Глава 7

Проектирование пользовательского интерфейса

Дни, когда экран компьютера напоминал “великого немого” и светился скучным зеленым цветом, а клавиатура служила единственным устройством ввода информации, ушли в прошлое. Сегодня экраны отличаются большей “смышленостью” и практически неограниченным цветовым разнообразием, а пользователь для управления выполнением программы вооружен мышью (не говоря уже о речевом и тактильном вводе). Конечно, программы по-прежнему создаются в расчете на то, чтобы не допустить неверных или запрещенных действий, однако сдвиг в управлении от алгоритма к пользователю заметно повлиял на способы проектирования и реализации GUI-систем.

Разработка пользовательского интерфейса начинается с ранних набросков диалоговых GUI-окон на этапе анализа требований. Эти наброски используются в процессе сбора требований, при разборе возможных сценариев работы системы с заказчиками, для создания прототипов и для включения документов описания прецедентов. В процессе проектирования осуществляется дальнейшая разработка окон GUI-интерфейса для приложения в соответствии с основными возможностями презентационного ПО GUI-интерфейса, а также особенностями и ограничениями выбранной программной среды. Эти вопросы являются предметом рассмотрения данной главы.

7.1. Проектирование интерфейса как междисциплинарная деятельность

Проектирование графического интерфейса пользователя (GUI — Graphical User Interface) представляет собой *междисциплинарную деятельность*. Оно требует усилий многофункциональной бригады — один человек, как правило, не обладает знаниями, необходимыми для реализации многоаспектного подхода к проектированию GUI-интерфейса. Надлежащее проектирование GUI-интерфейса требует объединения навыков художника-графика, специалиста по анализу требований, системного проектировщика, программиста, эксперта по технологии, специалиста в области социальной психологии, а также, возможно, некоторых других специалистов, в зависимости от характера системы.

Типичный процесс проектирования GUI-интерфейса для приложений ИС начинается с *прецедентов*. *Аналитик*, занимающийся описанием потока событий для прецедента, обладает некоторым зрительным образом GUI-интерфейса для поддержки человеко–машинного взаимодействия. Сложные человеко–машинные взаимодействия нельзя верно передать, пользуясь лишь "языком прозы". Иногда процесс сбора и согласования требований заказчика делает необходимым выработку эскизов GUI-интерфейса.

Проектировщик, участвующий в определении кооперативных взаимодействий для реализации прецедентов, должен иметь ясное зрительное представление о том, как выглядят экранные отображения, выраженные с помощью GUI-интерфейса. Если до него этого уже не сделал аналитик, проектировщик становится первым человеком, который вырабатывает графические представления пользовательского интерфейса. Эти графические представления, предлагаемые проектировщиком, должны соответствовать технологии, на которой базируется GUI-интерфейс — средствам организации окон и элементам управления окнами, Internet-броузерам и т.д. Чтобы успешно воспользоваться технологическими возможностями, может потребоваться прибегнуть к консультации *эксперта в области технологии*.

Прежде, чем проект кооперативных действий классов попадет к программистам для реализации, необходимо сконструировать прототип GUI-экранов, "дружественный" пользователю. Для решения этой задачи привлекаются *художники-графики* и *специалисты по социальной психологии*. Совместно они могут предложить привлекательный и удобный GUI-интерфейс.

Задача *программиста* заключается не в том, чтобы слепо реализовать экраны, но также предложить изменения, обусловленные средой программирования. В некоторых случаях изменения могут привести к улучшению GUI-интерфейса, в других случаях изменения отрицательно сказываются на проекте из-за ограничений, связанных с программированием или продуктивностью.

Из приведенного выше краткого обсуждения ясно, что проектирование GUI-интерфейса — очень обширная задача. На тему разработки GUI-интерфейса написано много книг, уделяющих основное внимание различным аспектам этой деятельности [16], [27], [26], [59], [73], [92]. В данной главе мы сконцентрируемся на том, что должен знать системный проектировщик, чтобы в *сотрудничестве с другими специалистами разработать* удачный интерфейс.

7.2. Интерфейс: от прототипа к реализации

Основной момент в проектировании GUI-интерфейса заключается в том, что *контроль находится на стороне пользователя* (при условии, что система, а не пользователь, контролирует системную целостность, защиту и безопасность). Современные объектно-ориентированные программы *управляются событиями*. Объекты реагируют на события (сообщения). Внутренние взаимодействия между объектами запускаются внешними событиями, инициируемыми пользователем.

"Впечатление и ощущение" от GUI-интерфейса служит "двигателем торговли" при продаже программного продукта заказчику. Прототипы GUI-интерфейса могут служить двоякой цели оценки "ощущения" от GUI-экранов и передачи его функций. Истинное "впечатление" становится доступным на этапе реализации.

Прототип GUI-интерфейса для класса `Organization` представлен на рис. 7.1, его основной целью является визуализация данных и контроль объектов, расположенных

в окне. Обратная связь с пользователями, получаемая в виде их откликов, а также идеи бригады разработчиков GUI-интерфейса изменяют "впечатление" от окна, а, возможно, и "ощущение" от него.

Пример 7.1. Управление контактами с клиентами

Обратитесь к постановке задачи для приложения *Управление контактами с клиентами* (разд. 2.3.3), а также к решенным примерам в главе 4. В частности, рассмотрите проект класса `Organization` в примере 4.8 (разд. 4.2.2.3).

Цель этого примера — продемонстрировать то, каким изменениям может подвергнуться GUI-экран для класса `Organization` от первоначального прототипа до конечной реализации. Мы предполагаем, что в качестве базовой технологии создания GUI-интерфейса выступает технология Microsoft Windows.

На рис. 7.2 показана возможная реализация окна `Organization` как диалогового. Как можно видеть из рисунка, программист предпочел реализовать окно в виде вкладок; кроме того, в окно был внесен ряд других изменений, касающихся "впечатления и ощущения" от него, таким образом, чтобы привести их в соответствие с принципами проектирования GUI-интерфейса для Microsoft Windows.

Рис. 7.1. Прототип окна для класса Organization (Управление контактами с клиентами)

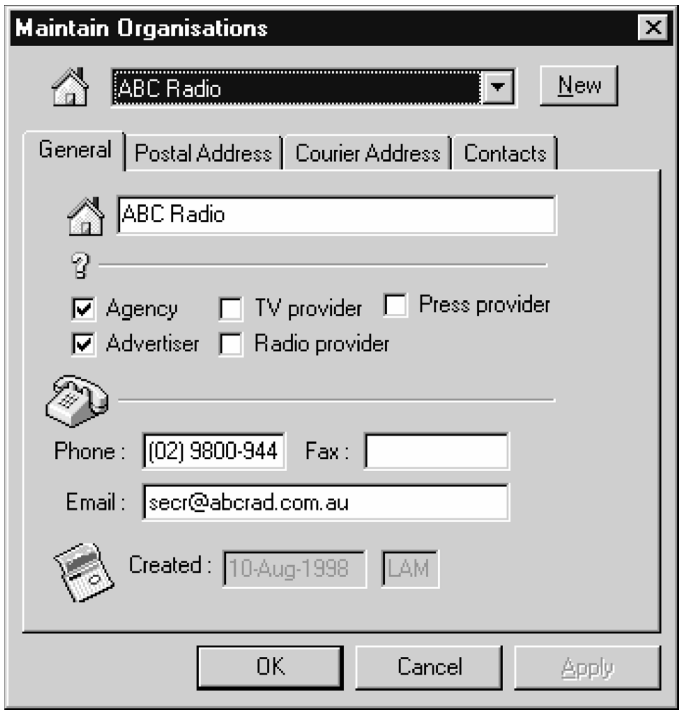

Рис. 7.2. Реализованное окно для класса Organization (Управление контактами с клиентами)

7.3. Руководящие принципы проектирования интерфейса, ориентированного на пользователя

Центральным звеном при проектировании GUI-интерфейса выступает пользователь. Это замечание послужило основанием для ряда приводимых ниже рекомендаций разработчикам ПО. Руководящие принципы опубликованы производителями GUI-интерфейсов [92]. Они также служат предметом обсуждения многих книг [27], [73].

Руководящие принципы служат разработчикам основанием для построения *GUI-интерфейса*. При принятии любых проектных решений в отношении GUI-интерфейса они должны использоваться разработчиками на подсознательном уровне. Некоторые из этих руководящих принципов выглядят как хорошо известные старые истины, другие основаны на современной GUI-технологии.

7.3.1. Контроль — на стороне пользователя

"*Контроль – на стороне пользователя*" — вот главнейший принцип построения GUI-интерфейса. Лучше было бы назвать этот принцип *пользовательским восприятием контроля*. Некоторые называют его принципом *отсутствия какой-либо* "*материнской опеки*" — программа не должна действовать как ваша заботливая мать, делая многие вещи

за вас [89]. Основной смысл этого принципа заключается в том, что пользователь инициирует действия, и если в результате этого контроль переходит к программе, то пользователь получает необходимую *обратную связь* (в виде курсора в форме песочных часов, индикатора ожидания или аналогичным способом).

