. 4.

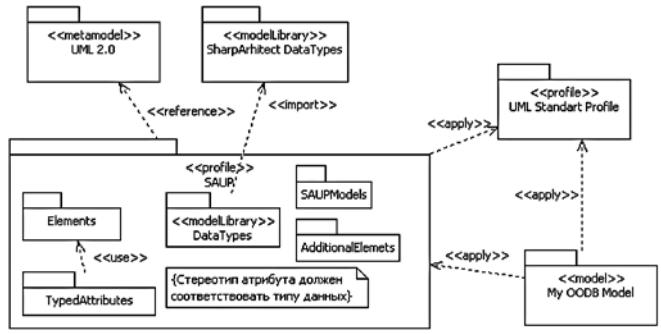

Рис. 4. Архитектура UML-профиля SharpArchitect UML Profile (SAUP)

Пакет Elements определяет элементы профиля, моделирующие сущности предметной области. Пакет TypedAttributes определяет набор стереотипов для атрибутов с типами данных, определенных в пакете DataTypes, который, в свою очередь, импортирует типы данных из среды разработки SharpArhitect RAD Studio (см. рис. 4). В профиле задано ограничение — стереотипы атрибутов должны соответствовать типам данных. Это ограничение задается специальной таблицей, где каждому типу ставится в соответствие стереотип. Пакет AdditionalElements определяет стереотипы, предназначенные для моделирования вспомогательных классов. Пакет SAUPModels дает определение моделям SAUP. Для построения метамодели профиля использованы элементы стандартного профиля UML (стереотипы и типы данных).

Структура пакета Elements показана на рис. 5. Каждый стереотип определяет некоторое множество помеченных значений, которые поставляют данные классам среды проектирования SharpArhitect RAD Studio (см. рис. 1, 2). Стереотип **DomainClass** определяет ряд помеченных значений, которые поставляют данные классу предметной области DomainClass. Стереотип **DomainAssociation** поставляет данные классам Association и AssociationEnd. Стереотип **EnumClass** в качестве метакласса использует элемент **Class**, а не элемент **Enumeration**. Стереотип ***TypedAttribute*** определяет общие помеченные значения для классов типов данных (рис. 6). Поскольку все эти стереотипы расширяют один и тот

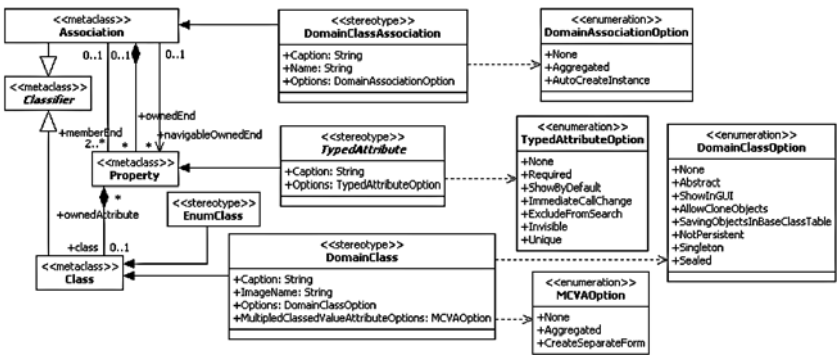

Рис. 5. Структура пакета Elements

же UML-элемент **Property**, все они сгруппированы в пакете TypedAttributes. Стереотипы атрибутов, определяемые в пакете TypedAttributes, находятся в отношении обобщения со стереотипом ***TypedAttribute***, что отражено в зависимости пакета TypedAttributes от пакета Elements.

Девять стереотипов (**DecimalAttribute**, **EnumAttribute**, **FileDataAttribute**, **HelperClassAttribute**, **IntAttribute**, **MetaModelClassAttribute**, **TextAttribute**, **TypeAttribute**, **StringAttribute**) имеют дополнительные помеченные значения. Остальные стереотипы атрибутов просто наследуют помеченные значения от стереотипа ***TypedAttribute***.

Структура пакета AdditionalElements представлена на рис. 7.

