РАЗДЕЛ I

МОДЕЛИ ОБРАБОТКИ И ПРЕДСТАВЛЕНИЯ ДАННЫХ

О построении MOF репозитория метаданных

А. А. Белевцев, В. Е. Кривцов

Введение

Опыт использования метаданных привел к расширенному пониманию их роли в построении программного обеспечения [1]. Исходное определение метаданных как «данных о данных» в настоящее время существенно расширилось за счет включения в него моделей программных систем [1)], и акцент в исследованиях сместился на изучение возможностей применения этих моделей не только на этапе проектирования ПО, но также на всех последующих этапах разработки.

Модели, вследствие своей независимости от технологий программирования, в принципе позволяют наладить взаимодействие между разными платформами и поэтому выступают главным инструментом интеграции разнородных ресурсов. Эта их особенность уже сейчас активно используется в таких технологиях, как CORBA, Web-сервисы, Net и т. п.

Компании, разрабатывающие программное обеспечение, используют самые разные модели. Например, набор средств моделирования при работе с реляционными данными включает таблицу, столбец, ключ. Набор конструкций моделирования при описании потоков работ включает активность, исполнителя, передачу, соединение. UML моделирование использует такие конструкции как класс, атрибут, операцию, ассоциацию. Набор моделирующих элементов при определении CORBA интерфейсов включает интерфейс, тип-значение и т. д.

Общее количество используемых в программировании моделей лавинообразно растет, что делает чрезвычайно актуальной задачу организации их хранения и работы с ними. Для ее решения создаются репозитории метаданных, поддерживающие основные необходимые операции с моде-

[1)] Фактически, сегодня термины «модель» и «метаданные» используются как эквивалентные. Такая их трактовка принята и в настоящей работе.

лями, такие как создание, чтение, изменение, удаление (CRUD = create, read, update, delete).

Главная проблема, связанная с разработкой репозиториев метаданных, это разнородность моделей и многообразие форматов их представления. Из-за отсутствия единых подходов компании создают собственные структуры хранения — только под свои задачи и под свои форматы используемых метаданных. В результате, при организации поиска, доступа и обмена метаданными, т. е. при решении именно тех задач, на поддержку которых ориентированы репозитории, возникают значительные проблемы.

Требования к репозиториям метаданных

Основные требования, предъявляемые к репозиториям метаданных, определяются необходимостью использования репозиториев в распределенных приложениях, для которых характерна уже отмечавшаяся выше крайняя разнородность используемых моделей.

Универсальность. Создание отдельных репозиториев под каждый тип моделей нецелесообразно, так как:

- затрудняет последующую организацию взаимодействия репозиториев;
- не автоматизировано и требует индивидуального подхода;
- требует значительных затрат ресурсов.

Переносимость. При работе распределенных приложений, важной проблемой организации их взаимодействия является обеспечение экспорта, импорта и обмена метаданными (моделями, интерфейсами и т. д.) разных форматов.

Интероперабельность. Важной возможностью является организация доступа к репозиторию из любого приложения, независимо от того, на каком языке программирования оно написано и в какой операционной среде исполняется. Иными словами, репозитории должны поддерживать возможность прозрачного CRUD доступа из разных приложений к моделям разных форматов.

Использование MOF при создании репозиториев

Последние разработки в области программной инженерии, в особенности концепция MDA (Model-Driven Architecture) [2], позволяют сформулировать общий подход к созданию репозиториев. Его идея опять базируется на использовании моделей, но уже более высокого — «мета» — уровня. Суть подхода состоит в построении абстрактной метамодели управления и обмена метаданными и задании способов ее трансформации в поддерживаемые технологии программирования (Java, CORBA, XML и др.).

Разработка метамодели опирается на технологию моделирования MOF (Meta Object Facility) [3]. MOF — это не зависящий от платфор-

мы, универсальный способ описания конструкций моделирования. MOF построен на нотации UML и спроектирован специально для поддержки моделирования метаданных. Поскольку модели метаданных называются метамоделями, MOF является общим средством метамоделирования.

