ДВИЖУЩИЕ СИЛЫ РАЗВИТИЯ CASE-СРЕДСТВ

Создание и применение среды программирования на базе декларативного языка представления знаний о предметной области не является панацеей от всех бед на этапе проектирования ИС. Однако определение «удачного» языка, на базе которого будет строиться логическая модель предметной области, создание и программная реализация процедур перевода из этого языка в языки более низкого уровня (для которых уже есть компиляторы в машинные коды) — задача вполне разрешимая.

И.В. Данылив, С.О. Лукашин, С.Г. Назаренко

В последнее время область инструментальных программных средств для проектирования, организационной поддержки разработки и сопровождения информационных систем (ИС) претерпевает значительные изменения. Постоянное совершенствование существующих CASE-средств (ERwin, Silverrun, Oracle Developer и т. п.), доработка под нужды проектирования интегрированных средств разработки программ (Delphi, Visual C++ и т. п.), появление новых проектных методов и средств их автоматизации — все эти количественные накопления явно предполагают качественный скачок. В чем же должна состоять суть таких качественных изменений?

СИСТЕМЫ ПРОГРАММИРОВАНИЯ НОВОГО ПОКОЛЕНИЯ

К сожалению, многие публикации изданий из области информационных технологий концентрируют внимание на проблемах использования существующих CASE-средств только на этапе проектирования ИС. Причем рассматриваются они с точки зрения «высокого искусства», хотя жизнь требует промышленного производства даже в области корпоративных информационных систем под заказ. Основное противоречие процесса проектирования сводится к проблемам взаимопонимания постановщиков задачи (от заказчика) и системных аналитиков (от исполнителя), которое должно разрешаться не в области точных наук и компьютерных технологий.

По мнению авторов, движущим противоречием, лежащим в основе процесса развития CASE-систем, является противоречие между высоким уровнем декларативного языка, который использует человек в процессе проектирования программного обеспечения, и необходимостью низкого уровня языка машинной реализации программ[1]. То есть наработки в области CASE-средств необходимо рассматривать не как автоматизацию творческого и коммуникативного труда системных аналитиков, а как подготовку к созданию языков программирования нового поколения – декларативных языков программирования.

Создание и применение среды программирования на базе декларативного языка представления знаний о предметной области не является панацеей от всех бед на этапе проектирования ИС. Тем более, такая среда не сможет заменить творческий труд коллективов аналитиков по созданию некоего комплекса моделей, описывающих предметную область. Однако определение «удачного» языка, на базе которого будет строиться логическая модель предметной области, создание и программная реализация процедур перевода из этого языка в языки более низкого уровня (для которых уже есть компиляторы в машинные коды) — задача вполне разрешимая.

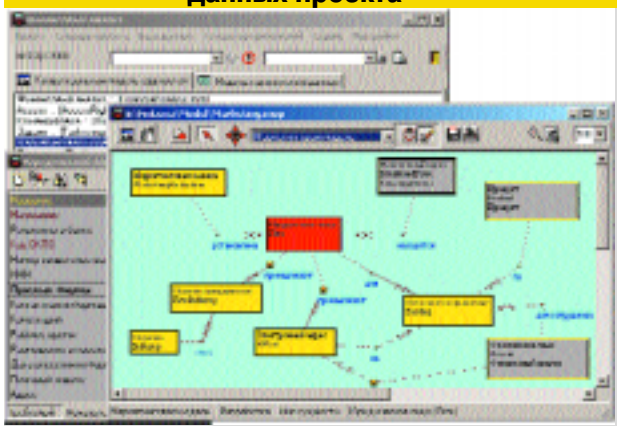
Она столь же актуальна, как в свое время была актуальной задача создания алгоритмических языков программирования высокого уровня. В необходимости решения последней уже никто не сомневается, хотя во время ее решения было немало сказано об узости выразительных средств таких языков и невозможности заменить ими труд программистов, «творчески» работающих в машинных кодах и различных Ассемблерах.

ИНСТРУМЕНТАЛЬНЫЕ ПРОГРАММНЫЕ СРЕДСТВА

Однако предмет данной статьи состоит не в теоретических основах развития современных CASE-средств до уровня интегрированных сред программирования на базе декларативного языка.

[1] *Основное направление развития языков программирования строилось на этом противоречии. Очевидно, что этот процесс продолжается и дальше.*

1 Редактор концептуальной модели данных проекта



Далее будет рассмотрен конкретный образец инструментального программного средства, система ТОПИС (Технология Объектного Проектирования Информационных Систем). Ее реализация началась в ноябре 1997 г., а начиная с сентября 1998 г. вся деятельность по производству программного обеспечения ЗАО «Бизнес Автоматика» проводится на базе этой системы (в настоящий момент — версии 2).

