САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

Иванов Александр Николаевич

АВТОМАТИЗИРОВАННАЯ ГЕНЕРАЦИЯ ИНФОРМАЦИОННЫХ СИСТЕМ, ОРИЕНТИРОВАННЫХ НА ДАННЫЕ

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Санкт-Петербург
2005

Работа выполнена на кафедре системного программирования математико-механического факультета Санкт-Петербургского Государственного Университета.

Научный руководитель: доктор физико-математических наук, профессор Терехов Андрей Николаевич

Официальные оппоненты: доктор технических наук, профессор Лисс Александр Рудольфович

кандидат технических наук Ицыксон Владимир Михайлович

Ведущая организация: Институт систем информатики им. А.П. Ершова Сибирского Отделения Российской Академии Наук

Защита диссертации состоится "___" _____ 2005 года в ___ часов на заседании диссертационного совета Д212.232.51 по защите диссертаций на соискание ученой степени кандидата наук при Санкт-Петербургском государственном университете по адресу 198504, Санкт-Петербург, Старый Петергоф, Университетский пр., д. 28, математико-механический факультет Санкт-Петербургского Государственного Университета.

С диссертацией можно ознакомиться в Научной библиотеке Санкт-Петербургского государственного университета по адресу: 199034, Санкт-Петербург, Университетская наб., д. 7/9.

Автореферат разослан " " _____ 2005 года

Ученый секретарь
диссертационного совета
доктор физико-математических наук,
профессор Б. К. Мартыненко

Общая характеристика работы

Актуальность темы

Хотя создание различных моделей ПО при его анализе и проектировании является общепринятой практикой, возможность использования моделей для генерации кода до сих пор остается предметом дискуссий [6].

Создатели CASE-пакетов 70х-90х годов XX века пытались представить процесс разработки системы как последовательное построение взаимосвязанных моделей, на основе которых осуществляется генерация программного кода. Характерными чертами данного подхода являются необходимость детального моделирования системы с разных точек зрения, внутренняя согласованность и взаимосвязь моделей, а также однозначно определенная семантика каждого элемента модели. Однако практика показала, что такой подход требует очень больших накладных расходов, а трудности поддержки итеративного процесса разработки делает его плохо применимым в большинстве реальных проектов. Это привело, в значительной степени, к разочарованию в идее генерации кода по моделям. Многие современные исследователи и практики считают, что модели можно использовать только как наброски, используемые в процессе обсуждения или изложения проблем и концепций. Однако, очевидные преимущества кодогенерационного подхода – снижение времени на разработку и числа ошибок за счет отказа от ручного кодирования – заставляют искать новые способы реализации этого подхода, с учетом предыдущих неудач.

Цели работы

Исследовать методы модельно-ориентированной разработки ПО в применении к информационным системам, ориентированным на данные. Разработать модели пользовательского интерфейса и ограничений на данные. Разработать методику поддержки итеративного процесса разработки при использовании модельно-ориентированного подхода. Реализовать предложенные решения в виде промышленной технологии создания информационных систем.

Основные результаты

В работе получены следующие результаты:

1. Создана модель пользовательского интерфейса ИС, ориентированных на работу с данными, методика ее построения и алгоритм генерации кода по предложенной модели.
2. Расширена модель классов UML для проектирования объектно-реляционных баз данных.

3. Создан диаграммный язык для описания ограничений на модель классов.
4. Разработана методика поддержки итеративной разработки ПО при использовании генераторов кода ИС по визуальным моделям.
5. Выполнена промышленная реализация выработанных подходов.
6. Проведена апробация представленных решений в ряде промышленных проектов.