MOF реализует общий взгляд на управление метаданными и конструктивно поддерживает возможность взаимодействия моделей разных видов, включая новые модели, разработка которых еще даже не начиналась. При этом MOF отталкивается от двух базисных предположений:

- в будущем всегда будут использоваться модели разных типов;
- совместимая поддержка и управление разными типами моделей возможна.

MOF использует совершенно иной подход к интеграции метаданных в сравнении с более ранними попытками создания интегрированных репозиториев метаданных. Основное отличие состоит в том, что вместо попытки унифицировать все моделирование посредством одного языка моделирования, MOF постулирует необходимость использования множества языков для описания различных системных аспектов на разных уровнях абстракции. MOF — это ключевая основа MDA, он обеспечивает высокую степень независимости от различий языков моделирования, что дает возможность управлять различными метаданными по единому принципу.

Очень важным следствием архитектуры MOF, при которой все метамодели описываются согласно единой метаметамодели, является возможность единообразного описания отображений между метамоделями. В настоящее время имеется уже несколько стандартов MOF, поддерживающих такие трансформации. К их числу относятся:

- MOF-XML отображение, которое называется XML Metadata Interchange (XMI) [4];
- MOF-Java отображение, называемое Java Metadata Interchange (JMI) [5];
- MOF-CORBA отображение, определенное OMG [6]. У этого отображения специального названия нет.

Перечисленные отображения содержат информацию, необходимую генераторам кода для автоматической трансформации абстрактного синтаксиса метамоделей в его конкретные представления, основанные на Java, CORBA и XML. Генераторы кода, основанные на MOF, должны применять эти отображения при чтении метамодели и создании Java API, CORBA API и XML DTD или Schema для представления метаданных в этих разных языках и системах программирования.

В настоящее время MOF уже применяется для описания различных метамоделей, например, реляционных данных или классов UML. Точно так же он может быть использован и для построения метамодели управления и обмена метаданными. Эта метамодель должна определять общий подход к описанию различных типов моделей и операций над ними. При

наличии такой метамодели возникает принципиальная возможность замкнуть ее на генератор кода, который, используя заданные универсальные трансформации, создаст ПО для управления и обмена моделями самых разных типов, подчиняющихся этой метамодели [7].

В итоге, реализация принципов MDA при разработке репозиториев метаданных позволит унифицировать и автоматизировать процесс создания репозиториев, интегрирующих в себе хранение, доступ, управление и обмен различными типами моделей.

Возможности MOF репозиториев

Репозитории, созданные и функционирующие на основе MOF, называются далее MOF репозиториями. На рис. 1 представлен такой репозиторий, содержащий метаданные (модели) разных типов, т. е. подчиненные разным MOF метамоделям. Каждый тип моделей имеет, вообще говоря, специфический интерфейс CRUD доступа. Клиенты репозитория используют эти интерфейсы для того, чтобы создавать новые модели в репозитории, читать уже находящиеся в репозитории модели, изменять их и удалять из репозитория.

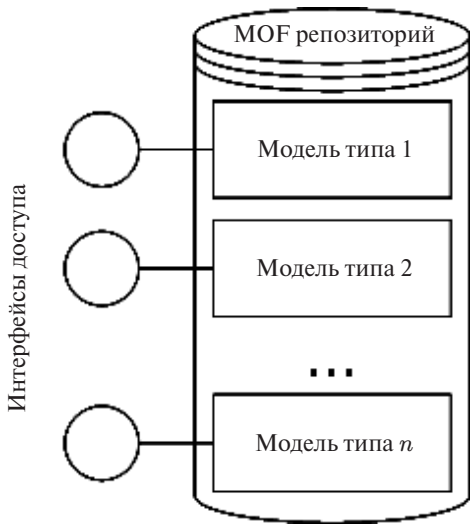

Рис. 1. MOF репозиторий

Репозиторий должен предоставлять клиентам интерфейсы для манипулирования моделями любых типов. В том числе возможна динамическая генерация интерфейсов («на лету») в ответ на соответствующий запрос. MOF генератор кода создает эти специфические интерфейсы и их реализации, используя для каждого типа моделей задающую его MOF метамодель.

