УДК: 681.518.339.13

С.А. Майстренко, м. н. с., Международный научно-учебный центр информационных технологий и систем НАН и МОН Украины

КОНЦЕПЦИЯ УРОВНЕЙ В РАЗРАБОТКЕ МОДЕЛИ ПРЕДМЕТНОЙ ОБЛАСТИ

Введение. В данной работе мы описываем концепцию уровней, где выделяем модель предметной области (МПО). Концепция уровней используется нами для разделения всей архитектуры репозитария информационных ресурсов (РИР) для упрощения задачи разработки, сопровождения, сопоставления и представления. В рамках данной статьи уровень экземпляра (или представления) отвечает в приложении за интерфейс пользователя, уровень модели предметной области (или предметной логики) отвечает за организацию семантики, уровень источника данных отвечает за хранение и интеграцию с существующими приложениями. Дополнительные уровни устанавливают соответствие между этими основными уровнями: уровень контроллера/медиатора, который сопоставляет представление с предметной логикой, и уровень отображения данных, сопоставляющий предметный уровень с источниками данных.

Цель. Целью данной статьи является рассмотрение уровня предметной области, какую роль он играет в РИР, как создается, как интегрируется с уровнем источника данных и тестируется.

Проблема. Достаточно сложно использовать существующие в сети предметные абстракции, в разработке оптимального и универсального решения, при построении адекватной и гибкой МПО, поскольку она содержит семантику, которая должна быть использована определенным способом, влияющим на возможности сопоставления модификации и развития, в соответствии с меняющимися и возникающими взглядами и требованиями.

Пути решения. Сбор и хранение предметных абстракций в объектах предметной логики значительно увеличивает возможности приложений РИР, сделав их более гибкими для сопоставления и повторного использования в динамически расширяемом приложении, которое будет легко модифицироваться в связи с меняющимися и возникающими взглядами и требованиями.

Обоснование архитектуры. Построение многоуровневой архитектуры РИР, основывается на поддержке многократно используемой модели предметной области, допускает независимое построение и проверку каждого из уровней. Первичной целью представления многоуровневой архитектуры есть отделение уровня предметной логики от уровней представления и источника данных. Многократное использование МПО достигается за счет достижения изоляции уровня предметной логики.

Ранее было представлено три иерархически соположенных уровня: мета уровень, уровень предметной области, уровень экземпляра [1]. Уровень предметной области и уровень экземпляра мы привели к архитектуре из пяти уровней [2], где введение двух дополнительных уровней между уровнями представления и источника данных выделяет уровень предметной логики, отделяя ее от требований представления данных и источника данных рис. 1.

Уровень представления (экземпляра) состоит из объектов, определенных для ввода пользователя и вывода информации приложением. Типичными технологиями представления в рамках J2EE являются: HTML/JSP (сервлеты в, действующие качестве контроллеров), XML/XSLT (сервлеты в роли контроллеров), аплеты (использующие AWT/Swing), приложения (использующие AWT/Swing)

Уровень Контроллера/медиатора – это способ изолированного взаимодействия пользователя (уровень представления) с уровнем предметной логики, служит для реализации согласованной логики приложения, включающей тесно взаимосвязанные транзакционные объекты МПО. Он предоставляет уровень для абстрагирования обычных интерфейсов пользователей и логики контроллера приложений РИР, которую можно разделять среди нескольких реализаций интерфейсов пользователей. Отсюда главная цель многоуровневого приложения – это многократное использование уровня предметной логики для других представлений.

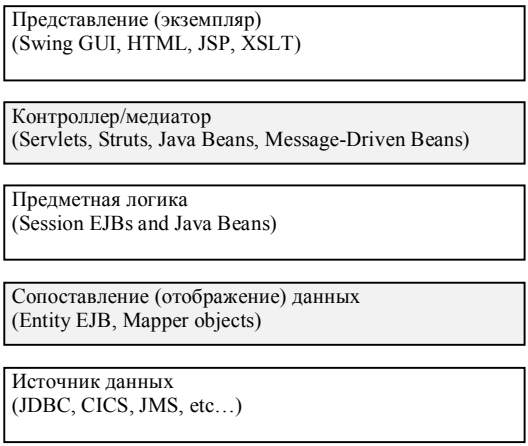

Рис. 1. Многоуровневая архитектура

Запросы состояния и поведения предметной логики осуществляются через объекты контроллера, определенного для конкретных требований уровня представления. Медиаторы собирают и отделяют специфический для приложения функционал от технологий уровня представления.