Научная новизна

1. Модель пользовательского интерфейса ИС отличается от аналогичных подходов [1,4] богатой микромоделью, позволяющей создавать сложные экранные формы, содержащие такие дополнительные элементы, как фильтры в списках, встроенные списки в карточках и т.д., с привязкой этих элементов к соответствующим элементам модели данных. Кроме того, новым является выделение связей многие-ко-многим как отдельных элементов модели данных, требующих специального представление в пользовательском интерфейсе, и предложение в качестве такого представления экранной формы специального вида – формы-отношения.
2. Методика построения модели интерфейса отличается тем, что в качестве основного способа формирования модели предлагается использовать не работу в графическом редакторе CASE-пакета, а использование специальных мастеров (Wizards), позволяющих построить модель для одной экранной формы и, при необходимости, сразу же сгенерировать код для него. Такой подход позволяет сгладить разрыв между моделированием и WYSIWYG («What You See Is What You Get») подходом, использующемся в большинстве современных средств разработки интерфейса.
3. Расширение модели классов UML для проектирования объектно-реляционных баз данных отличается тем, что является расширением метамодели UML, а не UML-профилем [1]. Преимущества такого подхода заключаются в возможности поддерживать целостность представлений и модели данных средствами CASE-пакета, что невозможно в случае профиля.
4. Язык для описания ограничений на модель классов отличается от аналогичных подходов [3,7] тем, что предлагаемые диаграммы обычно «визуально похожи» на ограничиваемый фрагмент модели классов (эта похожесть достигается за счет наличия гомоморфизма между моделью классов и моделью ограничения). Кроме того, синтаксис предлагаемого языка основан на синтаксисе диаграмм

кооперации языка UML, что позволяет описать его как профиль UML и использовать в большинстве CASE-пакетов, поддерживающих UML.

5. Методика поддержки итеративной разработки отличается тем, что позволяет производить повторную генерацию кода в случае изменения исходных моделей, сохраняя при этом, в большинстве случаев, «ручные» доработки сгенерированного кода.

Практическая ценность

Предложенные решения были положены в основу технологии разработки информационных систем REAL-IT, созданной под руководством автора и использованной для разработки ряда ИС для отечественных и зарубежных заказчиков, в том числе линейки продуктов «Студент», «Стипендия», «Аспирант», использующихся на ряде факультетов СПбГУ, АИС БТИ Ленинградской области, а также системы автоматизации управления страховыми контрактами, разработанной в компании BridgeQuest.

Апробация работы и публикации

Результаты диссертации были доложены на всероссийских научно-методических конференциях ТЕЛЕМАТИКА'2002, ТЕЛЕМАТИКА'2003 и семинарах кафедры системного программирования СПбГУ. По теме диссертационной работы было опубликовано 10 научных работ. Работа поддержана грантом РФФИ № 05-0951/07.

Структура и объем диссертации

Диссертационная работа состоит из введения, семи глав, заключения и списка литературы. Работа содержит 130 страниц, 23 рисунка, список литературы из 104 наименований.

Содержание работы

Во введении обосновывается актуальность темы исследования, формулируются цели и задачи работы.

Данная диссертация посвящена разработке подхода к генерации кода информационных систем, ориентированных на данные, на основе визуальных моделей. Характеристическим свойством этого класса систем является тот факт, что наиболее важным архитектурным элементом системы является модель данных и ее реализация – схема БД, а также, что основными компонентами системы являются база данных и интерфейс пользователя.

Направленность на один конкретный класс систем позволяет сузить число используемых моделей и генерировать по ним полностью работоспособное приложение, а использование мастеров (Wizards) –

облегчить создание этих моделей, соблюдая их согласованность. Специальное внимание уделяется интеграции сгенерированного и написанного «вручную» (в т.ч. «поверх» сгенерированного) кода, позволяющей применять предложенный подход в проектах, использующих итеративные процессы разработки.

В первой главе рассматриваются определение и классификация информационных систем, определятся функциональность ИС, ориентированных на данные. В качестве основных функций выделяются функции непосредственной работы пользователя с данными – ввод, просмотр и редактирование.

Во второй главе описываются предлагаемый процесс разработки ИС (рис.1) в технологии REAL-IT.

Рисунок 1. Процесс создания ИС в технологии REAL-IT

Разработка системы состоит, главным образом, в моделировании основных ее элементов – базы данных и пользовательского интерфейса. Модель данных состоит из двух подмоделей – схемы данных, которая представляет собой стандартную модель классов UML, и модели

ограничений на данные, которые описываются с помощью предложенного в работе визуального языка.