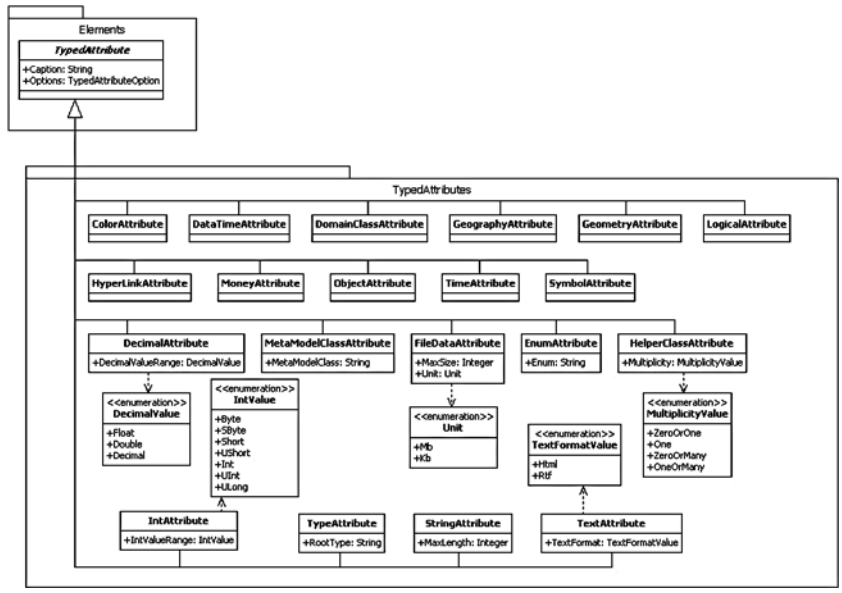

Рис. 6. Структура пакета *TypedAttributes*

Пакет AdditionalElements определяет ряд дополнительных стереотипов для определения сущностей, отличающихся от сущностей предметной области. Стереотип **HelperClass** помечает вспомогательные классы, которые не хранятся в ООБД. Стереотип **SystemClass** помечает системные классы, например BaseRunTimeTreeNodeDomainClass. Эти классы предназначены для реализации типовых решений, таких как паттерны проектирования. Два стереотипа **Parameter** и **MethodParameterClass** предназначены для отображения класса-параметра MethodParameterClass. Кроме того, в этом пакете определен стереотип **N-AryAssociation**, который обозначает *n*-арные ассоциации. В качестве метаклассов использованы UML-элементы **Class** и **Association**. В отличие от других стереотипов, этот стереотип не имеет аналогов в метамодели и предназначен для концептуального проектирования.

Пакет SAUPModels содержит определение всего одного стереотипа **SAUPModel** на основе метакласса **Model** (из пакета Models). Тем самым модель SAUP определяется как пакет для размещения как элементов профиля, так и элементов UML.

Как видно из краткого описания SAUP, разработанный профиль позволяет проектировать объ-

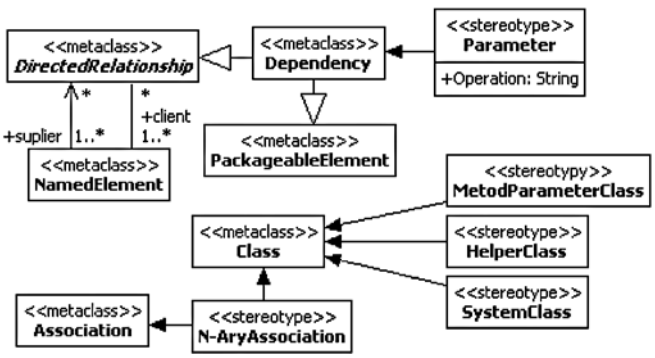

Рис. 7. Структура пакета AdditionalElements

ектно-ориентированные приложения баз данных для различных прикладных предметных областей. При этом проектирование с применением UML профиля по сути представляет собой построение диаграммы классов, что вполне логично и естественно по сравнению с построением диаграмм объектов метаклассов объектной системы, предложенным в работах [21, 22]. Авторы считают это решение оптимальным и намерены использовать в будущем только его.

5. Примеры использования разработанного UML-профиля

Рассмотрим несколько примеров применения профиля. Для демонстрации возможностей профиля необходимо рассмотреть относительно большую модель, содержащую множество классов и отношений между ними. Некоторое время назад для демонстрации собственной среды разработки SharpArchitect RAD Studio и реализованных в ней функциональных возможностей потребовалось спроектировать тестовую предметную область. Для модели были выделены критерии оптимальности и получена соответствующая реализация. В результате подход был обобщен, и появилась унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений баз данных [23, 24]. Именно эта модель изображена в нотации SAUP на рис. 8.