Рис. 7.3 демонстрирует типичный поток управления в человеко–машинном взаимодействии. Событие, инициированное пользователем (выбор пункта меню, щелчок мышью, перемещение указателя мыши по экрану и т.д.), может привести к открытию окна GUI-интерфейса или вызову программы — как правило, программы на языках типа 4GL или SQL в рамках приложения ИС. Программа временно перехватывает контроль у пользователя.

Процесс выполнения программы имеет возможность вернуть управление назад тому же или другому окну. В другом случае он может вызвать другой модуль на языках типа 4GL или SQL или вызвать внешнюю процедуру. В некоторых случаях программа может кое-что делать для пользователя. Это возможно, к примеру, если программе требуется выполнить вычисления, которые обычно связаны с явным пользовательским событием, или если программа перемещает указатель на другое поле экрана, а для события перемещения с исходного поля предусмотрен обработчик события выхода, связанный с ним.

7.3.2. Согласованность

Согласованность несомненно является вторым основным принципом разработки качественного интерфейса. Фактически согласованность означает соблюдение стандартов и следование некоторым общепринятым правилам работы с GUI-интерфейсом. Согласованность может рассматриваться, по меньшей мере, в двух аспектах.

- Соответствие стандартам поставщика GUI-интерфейса.
- Соответствие стандартам в области именования, программирования и другим, разработанным внутри организации стандартам, которые связаны с GUI-интерфейсом.

Оба аспекта одинаково важны, и второй (на который оказывают влияние разработчики) не должен противоречить первому. Если приложение разрабатывается для Windows, то следует обеспечить “впечатление и ощущение”, свойственные работе в системе Windows. Для системы Macintosh замена знаменитого “яблочного” меню вложенным меню (типа “кенгуру”), как это раз попытался сделать автор, вообще никудышная идея!

Разработчик GUI-интерфейса не должен слишком увлекаться творчеством и предлагать необычные новшества. Это может плохо сказаться на уверенности и умении пользователей. Пользователям следует представлять знакомую среду, поведение которой предсказуемо. Как заметил Трайсман (Treisman) [89], можно представить, какова была бы реакция автомобилистов, если бы производитель автомобилей выпустил на рынок новую машину, у которой поменяли местами педали скорости и тормоза!

Не следует также недооценивать соответствие внутренним стандартам в области именования, программирования, аббревиатур и т.п. Сюда относятся именование и программирование меню, командных кнопок, полей экранов, также стандарты по расположению объектов на экране и последовательному использованию элементов GUI-интерфейса в рамках всех приложений, разрабатываемых собственными силами.

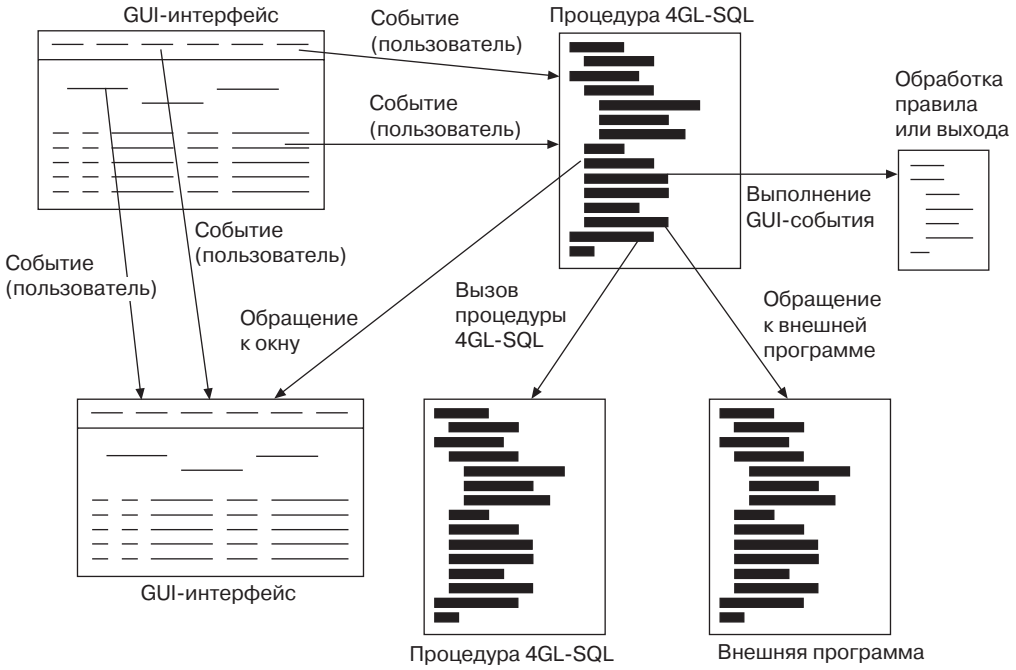

Рис. 7.3. Поток управления GUI-программы

7.3.3. Индивидуализация и настройка

Индивидуализация и настройка — два взаимосвязанных принципа разработки GUI-интерфейса. Индивидуализация GUI-интерфейса — это просто его настройка под персональные нужды, в то время как настройка — так, как мы понимаем ее здесь — административная задача приспособления ПО к требованиям различных групп пользователей.

Примером индивидуализации является изменение пользователем порядка и размеров колонки в программе просмотра строк (так называемые *сетки* — *grid*) с последующим сохранением этих изменений как его личных предпочтений. При обращении к этой же программе в будущем данные предпочтения учитываются.

Примером настройки является различие в функционировании программы по отношению к опытным пользователям и пользователям-новичкам. Например, пользователю-новичку программа может предложить явную помощь и дополнительные предупреждающие сообщения, указывающие на потенциальную опасность некоторых событий, инициируемых пользователем.

Во многих случаях различия между индивидуализацией и настройкой весьма размыты и малозаметны. Изменение пунктов меню, создание новых меню и т.п. попадают в обе категории. Если они предназначены для индивидуального использования — это индивидуализация. Если они осуществляются системным администратором в интересах всего коллектива пользователей — это настройка.

7.3.4. Терпимость к ошибкам

Хорошо спроектированный интерфейс должен позволять пользователям экспериментировать и совершать ошибки, проявляя *терпимость к ошибкам*. Подобная терпимость стимулирует исследовательскую активность пользователя, поскольку позволяет ему выполнять ошибочные последовательности действий с возможностью в любой момент совершить при необходимости "откат" в начало. Терпимость к ошибкам подразумевает многоуровневую систему отмены операций.

Об этом принципе легко говорить, однако, он трудно поддается реализации. Реализация терпимости к ошибкам в интерфейсе отличается особой сложностью для многопользовательских приложений баз данных. Пользователь, который снял (и потратил) деньги с банковского счета, не имеет возможности отменить эту операцию! Единственное, что он в состоянии сделать — исправить ошибку (если, конечно, это было ошибкой!), положив деньги назад на счет, осуществив другую операцию. Должен или не должен терпимый к ошибкам интерфейс предупреждать пользователя о последствиях снятия денег со счета — вопрос спорный (и относится он к принципу индивидуализации интерфейса).

7.3.5. Обратная связь

Принцип *обратной связи* дополняет первый принцип — контроль должен находиться на стороне пользователя. Контролировать ситуацию — значит знать, что происходит, когда контроль временно передается программе. Разработчик должен встроить в систему визуальные или/аудиоподсказки для каждого события, инициируемого пользователем.

В большинстве случаев указатель в форме песочных часов или индикатор ожидания представляет достаточный уровень обратной связи, чтобы понять, что программа что-то делает. Для тех компонент приложения, которые иногда могут стать источником проблем с производительностью, может потребоваться более ясная обратная связь (например, в виде отображения поясняющего сообщения). В любом случае, разработчик никогда не должен предполагать, что приложение выполняется настолько быстро, что обратная связь не потребуется. Любые отклонения в рабочей нагрузке приложения докажут, что разработчик, к сожалению, ошибался.

7.3.6. Эстетичность и удобство

Эстетичность интерфейса влияет на зрительное восприятие системы. *Удобство* касается легкости, простоты, эффективности, надежности и продуктивности в использовании интерфейса. Безусловно, оба принципа касаются удовлетворенности пользователя. Именно в этом вопросе разработчик GUI-интерфейса нуждается в помощи художника-графика и эксперта по социальной психологии.

Существует много разнообразных "золотых правил", касающихся создания эстетичного и удобного интерфейса [27], [16]. При этом следует учитывать такие вопросы, как фиксация и движение человеческого глаза, использование цветов, чувство уравновешенности и симметрии, выравнивание и отступ между элементами, чувство пропорции, группирование связанных элементов и т.д.

Принцип эстетичности и удобства превращает разработчика GUI-интерфейса в художника. В этом смысле неплохо помнить о том, что "простота — эталон красоты". В

действительности *простоту* часто рассматривают как еще один принцип создания GUI-интерфейса, прочно связанного с принципами эстетичности и удобства. В сложных приложениях простота лучше всего достигается с помощью подхода "разделяй и властвуй" за счет последовательного раскрытия информации таким образом, что она отображается только тогда, когда в ней возникает необходимость, возможно, в различных окнах.