Модель интерфейса можно разделить на модель представлений, описывающую логическое представление данных для пользователя, и модель экранов, описывающую оконный интерфейс пользователя.

Все эти модели основаны на языке UML и хранятся в репозитории объектно-ориентированного CASE-пакета. Для их построения используются либо редактор диаграмм CASE-пакета, либо специальные мастера (Wizards), входящие в состав REAL-IT.

Наличие собственных моделей означает специализацию UML. В большинстве случаев, для этого достаточно использовать механизм UML-профилей. В частности, как профили можно описать специализации для модели ограничений и модели интерфейса. Однако, использование такого подхода для представлений не обеспечивает необходимого контроля целостности модели, поэтому в работе предлагается использовать в данном случае не профиль, а расширение метамодели UML специальными элементами, приводится схема метамодели для этого расширения.

По разработанным моделям REAL-IT позволяет 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, архитектура системы предусматривает широкий набор средств для встраивания дополнительных компонент и стыковки их со сгенерированным кодом.

Третья глава посвящена вопросу визуального моделирования ограничений на данные.

Рассматриваются существующие подходы – такие как Visual OCL [3] и Constraint Diagrams [7], анализируются их возможности и недостатки.

Выделяется класс ограничений следующего вида: выделяются два объекта (экземпляры классов), связанные ассоциацией, а также другие, связанные с ними, объекты и ассоциации (их мы будем называть контекстом ассоциации). При этом ассоциация допустима только в том случае, если такой контекст можно построить, т.е. существуют объекты, связанные определенными в контексте ассоциациями.

Для данного класса ограничений предлагается язык описания, основанный на использовании диаграмм кооперации UML.

Каждая диаграмма ограничений представляет собой граф, вершинам которого соответствуют классы в модели классов, на которую накладываются ограничения. При этом одному классу может соответствовать произвольное количество вершин. Ребрам графа соответствуют ассоциации между классами. При этом одно из ребер выделяется – оно соответствует ограничиваемой ассоциации. Используются два типа элементов диаграмм кооперации – объекты и связи, сообщения не используются. Для связей, соответствующих ограничиваемым ассоциациям вводится стереотип «Limited». При этом считается, что диаграмма кооперации является диаграммой ограничений в том и только том случае, если на ней присутствует хотя бы одна связь со стереотипом «Limited» (в предлагаемой версии нотации такая связь на диаграмме может быть только одна). У всех объектов на диаграмме ограничений должны быть указаны их классы, а у связей – ассоциации.

В качестве примера рассмотрим модель данных, состоящую из трех классов – подразделения, сотрудника и компьютера (рис.2). Предположим, что сотруднику может быть предоставлено право работать только компьютерах того подразделения, в котором он работает. Это ограничение невозможно выразить средствами модели данных, с помощью ограничений ссылочной целостности или как-либо еще. Спецификация этого ограничения с помощью предлагаемого языка показана на рис.3.

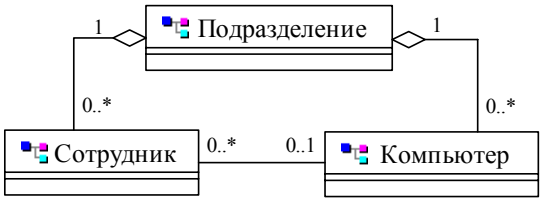

Рисунок 2. Пример модели классов для ограничения

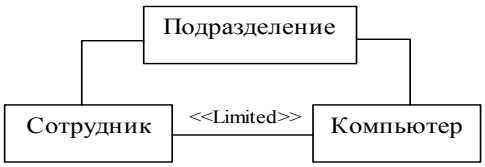

Рисунок 3. Пример ограничения

Рассматривается несколько расширений предлагаемой модели:

- Модель с отрицаниями. Нотация диаграмм ограничений расширяется связями с новым стереотипом – «Absent», позволяющим описывать

отрицательные условия, т.е. элементы контекста, которые должны отсутствовать для выполнения ограничения.

