Технологическое решение REAL-IT: создание информационных систем на основе визуального моделирования

А.Н.Иванов
iw@tercom.ru
Санкт-Петербургский государственный университет
198504, Университетский пр., 28
Санкт-Петербург, Россия

Аннотация

В данной статье описывается технологическое решение для разработки информационных систем, основанное на использовании визуального моделирования и генерации кода по моделям. Областью применения технологического решения являются информационные системы, ориентированные на обработку данных. Технологическое решение апробировано при создании нескольких промышленных информационных систем.

Введение

Одним из распространенных подходов к разработке информационных систем (ИС) является визуальное моделирование, которое предполагает использование CASE–средств для автоматизации построения моделей системы.

CASE-пакеты 1970-1990 гг. XX в., такие, как ADW[1], Seer HPS[2] и др., предназначались для разработки произвольных информационных систем. В этом смысле их можно назвать универсальными. Их создатели пытались представить процесс разработки системы как последовательное построение взаимосвязанных моделей, на основе которых осуществляется генерация программного кода. Характерными чертами данного подхода являются

[1] Application Development Workbench, продукт компании KnowledgeWare.

[2] Seer High Productivity Systems, продукт компании Seer Technologies Inc.

необходимость детального моделирования системы с разных точек зрения, внутренняя согласованность и взаимосвязь моделей, а также однозначно определенная семантика каждого элемента модели. Практика показала, что такой подход требует очень больших накладных расходов.

В настоящее время можно выделить два основных способа использования визуального моделирования при разработки информационных систем.

1. Создание спецификаций на этапах анализа и проектирования, которые используются как иллюстрации к отдельным проектным решениям. В данном случае не требуется описывать систему детально и согласовывать между собой различные виды моделей, но и не ставится вопрос о генерации кода.

2. Создание специализированных решений с генерацией кода по диаграммам для отдельных классов систем. Например, использование диаграмм "сущность-связь" [7] для разработки схемы базы данных или моделирование и генерация событийно-ориентированной бизнес-логики телекоммуникационных систем [14]. Специализированные решения уже включают в себя реализацию общих свойств разрабатываемого класса систем, тем самым проектирование и реализация целевой системы в рамках этого решения упрощаются. Такие решения не универсальны, однако более дешевы и просты в использовании.

В данной работе представлено специализированное технологическое решение[3] REAL-IT, основанное на использовании универсального объектно- ориентированного CASE-пакета REAL [6] и включающее в себя набор генераторов программного кода и дисциплину их использования. REAL-IT предназначено для разработки класса информационных систем обработки транзакций (transaction processing systems), часто называемых также системами обработки данных (data processing systems) [8], в которых

[3]Технологическое решение [4] представляет собой набор программных средств и методик их применения, реализующих стратегию использования CASE-пакета в рамках конкретного производственного процесса, направленную на разрешение проблем этого процесса.

основной подсистемой является картотека данных. Практика показывает, что во многих информационных системах картотека занимает центральное место (картотека сотрудников в системе учета кадров, картотека товаров складской системы и т.д.), что делает актуальной задачу автоматизации процесса разработки систем этого класса.

Традиционно в процессе создания программных систем выделяют фазы анализа, проектирования и реализации. REAL-IT предназначено для автоматизации этапа реализации ИС. Этапы анализа и проектирования остаются за рамками данного технологического решения (они покрываются методологией REAL [6]). При этом, использование диаграмм UML [13] в качестве основы для генерации исполняемого кода позволяет уменьшить разрыв между этапами проектирования и реализации, поскольку именно UML является в настоящее время наиболее часто используемым языком для описания результатов проектирования.

В работе описан процесс реализации информационных систем в рамках REAL-IT. Основными этапами являются визуальное моделирование предметной области, включая моделирование расширенных ограничений ссылочной целостности [1], автоматизированная генерация базы данных и пользовательского интерфейса [3].

1 Процесс создания системы

Схема, представленная на рисунке 1, отражает основные этапы процесса реализации информационной системы в рамках технологического решения REAL-IT.