7.4. Оконный интерфейс

Проектирование GUI-интерфейса характеризуется двумя основными аспектами — проектированием окон и проектированием элементов ввода и редактирования информации в окна. Оба аспекта зависят от базовой среды GUI. Последующее рассмотрение концентрируется на среде Microsoft Windows [92].

Типичное Windows-приложение состоит из единственного *главного окна* приложения (*primary window*). Главное окно поддерживается набором *всплывающих окон* (*pop-up window*), *вторичных окон* (*secondary window*). Вторичные окна поддерживают действия пользователя с главным окном. Многие действия, поддерживаемые вторичными окнами, представляют собой набор основных операций над базой данных — так называемый набором CRUD-операций. (CRUD — популярная аббревиатура, которая обозначает четыре основных операции над данными: Create, Read, Update, Delete ("Создать", "Читать", "Обновить" "Удалить").

7.4.1. Главное окно

Главное окно имеет границу (рамку). Рамка содержит строку заголовка для окна, строку меню, панели инструментов, строку состояния, а также отображаемое и модифицируемое содержимое окна. Горизонтальные и вертикальные полосы прокрутки используются для прокрутки, при необходимости, содержимого окна.

Отображаемое и модифицируемое содержимое может быть организовано в виде подокон (панелей). Панели позволяют осуществлять просмотр и манипулирование различными, но связанными между собой информационными элементами содержания. На рис. 7.4 показано главное окно, которое отображается после успешного входа в систему и открытия приложения. Панель в левой части окна содержит схему приложения в стиле программы the Windows Explorer (кнопка закрытия в правом верхнем углу панели говорит пользователю о том, что при желании панель можно убрать из окна). Комментарии соответствуют хорошо известной терминологии Windows.

Типичными отличительными чертами главного окна являются наличие строки меню и панели инструментов. Панель инструментов содержит командные кнопки для наиболее часто используемых пунктов меню. Пиктограммы панели инструментов дублируют эти пункты меню. Они предназначены для быстрого доступа к наиболее часто используемым командам.

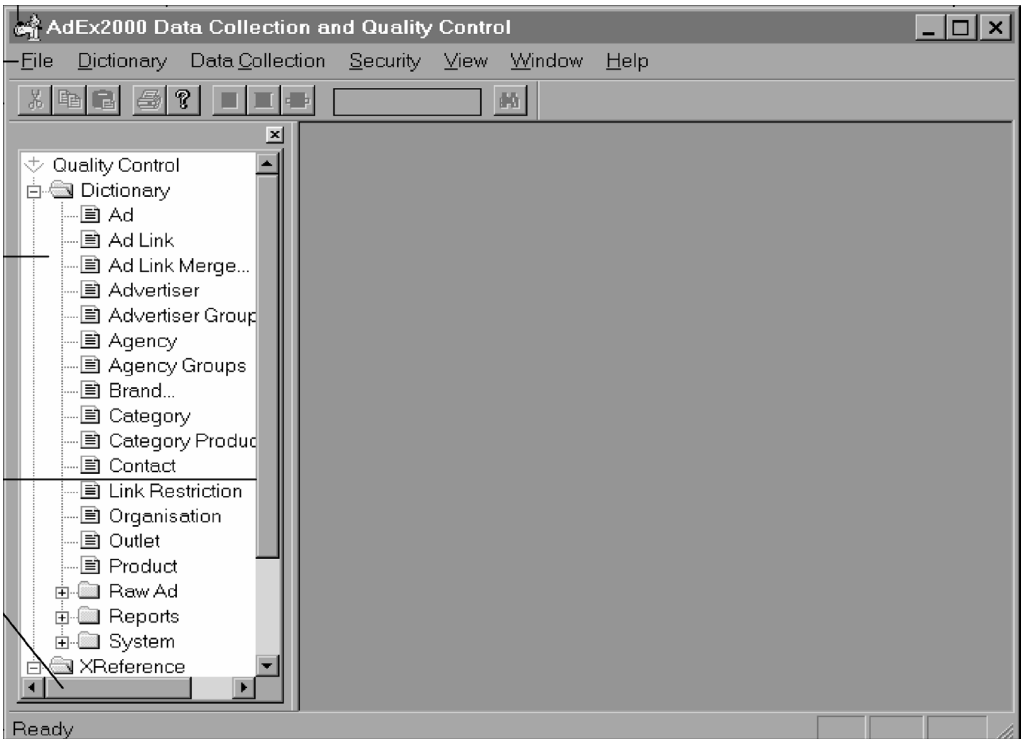

Рис. 7.4. Главное окно приложения

Пример 7.2. Управление контактами с клиентами

Обратитесь к постановке задачи для приложения *Управление контактами с клиентами* (разд. 2.3.3), а также к решенным примерам в главе 4. Одно из требований пользователя заключается в том, чтобы приложение *Управление контактами с клиентами* было построено на функциональной модели диалогового окна Calendar программы Microsoft Outlook (рис. 7.5).

Главное окно должно отображать виды деятельности, запланированные на день для работника, использующего систему. Элемент управления Calendar может отображать деятельность, относящуюся как к прошлому, так и к будущему. Мероприятия, запланированные на определенное время дня (*спланированные по времени*), должны отображаться в левой панели окна Microsoft Outlook. Однако, пользователи выдвигают также требование об отображении и обработке *не спланированных по времени*, *просроченных* (“ожидаемых” в прошлом) и *завершенных* мероприятий.

Наша задача в этом примере заключается в проектировании главного окна для приложения *Управление контактами с клиентами*, удовлетворяющем приведенным выше требованиям.

Рис. 7.5. Окно Calendar программы Microsoft Outlook

На рис. 7.6 показано главное окно приложения *Управление контактами с клиентами*. В целях экономии места элемент управления Calendar спроектирован как "съемное" плавающее окно. При желании его можно закрыть. Для каждого мероприятия в левой панели отображается краткое описание мероприятия и либо название организации, либо имя контактного лица. Кроме того, для некоторых мероприятий может отображаться дополнительная информация, например, телефонный номер организации или контактного лица, номер факса или адрес. Используемая в этой книге черно-белая печать не дает возможности в полной мере оценить использование цветов в левой панели главного окна. В частности, цвета используются для обозначения приоритетов, назначенных мероприятиям (высокий — красный, обычный — черный и низкий — синий).

Правая панель служит трем целям. Она отображает три типа мероприятий. *Завершенные* мероприятия переносятся из левой панели и размещаются в верхней части правой панели. Текст описания этих мероприятий отображается в синем цвете и зачеркивается. Основная причина, по которой завершенные мероприятия вообще не удаляются с экрана, состоит в том, что иногда требуется придать завершенным мероприятиям статус "незавершенных" (положим, что мы преждевременно решили, что работа выполнена, однако позже выяснили, что она еще не вполне закончена).

Просроченные мероприятия перечислены в правой панели. Они отображаются красным шрифтом. Наконец, мероприятия, которые *не спланированы по времени*, отображаются черным цветом и перечисляются в нижней части правой панели. Левая колонка панели спроектирована в расчете на людей, которые не различают цветов. Соответствующие пиктограммы обозначают три типа мероприятий, которые могут присутствовать в правой панели.

Рис. 7.6. Главное окно приложения "Управление контактами с клиентами"

7.4.1.1. Окно просмотра строк

Часто главное окно приложения ИС используется в качестве *окна просмотра строк* для отображения записей базы данных, содержащих, к примеру, данные о сотрудниках. Подобное окно иногда называют *броузером строк* (*row browser*) по аналогии с Web-броузером. Пользователь имеет возможность "прокручивать" строки вверх и вниз с помощью вертикальной полосы прокрутки или клавиш клавиатуры (<Page Up>, <Page Down>, <Home>, <End>, а также стрелок "вверх" и "вниз").

На рис. 7.7 показан пример окна просмотра строк. Документ, помеченный как `Ad Link`, внутри главного окна является *дочерним окном* (*child window*) (это понятие объясняется позже). Дочернее окно обладает собственным набором *кнопок управления окном* (*window button*) (для минимизации, восстановления, закрытия), расположенных в правом углу строки меню. Ширину колонок сетки броузера *можно изменять*, также как и *порядок их расположения*. Округлые вмятинки рядом с именами колонок указывают на то, что колонки можно *сортировать* — щелчок мышью на колонке вызывает сортировку записей в порядке возрастания или убывания значений в этой колонке.

В любой данный момент времени в окне просмотра строк активна только одна строка (запись). Двойной щелчок мышью на этой строке обычно приводит к отображению "*окна редактирования*" (*edit window*), которое содержит детали, относящиеся к этой записи. Окно редактирования позволяет модифицировать содержимое записи.

Панели могут использоваться для разделения окна по вертикали или по горизонтали, либо в обоих направлениях. Рис. 7.8 иллюстрирует разделение окна по горизонтали. Как ясно из заголовка окна, три панели используются для отображения товаров по рекламодателям и рекламным агентствам. Средняя панель содержит перечень рекламодателей, связанных с текущим выбранным (выделенным) рекламным агентством, показанным в верхней панели. Нижняя панель, в свою очередь, отображает товары, рекламируемые выбранным рекламодателем.