Стереотипом "N-AryAssociation" помечена *n*-арная ассоциация **Position**. На диаграмме также присутствуют стандартные UML-элементы, например класс-ассоциация. Стереотипы отчетливо показывают, какие элементы модели должны быть доопределены. На диаграмме в качестве примера для классов со стереотипом DomainClass явно показаны

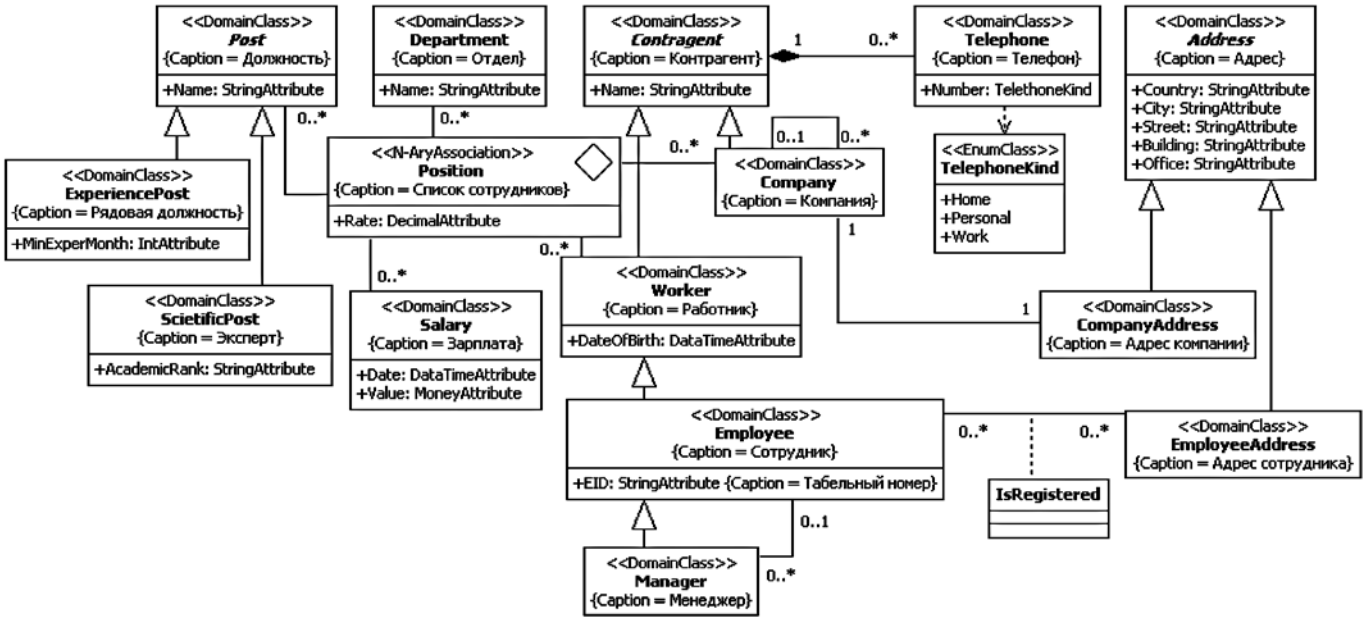

Рис. 8. UML-диаграмма классов унифицированной модели тестирования инструментов разработки объектно-ориентированных приложений, построенная с применением профиля SAUP

