УДК 381.3.06

Т.Г. Шемсединов, А.А. Стенин

РАСПРЕДЕЛЕННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ В ЗАДАЧАХ МОДЕЛИРОВАНИЯ

Разнообразие моделируемых объектов реального мира, привело к созданию немалого количества моделей данных. При этом, для представления различных объектов, совершенно естественным может оказаться применение разных моделей, более близких к ним по своей природе. Насильственное отображение предметной области в чужеродную модель, приводит к появлению внутренних противоречий, и к конфликту ИС с пользователем, который интуитивно желает видеть на экране данные в их естественном виде. Даже в рамках одной предметной области часто встречается необходимость в двух и более моделях. Таким образом, становится необходимой разработка гибридной информационной модели, которая, одновременно должна отображаться в родственную ей модель данных. Различие в понятиях информационной модели и модели данных раскрыты в работе А. Праса и Ю. Шоэнвалдера [1], основанной на результатах рабочих групп IETF, IRTF, ITU, DMTF, NMRG, других специализированных конференций и семинаров.

В разрабатываемом подходе предпринята попытка, построить гибридную модель на основе объектного подхода. Но, тем не менее, она должна много почерпнуть из других моделей данных: иерархической, сетевой, реляционной и из семантической и ER информационных моделей. Обзоры различных моделей и их сравнительный анализ даются в [2-4].

Разрабатываемая гибридная модель предусматривает создание, сквозного для всех слоев ИС, формата представления данных. Это необходимо в целях ухода от сложных преобразований данных между уровнями системы. Каждый из уровней отвечает за разные уровни абстракции и за разные семантические разрезы единой модели предметной области. Следует, так же, отметить, что полное семантическое осознание предметной области, может быть только у пользователя, поэтому его можно считать частью ИС.

Модель данных и информационная модель должны быть динамическими, т.е. позволять пользователю изменять общую модель предметной области и логику работы ИС так, как этого требует постоянно развивающиеся реальные объекты в динамических предметных областях. К сожалению, большинство из современных CASE средств и инструментария для UML и ER моделирования (применяемых как для управления схемами баз данных, так и программными продуктами) имеют в своей основе принцип разового обновления модели, подобный компиляции. Во многих случаях модель хранится отдельно от системы, во внутреннем формате CASE средства. Это не дает возможности динамического

изменения модели предметной области, без временной остановки функционирования информационной системы, а так же, часто приводит к необходимости изменять программное обеспечение для доступа к данным.

В этом месте проблема моделирования данных плавно переходит в задачу усовершенствования архитектуры ИС. Основываясь на изложенных положениях, мы видим, что часто программные модули жестко привязаны к определенным типам и классам данных и к модели предметной области, к схемам и структурам хранения, с одной стороны, и к интерфейсам пользователя, с другой. Т.е. программное обеспечение, частично содержит в своем теле данные о модели. Такое "размазывание" модели предметной области по слоям ИС противоречит принципам гибких систем. Все это ведет к тому, что выигрыш от модульности ИС практически полностью пропадает. Как только изменяются схемы хранения, появляются новые классы и типы данных, новые связи, новая бизнес логика, то это приводит к модификации всех слоев ИС. Таким образом, можно утверждать, что в гибкой ИС, информационная модель должна управлять логикой обработки данных. Все модули должны черпать метаинформацию из обобщенной модели предметной области, а при ее изменении, должны автоматически корректироваться все слои ИС; изменения схемы БД, должны отображаться во всех модулях системы. В идеальном случае, ИС должна быть на 100% управляема информационной моделью, которая непрерывно корректируется и развивается в процессе своего функционирования, получая новую информацию от пользователя и аккумулируя ее в хранилище. Подобный подход, накопления знаний, применяется в EME и описан в [5]. Однако, в реализации EME, для изменения модели требуется вмешательство программистов, т.к. бизнес-логика пишется на компилируемом языке.