Интерфейсы, предоставляемые репозиторием, зависят не только от типа модели, они также специфичны для используемой платформы. Например, если средство моделирования является Java клиентом, то оно должно иметь возможность использовать Java API, чтобы записать метаданные в репозиторий, когда разработчик сохраняет модель, и извлечь метаданные из репозитория, когда разработчик хочет выгрузить модель. Это значит, что модели в репозитории должны быть представлены как Java

объекты (или CORBA объекты, или иметь другое требуемое представление) соответствующими Java (или CORBA, или другими) интерфейсами.

Кроме этого, используя MOF метамодель, репозитории могут создавать XML DTD и Schema, которые определяют формат XML документов, используемых для экспорта и импорта моделей. Импорт и экспорт XML документов, как правило, является наилучшим способом обмена метаданными между репозиториями и приложениями, существующими в разных локальных сетях. Это вызвано наличием сетевых экранов и других ограничений, из-за которых транзакционные сетевые взаимодействия через интерфейсы репозитория могут оказаться проблематичными.

Поэтому, когда нужно сохранить модель, удаленное средство моделирования могло бы экспортировать в репозиторий XML документ, в который трансформируется эта модель в результате применения XMI преобразования, и импортировать XML документ из репозитория, когда нужно извлечь модель.

Эти возможности работы с MOF репозиториями проиллюстрированы на рис. 2.

MOF ничего не говорит о том, как модели физически должны храниться в репозитории. Это означает, что допустим любой способ хранения: в реляционной базе данных, в ОО базе данных, в виде плоских файлов в файловой системе, непосредственно как XML файлы, просто в памяти

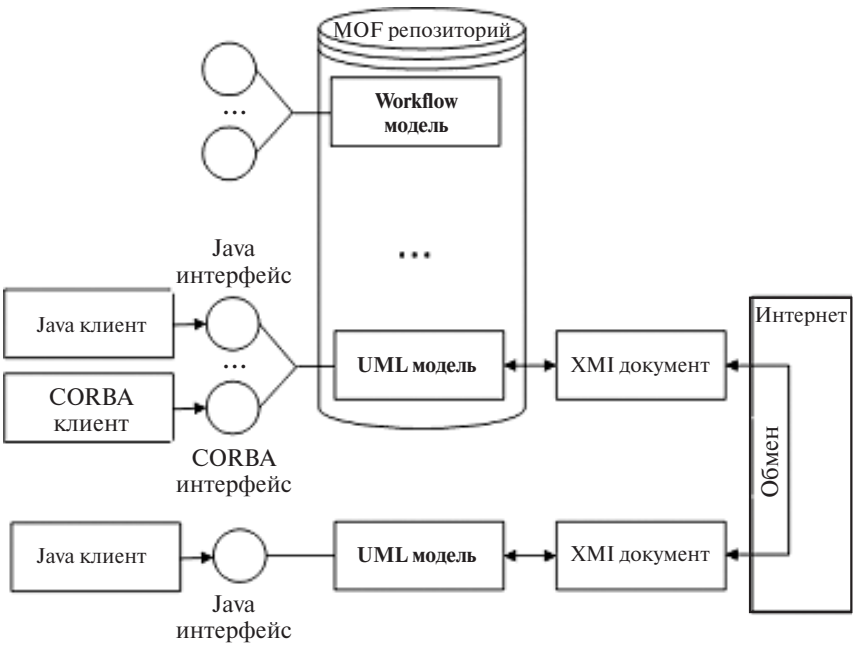

Рис. 2. Управление и обмен моделями в MOF репозитории

и т. д. — в зависимости от конкретной реализации репозитория. Некоторые MOF репозитории отделяют слой постоянного хранения метаданных, обеспечивая его реализацию по умолчанию и сохраняя при этом возможность plug in для других реализаций. Некоторые предоставляют несколько вариантов постоянного хранения. В любом случае, MOF стандартизирует только API и XML форматы, используемые для взаимодействия с репозиториями, — и не касается внутреннего физического формата хранения метаданных.