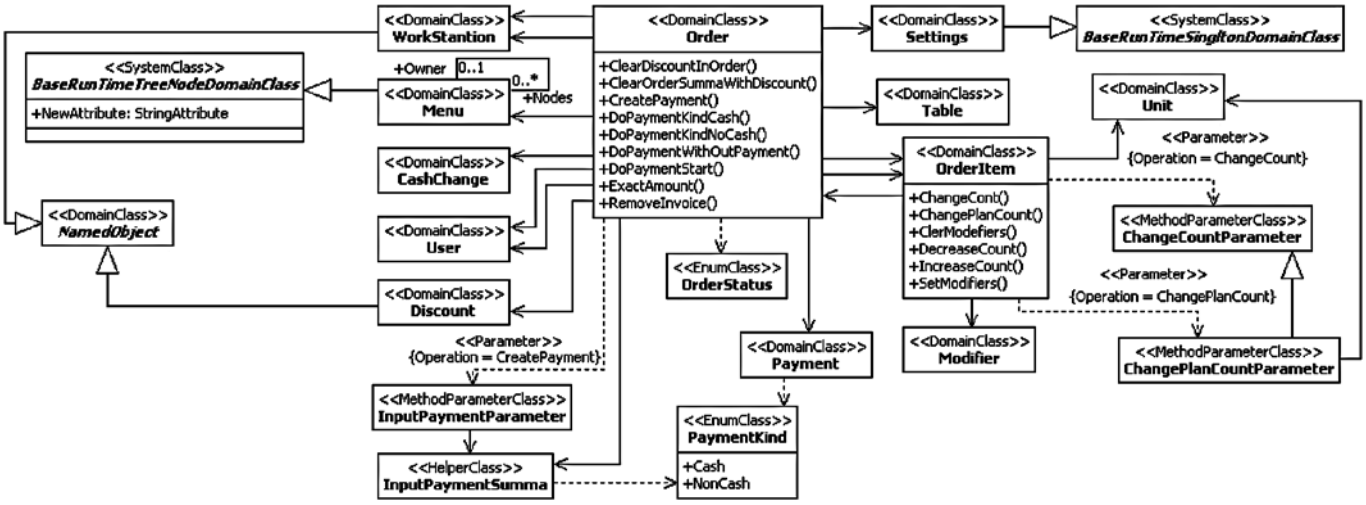

Рис. 9. Фрагмент UML-диаграммы классов информационной системы для ресторанов быстрого питания

помеченные значения Caption, а для класса **Employee** также одно из помеченных значений стереотипа атрибута ED. Хотя все атрибуты проектной модели должны быть помечены стереотипами, их можно не отображать на диаграммах. Точно так же можно скрывать помеченные значения стереотипов. Это позволяет проводить концептуальное и логическое проектирование, не детализируя спецификацию UML-элементов.

Как видно из рис. 8, на нем представлены только классы предметной области, позволяющие описать сохраняемые типы сущностей. В работе [26] при прототипировании графической формы заказа в приложении, разработанном для сенсорного экрана, потребовалось объявить множество дополнительных классов (HelperClass и др., см. рис. 1). У авторов возник ряд проблем при отображении диаграммы классов. Ниже представлено решение с помощью профиля SAUP (рис. 9).

Данная диаграмма иллюстрирует использование стереотипов пакета AdditionalElements профиля SAUP. Системный класс ***BaseRunTimeTreeNodeDomainClass*** реализует паттерн *Composite* и помечается стереотипом "SystemClass". Иногда возникает необходимость добавить дополнительные атрибуты в этот и другие системные классы. Это показывается просто как дополнительный атрибут этого класса с типом из библиотеки SharpArhitect Data Types. Класс-параметр **InputPaymentParameter** помечен стереотипом "MethodParameterClass" и находится в отношении зависимости с классом **Order**. Отношение зависимости помечается стереотипом "Parameter" и имеет помеченное значение Operation, которое называет операцию, для которой соз-

дается параметр. Вспомогательный класс **InputPaymentSumma** используется при оплате заказа как наличным, так и безналичным расчетом и представлен для демонстрации использования параметров методов.

Приведенные примеры наглядно демонстрируют, что применение профиля SAUP позволяет организовать мышление проектировщика в понятиях метамодели объектной системы, сохраняя возможность проектирования в диаграммах классов UML.

Заключение

Обзор литературы показал, что благодаря своей гибкости UML можно применять для приложений различных назначений, для этого достаточно разработать собственный профиль. В данной работе представлен процесс метамодельно-ориентированного проектирования приложений баз данных, суть которого заключается в создании экземпляров метаклассов объектной системы. При этом авторы разработали собственный UML-профиль, позволяющий упростить представленный подход. Описанные примеры доказали зрелость предложенного подхода. Дальнейшим развитием профиля является представление динамической составляющей, а именно методов классов, валидационных и визуализационных правил.