Каждый проект, разрабатываемый в ТОПИС, содержит три основные модели:

- концептуальную модель данных (КМД);
- модель данных приложений (МДП);
- модель представления интерфейса (МПИ).

Концептуальное моделирование данных

Концептуальная модель данных позволяет описать структуру базы данных проекта: таблицы, атрибуты, ограничения, связи между таблицами (рис. 1). Технология ТОПИС при разработке КМД основывается на стандарте IDEF1X. Графическое представление концептуальных единиц этого стандарта в технологии ТОПИС может показаться несколько необычным по сравнению с общепринятым видом ER-диаграмм. Однако эта специфика отражает реальные потребности программистов при использовании ER-диаграмм в процессе моделирования других аспектов предметной области.

При разработке КМД ТОПИС предоставляет большое количество сервисных функций, целью которых является обеспечение возможности для программиста сконцентрироваться на логической модели информационной структуры. Физическая интерпретация логической структуры на конкретную платформу системы управления базами данных (в настоящей версии поддерживаются платформы Oracle, MS SQL Server, InterBase) — одна из задач ТОПИС. Например, связи и автоинкрементный атрибут ID каждой таблицы генерируются автоматически. Каждой связи на ER-диаграмме соответствует один атрибут, принадлежащий сущности, в которую входит стрелка. Имена таких атрибутов формируются автоматически при соединении имени сущности, из которой выходит связь, и постфикса ID. Например, FirmID — имя атрибута для связи, соединяющей таблицы Email и Firm (см. рис. 1). В случае, если между двумя сущностями более одной связи, для формирования имен атрибутов связей используется дополнительный префикс, определяемый проектировщиком и указывающий на семантические особенности каждой связи. Связи могут быть каскадно удаляющими. Такие связи помечены на ER-диаграмме квадратиком. Удаляющие связи обеспечивают целостность БД с помощью автоматически генерируемого каскадного триггера удаления.

Атрибуты сущности описываются следующими параметрами:

- *название* — используется для генерации визуальных элементов, соответствующих данному атрибуту в клиентском приложении. К названию не предъявляется никаких дополнительных требований. Обычно для названия используется язык интерфейса ИС. Допускается использование пробелов;
- *имя* — используется для формирования идентификаторов, соответствующих атрибуту. В связи с этим к имени предъявляются все требования, свойственные обычному идентификатору, а также требование несовпадения со служебными словами системы управления базами данных (СУБД);
- *тип* — для описания типов атрибутов используются внутренние типы данных, которые в процессе физического создания БД интерпретируются в типы данных той СУБД, на которой производится создание БД;
- *начальное значение*;
- *набор признаков* — требуемый, уникальный, поисковый (включен в первичный ключ таблицы), индексируемый (включен в индекс).

В зависимости от типа атрибута могут указываться дополнительные характеристики (например, для символьных атрибутов указывается длина, для целочисленных — количество знаков, для атрибутов перечислимого типа — перечень допустимых значений).

Для удобства визуального восприятия большая КМД может быть разбита на подмодели. Каждая подмодель, обычно, описывает некоторый логический фрагмент общей КМД проекта (например, подмодель «Маркетинг», «Финансовый анализ» и т. д.). Сущности могут быть легко перенесены из одной подмодели в другую. В случае, если из одной подмодели необходимо установить связь на сущность другой подмодели, то для обеспечения такой возможности предусмотрено понятие «внешняя сущность» (на рис. 1 внешние сущности изображены прямоугольниками с двойной линией контура).

В ТОПИС предусмотрено большое количество функций работы с объектами физической БД, полезными для разработчиков как на этапе создания ИС, так и на этапе ее сопровождения, а именно: создание, удаление и обновление объектов БД (сохраненных процедур, триггеров, просмотров, ограничений); автоматическое обновление БД с восстановлением всех

данных в новой структуре; передача данных с одной платформы СУБД на другую и т. п.

Моделирование данных приложений

В настоящей версии ТОПИС Модель данных приложений проектируется после создания КМД. Модель данных приложения представляет собой список накопителей данных (НД). Для каждого НД из списка при помощи редактора «Дерево достижимости» (рис. 2) специфицируется его представление атрибутами КМД.