Уровень предметной логики (МПО) служит для собирания предметных абстракций в объектах. Объекты уровня предметной логики обычно реализуются в виде классов Java.

Определенная предметная область предоставляет основу для разработки уровня предметной логики (МПО). В разных приложениях РИР, можно использовать одну и ту же МПО, для нескольких целей или повторно использовать ее в других приложениях, где каждый случай использования

соответствует некоторому своему аспекту представления. Часто эти различные цели требуют рассмотрения с разных точек зрения. Можно внедрить эти различные точки зрения непосредственно в предметную модель, но иногда лучше разделить их на отдельные пакеты и классы. Для реализации такой модели существует шаблон уровня служб [2], являющийся средством предоставления нескольких интерфейсов для одной и той же МПО для удовлетворения определенных требований или представлений. Этот шаблон используется для предоставления API определенного назначения, чтобы МПО не стала чрезмерно усложненной различными функциональными представлениями. Уровни служб нужны, когда одна и та же МПО используется для различных целей.

Уровень сопоставления (отображения или персистентности) берет на себя функцию перемещения данных от объектов уровня предметной логики к серверным источникам данных и обратно (Apache Castor, Universe, CrossLogic, API Entity EJB).

Уровень доступа к источнику данных с помощью механизма подобного к JDBC API (либо J2EE-JMS-EIS – асинхронный доступ к данным; J2EE-EIS-EIS – синхронный доступ к данным) дает возможность создавать и выполнять запросы на независимом от разработчиков языке структурированных запросов (SQL). Также, можно получить доступ к данным внешних систем посредством веб-сервисов, которые являются самым прогрессивным механизмом предоставления открытого доступа к службам предприятий.

Уровень предметной логики. Уровень предметной логики (МПО) представляет абстракцию проблемного пространства. Он формализует знания, полученные относительно определенной интересующей предметной области. Эрик Эванс (Eric Evans) определяет МПО, как модель, которая «...служит для добычи знаний, улучшения коммуникаций и предоставления прямого пути к реализации и сопровождению функционального программного обеспечения» [3]. В своей работе [4] Фред Брукс (Fred Brooks) приводит различие между «сущностью» и «частностью» в разработке программного обеспечения (ПО). Сущностью проблемы является то, что делает ее внутренне интересной или трудной. Частности реализации проистекают от неадекватности языков или инструментов или от ограниченного понимания возможностей инструментов и языков. Отсюда следует, что нужно фокусироваться на сущности проблемы, не беспокоясь о частностях определенной технологии реализации.

Общими или характерными целями МПО являются: тщательно и недвусмысленно собирать, поддерживать и развивать знание о предметной области проблемы; использовать общий, разделяемый язык, чтобы улучшить взаимодействие между представленными проблемами; предоставить надежный базис, для реализации и сопровождения приложения.

Поскольку предметная область подвержена быстрым изменениям, важно построить приложения РИР указанным выше способом, который

минимизирует связи с другими уровнями. Уровень предметной логики изолирован от механизмов уровней представления или источника данных таким образом, что все они могут меняться независимо друг от друга.

