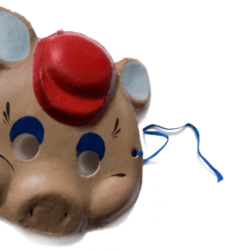
ПРОФЕССИОНАЛЬНО

Алан Купер

ОБ ИНТЕРФЕЙСЕ

основы проектирования взаимодействия



Алан КУПЕР
Роберт РЕЙМАН
Дэвид КРОНИН

ABOUT FACE 3

СИМВОЛ®

About Face 3

The Essentials of Interaction Design

Alan Cooper, Robert Reimann, Dave Cronin

Wiley Publishing, Inc.

ПРОФЕССИОНАЛЬНО

Алан Купер об интерфейсе

Основы проектирования взаимодействия

Алан Купер, Роберт Рейман, Дэвид Кронин

СИМВОЛ®

Санкт-Петербург — Москва
2009

Серия «Профессионально»

Алан Купер, Роберт Рейман, Дэвид Кронин

Алан Купер об интерфейсе
Основы проектирования взаимодействия

Перевод М. Зислиса

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*А. Копылов*
Редактор	*В. Подобед*
Художник	*О. Макарова*
Корректор	*С. Минин*
Верстка	*Д. Орлова*

Купер А., Рейман Р., Кронин Д.

Алан Купер об интерфейсе. Основы проектирования взаимодействия. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 688 с., ил.

ISBN 978-5-93286-132-5

Когда в 1995 году увидело свет первое издание «About Face», идея проектировать продукты исходя из целей людей казалась революционной. Благодаря работам Алана Купера и других первопроходцев, проектирование взаимодействия получило сегодня широкое признание как уникальная и крайне важная дисциплина, однако эта работа далека от завершения.

Авторы полностью обновленного издания, признанные мировые эксперты в вопросах создания интерфейсов, детально описывают разработанный в компании Cooper и примененный во множестве проектов целостный подход к проектированию взаимодействия, ориентированный на цели пользователя. Отличительной чертой книги является ее практическая направленность – значительную часть издания занимает подробный разбор принципов и шаблонов проектирования взаимодействия. Большое внимание уделено новым информационным средам: веб-приложениям, мобильным приложениям, киоскам и т. п.

Книга адресована всем специалистам, по роду деятельности соприкасающимся с процессом создания цифровых продуктов. Проектировщикам взаимодействия и дизайнерам интерфейсов она послужит настольным справочником по организации процесса и повседневным подручным инструментарием.

ISBN 978-5-93286-132-5
ISBN 978- 0-470-08411-3 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Подписано в печать 06.04.2009. Формат 70x100$^{1}/_{16}$. Печать офсетная.
Объем 43 печ. л. Тираж 1500 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Сью, моей лучшей подруге во всех жизненных испытаниях.

Максвеллу Аарону Рейману.

Гретхен.

А также куперистам прошлого, настоящего и будущего
и тем умеющим мечтать практикам, чьими усилиями
проектирование взаимодействия превратилось в профессию.

Оглавление

Об авторах

Алан Купер – новатор в области программного обеспечения, программист, проектировщик и теоретик. Его упоминают как создателя первых серьезных деловых программ для микрокомпьютеров и часто называют «отцом языка Visual Basic». Последние 15 лет компания Алана, Cooper, занималась консультированием в области проектирования взаимодействия, помогая другим компаниям создавать новые продукты и совершенствовать поведение цифровых технологий. В компании Cooper Алан возглавлял работы по созданию новой методологии проектирования успешного программного обеспечения, которую он называет целеориентированным процессом. В ходе этой работы, в частности, появились персонажи – инструмент, получивший широкое распространение после выхода в 1998 году второй книги Алана «Психбольница в руках пациентов». Купер известен также как писатель, лектор и горячий сторонник гуманизации технологий.

Роберт Рейман посвятил последние 15 лет расширению границ цифровых продуктов в роли проектировщика, писателя, лектора и консультанта. Он возглавлял десятки проектов по проектированию взаимодействия в таких областях, как электронная коммерция, порталы, персональная производительность, среды создания контента, медицинские и научные приборы, беспроводные технологии и портативные устройства, работая как с начинающими компаниями, так и с компаниями из числа Fortune 500. В качестве главы исследовательского отдела в Cooper Рейман руководил разработкой и совершенствованием многих из целеориентированных методов, описанных в данной книге. В 2005 году Рейман стал первым президентом ассоциации проектирования взаимодействия (IxDA, www.ixda.org) – глобальной некоммерческой организации, объединяющей проектировщиков взаимодействия. В настоящий момент он отвечает за проектирование опыта взаимодействия в Bose Corporation.

Дэйв Кронин – руководитель отдела проектирования взаимодействия в Cooper. Он помогает проектировать продукты, служащие нуждам таких групп людей, как хирурги, посетители музеев, маркетологи, инвестиционные аналитики, интернет-покупатели, персонал больниц, водители, стоматологи, финансовые аналитики, логистики, пожилые и немощные люди. Работая в компании Cooper, он также внес существенный вклад в процесс создания и совершенствования методов целеориентированного проектирования, описываемых в этой книге.

Предисловие. Постиндустриальный мир

Промышленная эра завершилась. Производство, бывшее основным стимулом развития экономики в последние 175 лет, сдало свои позиции. Да, объемы производства сегодня велики как никогда, однако лидерство принадлежит цифровым технологиям, а в экономике ключевую роль играет программное обеспечение. Произошел переход от атомов к битам. На дворе постиндустриальная эра.