- Модель с ограничениями на отдельные объекты. В ее рамках можно определять допустимость связи между двумя объектами не только в зависимости от наличия других связей, но и, например, в зависимости от значений атрибутов этих объектов (или связанных с ними). Эти дополнительные ограничения могут быть специфицированы с помощью OCL или любым другим способом.
- Дизъюнктная модель. Все ограничения, описанные для системы, разбиваются на группы, в каждую из которых попадают ограничения, содержащие одну и ту же ограничиваемую ассоциацию. Ограничения в каждой группе считаются связанными дизъюнкцией, в то время как между собой группы связаны конъюнкцией.

Для предлагаемого языка, включая все перечисленные выше расширения, описывается формальная семантика.

Четвертая глава посвящена моделированию пользовательского интерфейса ИС. Рассматриваются существующие подходы к модельно-ориентированной разработке пользовательского интерфейса (Model-Based User Interface Development, MBUID) [5,8]. Как отдельное направление рассматриваются средства моделирования Web-интерфейсов, проводится сопоставление моделей, используемых в MBUID и Web-моделировании. Выделяются два основных способа построения модели интерфейса – на основе модели задач и на основе модели предметной области.

Выделяются два уровня в оконном интерфейсе пользователя:

- Макроуровень, описывающий набор окон и связей между ними. Связи обозначают возможность вызова одного окна из другого.
- Микроуровень, описывающий набор элементов управления каждого окна и свойства этих элементов управления.

Соответственно, модель интерфейса можно разделить на макромодель и набор микромоделей (для каждого окна).

На макроуровне выделяются несколько типов наиболее часто встречающихся окон:

1. Список, отображающий множество однородных объектов.
2. Карточка, позволяющая просматривать и редактировать свойства одного объекта.
3. Форма-отношение, реализующая связь «многие ко многим».

Формы этих типов могут быть связаны следующим образом:

1. Список с карточкой для просмотра и редактирования элемента списка. Позволяет пользователю просматривать/редактировать информацию об отдельном объекте, а также создавать новые объекты.
2. Карточка со списком или отношением для связанных объектов. Позволяет из карточки объекта переходить к просмотру/редактированию его связей с другими объектами. Связи этого вида можно инкапсулировать в составное окно – т.е. можно сгенерировать окно, состоящее из карточки и связанных с нею списков.
3. Отдельные элементы одного окна с другим окном – например, возможность вызова списка или карточки из фильтра или поля выбора.

Набор типов окон и связей между ними составляют макрометамодель интерфейса. Для каждого типа окон, кроме того, разработана его микрометамодель, определяющая возможности разработчика по управлению содержимым окна.

Наполнение модели интерфейса ведется, главным образом, с помощью мастеров (wizards), позволяющих создать или изменить модель отдельного окна, включая связи его с другими окнами. По окончании моделирования, мастер позволяет сразу же сгенерировать код окна, что позволяет разработчику контролировать результаты свой работы не только на уровне модели, но и непосредственно посмотреть на смоделированное окно. Можно также сгенерировать код по всей модели сразу или по выбранному фрагменту – особенно актуально это бывает при изменении модели данных или шаблонов генерации.

Генерация кода по модели происходит на основе шаблонов, изменение которых позволяет изменить внешний вид всего приложения в соответствии с требованиями конкретного заказчика.

Пятая глава посвящена поддержке итеративного процесса разработки при использовании генерации кода по моделям. Рассматриваются существующие методики поддержки итеративной разработки, для данного класса систем выделяются две основные задачи – учет «ручных» изменений кода, сгенерированного по моделям фазы проектирования (переход с фазы проектирования на фазу реализации), и миграции данных при эволюции схемы БД (переход с фазы реализации на фазу эксплуатации).

При рассмотрении первой задачи формулируется противоречие между необходимостью внесения «ручных» изменений в порожденную программу и необходимостью периодически производить перегенерацию. Рассматриваются следующие способы разрешения этого противоречия:

1. Разделение сгенерированного кода на компоненты, одни из которых не требуют модификации, а другие подвергаются модификации в процессе доработки и сопровождения программного кода. Перегенерация используется только для компонент первого вида, компоненты второго вида сопровождаются «вручную».