Теперь перейдем к вопросам архитектуры ИС. Широкое распространение получила модульная архитектура, хороший обзор которой можно найти в [3,6,7]. Однако, многие положения и идеи, описанные в многочисленных источниках, и широко используемые при построении современных ИС, имеют принципиальные ограничения и должны быть подвергнуты серьезному пересмотру. Например, распространена точка зрения, что если зафиксировать интерфейсы между модулями, четко их документировать в правилах декларативного языка описания интерфейсов, то это сделает модули независимыми друг от друга, легко заменяемыми и независимо модифицируемыми. Сложно не согласиться с таким положением, однако, его преимущества весьма спорны. Действительно, такой подход позволяет оптимизировать каждый модуль как "черный ящик", улучшить его показатели по потреблению ресурсов, быстродействию, надежности, и наконец, исправить ошибки в модуле, без внесения изменений в соседние модули ИС. Но расширить функциональность системы в целом, вряд ли удастся, если оставить интерфейсы фиксированными. Если модуль развивается и получает новую функциональность, то это требует расширения интерфейса, в лучшем случае, добавления в ин-

терфейс новых функций, а в худшем - изменения уже существующих функций, параметров вызова, типов входных и выходных данных. Можно допустить, что при таком расширении интерфейса, можно, все же, оставить совместимость со старым его вариантом. Хотя это порождает проблемы поддержки старых версий интерфейса, проблемы совместимости, решение которых приводит к потере надежности и производительности. Противоречие на лицо, а решения пока что являются частными и искусственными. Кроме того, остальные модули ИС не могут использовать новую функциональность измененного звена системы, до тех пор, пока сами не будут расширены. Т.е. даже при обеспечении совместимости, новые функции остаются невостребованными. Единственным адаптивным звеном в ИС остается человек, который обучается использованию новых функций по мере их появления. Но человек взаимодействует только с верхним слоем ИС, а хорошо было бы распространить подобную возможность на все стыки модулей. Сложность заключается еще и в том, что вызовы функций одного интерфейса разбросаны по телу вызывающего модуля и зашиты в нем. Как же “научить” ИС использовать новую функциональность, в каком бы слое она не появилась? Возвращаясь к идеям, изложенным ранее, можно предложить возложить функции описания интерфейсов межмодульного взаимодействия и функциональности, на модель предметной области, расширив ее до информационной модели ИС. Такая метамодель должна находиться в БД (на самом нижнем слое ИС), и автоматически распространяться на все слои, вплоть до пользователя, постепенно “обучая” каждый слой его функциональности и методам общения с другими модулями.

Это приводит к необходимости включить в каждый слой интерпретатор модели, который на лету модифицировал бы функциональность слоя. Но, как мы знаем, любая интерпретация приводит к потере производительности по сравнению с откомпилированным кодом. В данном случае, можно прибегнуть к частному решению, которое, характерно только для современной архитектуры вычислительной техники, и вызвано принципиальными ограничениями этой структуры, о чем было уже много сказано в различных работах, например в [8].

Автору представляется следующее решение: каждый слой должен содержать библиотеку абстрактных алгоритмов обработки форматов данных, написанных на низкоуровневых компилирующих языках. Следует обратить внимание, что это не библиотека, реализующая логику конкретной системы, а набор алгоритмов обработки допустимых форматов данных. Таким образом, интерпретатор, на основе декларативной модели, может построить необходимую для этого слоя логику “на лету”, без перекомпиляции системы, при этом, не потеряв в производительности.

Высказанное выше предложение, подводит нас к необходимости разграничения информационной модели и алгоритмического обеспечения, максимально абстрагировав его от предметной области. К сожалению, даже в современных объектно-ориентированных языках программирования, часто смешивают в одном классе, информацию о предметной

области, бизнес логику, методы и чисто компьютерную функциональность, направленную на необходимости технологической среды. Например, один класс может отвечать за представление в памяти параметров моделируемого объекта и, одновременно, обеспечивать его визуализацию, функции сетевого взаимодействия, для передачи данных объекта по сети, или функции взаимодействия с СУБД, направленные на связь объекта с хранилищем. Эта смесь, крайне сложна в модификации, и требует перекомпиляции всего проекта, каждый раз, при изменении чего-либо в структуре или в бизнес логике. Смешение данных предметной области, и технологически зависимой функциональности приложения, сводит к нулю преимущества объектного подхода.