Mini-max подход при моделировании предметной области. Проводится сбор данных о предметной области. Выделяются бизнес-процессы определенные в тексте, сценариях прецедентов, псевдокоде или взаимодействиях UML или диаграммах взаимодействия. Идентифицируются логические подмножества предметной области, которые более или менее независимы, и используются в качестве ориентира при разделении предметной области на отдельные пакеты для снижения сложности. Пакеты представляются в виде классов UML, Java. Производися идентификация различных функциональных областей, которыми можно управлять независимо. Для каждого отдельного бизнес-процесса, идентифицируются объекты из описаний по именам существительным, тогда как функции являются глаголами, описывающими обработку. Некоторые глаголы могут, в конечном счете, стать объектами служб пли контроллеров предметных областей. Выбрать называния объектов, основываясь на роли, которую они играют в предметной области. Из идентифицированных объектов, обладающих новыми сведениями или поведением, информация о новых сведениях или поведении перемещается как можно выше по иерархии классов, не нарушая целостности всего решения. При проектировании иерархии классов это подход называется подходом mini-max, который приводит к эффекту неглубокой иерархии классов, что минимизирует связь между объектами и максимизирует показатель целостности информации о поведении каждого выделенного объекта. Объекты предметной области сопоставляются с лежащим в основе уровнем источника данных, для обеспечения поддержки системы структурированного хранения и обеспечения аналитики [5], что гарантирует непротиворечивость МПО с уровнем (уже существующих) источников данных.

После завершения первой итерации включающей в себя описанные шаги, итерация повторяется снова, где идентифицируются новые функции и системные требования для отыскания новых знаний [6].

Во время каждого шага разработки реализуются специфические функции, которые были идентифицированы как системные требования, где при необходимости вновь раскладывается МПО на составные части, чтобы включить новые знания, для непрерывного улучшения МПО в течение полного жизненного цикла приложения РИР.

Уровень сопоставления данных. Этот уровень предоставляет сохраненные данные в специфическом хранилище данных для уровня предметной логики, который, в свою очередь, предоставляет эти данные в более естественном и приемлемом формате для других предметных объектов. Уровень сопоставления данных является шлюзом, который сопоставляет уровень предметной логики с уровнем источников данных. В нашем случае уровень источника данных представлен многомерной базой данных (МБД),

где хранилище данных служит поставщиком информации для МБД, представляющее собой реляционную базу данных (РБД), организованную по схеме “звезда”, содержащее набор объектов, которые позволяют манипулировать аналитическими данными. [5].

Такая база данных присутствует т.к. при росте МПО будет проводиться сопоставление многих объектов предметной логики, что будет усложнять их сопровождение и тестирование при изменениях МПО и реляционных схем вследствие рефакторизации и перемещений схем (изменчивость).

Инкапсуляция сохраняемых данных является типичным методом управления изменчивостью. Уровень сопоставления данных управляет изменчивостью на уровне источника данных и реальных данных, сохраняя независимость уровней предметной логики и источника данных.

Объектно-реляционное сопоставление. Объектно-реляционное сопоставление не тривиальная операция [2,3]. На примере интерфейса Depot покажем операцию сопоставления, которая независима от МПО и от системы хранения данных. JDBCDepot является абстрактной реализацией интерфейса Depot для JDBC. Этот класс предоставляет реализацию операции сопоставления, специфичного для JDBC, но независимого от любого конкретного сопоставляемого объекта предметной логики.

В интерфейсе Depot, показанном на рис. 2, объявлены методы, которые должны быть у определенного преобразователя для любого объекта предметной логики и источника данных. Эти методы соответствуют типичным операциям CRUD, которые необходимы для сохранения и обновления экземпляров объектов в их источнике данных. Для обнаружения всех экземпляров объекта предметной логики или обнаружения конкретного экземпляра с данным идентификатором объекта используются следующие методы. Метод Depot в интерфейсе Depot, реализует характерное для JDBC поведение, главным образом, управление сопоставлением и обработкой исключений JDBC. Этот метод вызывает абстрактный метод с тем же именем, но с суффиксом Impl, представляющим характерное для объекта предметной логики поведение. Именно этот абстрактный метод Impl должен реализовать характерный для объекта преобразователь, чтобы создавать, читать, обновлять и удалять экземпляры этого объекта в реляционной базе данных. Абстрактный метод activate реализован характерными для объекта предметной логики подклассами для перемещения данных из уровня источника данных к атрибутам объекта предметной логики. JDBCDepot также управляет соединениями с системой хранения данных. Также существует конкретная реализация JDBCDepot для каждого объекта предметной модели. Каждый абстрактный преобразователь источника данных требует конкретной реализации для каждого объекта предметной модели. Доступ к специфичным для объекта преобразователям.