Рис. 7.7. Окно просмотра строк

Рис. 7.8. Многопанельное окно просмотра строк

Рис. 7.9. Окно с панелями просмотра дерева и строк

7.4.1.2. Окно просмотра деревьев

Еще один популярный способ использования главного окна — *окно просмотра деревьев*, которое иногда называют *броузером деревьев* (*tree browser*). Программа просмотра деревьев отображает связанные записи в виде схемы с отступами. Схема содержит элементы управления, которые позволяют сворачивать и разворачивать дерево. Хорошо известный пример броузера деревьев — отображение каталогов файлов компьютера с помощью программы Windows Explorer.

В отличие от броузера строк броузер деревьев должен допускать модификацию на месте, т.е. должен допускать модификацию содержимого окна без активизации окна редактирования. Модификации в окне просмотра деревьев осуществляются с помощью операции "перетащить и поместить" ("drag and drop").

На рис. 7.9 показано окно просмотра деревьев, распложенное в левой панели главного окна. Правая панель представляет собой окно просмотра строк. Выбор записи для группы агентств приводит к отображению агентств, входящих в эту группу в окне просмотра строк.

7.4.1.3. Web-страница

Web-страницу также можно трактовать как особый вид главного окна, если она используется в качестве точки входа в Web-приложение. В отличие от традиционных приложений ИС строка меню и панель инструментов Web-страницы не используются для задач приложения. Они используются для навигации по Web — это общий вид деятельности для всех Web-броузеров. Для Web-приложений события, инициируемые пользователями, обычно программируются с помощью командных кнопок и активных гиперссылок.

Рис. 7.10. Окно Web-страницы

На рис. 7.10 показана Web-страница, представляющая собой точку входа для Web-узла Макуарского университета. Строка меню и панель инструментов не применимы к этой Web-странице. Гиперссылки используются для поиска в библиотечной базе данных, а также для того, чтобы "брать" книги или другие библиотечные материалы.

7.4.2. Вторичное окно

За исключением некоторых простейших приложений ИС *вторичное окно* дополняет свое главное окно. Оно расширяет функциональные возможности главного окна, в частности, в отношении операций, которые модифицируют базу данных (т.е. за счет операций вставки, удаления или обновления).

Вторичное окно обычно является *модальным* по отношению к его главному окну. Прежде, чем приступить к работе с любым другим окном приложения, пользователь должен ответить и закрыть вторичное окно. *Немодальные* вторичные окна допустимы, однако, их использование не рекомендуется.

Окно входа в систему — простой пример вторичного окна. Пример экрана входа в систему, показанный на рис. 7.11, иллюстрирует основные видимые отличия между главным и вторичным окном. У вторичного окна отсутствуют какие-либо "строки": строка меню, панель инструментов, полосы прокрутки или строка состояния. События инициируются пользователем посредством командных кнопок (кнопок запуска действий), таких как кнопки OK, Cancel (Отмена), Help (Помощь).

Вторичные окна выступают в иде различных экранных форм и масок. Ниже перечислены основные типы вторичных окон.

- Диалоговое окно.

- Папка с вкладками.
- Выпадающий список
- Окно сообщений.

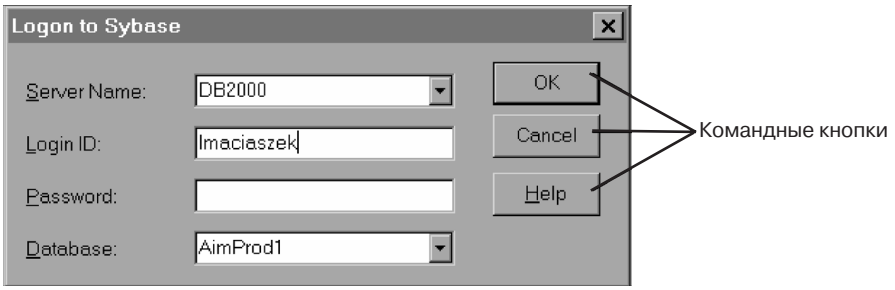

Рис. 7.11. Простое вторичное окно – окно входа в систему

7.4.2.1. Диалоговое окно

Диалоговое окно (*dialog box*) является почти синонимом понятия вторичного окна, заключает в себе наиболее часто требуемые свойства вторичных окон. Оно поддерживает диалог между пользователем и приложением. Диалог предполагает ввод пользователем некоторой информации, которая обрабатывается приложением.

На рис. 7.12 приведен пример диалогового окна. Это окно обновления. Оно отображает рекламную продукцию, соответствующую текущему товару, выбранному в подокне просмотра строк главного окна. Пользователь может модифицировать любые значения редактируемых полей, ограниченных рамками с белым заполнением внутри.

Пример 7.3. Управление контактами с клиентами

Обратитесь к постановке задачи для приложения *Управление контактами с клиентами* (разд. 2.3.3), а также к решенным примерам в главе 4. Обратитесь также к примерам 7.1 и 7.2 в этой главе.

Главное окно приложения (рис. 7.4) не позволяет выполнять определенные манипуляции с мероприятиями. Например, ввод нового мероприятия или обновление существующего мероприятия должны осуществляться через вторичное, в данном случае диалоговое, окно.

После двойного щелчка мышью на строке мероприятия должно появиться диалоговое окно, отображающее полный список деталей, относящихся к этому мероприятию. Диалоговое окно отображает не только информацию по мероприятию, но также данные о его сопутствующих задачах, равно как и информацию об организации и контактном лице, к которому относится мероприятие.

Затем относящиеся к мероприятию детали могут быть отображены и, возможно, модифицированы, включая тип мероприятия (поле Action (Действие) диалогового окна Details (Подробности), более длинное описание (поле Notes этого же окна), дату и время создания мероприятия, работника, создавшего мероприятие, а также работника, которому запланировано выполнение мероприятие и срок его выполнения (группа полей Due), и работника и время фактического завершения мероприятия (группа полей Completed).

Наша задача заключается в том, чтобы спроектировать диалоговое окно для манипулирования мероприятиями, удовлетворяющее приведенным выше требованиям.

Рис. 7.12. Диалоговое окно

На рис. 7.13 представлен вариант решения нашего примера. Обратите внимание, что поля Organization (Организация), Contact (Контактное лицо) не являются редактируемыми, поскольку "адресат" мероприятия не подлежит изменению. Аналогично значения группы полей, следующих за приглашением Created (Создано), также нельзя редактировать.

Значения полей, примыкающих к приглашению Completed (Завершено), не редактируемы в том смысле, что пользователь не может ввести их. Однако, нажатие кнопки Complete (Завершить) автоматически вставляет в эти поля значения даты, времени и имени пользователя. После завершения мероприятия пользователь по-прежнему имеет возможность вернуться к нему как к "несостоявшемуся", поскольку в таком случае кнопка Complete превращается в кнопку Uncomplete (Не завершать).

Пользователь имеет возможность сохранить изменения в базе данных и вернуться в главное окно, щелкнув на кнопке OK. Вместо этого пользователь может отменить (кнопка Cancel) внесение изменений и остаться в диалоговом окне. Наконец, пользователь может нажать кнопку New Event (Новое мероприятие), что приведет к сохранению изменений (после подтверждения их пользователем), очистке всех полей и даст пользователю возможность создать качественно новое мероприятие (не возвращаясь в главное окно).

7.4.2.2. Папка со вкладками

В тех случаях, когда объем информации, которую необходимо отобразить во вторичном окне, превышает "полезную площадь", а предмет рассмотрения можно логически разделить на несколько информационных групп, бывает полезным использование *папки со вкладками* (*tab folder*). В каждый данный момент времени видим только один из листов вкладок, расположенный на вершине этой "стопки" листов. (В системе Microsoft Windows папка со вкладками называется *листом свойств, снабженным вкладками* (*tabbed property sheet*), а каждая из вкладок называется *страницей свойств* (*property page*)).

Рис. 7.13. Диалоговое окно (Управление контактами с клиентами)

Пример 7.4. Управление контактами с клиентами

Обратитесь к постановке задачи для приложения *Управление контактами с клиентами* (разд. 2.3.3), а также к решенным примерам в главе 4. Обратитесь также к примерам 7.1, 7.2 и 7.3 в этой главе.

Рассмотрите папку со вкладками для окна Maintain Organizations, приведенную на рис. 7.2 (пример 7.1). Одна из вкладок называется Contacts (Контактные лица). Она предназначена для осуществления доступа и модификации данных поля Contact (класс `Contact`) с помощью этой папки со вкладками. При ее отсутствии пользователь вынужден был бы всегда возвращаться в главное окно и активизировать отдельное вторичное окно для ведения учетной информации по контактам.

Цель этого примера заключается в проектировании содержимого вкладки Contacts.

На рис. 7.14 показана папка со вкладками для ввода информации о новых организациях, занимающихся рекламой. Четыре вкладки разделяют большой объем информации, вводимой пользователем, на четыре группы. Командные кнопки в нижней части экрана применимы ко всему окну, а не только к текущей видимой странице вкладки.