Разработка системы состоит, главным образом, в моделировании основных ее элементов — базы данных и пользовательского интерфейса. По этим моделям можно автоматически сгенерировать работающее приложение. Возможность такой генерации обеспечивается двумя факторами: стандартизацией пользовательского интерфейса и отсутствием нетривиальной логики обработки данных. В тех случаях, когда эти условия нарушаются, автоматически сгенерированный код приходится дополнять кодом, написанным программистами “вручную”. Поскольку в реальных системах такие места обязательно найдутся, архитекту-

Рис. 1: Процесс создания ИС в технологическом решении REAL-IT

ра системы предусматривает широкий набор средств для встраивания дополнительных компонент и стыковки их со сгенерированным кодом.

В следующих разделах этапы разработки системы будут рассмотрены более подробно.

1.1 Моделирование схемы данных

Реализация ИС в REAL-IT начинается с проектирования схемы данных системы. Это самый первый и самый ответственный этап — все дальнейшие шаги по реализации системы так или иначе зависят от построенной схемы. Конечно, в процессе создания системы схема данных может претерпевать изменения, тем более что большинство современных процессов разработки ПО носит итеративный характер, поэтому в REAL-IT присутствуют специальные средства для поддержки итеративного процесса [2].

В REAL-IT для моделирования схемы базы данных используется модель классов REAL, являющаяся расширением модели классов UML (по сравнению с UML, в модели классов REAL добавлены специальные конструкции для поддержки моделиро-

вания баз данных — индексы и представления, а также для моделирования систем реального времени — таймеры, двунаправленные интерфейсы, сообщения и порты). Поскольку REAL-IT ориентировано на использование в качестве хранилища данных реляционной СУБД, при описании схемы данных не используются методы класса, соответственно отсутствуют такие свойства объектно-ориентированного подхода, как полиморфизм и инкапсуляция. Таким образом, схема данных в REAL-IT хотя и использует объектно-ориентированную нотацию, на деле является диаграммой сущность-связь, расширенной отношением тип-подтип (такое расширение для логической модели данных уже давно поддерживается большинством структурных CASE-пакетов).

1.2 Описание ограничений целостности

Модель классов является спецификацией, определяющей множество возможных наборов объектов и связей между ними. Ее недостатком является то, что множество различных наборов оказывается слишком широким: все связи являются формально независимыми друг от друга, и если существует ассоциация между классами, то допустимой является связь между любой парой экземпляров этих классов. В то же время, семантика предметной области обычно содержит ограничения на возможные связи. Рассмотрим, например, два класса — "Студент" и "Учебная группа", и две ассоциации между ними — "Член группы" и "Староста". Староста группы обязан принадлежать этой же группе, однако данное ограничение не выразимо средствами модели классов. В технологическом решении REAL-IT для описания подобных ограничений используется специальная визуальная нотация, разработанная на основе диаграмм кооперации UML[4]. Подробнее эта нотация изложена в [1]. Описанные с помощью данной нотации ограничения используются при генерации программного кода системы.

[4]В CASE-пакете REAL роль диаграмм кооперации играют диаграммы объектов языка REAL.

1.3 Описание экземпляров

При разработке системы могут возникать ситуации, когда уже на стадии построения модели можно выделить отдельные элементы данных (экземпляры классов), существование которых необходимо для правильного функционирования системы. Мы будем называть такие элементы данных *предопределенными объектами*. К предопределенным объектам можно отнести также элементы некоторых элементы справочников, общие для всех экземпляров системы.

Предопределенные объекты описываются на диаграммах кооперации UML, при этом для их выделения у экземпляров классов на диаграммах кооперации вводится специальный стереотип — "Описание экземпляра". Кроме того, естественным необходимым условием для предопределенного объекта является его привязка к долгоживущему классу, не являющемуся абстрактным. При описании предопределенных объектов на диаграмме можно указывать их связи друг с другом и значения их атрибутов.

1.4 Создание представлений

При создании пользовательского интерфейса используются представления. Представление в REAL — это выборка объектов и их атрибутов, которая может содержать, в том числе, вычислимые атрибуты с учетом связей между объектами (SQL-запрос). Представления позволяют конструировать дополнительные логические срезы, или интерфейсы, элементов предметной области. Использование представлений позволяет задавать набор данных, отображаемых в элементе интерфейса, и формат этих данных. Например, разработчик может указать представление для определения набора столбцов в списке, при этом среди них могут быть указаны атрибуты различных классов.

1.5 Создание экранов

Основной принцип, принятый в REAL-IT для определения архитектуры пользовательского интерфейса — создание оконного интерфейса на основе набора конструктивных элементов. Пользовательский интерфейс конструируется из элементов различных

типов, каждый из которых визуализирует часть модели данных по заданным правилам.