Список литературы

1. **The UML** and Data Modeling, White Paper: Technical report, Rational Software, 2003.
2. **Object** Management Group. UML 2.0 Infrastructure Specification, OMG document number formal/2011-08-05, 2011.
3. **Brdjanin D., Maric S.** An Example of Use-Case-driven Conceptual Design of Relational Database // EUROCON, 2007. The International Conference on "Computer as a Tool", 9—12 Sept. 2007, Warsaw, pp. 538—545, DOI: 10.1109/EURCON.2007.4400437
4. **Ritala T., Kuikka S.** UML Automation Profile: Enhancing the Efficiency of Software Development in the Automation Industry // 5th IEEE International Conference on Industrial Informatics, 23—27 June 2007, Vienna. P. 885—890. DOI: 10.1109/INDIN.2007.4384890
5. **Vepsalainen T., Hastbacka D., Kuikka S.** Tool Support for the UML Automation Profile — For Domain-Specific Software Development in Manufacturing // The Third International Conference on Software Engineering Advances, ICSEA '08, 26—31 Oct. 2008, Sliema. P. 43—50. DOI: 10.1109/ICSEA.2008.22
6. **Hengye Zhu, Guangyao Li, Liping Zheng.** A UML profile for HLA-based simulation system modeling // 6th IEEE International Conference on Industrial Informatics, INDIN 2008, 13—16 July 2008, Daejeon. P. 1602—1607. DOI: 10.1109/INDIN.2008.4618360
7. **Idoudi N., Duvallet C., Bouaziz R., Sadeg B., Gargouri F.** How to Model a Real-Time Database? // Proc. of IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC '09, 17—20 March 2009, Tokyo. P. 321—325. DOI: 10.1109/ISORC.2009.17
8. **Xudong Song, Xiaolan Yan, Liguo Yang.** Design ETL Metamodel Based on UML Profile. Second International Symposium on Knowledge Acquisition and Modeling, KAM '09, Nov. 30. Dec. 1, 2009, Wuhan. P. 69—72. DOI: 10.1109/KAM.2009.112
9. **Torres A., Galante R., Pimenta M. S.** Towards a UML Profile for Model-Driven Object-Relational Mapping // Proc. of XXIII Brazilian Symposium on Software Engineering, SBES '09, 5—9 Oct. 2009, Fortaleza, Ceara. P. 94—103. DOI: 10.1109/SBES.2009.22
10. **Ferreira T. B., Stempliuc S. M., Lisboa-Filho J.** Geographical data modeling with the UML GeoProfile and MDA transformations on the Enterprise Architect Tool // Proc. of 9th Iberian Conference on Information Systems and Technologies (CISTI), 18—21 June 2014, Barcelona. P. 1—6. DOI: 10.1109/CISTI.2014.6876987
11. **Mubin S. A., Jantan A. H.** A UML 2.0 profile web design framework for modeling complex web application. International Conference on Information Technology and Multimedia (ICIMU), 18—20 Nov. 2014, Putrajaya. P. 324—329. DOI: 10.1109/ICIMU.2014.7066653
12. **Usman M., Iqbal M. Z., Khan M. U.** A Model-Driven Approach to Generate Mobile Applications for Multiple Platforms // Proc of 21 st Asia-Pacific Software Engineering Conference (APSEC), 1—4 Dec. 2014, Jeju. P. 111—118. DOI: 10.1109/APSEC.2014.26
13. **Гурьянов В. И.** Моделирование классификаций на визуальном языке имитационного моделирования UML SP // Сборник докладов Шестой всероссийской научно-практической конференции "Имитационное моделирование. Теория и практика" (ИММОД-2013). Т. 1. Казань: Изд-во "ФЭН" АН РТ, 2013. С. 128—132.
14. **Гурьянов В. И.** Имитационное моделирование на UML SP. Чебоксары: Изд-во Филиала СПбГЭУ в г. Чебоксары, 2014. 135 с.