Рис. 7.14. Папка со вкладками

Как показано на рис. 7.15, вкладка Contacts отображает только имена контактных лиц в организации. Однако, вкладка обладает собственным набором командных кнопок (Add, Edit и Delete) для добавления, редактирования и удаления выделенных в текущий момент элементов из базы данных по учету информации о контактах. Действия по добавлению (Add) или редактированию (Edit) контактной информации приводят к открытию вторичного окна Maintain Contacts поверх окна Maintain Organizations. Окно Maintain Contacts является модальным по отношению к окну Maintain Organizations.

7.4.2.3. Выпадающий список

В некоторых случаях удобной заменой странице вкладки может служить *выпадающий список* (*drop-down list*) или набор выпадающих списков. Выпадающий список обеспечивает список выбора элементов, из которого пользователь может выделить один подходящий элемент. Для выполнения операции вставки нового элемента в список пользователь может ввести новое значение, которое добавляется в список и отображается при открытии списка в следующий раз.

Как видно из рис. 7.16, выпадающий список не обязательно должен быть простым списком, он может представлять собой окно просмотра деревьев.

Рис. 7.15. Папка с вкладками (Управление контактами с клиентами)

Рис. 7.16. Выпадающий список

Рис. 7.17. Окно сообщения

На рис. 7.17 показано сообщение, которое требует подтверждения (кнопка OK) от пользователя.

7.5. Зависимость между окнами

С точки зрения пользователя приложение выглядит как набор взаимодействующих окон. Задача разработчика GUI-интерфейса состоит в том, чтобы организовать зависимости между окнами в виде последовательной легко понятной структуры. Пользователь никогда не должен чувствовать себя заблудившимся среди открытых окон.

В идеале между главным окном и верхним текущим открытым вторичным окном должна устанавливаться простая, даже не иерархическая связь. Это достигается за счет придания вторичному окну статуса *модального* по отношению к предыдущему окну. Многодокументный интерфейс (multiple document interface — MDI), рассматриваемый в разд. 7.5.3, должен позволять обрабатывать любую сложную ситуацию, не прибегая к необходимости использования немодальных окон.

В то время, как проектирование GUI-интерфейса должно способствовать исследованию интерфейса пользователем, хороший проект *строки меню* остается принципиальным методом объяснения возможностей приложения. Команды меню, доступные пользователю при использовании выпадающих или выдвигающихся меню, служат в качестве косвенного объяснения зависимостей между окнами.

7.5.1. Документ и его представление

Проектирование GUI-интерфейса в среде Microsoft Windows непосредственно зависит от классов, поставляемых компанией Microsoft, которые реализуют объекты и элементы управления Windows — API-интерфейса Windows (application programming interface — интерфейс прикладного программирования). Соответствующая библиотека классов называется библиотекой MFC (Microsoft Foundation Classes — базовые классы Microsoft).

Программирование для Windows связано с порождением и использованием объектов MFC, а также с созданием специфичных для приложения классов, наследующих исходные функциональные возможности классов MFC. Программирование для Windows требует также принятия на вооружение специфической структуры, определяющей зависимости и взаимодействие между окнами. Эта структура известна как подход к программированию на основе механизма *документ/представление* (*document/view*) [38].

Документ — это механизм MFC, позволяющий собрать данные в приложении таким образом, что пользователь может взаимодействовать с ними. Документ может содержать помимо текстовых любые другие типы данных. Объект-документ является производным объектом класса `CDocument` библиотеки MFC.

Обычно на экране отображается только фрагмент данных, хранимых в объекте `CDocument`. Этот фрагмент называется *представлением* (*view*). Объект представления является производным объектом класса `CView`. Для одного документа может существовать множество представлений. С технической точки зрения объект `CView` и окно (фрейм), в котором он отображается, — это не одно и то же.

На рис. 7.18 показано, в чем состоит видимое различие между документом и представлением. В данном случае документ представляет собой документ, созданный с помощью текстового процессора, но в общем случае он может быть любым файлом, содержащим информацию, которая извлечена из базы данных.

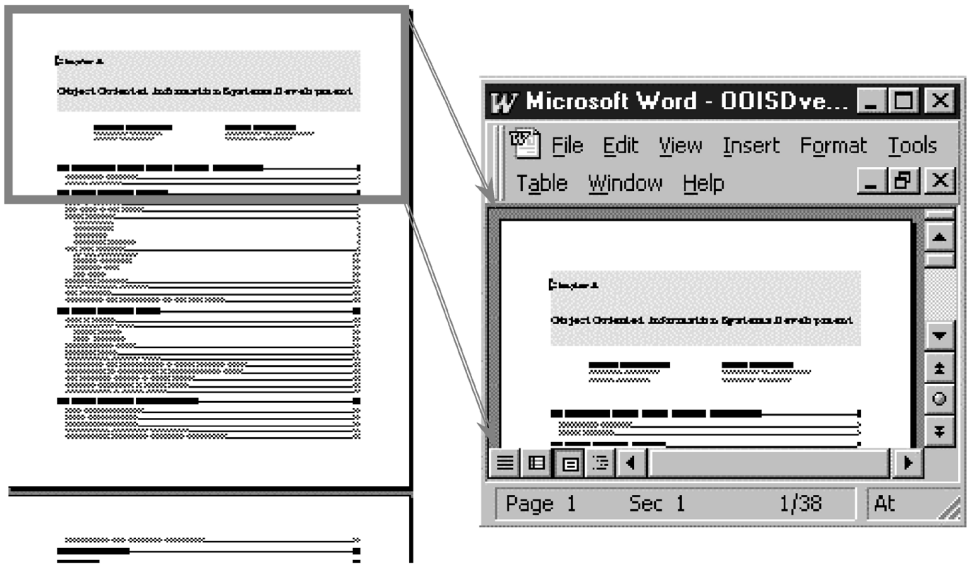

Рис. 7.18. Документ и его представление

7.5.2. Однодокументный интерфейс

В случае некоторых простых приложений проект GUI-интерфейса может состоять из единственного главного окна, в котором одновременно отображается только один открытый документ. Библиотека MFC поддерживает эту возможность, которая получила распространение под названием *SDI-интерфейса* (*single document interface — однодокументный интерфейс*).

В проекте GUI-интерфейса для приложения "Управление контактами с клиентами" используется SDI-приложение. В любой момент времени пользователю представлены события, относящиеся к одному дню (рис. 7.19).

7.5.3. Многодокументный интерфейс

Для более сложных приложений безусловно может потребоваться открывать одновременно несколько документов. Документы могут быть однотипными, однако, зачастую они принадлежат разным типам. Библиотека MFC поддерживает возможность, которая получила распространение под названием *MDI-интерфейса* (*multiple document interface — многодокументный интерфейс*).

Рис. 7.19. Приложение с однодокументным интерфейсом (SDI) (с любезного разрешения компании ACNielsen AdEx, Сидней, Австралия)

MDI-приложение все так же использует *только одно главное окно*. Это окно называется *родительским окном* (*parent window*). Однако, MDI-приложение позволяет открывать несколько документов внутри фрейма родительского окна. Каждый документ называется *дочерним окном* (*child window*). Логически каждое дочернее окно ведет себя так, как если бы оно было главным окном, которое может появиться только внутри родительского окна (но не на рабочем столе компьютера).

Тот факт, что архитектура MDI обладает только одним главным окном, по существу, выражается в том, что все дочерние окна совместно используют *одну строку меню*. Аналогично дочерние окна, как правило, совместно используют панель инструментов и строку состояния. Однако, существует возможность внести модификации в доступные пользователю действия, содержащиеся в меню и панели инструментов, чтобы отразить функциональные возможности текущего активного дочернего окна.

На рис. 7.20 показано MDI-приложение. Внутри фрейма главного окна открыто четыре документа (броузера строк). Пятый документ, расположенный в левой части главного окна, представляет собой броузер деревьев.

7.6. Навигация по окнам

Графическая картина GUI-окон, построенная с использованием прототипов или других средств визуализации, ничего не говорит о том, каким образом пользователь может фактически *перемещаться* по окнам. Нам еще требуется спроектировать *систему навигации по окнам*. Диаграмма навигации по окнам призвана визуализировать окна приложения и управляющие объекты, которые позволяют пользователю перемещаться от одного окна к другому.

К сожалению, UML не обладает методами графического моделирования для моделирования навигации по окнам. Необходимо разработать собственную модель или воспользоваться преимуществами стереотипов UML и настроить одну из диаграмм UML для представления навигации по окнам.

Рис. 7.20. Приложение с многодокументным интерфейсом (MDI)

7.6.1. Введение навигационных стереотипов в диаграмму видов деятельности

Оказывается, диаграмма видов деятельности является неплохим потенциальным стереотипом для оконной навигации. Диаграмма видов деятельности показывает переходы между видами деятельности (разд. 2.2.3.2). Однако, диаграмма видов деятельности, будучи разновидностью конечного автомата, может также обрисовать состояния объектов. Эту двойственность диаграмм видов деятельности можно использовать в изобразительных целях — для интерпретации аналогичной двойственности, присущей GUI-объектам. GUI-окна похожи на *состояния* в ожидании *событий* (*видов деятельности*).

Состояния и виды деятельности диаграммы видов деятельности можно превратить в стереотипы. *Стереотипы состояния* могут обозначать различные типы окон и другие GUI-объекты, которые "продолжают существовать" между событиями, т.е. которые обладают длительностью. На рис. 7.21 показано состояние, играющее роль стереотипа главного окна. Состояние представляет *окно просмотра* (*сетку*) для товаров, отображаемое в главном окне приложения. Это также и начальное *состояние модели*.