Класс DepotRegistry, показанный на рис. 2, предоставляет доступ к преобразователям, которые должны использоваться для каждого объекта в классе МПО. Класс DepotRegistry используется в приложениях и указывает,

какой источник данных применить. Затем он заполняет таблицу нужного преобразователя, для использования каждого объекта предметной логики, привязанного к своему классу. Метод getDepot ищет экземпляр преобразователя для использования созданным классом объекта МПО.

Методы приложения должны иметь возможность получать доступ к экземплярам объектов для того, чтобы манипулировать ими. Для получения доступа к экземплярам корневых объектов используются методы обнаружения объектов. Для каждой независимой части предметного уровня, которым нужно манипулировать, должен быть отдельный объект корневого уровня. Приложение использует методы обнаружения для определения местонахождения объектов корневого уровня по идентификаторам объекта. Поскольку методы обнаружения должны запрашивать источники данных и возвращать объекты МПО, то они реализуются в преобразователях, и их расположение определяется через DepotRegistry. Характерный (специфичный) для объекта предметной логики преобразователь должен иметь по крайней мере один метод обнаружения. Методы обнаружения должны быть статическими, поскольку для их работы не требуется экземпляр преобразователя. У каждого объекта предметной логики должен быть метод, который находит этот объект по его идентификатору. Абстрактный метод findByPrimaryKeyImpl осуществляет это для всех JDBCDepot. Специфичные для объекта предметной логики преобразователи реализуют этот метод, выполняя запрос SQL, который ищет соответствующую таблицу для записи, ключевой столбец, соответствующий идентификатору объекта.

При построении объектно-ориентированной МПО реляционной базы данных выполняется сопоставление каждого объекта предметной модели с реляционными таблицами. Существуют стратегии для сопоставления объектно-ориентированной МПО с реляционной базой. В стратегии «сверху вниз» МПО используется для выведения схемы для базы данных. В стратегии «снизу вверх» объектная МПО выводится из реляционной схемы. Стратегия «сходящийся к середине» используется, когда МПО и реляционная схема не очень хорошо соответствуют друг другу. Стратегия «сходящийся к середине» является более актуальной стратегией, поскольку часто уже имеется по крайней мере частичная реляционная схема, а в предметной модели можно использовать эти данные и расширить их для поддержки новых функций. На реляционные схемы часто оказывают влияние требования нормализации и производительности поэтому они могут не отражать идеальные МПО

Сопоставление связей между объектами и реляционной базой данных является одним из более сложных аспектов объектно-реляционного сопоставления.

Связи между объектами в предметной модели поддерживаются посредством объектных ссылок языка программирования. Эти объектные ссылки должны быть сопоставлены с механизмом, поддерживаемым уровнем источника данных. Для систем реляционных баз данных это внешние ключи.

На рис. 2 использовано для каждого объекта предметной логики поле идентификатора.

Поля идентификаторов сопоставляются с первичными ключами объекта предметной логики в соответствующих таблицах. Эти идентификаторы могут также использоваться для предоставления значений для внешних ключей, необходимых для отображения связей между объектами предметной логики.

Для обеспечения непротиворечивости МПО при сохранении данных в источнике данных используются транзакции, но важно также поддерживать непротиворечивое состояние в памяти при исполнении транзакции. Одним из способов избежать противоречивости является обеспечение отсутствия