Подход к построению MOF репозитория

Рассматривается задача создания универсального репозитория для манипулирования и обмена метаданными любых типов. Как отмечалось выше, принципиальная возможность реализации подобных систем основывается на MOF и на стандартных MOF трансформациях метамоделей различных типов.

При этом должна быть обеспечена универсальная возможность трансформации моделей на разные технологические платформы (Java, CORBA и т. д.) для предоставления интерфейсов доступа к хранимым моделям из приложений этих платформ. Также необходимо обеспечить возможность преобразования в XMI формат для поддержки сетевых обменов.

На рис. 3 приведена общая модель MOF репозитория. Физическое хранение моделей отделено от логической схемы. Работа с моделями в репозитории ведется под управлением метамоделей, которым подчинены эти модели, например метамодель UML, метамодель Workflow, метамодель CWM. Каждая из этих метамоделей определяет:

- физическую схему хранения;
- представление интерфейсов логического доступа;
- схемы XML документов.

Создание перечисленных схем происходит на основе следующих трансформаций:

- MOF-Платформа — отображение MOF в технологические платформы, такие как Java, CORBA, Web-сервисы и т. д.;
- MOF-CWM — отображение MOF в метамодели CWM, такие как реляционная база данных, объектное хранение, многомерная база данных и т. д.;
- MOF-XML — преобразование MOF в XML для поддержки обмена моделями.

При такой организации клиенты репозитория получают возможность работать с любыми моделями, содержащимися в нем, с использованием специфических для клиентских платформ интерфейсов. Поскольку все преобразования ведутся на основе метамоделей, возникает возможность динамической генерации схем хранения и интерфейсов доступа по запросу клиента.

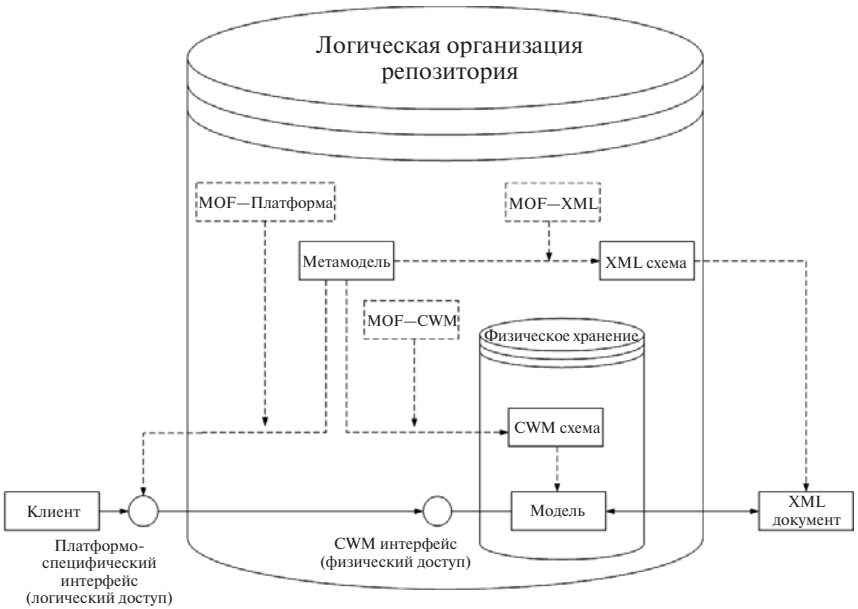

Рис. 3. Общая модель MOF репозитория

В настоящее время стандартизованы трансформации MOF-CORBA, MOF-Java (JMI), MOF-XML (XMI), и создаются инструментальные средства поддержи этих трансформаций.