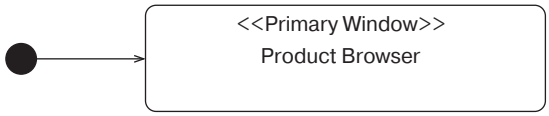

Рис. 7.21. Состояние, обозначенное как стереотип GUI-объекта

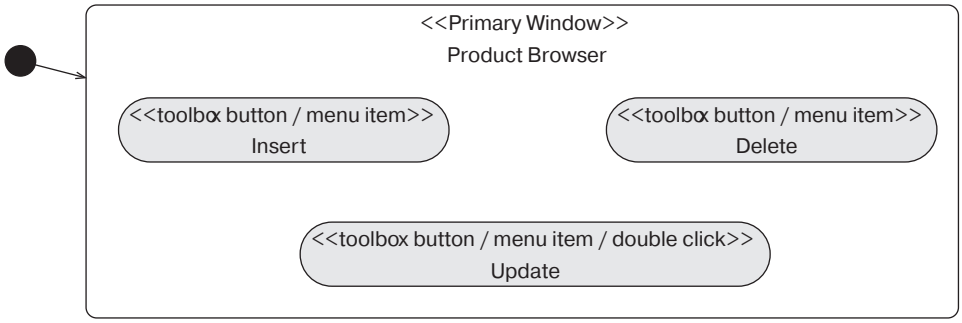

Рис. 7.22. Вид деятельности, обозначенный как стереотип управляющего GUI-объекта

Стереотипы видов деятельности могут обозначать различные типы управляющих элементов GUI-интерфейса, которые можно использовать для запуска событий на состояниях (теперь обозначенных как объекты GUI-окон). В противоположность состоянию деятельность обладает очень короткой длительностью — можно сказать, что на нашей относительной временной шкале они обладают нулевой длительностью.

Виды деятельности (овалы) можно представить внутри состояния (прямоугольник со скругленными краями), к которому они применяются. На рис. 7.22 показано три вида деятельности в состоянии `Product Browser`. Окно `Product Browser` можно использовать для инициирования события вставки, удаления или обновления товара. Все три события можно запустить с помощью командной кнопки панели инструментов или пункта меню. Двойной щелчок мышью на строке товара в окне просмотра также поддерживает событие обновления.

Полный список стереотипов состояний и видов деятельности для поддержки проектирования системы навигации по окнам, как правило, зависит от выбранного GUI-интерфейса. Разработчики могут также подумать над использованием стереотипов в виде пиктограмм. Ниже приведен неполный перечень стереотипов для GUI-интерфейса Microsoft Windows.

- *Состояния* (*окна*)
  - Главное окно
  - Панель в главном окне
  - Окно просмотра строк
  - Окно просмотра деревьев
  - Web-страница
  - Вторичное окно
  - Диалоговое окно

- Окно сообщений
- Папка со вкладками
  - Типы данных окон
- Текстовое поле
- Комбинированное окно
- Спиновое окно
- Колонка
- Строка
- Группа полей

- *Виды деятельности* (*элемент управления окном*)
  - Пункт выпадающего меню
  - Пункт всплывающего меню
  - Кнопка панели инструментов
  - Командная кнопка
  - Двойной щелчок мышью
  - Выбор из списка
  - Клавиша клавиатуры
  - Функциональная клавиша клавиатуры
  - Комбинация клавиш клавиатуры
  - Бегунок полосы прокрутки
  - Кнопка закрытия окна

7.6.2. Диаграмма навигации по окнам

После определения стереотипов состояний и видов деятельности можно использовать линии *перехода* для соединения видов деятельности и состояний. В результате получится диаграмма навигации по окнам, в которой виды деятельности запускают переходы на состояниях.

Рис. 7.23 расширяет предыдущий пример и показывает состояния, которые являются результатом запуска трех видов деятельности, присутствующих в окне `Product Browser`. Состояние, запускаемое событием `Update`, расширено. Окно `Update Product` (`<<dialog box>>`) содержит четыре деятельности (`<<command buttons>>`). Нажатие на кнопке `OK` или `Cancel` вызывает обратный переход в окно `Product Browser`. Нажатие на кнопке `Save` или `Clear` не изменяет активное окно (если нам требуется зафиксировать изменение состояния внутри активного окна, необходимо расширить нашу модель за счет дополнительных стереотипов).

Модель на рис. 7.23 ограничена объектами GUI. В главе 9 будет показано, каким образом можно дальше развить стереотипы диаграммы видов деятельности, чтобы визуализировать всю логику приложения, включая доступ к базе данных.

Пример 7.5. Управление контактами с клиентами

Обратитесь к постановке задачи для приложения *Управление контактами с клиентами* (разд. 2.3.3), а также к решенным примерам в главе 4. Обратитесь к примерам 7.1, 7.2 и 7.3 в этой главе.

Рассмотрите главное окно на рис. 7.6 (пример 7.2) и диалоговое окно на рис. 7.13 (пример 7.3). Разработайте диаграмму навигации по окнам в этих двух окнах. Диаграмма должна обозначать и представлять модель основных событий, инициируемых пользователем, для главного и диалогового окон. Следует также обратиться к использованию окна Calendar — "плавающего" комбинированного окна.

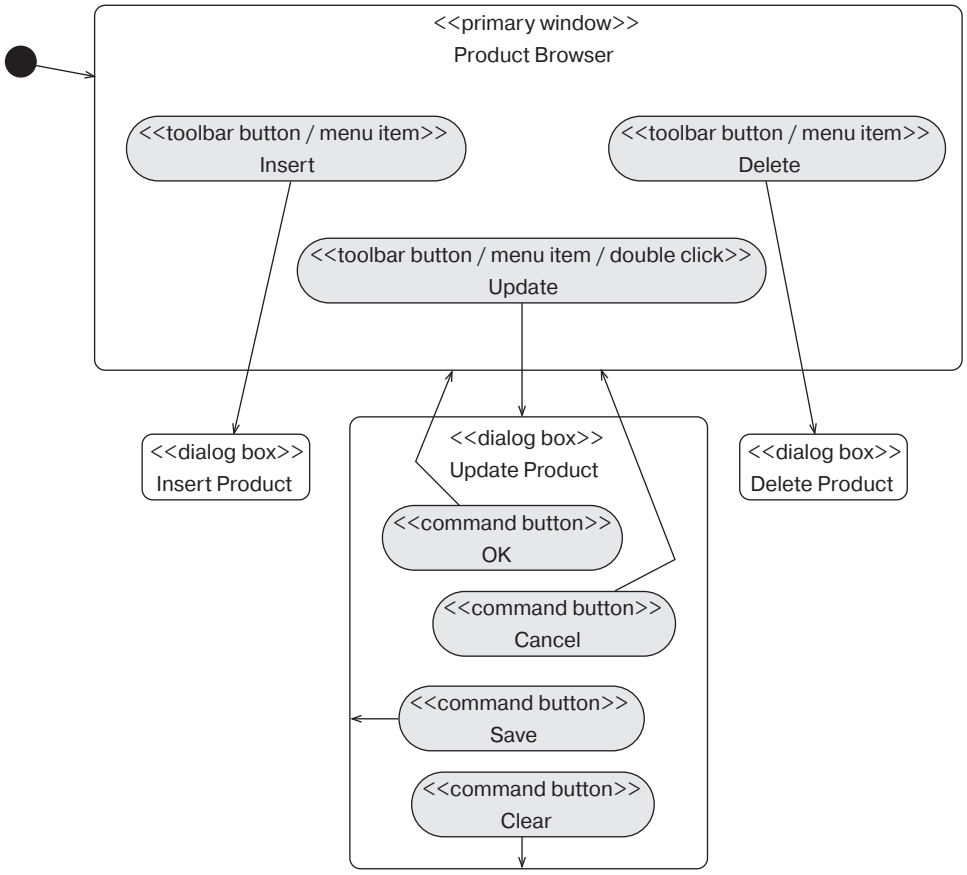

Рис. 7.23. Диаграмма навигации по окнам

Диаграмма навигации по окнам для нашего примера должна рассматривать в качестве состояний три окна `Contact Management` (`<<primary window>>`), `Task/Event Details` (`<<dialog box>>`) и `Calendar` (`<<combo box>>`). Левую и правую панели главного окна также можно рассматривать как возможные состояния (подокна) главного окна-состояния. Однако, в предлагаемом решении мы пропускаем панели.

Как следует из панели инструментов, показанной на рис. 7.6, существует большое количество видов деятельности (мероприятий), которые можно запустить на главном окне. Однако, наша задача заключается в определении только основных инициируемых пользователями событий, включая те, которые активизируют вторичные окна Task/Event Details и Calendar.

На рис. 7.24 показана диаграмма навигации по окнам для нашего примера. Событие Calendar (<<toolbar button>>) отображает деятельность Calendar (<<combo box>>). Комбинированное окно может "плавать" по экрану. Щелкнув на квадратике закрытия окна можно избавиться от него. Выбор месяца (<<scroll>>) или выбор даты (<<select>>) не удаляют комбинированное окно.