15. **Эванс Э.** Предметно-ориентированное проектирование (DDD). Структуризация сложных программных систем. М.: Вильямс, 2010, 423 с.
16. **Олейник П. П.** Программа для ЭВМ "Унифицированная среда быстрой разработки корпоративных информационных систем SharpArchitect RAD Studio". Свид-во о гос. регистрации № 2013618212 от 04 сентября 2013 г.
17. **Олейник П. П.** Элементы среды разработки программных комплексов на основе организации метамодели объектной системы // Бизнес-информатика. 2013. № 4 (26). С. 69—76. URL: http://biiournal.hse.ru/data/2014/01/16/1326593606/IBI%204(26)%202013.pdf
18. **Олейник П. П., Кураков Ю. И.** Концепция создания обслуживающей корпоративной информационной системы экономического производственно-энергетического кластера // Прикладная информатика. 2014. № 6. С. 5—23.
19. **Oleynik P. P., Nikolenko O. I., Yuzefoya S. Yu.** Information System tor Fast Food Restaurants // Engineeiring and Technology. 2015. Vol. 2, No. 4. P. 186—191. http://article.aascit.org/file/pdf/9026895.pdf
20. **Oleynik P. P.** Metamodel-Driven Design of Database Applications. // Journal of Computer Science Technology Updates. 2015. Vol. 2,_ No. 1. P. 15—24. http://www.cosmosscholars.com/images/JCSTU-v1n1/JCSTU-V2-N1/JCSTU-V2N1A3-Oleynik.pdf
21. **Олейник П. П.** Предметно-ориентированное проектирование структуры базы данных в понятиях метамодели объектной системы // Объектные системы — 2014: материалы VIII Международной научно-практической конференции (Ростов-на-Дону, 10—12 мая 2014 г.) / Под общ. ред. П. П. Олейника. Ростов-на-Дону: ШИ (ф) ЮРГПУ (НПИ) им. М. И. Платова, 2014. С. 41—46. URL: http://obiectsystems.ru/files/2014/Obiect_Systems_2014_Proceedings.pdf
22. **Oleynik P. P.** Using metamodel of object system for domain-driven design the database structure // Proc. of 12th IEEE East-West Design & Test & Symposium (EWDTS'2014), Kiev, Ukraine, September 26—29, 2014. DOI: 10.1109/EWDTS.2014.7027052
23. **Гурьянов В. И., Олейник П. П.** UML-профиль проектирования структуры объектно-ориентированной базы данных // Объектные системы — 2015: материалы X Международной научно-практической конференции. Ростов-на-Дону, 10—12 мая 2015 г. / Под общ. ред. П. П. Олейника. Ростов-на-Дону: ШИ (ф) ЮРГПУ (НПИ) им. М. И. Платова, 2015. URL: http://objectsystems.ru/files/2015/Object_Systems_2015_Proceedings.pdf
24. **Олейник П. П.** Унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений // Объектные системы — 2014 (Зимняя сессия): материалы IX Международной научно-практической конференции. Ростов-на-Дону, 10—12 декабря 2014 г. / Под общ. ред. П. П. Олейника. Ростов-на-Дону: ШИ (ф) ЮРГПУ (НПИ) им. М. И. Платова, 2014. — С. 25—35. URL: http://obiectsystems.ru/files/2014WS/Obiect_Systems_2014_Winter_session_Proceedings.pdf
25. **Николенко О. И., Олейник П. П., Юзефова С. Ю.** Прототипирование и реализация графической формы заказа для информационной системы ресторанов быстрого питания // Объектные системы — 2015: материалы X Международной научно-практической конференции. Ростов-на-Дону, 10—12 мая 2015 г. / Под общ. ред. П. П. Олейника. Ростов-на-Дону: ШИ (ф) ЮРГПУ ((НПИ) им. М. И. Платова, 2015. URL: http://obiectsystems.ru/files/2015/Obiect_Systems_2015_Proceedings.pdf