Основной проблемой, осложняющей создание универсальных репозиториев такого типа, является отсутствие стандартизованных преобразований MOF-CWM, CWM–XML, а также «моста» Платформа-CWM, определяющего преобразование вызовов логических интерфейсов в операции CWM интерфейса с объектами в физическом хранилище.

Реализация архитектуры репозитория

Одним из основных вопросов при реализации репозитория является выбор технологии физического хранения. Предлагается в качестве такой технологии использовать реляционную базу данных (RDB). Причинами для выбора данного решения являются:

- удобство хранения и конвертации метаданных различного типа;
- удобство обработки данных в хранилище;
- удобство доступа из различных программных надстроек;
- масштабируемость.

На рис. 4 показан пример работы с UML и Workflow моделями. Клиенты репозитория имеют возможность доступа к моделям из среды

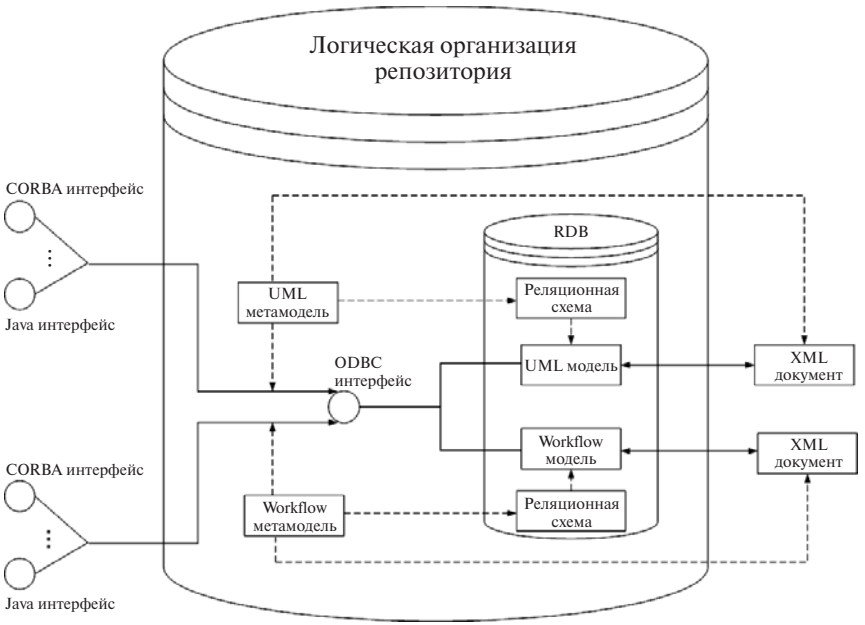

Рис. 4. Архитектура репозитория

Java и CORBA. Физически все операции доступа к моделям реализованы как операции над RDB представлениями моделей.

Принципиально, подобная архитектура позволяет репозиторию произвольно наращивать количество поддерживаемых MOF совместимых типов моделей, а также обеспечить доступ к ним с любых технологических платформ.

Литература

1. *Белевцев А. А.*, *Кривцов В. Е.* Метаданные в распределенных системах // Проблемы вычислений в распределенной среде: организация вычислений в глобальных сетях. Сборник трудов ИСА РАН. М.: УРСС, 2004.
2. *Kleppe A.*, *Warmer J.*, *Bast W.* MDA explained: the model driven architecture: practice and promise. Adison-Wesley, 2004.
3. www.adtmag.com/java/articleold.asp?id=1246&mon=3&yr=1999.
4. XML Metadata Interchange (XMI) Specification v2.0; www.omg.org/docs/formal/03–05–02.pdf.
5. java.sun.com/products/jmi.
6. MOF Specification v. 1.4; www.omg.org/technology/documents/formal/mof.htm.
7. *Frankel D.* Using Model-Driven Architecture to Manage Metadata: White Paper IONA Technologies PLS, 2002.