Среди множества конструктивных элементов, из которых строится интерфейс пользователя, учитывая специфику данного класса систем, т.е. обработку данных и учет статистики, мы выделяем несколько наиболее часто встречающихся элементов:

- список, отображающий множество объектов;
- карточка, позволяющая посмотреть и отредактировать свойства одного объекта;
- форма-статистика, отображающая статистические данные в виде таблицы;
- форма-отношение, реализующая связь “многие-ко-многим”;
- иерархический список, представляющий иерархию однородных объектов в виде дерева.

Исходя из этого положения, в REAL-IT используется генератор для создания пользовательского интерфейса на основе шаблонов элементов интерфейса, определяющих типы элементов интерфейса. Подробнее структура и функциональные возможности интерфейса пользователя, разрабатываемые в REAL-IT, описаны в [3].

Благодаря тому что пользовательский интерфейс состоит из элементов предопределенных типов, становится возможным строить его на базе модели предметной области, используя для этого соответствующие генераторы.

Основное отличие генераторов интерфейса REAL-IT от аналогичных подходов [9, 11, 12] заключается в том, что интерфейс рассматривается не просто как непосредственное следствие модели данных, но как проекция этой модели на точку зрения пользователя системы, которая может варьироваться в зависимости от выполняемых им задач.

Генератор интерфейса REAL-IT организован в виде мастера, позволяющего разработчику в интерактивном режиме указать свойства этой проекции [3].

Генератор создает элемент интерфейса на основе информации, содержащейся в модели предметной области, шаблона для данного типа элемента и информации, внесенной пользователем

в процессе генерации. Настройка экранных форм в процессе генерации — конструирование внешнего вида интерфейса, определение свойств полей и т.д. — обеспечивает гибкость генераторов, позволяет пользователю создавать экраны, максимально удовлетворяющие его требованиям.

Модель сгенерированного экрана (включая сделанные пользователем настройки) сохраняется в репозитории, в модели классов. После генерации всех экранов приложения в модели классов будет находиться схема интерфейса, содержащая описание всех сгенерированных экранов и переходов между ними. В дальнейшем она может быть использована при перегенерации приложения. Существует набор массовых генераторов, создающих не один, а сразу несколько экранов на основе только модели интерфейса, что удобно для поддержки итеративного характера разработки интерфейса.

1.6 Генерация

По созданным моделям в REAL-IT генерируются следующие компоненты системы:

- база данных;
- программный интерфейс базы данных;
- экранные формы.

1.6.1 База данных

Для создания базы данных используется надстройка к CASE-пакету REAL — генератор базы данных. Это единственный из генераторов, который не зависит от языка реализации системы — база данных создается посредством предложений языка SQL DDL [10], причем настройка на конкретный диалект этого языка производится с помощью специального файла конфигурации.

По каждому классу модели данных в базе данных создается таблица. Исключением являются абстрактные классы — т.е. классы, у которых не может быть экземпляров, кроме экземпляров их наследников. По абстрактным классам генерируются хранимые запросы, выбирающие все экземпляры данного класса, собирая через конструкцию UNION языка SQL информацию из таблиц всех его наследников.

При генерации в каждом классе создается уникальный идентификатор объекта — поле Id, оно же становится первичным ключом. Это поле заполняется автоматически при создании каждой новой записи класса и впоследствии не может быть изменено.

Атрибуты класса переходят в поля соответствующей классу таблицы или запроса. При этом используется название атрибута и его тип. Кроме того, при генерации базы данных используется информация о предопределенных объектах –– по каждому такому объекту создается запись в соответствующей таблице.

1.6.2 Программный интерфейс базы данных

Программный интерфейс базы данных –– это компонента на языке реализации, позволяющая работать с содержимым базы данных в объектном стиле. По каждому классу модели данных в этой компоненте создается класс-оболочка (Wrapper). Эти классы позволяют работать с записями в таблицах базы данных как с объектами, реализуя стандартный набор операций: добавление, просмотр, модификация и удаление.

1.6.3 Экранные формы

Вся работа с экранными формами ведется из мастера создания экранных форм. Код формы можно сгенерировать непосредственно после ее моделирования, что позволяет сразу выявить недостатки модели и исправить их. Кроме того, мастер позволяет произвести массовую генерацию всех или выделенного множества экранных форм.