На самом деле, НД — это не результат «склеивания» атрибутов таблиц КМД, это некое хранилище данных, используемое в определенных бизнес-процессах предметной области ИС. Сформированные после моделирования бизнес-процессов хранилища данных должны превращаться в реляционную структуру КМД как с использованием существующих фрагментов КМД других проектов, так и с порождением новых сущностей, новых атрибутов и новых связей. Спецификация взаимосвязей между атрибутами хранилищ данных модели бизнес-процессов и атрибутами сущностей КМД и является моделью данных приложения. Такой подход к моделированию данных приложения предполагается в новой версии ТОПИС.

Спецификация НД используется для отображения информации в клиентском приложении. Для этого ТОПИС по НД (режимы «SQL» и «Менеджер класс» на рис. 2) автоматически генерирует в БД просмотр (view), а в клиентском приложении — менеджер-класс, обладающий методами отображения соответствующих данных на рабочих формах ИС.

Просмотр может быть дополнен вычислимыми атрибутами, построенными с использованием синтаксиса SQL для каждой поддерживаемой платформы БД.

Накопитель данных может иметь собственную модель состояний (режим «Модель состояний» на рис. 2), обеспечивающую следующие возможности для автоматической генерации клиентских приложений:

- визуализация распознанных состояний;
- управление доступом к действиям (в клиентском приложении — это пункты меню и кнопки на рабочих формах);
- отображение для пользователя информации о распознанных состояниях в текстовом виде.

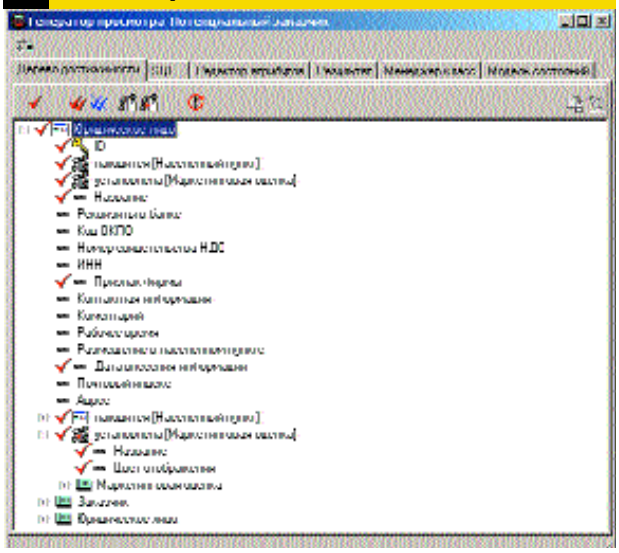
При описании модели состояний разработчик ИС имеет возможность указать способ распознавания состояния текущей строки накопителя, а также указать способы визуализации каждого распознанного состояния (можно использовать характеристики шрифта – цвет, стиль, размер, название шрифта и цвет фона).

Моделирование представления интерфейса

МПИ позволяет пользователю ТОПИС описать весь интерфейс клиентского приложения и сгенерировать работающий исходный код для среды программирования Delphi. Архитектура генерируемого клиентского приложения включает следующие типы классов:

- моделирующие классы (МК);
- менеджер-классы (МНК);

2 Спецификация накопителя данных



- классы форм редактирования (КФР);
- классы рабочих форм (КРФ);
- класс главной формы (КГФ);
- классы библиотеки бизнес-компонентов, обеспечивающих отображение и изменение информации.

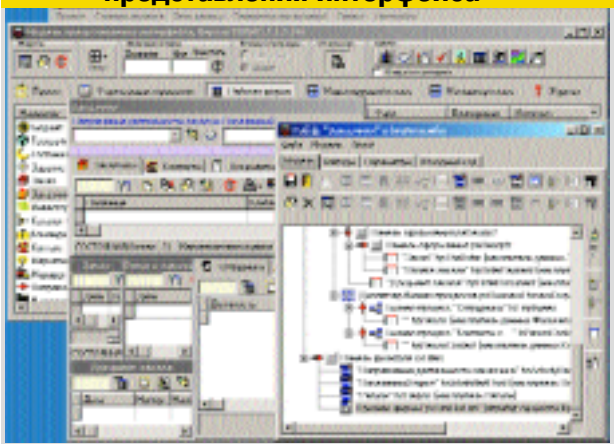
Моделирующие классы соответствуют сущностям КМД. Они являются контейнерами методов для взаимодействия компонентов интерфейса и БД. В процессе доработки программного кода после автоматической генерации макета ИС моделирующие классы дополняются прикладными методами, которые отрабатывают бизнес-правила предметной области. Несмотря на их неформализованное написание, прикладные методы учитываются ТОПИС как ресурсы проекта и воссоздаются при перегенерации программного кода клиентских приложений.