P. P. Oleynik, PhD, System Architect Software, Aston JSC, Associate Professor,
Shakhty Institute (branch) of Platov South Russian State Polytechnic University (NPI),
Russia, Rostov-on-Don, xsl@list.ru,
V. I. Gurianov, PhD in Technical Science, Assistant Professor, Saint-Petersburg State Economic University,
Branch in Cheboksary, Russia, Cheboksary, vg2007sns@rambler.ru

UML-Profile for Metamodel-Driven Design of Database Applications

Currently, most of the newly developed applications are database applications. At the same time for the functionality of the system most commonly used object-oriented programming languages that allow you to create re-usable pieces of code, which is achieved with the use of encapsulation, inheritance and polymorphism. Further development of the object-oriented approach is the extension of its principles for the development of the entire application. As a result, the developer provides a single development environment that allows you to create a target application. At the same time the developer tasked correct design and construction of domain models in terms of the selected tool.

This article presents a UML-profile that repeatedly used by the authors in the development of database applications in their own development environment. This profile allows you to metemodelno-oriented application design and describes the domain model in terms of the development of object metamodel. At the end of the article provides examples of the use of the profile and draw conclusions on ways of working.

Keywords: *UML, UML Profile, Databases, Object-Oriented Design*

References

1. **The UML** *and Data Modeling*, White Paper, Technical report, Rational Software, 2003.
2. **Object** *Management Group. UML 2.0 Infrastructure Specification*, OMG document number formal/2011-08-05, 2011.
3. **Brdjanin D., Marie S.** An Example of Use-Case-driven Conceptual Design of Relational Database, *EUROCON, 2007. The International Conference on "Computer as a Tool", 9—12 Sept. 2007, Warsaw*, pp. 538—545. DOI: 10.1109/EURCON.2007.4400437
4. **Ritala T., Kuikka S.** UML Automation Profile: Enhancing the Efficiency of Software Development in the Automation Industry, *Proc. of 5th IEEE International Conference on Industrial Informatics, 23—27 June 2007, Vienna*, pp. 885—890, DOI: 10.1109/INDIN. 2007.4584890
5. **Vepsalainen T., Hastbacka D., Kuikka S.** Tool Support for the UML Automation Profile — For Domain-Specific Software Development in Manufacturings, *Proc. of The Third International Conference on Software Engineering Advances, ICSEA '08, 26—31 Oct. 2008, Sliema*, pp. 43—50. DOI: 10.1109/ICSEA.2008.22
6. **Hengye Zhu, Guangyao Li, Liping Aheng.** A UML profile for HLA-based simulation system modelin, *Proc. of 6th IEEE International Conference on Industrial Informatics, INDIN 2008, 13—16 July 2008, Daejon*, pp. 1602—1607. DOI: 10.1109/INDIN.2008.4618360
7. **Idoudi N., Duvallet C., Bouaziz R., Sadeg B., Gargouri F.** How to Model a Real-Time Database? *Proc. of IEEE International Symposium on Object/Component/Service-Oriented Real/Time Distributed Computing, ISORC '09, 17—20 March 2009, Tokyo*, pp. 321—325, DOI: 10.1109/ISORC.2009.17
8. **Xudong Song, Xiaolan Yan, Liguo Yang.** Design ETL Metamodel Based on UML Profile, *Second International Symposium on Knowledge Acquisition and Modelibg, KAM'09, Nov. 30, 2009-Dec. 1 2009, Wuhan*, pp. 69—72. DOI: 10.1109/KAM.2009.112
9. **Torres A., Galante R., Pimenta M. S.** Towards a UML Profile for Model-Driven Object-Relational Mapping. *XXIII Brazilian Symposium on Software Engineering, SBES '09, 5—9 Oct., Fortaleza, Ceara*, pp. 94—103. DOI: 10.1109/SBES.2009.22
10. **Ferreira T. B., Stempliuc S. M., Lisboa-Filho J.** Geographical data modeling with the UMLGeoProfile and MDA transformations on the Enterprise Architect Tool, *9th Iberian Conference on Information Systems and Technologies (CISTI), 18—21 June 2014, Barcelona*, pp. 1—6. DOI: 10.1109/CISTI.2014.6876987
11. **Mubin S. A., Jantan A. H.** A UML 2.0 profile web design framework for modeling complex web application., *International Conference on Information Technology and Multimedia (ICIMU), 18—20 Now. 2014, Putrajaya*, 324—329 pp., DOI: 10.1109/ICIMU.2014.7066653