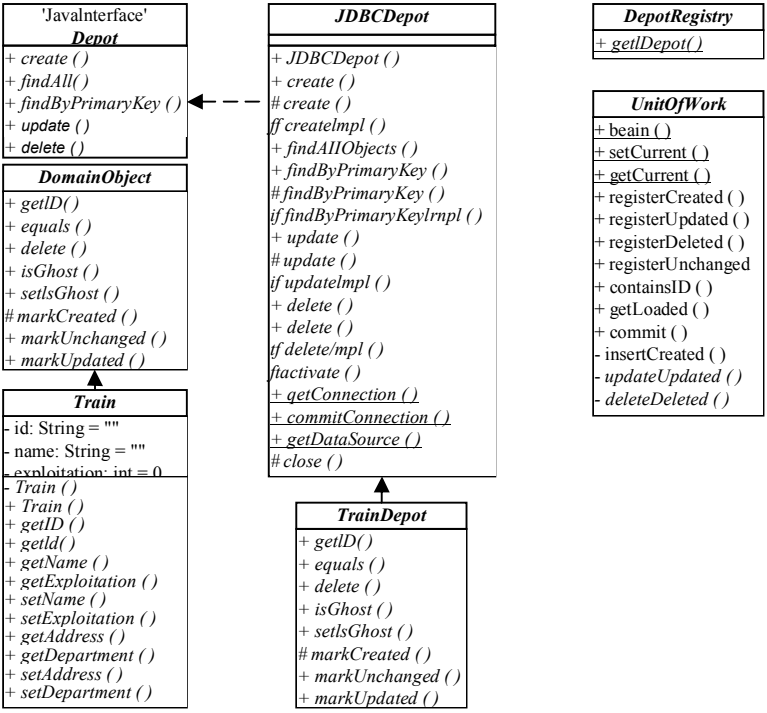

Рис. 2. Объектно-реляционное сопоставление

дублирующих экземпляров объектов предметной логики.

Реляционные источники данных не поддерживают непосредственно наследование, поэтому на уровне сопоставления данных нужно обеспечить преобразование иерархии наследования в данные в базы. Есть три подхода к реализации наследования сохраняемости и три соответствующих шаблона. Первый – наследование с одной таблицей (Single Table Inheritance) [2].

Второй – наследование с таблицей классов (Class Table Inheritance) [2]. Третий – наследование с конкретной таблицей (Concrete Table Inheritance) [2].

В начале тестирования разрабатываются контрольные примеры, основываясь на ожидаемом поведении МПО, а затем разрабатываются МПО до тех пор, пока тесты не будут пройдены. Контрольные примеры основываются на прецедентах или бизнес-процессах пользователя. Для тестирования предметной семантики прежде, чем создавать весь код сопоставления используется некоторый черновой уровень сопоставления, не требующий объектно-реляционного сопоставления. Данные МПО, просто сохраняются в памяти во время запуска контрольного примера, а потом выбрасываются. Если контрольным примерам требуется, чтобы некоторые объекты уровня предметной логики сохранялись за пределами набора тестов таким образом, чтобы их можно было повторно использовать, тогда используется простой преобразователь, сериализующий объекты предметной логики в виде объектов Java или XML файлов. В этом случае DepotRegistry отвечает за определение того, какой преобразователь использовать для каждого объекта предметной логики. Для циклического тестирования особенно важно использование специальных тестовых наборов, когда во время разработки меняются как МПО, так и схема базы данных.

Заключение. В данной статье были рассмотрены уровень МПО и уровни сопоставления данных в многоуровневой архитектуре. Мы актуализировали важность предметной модели в установлении основы для проекта. Были представлены проблемы, которые нужно решать при сопоставлении предметной модели с источниками данных через уровень сопоставления данных. Было представлено использование инструментов объектно-реляционного сопоставления, для автоматизации генерации кода сопоставления.

1. *Майстренко С.А.* Возможности использования средств ведения онтологии экспертных взглядов на предметную область в экспертизе программных систем // Моделювання та інформаційні технології - 2005, с. 3 – 13.

2. *Martin Fowler with David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, and Randy Stafford //* Patterns of Enterprise Application Architecture, Reading, MA: Addison-Wesley, 2002.

3. *Eric Evans.* Domain Driven Design // Tacking Complexity at the Heart of Software, Reading, MA: Addison-Wesley, 2003.

4. *Frederic Brooks.* The Mythical Man Month // Essays on Software Engineering, Ann. Ed., Reading, MA: Addison-Wesley, 1995.

5. *Майстренко С.А.* Представление информации на основе OLAP технологий в общей метамодели хранилища данных. // // Зб. наук. пр. ІПМЕ НАН України – 2007. № 44, с. 126-134

6. *Майстренко С.А.* Анализ требований используемых для пошаговой разработки базы знаний // Моделювання та інформаційні технології – 2005, № 35, с. 127-135