Менеджер-классы предназначены для управления процессом получения информации НД и представления ее в интерфейсных компонентах приложения. Для каждого НД из МДП генерируется соответствующий ему МНК. Они являются управляющими классами для бизнес-компонентов, взаимодействующих с табличным представлением информации, поставляемой НД из БД. Основная задача МНК — генерация и запуск SQL кода, получение от сервера набора данных и передача его связанному бизнес-компоненту.

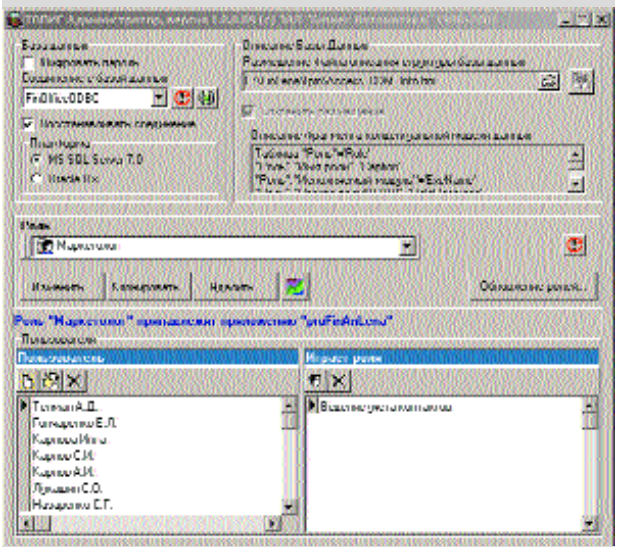
Классы форм редактирования и классы рабочих форм визуально проектируются в графическом редакторе МПИ (рис. 3). Этот редактор позволяет создать полное визуальное представление рабочих форм клиентских приложений. Компоненты, из которых конструируется форма, поступают из библиотеки бизнес-компонентов и имеют методы, необходимые для взаимодействия с соответствующими им МК и МНК.

Все бизнес-компоненты снабжены необходимой функциональностью представления интерфейса пользователя ИС, наработанной в реальных проектах. Ком-

3 Редактор модели представления интерфейса



4 Редактор ролевого доступа



понентная модель каждой формы хранится в отдельном файле, что позволяет обеспечивать коллективную работу над проектом (при помощи блокировок, устанавливаемых на редактируемые формы).

Моделирование главной формы приложения также производится из ТОПИС. В процессе создания главной формы ИС описываются главное меню и вызовы рабочих форм проекта. Отдельные элементы главного меню могут быть вынесены на панель инструментов главной формы для обеспечения быстрого доступа.

По созданной таким образом МПИ осуществляется общая генерация исходных файлов проекта на языке Object Pascal[2]. Сгенерированный код может быть сразу откомпилирован с помощью компилятора Delphi и проект запущен на выполнение. Созданный макет полностью работоспособен — можно выполнять операции просмотра, добавления, изменения и удаления данных, а также все виды фильтрации информации на рабочих формах.

Для создания полнофункциональной системы требуется доработка полученного кода приложения с целью программирования бизнес-правил предметной области. В технологии ТОПИС реализован принцип циклической разработки проекта, когда на каждом этапе работ допускается изменение проекта на любом уровне. Сгенерированный код на Object Pascal «не отрывается» от порождающих его моделей и может быть повторно пересоздан после внесения изменений в любые модели проекта. При этом изменения, которые были сделаны в исходном коде в процессе доработки макета, не теряются, что достигается благодаря применению специальной архитектуры построения системы (генерируемый код и ручные доработки кода размещаются в разных модулях и объединяются в единое целое только на этапе компиляции проекта).

Технология ТОПИС обеспечивает управление коллективной работой над проектом. Реализована возможность блокировки разрабатываемого ресурса в общей библиотеке проекта, которая находится на файловом сервере рабочей группы, копирования модуля на локальную машину разработчика для внесения изменений, а также разблокировки и помещения модуля в библиотеку проекта. После разблокировки внесенные изменения становятся доступными всей рабочей группе.

Определение структуры приложений информационной системы

ТОПИС не содержит средств моделирования структуры приложений ИС. Здесь принята идеология динамического распределения прав доступа к функциям системы. Приложение, созданное по технологии ТОПИС, при включении соответствующих опций генератора обладает возможностью ограничивать доступ пользователей ИС к любым функциональным элементам.