Доступ к окну Task/Event Details (<<dialog box>>) получают посредством события Update Event (<<toolbar button/menu item/double click>>). Командные кнопки OK и Cancel удаляют диалоговое окно с экрана и возвращают управление главному окну. Активизация кнопки Complete приводит к заполнению полей завершения мероприятия (Complete) в окне, но управление остается у диалогового окна. Это является следствием того, что пользователь может пожелать создать новое мероприятие. Нажатие кнопки New Event очищает поля экрана и позволяет пользователю ввести детальную информацию, относящуюся к новому мероприятию.

Пример 7.6. Телемаркетинг

Обратитесь к постановке задачи для приложения *Телемаркетинг* (разд. 2.3.4) и к решенным примерам для приложения *Телемаркетинг* в главе 4. Рассмотрите приведенные ниже уточняющие детали и спроектируйте диаграмму навигации по окнам.

Интерфейс главного окна для приложения *Телемаркетинг* представляет собой пустое окно со строкой названия приложения и двумя командными кнопками. Кнопки позволяют телемаркетеру осуществить следующий звонок (Next Call) благотворителю или выйти из приложения (Quit). Нажатие кнопки Next Call вызывает отображение сообщения с приветствием благотворителю и информационных полей окна с данными о текущем звонке.

В ходе разговора с благотворителем телемаркетер взаимодействует с системой, нажимая различные командные кнопки. На панели инструментов имеется ряд кнопок для записи результатов звонка (таких как заказ на билеты, перепланирование звонка или неудача). Имеется также ряд командных кнопок для просмотра более подробной информации, относящейся к кампании или благотворителю.

На рис. 7.25 представлена диаграмма навигации по окнам для приведенного выше примера. Хотя Entry Window и Call Window — это одно и то же главное окно (<<primary window>>), мы в явном виде представляем два состояния окна. Состояние Entry Window становится активным после начального запуска программы, а также в том случае, когда результат звонка неудачен (Unsuccessful). Возможна ситуация, при которой мероприятия во вторичном окне могут возвращать управление окну Entry Window, но это не отражено в модели. Нажатие кнопки выхода Quit в любом состоянии приводит к закрытию приложения.

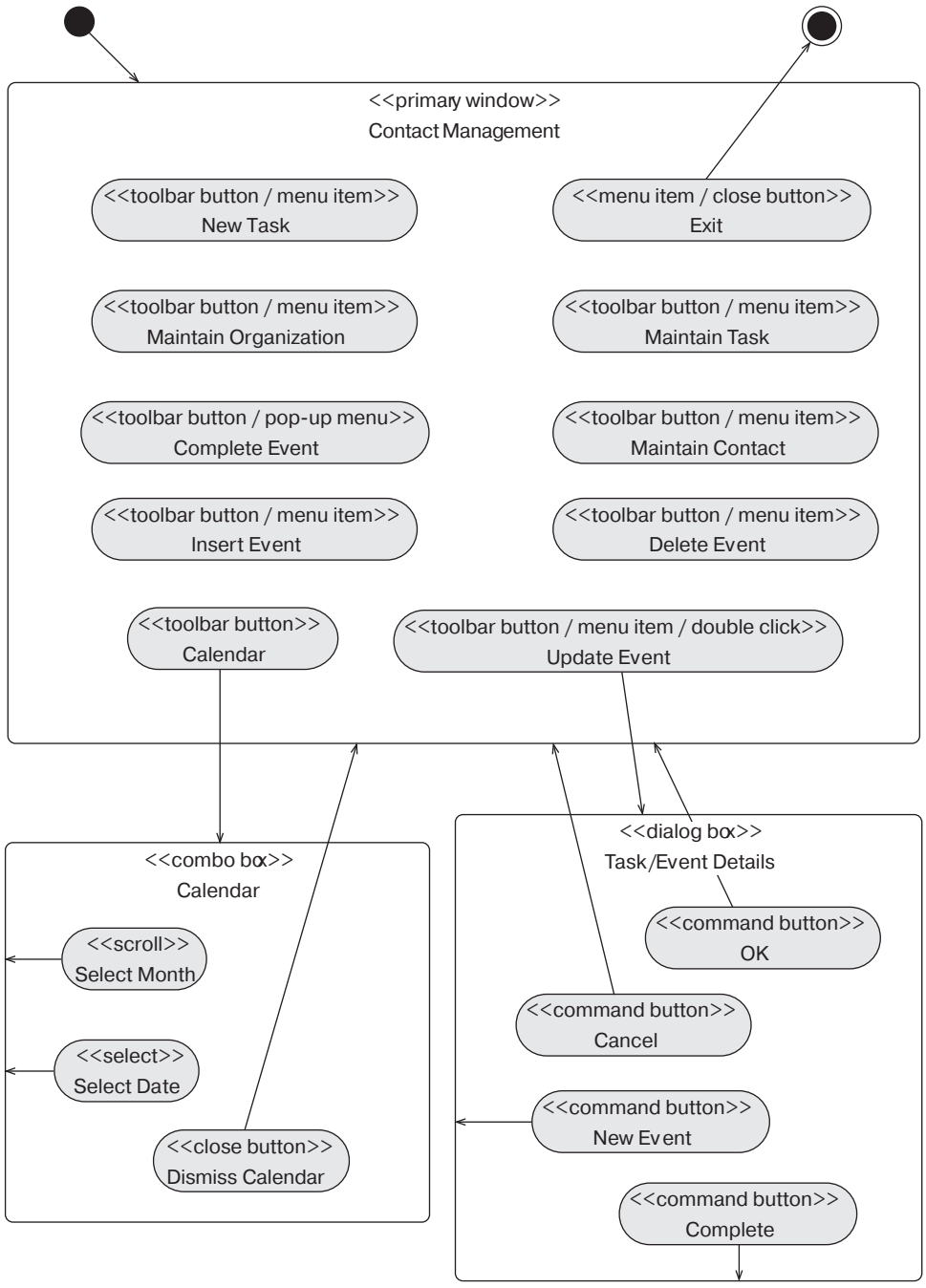

Рис. 7.24. Диаграмма оконной навигации для приложения "Управление контактами с клиентами"

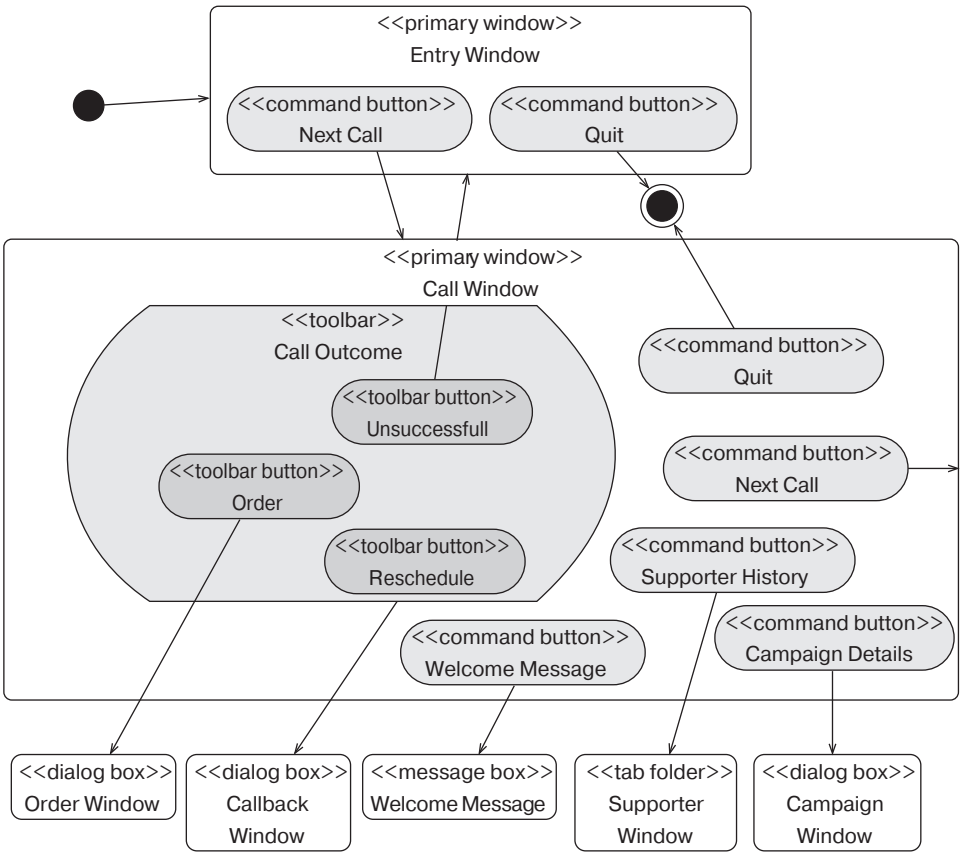

Рис. 7.25. Диаграмма оконной навигации для приложения "Телемаркетинг"

Резюме

Разработка GUI-интерфейса охватывает весь жизненный цикл производства программного продукта — она начинается на этапе анализа и продолжается до реализации. Данная глава обращена к проектированию GUI-интерфейса, в особенности для среды Microsoft Windows. В ней также вводится графическая нотация для представления навигации по окнам.