2. Ведение учета «ручных» изменений сгенерированного кода и их повторное внесение после каждой перегенерации.

3. Выделение модификаций кода в отдельные компоненты с автоматическим добавлением их в итоговую программу. Основные способы осуществления такого добавления – это использование препроцессора целевого языка, директив сборщика, специально написанных для этой цели программ (обычно, скриптов) или препроцессоров (например, фреймовых или аспектных), а при объектно-ориентированном подходе – использование полиморфизма (шаблон проектирования «Generation Gap»).

4. Расширение источников информации для кодогенерации – использование дополнительных моделей, расширение набора свойств элементов модели. При этом количество «ручных» изменений уменьшается за счет того, что специфичную семантику, которая вынуждает вносить эти изменения, удается внести в модель, используемую для генерации.

5. Выделение «ручных» изменений из кода системы путем автоматического анализа измененного текста программы и автоматическое внесение их при перегенерации.

Перечисленные выше способы анализируются с точки зрения применимости их к различным компонентам ИС.

Для решения задачи сохранения информации в БД предлагается не вносить изменения в существующую базу, а переносить информацию из старой базы в пустую базу с обновленной схемой, при необходимости, выполняя при этом перевод данных в новый формат. Рассматриваются проблемы такого подхода и пути их решения.

В шестой главе описывается реализация предложенных подходов на различных платформах – Windows (Visual Basic), J2SE и J2EE (Web).

Несмотря на наличие у каждой из этих платформ своих специфических особенностей, для каждой из них оказалось возможным создать набор инструментальных средств (генераторов кода, библиотек, вспомогательных утилит), реализующих предлагаемые в работе методики разработки информационных систем. При этом большая часть исходных текстов генераторов кода используется одновременно во всех версиях технологии. Основное отличие разных версий заключается в архитектуре

библиотек динамической поддержки и способе внесения «ручных» изменений в сгенерированный код.

Отдельно рассматриваются меры, направленные на оптимизацию времени исполнения сгенерированного приложения, поскольку технология REAL-IT позволяет создавать достаточно сложные по структуре экранные формы, содержащие большое количество элементов, инициализация которых требует выполнения запросов к базе данных. «Наивная» реализация, при которой все элементы инициализируются в момент загрузки формы, может вызвать длительную задержку при открытии окна. Использование техники «ленивой» инициализации позволяет разделить эту задержку на несколько коротких (приемлемых для пользователя) задержек, возникающих при взаимодействии пользователя с конкретными элементами формы. Кроме того, общая сумма этих задержек уменьшается за счет того, что обычно на сложных формах часть элементов редко используется и в большинстве случаев не требует полной инициализации. Дополнительная оптимизация достигается за счет учета зависимостей между полями и передачи дополнительной информации между формами (вместо повторного запроса к базе данных).

В седьмой главе сформулированы основные направления дальнейших исследований по теме работы. К ним относятся как развитие описанных в работе подходов и методик, так и расширение предлагаемой технологии за счет включения в нее новых методик.

В качестве конкретных направлений исследований определены следующие:

- В области моделирования расширенных ограничений ссылочной целостности – использование рекурсивных связей между объектами при описании контекста ограничения.
- В области моделирования пользовательского интерфейса – добавление новых элементов в его макрометамодель, выявление и формализация новых типов форм и новых возможностей по построению составных форм.
- Моделирование прав доступа на объекты системы. Возможность описывать распределение прав между пользователями системы с помощью высокоуровневых средств моделирования, на основе функциональной модели и модели предметной области.
- Поддержка разработки линеек (product line) и семейств (product family). Реализация такой поддержки в REAL-IT требует создания средств для их моделирования, включая описание общих частей и специализаций для каждого из членов семейства, а также использование этих моделей в генераторах для порождения кода.

- Использование в дополнение к модели данных для реализации системы модели бизнес-процессов. Генерации кода на основе нескольких независимых моделей, каждая из которых описывает систему со своей точки зрения, является одной из наиболее актуальных, на сегодняшний день, нерешенных проблем в области использования модельно-ориентированных подходов к разработке ПО.