Для ввода подобных ограничений вместе с приложением заказчику передается и дополнительное программное средство «ТОПИС — Администратор», предназначенное для администратора ИС и позволяющее ему создавать роли и назначать их пользователям.

Каждому зарегистрированному пользователю обязательно назначается роль. Если пользователь зарегистрирован, а роль ему не назначена, то он не сможет работать с приложением.

Процесс создания роли заключается во введении ограничений на доступ к элементам приложения. В первую очередь для роли разграничивается доступ к элементам главного меню приложения. Если требуется более тонкая настройка доступа, можно запретить доступ к любому элементу выбранной формы. На рис. 4 компоненты, которые будут недоступны пользователю, играющему роль «Маркетолог», показаны серым цветом и зачеркнутым шрифтом.

[2] *Выбор языка Object Pascal и платформы Delphi обусловлен тем, что авторы статьи и их коллеги чаще всего используют данную платформу в своей профессиональной деятельности. В общем случае, конечно, желательна многоплатформенность CASE-средства не только в плане генерации серверной части ИС, но и в плане генерации клиентских приложений.*

Следует подчеркнуть, что введение ограничений в роли не требует перекомпиляции клиентского приложения. Введенные администратором ограничения динамически отражаются на работе системы, и для того чтобы они вступили в силу, пользователю необходимо только перезагрузить приложение.

Ролевое распределение доступа к ресурсам приложения чаще всего применяется для организации коллективной сетевой работы с наиболее важными документами. Например, администратор создал роли «Менеджер», «Кладовщик» и «Бухгалтер». Все они могут пользоваться одной и той же формой приложения для работы с накладными, однако, должны иметь различные права по обработке информации. «Менеджер» должен иметь возможность создавать, редактировать и удалять, «Кладовщик» — делать отметку о выдаче товара со склада, а «Бухгалтер» — оприходовать товар. Все эти возможности могут быть легко реализованы администратором при закрытии соответствующих действий на форме для каждой из ролей.

Такой динамический подход к распределению ресурсов приложения оправдывает себя также в силу частой необходимости перераспределения существующих ролевых обязанностей (например, сотрудник уходит в отпуск и его обязанности должны быть переданы другим сотрудникам).

ЗАКЛЮЧЕНИЕ

Текущая версия ТОПИС представляет собой инструмент аналитика-программиста, который, даже не будучи особо искушенным в тонкостях современных языков программирования и СУБД, сможет успешно решать вопросы разработки ИС: от создания работоспособного макета системы до доведения ее к стадии промышленной эксплуатации. Хотя по описательным возможностям логической модели предметной области она еще не полностью достигла общепринятых стандартов.

Так, модель бизнес-процессов, при помощи которой описывается постановка задачи заказчиком и которая может служить формализованным средством распределенного внесения данных в проект, должна представляться не списком хранилищ данных (в терминологии ТОПИС — накопители данных), а полноценными графическими конструкциями в формате Data Flow Diagram. Элементы данной модели являются исходным пунктом формализованного проектирования состояний информационной базы и процессов системы.

Модель состояний системы, которая служит для формализованного описания состояний информационной базы системы и связи их с моделью бизнес-процессов, также должна представляться графическими конструкциями с мощным аппаратом их спецификации. Элементы этой модели являются отправными данными для генерации методов классов клиентских приложений (которые отвечают за переход информационной базы в новое состояние) и соответствующих визуальных форм отображения состояния информационной базы.

Отсутствует пока модель процессов системы, которая должна использоваться для формализованного описания алгоритмов проекта и связи их с моделью бизнес-процессов.

Отсутствует также модель спецификации приложений для формализованного описания структуры невизуальных классов клиентских приложений, а также их увязывания с элементами других моделей.

Новая версия ТОПИС, которая должна появиться во второй половине 2002 года, будет иметь указанные выразительные возможности, что позволит полностью генерировать коды программного обеспечения серверной и клиентских частей ИС, а также расширит возможности использования ее в качестве инструмента системного анализа бизнес-процессов на предприятии.

Данылив Иван Васильевич —
канд. техн. наук, ст. науч. сотр.,
технический директор,

Лукашин Сергей Олегович —
генеральный директор,

Назаренко Сергей Геннадиевич –
канд. техн. наук, ведущий программист,
ЗАО «Бизнес Автоматика» (г. Харьков).