Проектирование GUI-интерфейса представляет собой *междисциплинарную деятельность*, требующую объединения усилий различных специалистов. Проектирование должно вестись в соответствии с *руководящими принципами*, декларируемыми производителями оконного интерфейса, на использование которого ориентирован проект. Эти принципы направлены на решение таких вопросов, как принцип локализации управления на стороне пользователя, соответствие, индивидуализация, настройка, терпимость к ошибкам, обратная связь, эстетичность и удобство использования.

В интерфейсе системы Microsoft Windows различают главное окно и вторичное окно. *Главное окно* может быть окном просмотра строк, окном просмотра деревьев или

Web-страницей. *Вторичное окно* может быть диалоговым окном, папкой со вкладками, выпадающим списком или окном сообщений. Вторичное окно может быть по отношению к своему главному окну *модальным* или *немодальным*. Дальнейшее уточнение зависимостей между окнами осуществляется применительно к использованию *SDI-интерфейса* или *MDI-интерфейса*.

Визуальное проектирование отдельных окон представляет собой лишь один из аспектов разработки GUI-интерфейса. Второй основополагающий аспект разработки связан со схемами *перемещения по окнам*, которые призваны зафиксировать возможные пути навигации пользователя между окнами приложения. В данной главе были введены диаграммы навигации по окнам как один из подходов к решению этой проблемы. Диаграммы навигации по окнам расширяют диаграммы видов деятельности языка UML. В действительности, они представляют собой новый *профиль* UML для моделирования навигации по окнам.

Вопросы

В1. Перечислите и дайте краткое определение руководящих принципов разработки GUI-интерфейса.

В2. В чем отличие главного окна от вторичного окна?

В3. Что такое панель? В чем она может быть полезной при проектировании GUI-интерфейса?

В4. Что такое папка со вкладками? В чем ее отличие от диалогового окна?

В5. В чем заключается сущность подхода “документ/представление” к построению графического интерфейса?

В6. Допускает ли MDI-интерфейс использование нескольких главных окон? Объясните свой ответ.

В7. Каким образом окна и элементы управления окнами представляются на диаграммах навигации по окнам? Соответствует ли это представление назначению диаграмм видов деятельности в языке UML? Объясните свой ответ.

Упражнения

Дополнительные требования. Телемаркетинг

Рассмотрите следующие дополнительные требования для приложения *Телемаркетинг*.

1. Окно `Telemarketing Control` представляет собой главное окно интерфейса для приложения *Телемаркетинг*. Это окно выводит на экран компьютера телемаркетера список вызовов в текущей очереди. Когда телемаркетер запрашивает вызов из очереди, система устанавливает соединение, и телемаркетер получает возможность обработать телефонное подключение. При этом на экране отображается информация о звонке (`Call Summary`) — выводится время начала, завершения и продолжительность текущего звонка.
2. После установления соединения окно `Telemarketing Control` отображает информацию о текущем звонке — кому произведен звонок, в рамках какой кампании, какой тип звонка осуществлен. Если для текущего телефонного номера запланировано более одного звонка, телемаркетеру предоставляется возможность осуществить цикл этих звонков.
3. На любом этапе разговора телемаркетер может просмотреть предысторию благотворителя (окно `Supporter History`), относящуюся к предыдущим кампаниям.

Аналогично можно просмотреть подробности, относящиеся к кампании, в рамках которой произведен текущий звонок (окно `Campaign`).

4. GUI-интерфейс обеспечивает возможность быстрой фиксации результатов звонка. К возможным результатам относятся: “размещение” (т.е. билеты были заказаны), “обратный вызов”, “неудача”, “нет ответа”, “занято”, “автоответчик”, “факс”, “неверный номер” и “разрыв связи”.
5. Окно `Campaign` отображает подробности, относящиеся к кампании, билетам и призам, разыгрываемым в рамках кампании. Подробности, относящиеся к кампании, включают: идентификационный номер, название, даты начала, окончания и розыгрыша призов. Подробности, относящиеся к билетам, включают: количество билетов, участвующих в кампании, сколько из них продано и сколько еще имеется в наличии. Подробности, относящиеся к призам, включают: описание приза, цена приза и место среди других призов (первое, второе или третье).
6. Окно `Supporter History` отображает предысторию звонков и предысторию участия благотворителя в прошлых кампаниях. История звонков включает перечень последних звонков, типы звонков, результаты, а также идентификаторы кампаний и телемаркетеров. Предыстория кампаний содержит информацию о размещении билетов и выигрышах призов благотворителем.
7. При выборе действия `Placement` (Размещение) активизируется окно `Placement`, которое позволяет пользователю выделить билеты благотворителю и зафиксировать платеж.
8. При выборе действия `No Answer` или `Engaged` для каждого текущего звонка в системе в качестве результата записываются значения “нет ответа” и “занято” соответственно. Затем звонки перепланируются системой на другое время следующего дня при условии, что каждый звонок не выходит за пределы лимита попыток, установленного для данного типа звонка.
9. При выборе действия `Machine` в системе в качестве результата записывается значение “автоответчик”. Продолжительность звонка устанавливается только для первого из текущих звонков. Затем звонки перепланируются системой на другое время следующего дня при условии, что каждый звонок не выходит за пределы лимита попыток, установленного для данного типа звонка.
10. При выборе действия `Fax` или `Wrong` в системе в качестве результата записываются значение “факс” или “неверный номер”. Продолжительность звонка устанавливается только для первого из текущих звонков. После этого для каждого благотворителя с текущим номером телефона данные, относящиеся к благотворителю, обновляются, в них заносится признак “испорченный телефон”.
11. При выборе действия `Disconnected` в системе в качестве результата записывается значение “разрыв связи”. После этого для каждого благотворителя с текущим номером телефона данные, относящиеся к благотворителю, обновляются, в них заносится признак “испорченный телефон”.
12. При выборе действия `Callback` в системе в качестве результата записывается значение “обратный вызов”. Продолжительность звонка устанавливается только для первого из текущих звонков. Чтобы получить время и дату обратного вызова и поместить их, вызывается окно `Call Scheduling`. Затем звонок перепланируется системой (с присвоением ему нового приоритета) на время и дату, полученные с помощью окна `Call Scheduling`. Новому звонку присваивается тип “обратный вызов”.
13. Если при выходе из окна `Placement` все оставшиеся билеты кампании уже распределены, все последующие звонки благотворителям для данной кампании не имеют смысла. Все подобные звонки должны быть удалены из очереди звонков.

У1. *Телемаркетинг* — обратитесь к изложенным выше дополнительным требованиям, а также к решениям, предложенным в ходе разбора примера *Телемаркетинг*, как определено в диаграмме видов деятельности для книги. Рассмотрите диаграмму классов из примера 4.7 (разд. 4.2.1.2.3).

Модифицируйте и расширьте диаграмму классов, показанную на рис. 4.4, для поддержки дополнительных требований, которые сформулированы выше.

У2. *Телемаркетинг* — обратитесь к изложенным выше дополнительным требованиям, а также к решениям, предложенным в ходе разбора примера *Телемаркетинг*, как определено в диаграмме видов деятельности для книги.

Спроектируйте и сделайте набросок главного окна для приложения "Телемаркетинг", т.е. окна `Управление телемаркетингом`. Объясните, каким образом это окно удовлетворяет соответствующие требования к приложению.

У3. *Телемаркетинг* — обратитесь к изложенным выше дополнительным требованиям, а также к решениям, предложенным в ходе разбора примера *Телемаркетинг*, как определено в диаграмме видов деятельности для книги.

Спроектируйте и сделайте набросок окна `Supporter History` для приложения "Телемаркетинг". Объясните, каким образом это окно удовлетворяет соответствующие требования к приложению

У4. *Телемаркетинг* — обратитесь к изложенным выше дополнительным требованиям, а также к решениям, предложенным в ходе разбора примера *Телемаркетинг*, как определено в диаграмме видов деятельности для книги.

Изобразите диаграмму навигации по окнам для приложения *Телемаркетинг*. Если потенциальное решение оказывается слишком сложным, разделите диаграмму на несколько меньших диаграмм. Старайтесь не слишком поддаваться влиянию диаграммы навигации по окнам из примера 7.6 (разд. 7.6.2). Объясните, каким образом это окно удовлетворяет требования к приложению.

У5. *Internet-магазин* — обратитесь к решениям, приведенным в наставлении по приложению *Internet-магазин*, как определено в диаграмме видов деятельности для книги.

Спроектируйте и сделайте набросок Web-страницы, которая отображает состояние текущего заказа для клиента. Объясните, с какими сложностями вы встретились.

У6. *Internet-магазин* — обратитесь к решениям, приведенным в наставлении по приложению *Internet-магазин*, как определено в диаграмме видов деятельности для книги.

Изобразите диаграмму навигации по окнам для приложения *Internet-магазин*. Если потенциальное решение оказывается слишком сложным, разделите диаграмму на несколько меньших диаграмм. Объясните, каким образом это окно удовлетворяет требования к приложению.