12. **Usman M., Iqbal M. Z., Khan M. U.** A Model-Driven Approach to Generate Mobile Applications for Multiple Platforms, *21 st Asia-Pacific Software Enginemng Conference (APSEC), 1—4 Dec. 2014, Jeju*, pp. 111—118. DOI: 10.1109/APSEC.2014.26
13. **Gur'yanov V. I.** Modelirovanie klassifikatsiy na vizual'nom yazyke imitatsionnogo modelirovaniya UML SP, *Sbornik dokladov Shestoy vserossiyskoy nauchno-prakticheskoy konferentsii "Imitatsionnoe modelirovanie. Teoriya i praktika" (IMMOD-2013)*, vol. 1, Kazan', Izdatel'stvo "FEN" Akademii nauk RT, 2013, pp. 128—132.
14. **Gur'yanov V. I.** *Imitatsionnoe modelirovanie na UML SP.* Cheboksary, Filial SPbGEU v. g. Cheboksary, 2014. 135 p.
15. **Evans E.** *Predmetno-orientirovannoe proektirovame (DDD). Strukturizatsiya slozhnykh programmnykh sistem*, Moscow, Vil'yams, 2010, 423 p.
16. **Oleynik P. P.** *Programma dlya EVM "Unifitsirovannaya sreda bystroy razrabotki korporativnykh informatsionnykh sistem SharpArchitect RAD Studio",* svidetel'stvo o gosudarstvennoy registratsii № 2013618212 ot 04 sent. 2013 g.
17. **Qleynik P. P.** Elementy sredy razrabotki programmnykh kompleksov na osnove organizatsii metamodeli ob''ektnoy sistemy, *Biznes-informatika*, 2013, no. 4 (26), pp. 69—76. URL: http://bijournal.hse.ru/data/2014/01/16/1326593606/1BI%204(26)%202013.pdf
18. **Oleynik P. P., Kurakov Yu. I.** Kontseptsiya sozdaniya obsluzhivayushchey korporativnoy informatsionnoy sistemy ekonomicheskogo proizvodstvenno-energeticheskogo klastera, *Prikladnaya informatika*, 2014, no. 6, pp. 5—23.
19. **Oleynik P. P., Nikolenko O. I., Yuzefova S. Yu.** Information System for Fast Food Restaurants, *Engineering and Technology*, vol. 2015, 2, no. 4, pp. 186—191., http://article.aascit.org/file/pdf/9020895.pdf
20. **Oleynik P. P.** Metamodel-Driven Design of Database Applications, *Journal of Computer Science Technology Updates*, 2015, vol. 2, no. 1, pp. 15—24. URL: http://www.cosmosscholars.com/images/JCSTU-v1n1/JCSTU-V2-N1/JCSTU-V2N1A3-Oleynik.pdf
21. **Oleynik P. P.** Predmetno-orientirovannoe proektirovanie struktury bazy dannykh v ponyatiyakh metamodeli ob"ektnoy sistemy, *Ob''ektnye sistemy — 2014: materialy VIII Mezhdunarodnoy nauchno-prakticheskoy konferentsii, Rostov-na-Donu, 10—12 maya 2014 g.,* Pod obshch. red. P. P. Oleynika, Rostov-na-Donu: ShI (f) YuRGPU (NPI) im. M. I. Platova, 2014, pp. 41—46, URL: http://objectsystems.ru/files/2014/Object_Systems_2014_Proceedings.pdf
22. **Oleynik P. P.** Using metamodel of object system for domain-driven design the database structure, *Proc. of 12th IEEE East-West Design & Test Symposium (EWDTS'2014), Kiev, Ukraine, September 26—29, 2014,* DOI: 10.1109/EWDTS.2014.7027052
23. **Gur'yanov V. I., Oleynik P. P.** UML-profil' proektirovaniya struktury ob"ektno-orientirovannoy bazy dannykh, *Ob''ektnye sistemy — 2015: materialy X Mezhdunarodnoy nauchno-prakticheskoy konferentsii, Rostov-na-Donu, 10—12 maya 2015*, Pod obshch. red. P. P. Oleynika, Rostov-na-Donu, ShI (f) YuRGPU (NPI) im. M. I. Platova, 2015, http://objectsystems.ru/files/2015/Object_Systems_2015_Proceedings.pdf
24. **Oleynik P. P.** Unifitsirovannaya model' testirovaniya instrumentov razrabotki ob''ektno-orientirovannykh prilozheniy, *Ob''ektnye sistemy — 2014 (Zimnyaya sessiya): materialy IX Mezhdunarodnoy naucrhno-prakticheskoy konferentsii, Rostov-na-Donu, 10—12 dekabrya 2014 g.,* Pod obshch. red. P. P. Oleynika, Rostov-na-Donu, ShI (f) YuRGPU (NPI) im. M. I. Platova, 2014, pp. 25—35. URL: http://objectsystems.ru/files/2014WS/Object_Systems_2014_Winter_session_Pro ceedings.pdf
25. **Nikolenko O. I., Oleynik P. P., Yuzefova S. Yu.** Prototipirovanie i realizatsiya'graficheskoy formy zakaza dlya informatsionnoy sistemy restoranov bystrogo pitaniya, *Ob'ektnye sistemy — 2015: materialy X Mezhdunarodnoy nauchno-prakticheskoy konferentsii, Rostov-na-Dony, 10—12 maya 2015 g.*, Pod obshch. red. P. P. Oleynika, Rostov-na-Dony, ShI (f) YuRGPU (NPI) im. M. I. Platova, 2015, URL: http://objectsystems.ru/files/2015/Object_Systems_2015_Proceedings.pdf