В заключении приводятся основные результаты работы.

Список литературы

1. Максимчук Р., Нейбург Э. Проектирование баз данных с помощью UML: Пер. с англ. — М.: Вильямс, 2002. — 288 с.
2. Balzert H. From OOA to GUIs — the JANUS System // Journal of Object-Oriented Programming — 1996. — Vol. 8. N 9. — P. 43-47.
3. Bottoni P. at al. A Visualization of OCL using Collaborations // Unified Modeling Language — Modeling Languages, Concepts, and Tools (UML'2001). — Springer, 2001. — P. 257-271.
4. Ceri S., Fraternali P., Bongio A. Web Modeling Language (WebML): a modeling language for designing Web sites // Computer Networks. — 2000. — Vol. 33. N 1. — P. 137-157.
5. da Silva P. User Interface Declarative Models and Development Environments: A Survey // Lect. Notes Comput. Sci. — 2000. — Vol. 1946. — P. 207-226.
6. Fowler M. What Is the Point of the UML // Lect. Notes Comput. Sci. — 2003. — Vol. 2863. — P. 325.
7. Kent S. Constraint Diagrams: Visualising Invariants in Object-Oriented Models. // ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA '97). — ACM Press, 1997. — P. 327-341.
8. Schlungbaum E. Model-Based User Interface Software Tools. Current state of declarative models. — Atlanta, 1996. — (Tech. Rep. / Graphics, Visualization & Usability Center / Georgia Institute of Technology).

Работы автора по теме диссертации

9. Иванов А.Н. Графический язык описания ограничений на диаграммы классов UML // Программирование. — 2004. — N 4. — С. 204-208.

10. Иванов А.Н. Механизмы поддержки циклической разработки ИС в рамках модельно-ориентированного подхода // Системное программирование. — СПб, 2004. — С.101-123.

11. Иванов А.Н. Технологическое решение REAL-IT: создание информационных систем на основе визуального моделирования // Системное программирование. — СПб, 2004. — С.89-100.

12. Иванов А.Н., Стригун С.А. Технологическое решение REAL-IT: Моделирование и генерация пользовательского интерфейса // Системное программирование. — СПб, 2004. — С.124-147.

13. Стригун С.А., Иванов А.Н., Соболев Д.И. Технология REAL для создания информационных систем и ее применение на примере системы «Картотека» // Математические модели и информационные технологии в менеджменте. Выпуск 2. — СПб, 2004. — С.120-139.

14. Худякова Ю.М., Иванов А.Н., Васильева Н.С. Информационная система «Студент». Механизм распределения прав доступа. // Телематика-2003: Тез. докл. Всерос. науч.-методич. конф. — СПб, 2003.

15. Стригун С.А., Иванов А.Н. Опыт создания автоматизированной системы управления учебным процессом ВУЗа. // Телематика-2002: Тез. докл. Всерос. науч.-методич. конф. — СПб, 2002.

16. Иванов А.Н. Высокоуровневый механизм описания прав доступа на разрабатываемую программную систему // Объектно-ориентированное визуальное моделирование. — СПб, 1999. — С. 78-85.

17. Терехов А.Н., Романовский К. Ю., Кознов Дм.В., Долгов П.С., Иванов А.Н. Real: Методология и CASE-средство для разработки систем реального времени и информационных систем // Программирование — 1999. — N 5. — С. 44-51.

18. Терехов А.Н., Романовский К.Ю., Кознов Дм.В., Долгов П.С., Иванов А.Н. Объектно-ориентированная методология разработки информационных систем и систем реального времени // Объектно-ориентированное визуальное моделирование. — СПб, 1999. — С.4-20.

Подписано к печати . .2005. Формат бумаги 60x84 1/16.
Бумага офсетная. Печать ризографическая.
Объем 1 усл. п.л. Тираж 100 экз. Заказ .
Отпечатано в отделе оперативной полиграфии НИИХ СПбГУ
с оригинал-макета заказчика.
198504, Санкт-Петербург, Старый Петергоф, Университетский пр., 26.