Сейчас сложилась такая ситуация, когда в индустрии информационных технологий все чаще появляются волны новых подходов, концепций, архитектур, технологий и программных продуктов, претендующих на решение всех проблем пользователя. Складывается впечатление, что информационные технологии переживают стремительный подъем, но это обманчивое впечатление, т.к. все технологические новшества происходят на каком-то отдельном слое ИС, и имеют целью скорее улучшение технологических характеристик, чем создание концептуально новых решений. Все это, в большей степени похоже на маркетинговые волны, борьба различных реализаций одних и тех же концепций, происходит количественное развитие, а не качественное. Декларируется множество привлекательных, для разработчиков и пользователей, возможностей. Но без комплексного решения проблем моделирования и архитектуры систем, эти волны захлебываются, так и не принеся ожидаемого эффекта.

В доказательство приведу несколько мыслей из [8], которые были изложены еще в 1989 году, но с тех пор их актуальность только увеличилась.

1. СУБД часто навязывают модель данных для всей ИС. А от себя добавлю, что т.к. обычно, крупные промышленные СУБД поддерживают только одну модель данных (например, реляционную или иерархическую), то это становится определенным насилием над данными. Основываясь на опыте внедрения ИС в различных областях их применения, мы видим, что это оборачивается и насилием над организационной структурой предприятия. Во многих случаях это приводит к необходимости реорганизации предприятий, примеры могут быть найдены в работах [3,4,7].

2. Вторая идея из [8], заключается в том, что часто выбирают не модель данных, подходящую для каждого конкретного случая, а успешную СУБД, что не всегда совпадает.

Основываясь на этих положениях, мы видим, что существует глобальная проблема взаимоотношений между инфологической и даталогической моделями. Под инфологической понимается модель идущая от предметной области, а под даталогической - модель физического представления и хранения данных. Таким образом, ИС призвана “помирить”

 ISSN 1562-9945

эти две стороны. См [2].

Разработка гибридной модели может быть решением этой проблемы, где отображение предметной области в технических средствах будет естественным и не насильственным. Нагрузка же по семантической обработке данных, должна быть распределена между слоями ИС, при чем каждый нижний слой, должен быть ближе к даталогии, а каждый верхний - к инфологии. Однако в современных системах существуют слои, дублирующие уровни абстракций и семантические уровни друг друга, более того, вносящие в этот график лишний перегиб, и приводя к немыслимой неоптимальности использования, и без того ограниченной архитектуры вычислительной техники.

В ИС, в разработке которых мне довелось участвовать, и применять широко распространенные подходы и технологии, меня всегда поражало количество преобразований над данными, на протяжении их "путешествия" от хранилища к пользователю и обратно. Для любого технического специалиста, понимающего все эти внутренние процессы и преобразования, должно быть просто жалко "несчастные" данные, обреченные на столь замысловатые и порой абсурдные метаморфозы.

Позволю себе предположить, что нормальные формы (реляционной модели) являются только методами избавления от аномалий, вызванных проблемами отображения инфологической модели в даталогическую. Общие проблемы отображения хорошо раскрыты в [9]. Но одновременно, нормальные формы есть и методом декомпозиции, что приводит к расщеплению данных в целях обработки и хранения. Это абсолютно не соответствует идее инкапсуляции в объектно-ориентированном анализе (OOA) и моделировании (OOD), о чем можно прочитать у множества авторов, но все же предпочтительнее обратиться к классику жанра, Гради Бучу [10].

Нужно заметить, что принцип инкапсуляции иногда приносит пользу, а иногда и вносит противоречия в модель предметной области. Если посмотреть на моделируемые объекты в реальном мире, то можно выделить типы взаимодействия между ними, в которых объекты взаимодействуют как инкапсулированные, но на ряду с ними есть и множество примеров, когда часть объекта взаимодействует со внешними объектами или их частями, сами по себе, а не как часть чего-то целого. Можно проследить, что чем "меньше" часть объекта, тем выше степень ее интеграции в общую структуру макрообъекта. Например, четко можно увидеть понижение степени инкапсуляции в цепочке: "электрон", "молекула", "клетка", "орган", "человек", "семья", "организация", "страна". Вряд ли, "клетка" будет взаимодействовать со "страной", не как часть "человека". А вот человек, может взаимодействовать со "страной" и как часть "семьи", и сам по себе. Кажется, что степень инкапсуляции растет в зависимости от размеров объектов, но при детальном анализе, она изменяется в соответствии с их структурной сложностью. Это может и должно быть отображено в модели предметной области. Инкапсуляция становится не просто параметром взаимодействия, а диапазоном, в ко-

тором один объект может быть достижим другим в дереве абстрактных классов. При переходе к гибридной модели, пересмотру должны быть подвергнуты и другие принципы OOA и OOD.