2 Поддержка циклического процесса разработки

Жизненный цикл информационных систем, разрабатываемых с помощью REAL-IT, содержит стандартные фазы: анализа, проектирования, реализации и эксплуатации [5]. На каждой фазе создается своя модель системы, состоящая из документов, диаграмм, исходных кодов и т.д., причем модель каждой следующей

фазы строится на основе модели предыдущей. В том случае, когда процесс разработки носит итеративный, или циклический характер, на всех итерациях, кроме первой, при переходе к следующей фазе возникает задача построения ее модели на основе как модели предыдущей фазы, так и модели предыдущей итерации данной фазы. Если переход между фазами осуществляется разработчиком "вручную", то задача интеграции моделей целиком ложится на его плечи. Если же переход осуществляется с использованием автоматических или автоматизированных средств, то возникает задача расширения этих средств (или дисциплины их использования) возможностью учета старой модели.

В REAL-IT применяются генераторы базы данных и пользовательского интерфейса, автоматизирующие переход от фазы проектирования к фазе реализации. При этом в проблеме поддержки итеративной разработки можно выделить две задачи — учет "ручных" изменений кода, сгенерированного по моделям фазы проектирования и сохранение информации в базе данных при ее обновлении. Для решения каждой из этих задач в REAL-IT содержатся соответствующие методики и программные средства [2].

Заключение

В статье описано технологическое решение REAL-IT. Отличительными особенностями решения являются:

- Ограничение класса информационных систем — технологическое решение предназначено для разработки систем, в которых преобладает картотека данных.
- Использование модели предметной области для создания пользовательского интерфейса.
- Генерация пользовательского интерфейса с программным кодом, не требующим модификации. Каждая экранная форма создается на основе шаблона, параметризуемого, с одной стороны, фрагментом модели предметной области (набором используемых объектов и их связей), а с другой стороны — дополнительными настройками, устанавливаемыми разработчиком.

- Использование компонентной архитектуры для построения приложений.
- Поддержка итеративного процесса разработки.

Предложенное технологическое решение использовалось при создании ряда промышленных информационных систем на различных целевых языках (Java, Visual Basic). Практика использования REAL-IT, в том числе в многолетних проектах, показала эффективность его применения для выбранного класса систем.

Список литературы

[1] Иванов А.Н. Графический язык описания ограничений на диаграммы классов UML (готовится к выходу) // Программирование. — 2004. — № 3.

[2] Иванов А.Н. Механизмы поддержки циклической разработки ИС в рамках модельно-ориентированного подхода // Наст. сборник. — С. 101-123.

[3] Иванов А.Н., Стригун С.А. Технологическое решение REAL-IT: автоматизированная разработка пользовательского интерфейса информационных систем // Наст. сборник. — С. 124-147.

[4] Кознов Д.В. Визуальное моделирование компонентного программного обеспечения: Дисс. канд. физ.-мат. наук. — СПб.: 2000. — 82 с.

[5] Соммервилл И. Инженерия программного обеспечения. — М.: Вильямс, 2002. — 624 с.

[6] Терехов А.Н. и др. Real: методология и CASE-средство разработки информационных систем и программного обеспечения систем реального времени // Программирование. — 1999. — № 5. — С. 44-51.

[7] Чен П. Модель “сущность-связь” — шаг к единому представлению данных // Системы управления базами данных. — 1995. — № 3.

[8] Avison D.E., Fitzgerald G. Information Systems Development: Methodologies, Techniques and Tools (2nd ed.). — McGraw-Hill: 2002. — 505 p.

[9] Balzert H. From OOA to GUIs — the JANUS System // Journal of Object-Oriented Programming — 1996. — Vol. 8. № 9. — P. 43-47.

[10] Database Language SQL: ISO/IEC 9075:1992. — 1992. — 693 p.

[11] Janssen C., Weisbecker A., Ziegler J. Generating User Interfaces from Data Models and Dialogue Net Specifications // Proc. InterCHI'93. — 1993. — P. 418-423.

[12] Puerta A. The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development // Proc. Computer-Aided Design of User Interfaces. — 1996. — P. 19-36.

[13] Rumbaugh J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual. — Boston: Addison-Wesley, 1999. — 576 p.

[14] Specification and Description Language (SDL): ITU-T Recommendation Z. 100. — 1993. — 204 p.