Но одним из главных положений, на которых строится гибридная модель, является принцип интеграции данных, в отличие от принципа декомпозиции, применяемого в реляционной модели. Объекты помещаются в хранилище “целиком”, и могут быть извлечены “целиком”. При этом могут быть построены и сложные запросы, основывающиеся на множестве объектов, с условиями, функциями от полей и вложенными запросами. Но чаще всего для работы с объектом используется его естественное цельное представление.

Подводя итоги, перечислим основные, из рассмотренных нами, концепций построения гибридной ИС, которые частично реализованы в прототипе СУБД UOD и протокола взаимодействия распределенных приложений USP:

Интеграция различных моделей данных;

Необходимость сквозного формата представления данных для всех слоев ИС, и минимизация преобразований информации между модулями;

Обеспечение поддержки динамического изменения модели предметной области без приостановки и перекомпиляции системы, во время ее функционирования;

Создание архитектуры распределенной информационной системы в соответствии с, приведенным в данной статье, пересмотром концепции модульности и гибкости интерфейсов между слоями ИС.

В настоящее время разрабатываются подходы для решения следующих задач:

Взаимоотношения данных и логики в единой информационной модели ИС;

Создание гибких механизмов расширения функциональности системы, на основе интерпретируемой метаинформационной семантики;

Методы естественного отображения предметной области в информационную модель и отказ от насильственной реорганизации моделируемых объектов, для приведения их в соответствие с искусственными компьютерными структурами;

Средства представления множественных связей между объектами и применения к теоретико-множественных операций к объектам гибридной модели.

Каждая из этих проблем требует отдельной статьи. Дополнительную информацию о проекте UOD/USP можно найти по адресу http://niist.ntu-kpi.kiev.ua/USP

А связаться с разработчиками можно по e-mail: Timur@niist.ntu-kpi.kiev.ua

Литература

[1] A. Pras, J. Schoenwaelder, “On the Difference between Information Models and Data Models”, University of Twente, University of Osnabrueck, RFC-3444, January 2003.

[2] Д. Цикритзис, Ф. Лоховски, “Модели данных”, перевод с английского, Москва, “Финансы и статистика”, 1985.

[3] Г.Н. Калянов, “CASE-технологии: Консалтинг в автоматизации бизнес-процессов”, Москва, 2000.

[4] Лилия Козленко “Проектирование информационных систем”, “Часть 1. Этапы разработки проекта: стратегия и анализ”, КомпьютерПресс, № 9, 2001.

[5] Н.П. Мареев, “И все-таки это искусство”, http://www.eme.ru/statii/iskusst.htm, EME, 2000.

[6] Сергей Костяков, “Технология программных компонентов в корпоративных системах”, PC Magazine (Russian Edition), “СК Пресс” 1999, http://www.stu.ru/library/pc_mag/kostya4-99.html

[7] Лилия Козленко, “Проектирование информационных систем”, “Часть 2. Этапы разработки проекта: определение стратегии тестирования и проектирование”, КомпьютерПресс, № 11, 2001.

[8] С. Осуга “Обработка знаний” (перевод с японского), Москва, “Мир”, 1989.

[9] В.В. Кисиль, “Роль процессов отражения в самоорганизации сложных систем”, Одесский государственный университет, декабрь 1995.

[10] Grady Booch, “Object Oriented Design (with applications)”, Rational, 1991.

[11] Fabio Casati and Umeshwar Dayal, Hewlett-Packard, “Letter from the Special Issue Editors”, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, December 2002 Vol. 25 No. 4

[12] Gustavo Alonso, “Myths around Web Services”, Department of Computer Science Swiss Federal Institute of Technology (ETHZ), Switzerland, www.inf.ethz.ch/department/IS/iks/

Получено: 08.12.2002