Все больше и больше продуктов содержат в себе программный код. В моей духовке есть микросхема, управляющая подсветкой, вентиляцией и температурой нагрева. Когда мне доставляют товары на дом, я расписываюсь уже не на листке бумаги, а в компьютере. Выбирая автомобиль, в действительности я выбираю навигационную систему.

Все большее число компаний сегодня полностью зависят от программного обеспечения, и речь идет не только о таких очевидных примерах, как Amazon.com и Microsoft. Тысячи компаний от мала до велика, предлагающих продукты и предоставляющих услуги в самых различных областях бизнеса, используют программное обеспечение во всех сферах своей деятельности, будь то финансовые операции, управление, планирование или продажи. Системы бизнес-логики, поддерживающие работу крупных компаний, – это программные системы. Поиск и найм сотрудников, управление персоналом, инвестиции и арбитражные операции, закупки и организация поставок, финансовые операции и поддержка принятия решений – все эти системы сегодня основаны на программном коде. Продажи и маркетинг осуществляются преимущественно через Интернет. На «передовой» больше нет живых людей – теперь там находится программное обеспечение. Поставщики, клиенты, коллеги, сотрудники – все они общаются с компаниями посредством программ или при поддержке программ.

Организационные структуры и методы управления, хорошо зарекомендовавшие себя в прошлом в промышленном производстве, сегодня, в постиндустриальную эру, подводят нас. Подводят потому, что ориентированы на преобразования и перемещение таких вещей, которые сделаны из атомов. Нужные нам атомы существуют в ограниченных количествах и требуют значительных энергозатрат для преобразования и транспортировки. Программы сделаны из битов, а не из атомов,

и это качественное отличие. Существует бесконечное количество битов, а их транспортировка, преобразование и даже копирование практически не требуют затрат энергии.

Отличаются и люди, создающие программы. Склонности, отношение к работе, подготовка, язык, инструменты и система ценностей у среднего программиста и среднего рабочего на конвейере качественно различны. Наиболее эффективные способы управления программистами – радикально иные по сравнению с эффективными способами управления рабочим классом, применявшимися в прошлом. Чтобы добиться от программистов выполнения задач компании, требуются навыки, неведомые управленцам промышленной эпохи.

Снижение стоимости производства было главным достижением индустриализации. Поэтому лучшие и светлейшие умы этой эры бились над уменьшением расходов, связанных с созданием продукции. В постиндустриальную эру стоимость сырья, сборки и поставки продукции одинаково мала для всех игроков рынка. Единственный действенный путь снижения затрат на производство связан с автоматизацией, планированием и управлением знаниями, т. е. с программами. Другими словами, вместо того чтобы экономить доллар на создании каждой безделушки, можно сэкономить миллион, произведя наиболее желанный товар в нужном количестве.

Будучи единожды созданной, программа может копироваться бесконечное число раз практически бесплатно. Сокращение затрат на написание этой программы не приносит практически никакой выгоды и, как правило, выливается в снижение качества. Поэтому основное уравнение бизнеса промышленной эпохи сегодня оказывается перевернутым: лучшие и светлейшие головы посвящают себя повышению эффективности программ и качества взаимодействия с ними. Не стоит забывать, что все современные финансово-расчетные системы направлены на пристальное отслеживание стоимости производства и потому не способны корректно отражать состояние бизнеса, основанного на программном обеспечении. Когда руководство принимает решения на основе этих ошибочных показателей, теряется масса времени, тратятся лишние деньги и упускаются благоприятные возможности.

Неудивительно, что компании сталкиваются с огромными трудностями при создании программ. Талантливые управленцы обнаруживают, что их замыслы неуловимо, но существенно видоизменяются где-то на пути от концепции продукта к его выпуску. Хорошие с виду планы оказываются непригодными, когда дело доходит до управления созданием программного продукта. Пришла пора избавиться от устаревших методов управления и дать дорогу проектированию взаимодействия как ключевому инструменту создания программных продуктов и управления их разработкой.

Со дня первой публикации этой книги в 1995 году область проектирования взаимодействия значительно выросла и возмужала. Просуществово-

вав долгое время под гнетом метода проб и ошибок и решений, принимаемых задним числом, ныне проектирование взаимодействия вместе со своими вариациями и смежными областями превратилось в четкий, надежный, эффективный инструмент проектирования успешного поведения продуктов. С изобретением и внедрением методики персонажей, усовершенствованием текстовых сценариев взаимодействия и с появлением практики целеориентированного проектирования (Goal-Directed™ Design) в целом все компании получили в свое распоряжение средства, которые позволяют создавать программные продукты, ориентированные на высококачественное взаимодействие с пользователями.

Более того, проектирование взаимодействия превратилось в чрезвычайно мощный инструмент управления разработкой программного продукта. Предоставляя описание окончательного варианта продукта, проектирование взаимодействия снабжает разработчиков аналогом чертежей, который помогает программистам осознавать, что именно они создают, а руководителям – оценивать прогресс в работе программистов.

Проектирование взаимодействия проявило себя и как маркетинговый инструмент, предоставляющий предельно ясную и точную информацию о том, кто конкретно будет использовать продукт и для чего. Понимание источника мотивации покупателей – манна небесная для маркетологов. Качественные исследования и аналитические аспекты целеориентированного проектирования позволяют достичь глубокого понимания рынка.

После того как начался интернет-бум, породивший ощущение, что отбрасывание здравого смысла – самый верный путь к мгновенному обогащению, мне все чаще приходилось слышать от разумных и вроде бы вменяемых людей, что знать, чего хочет пользователь, попросту невозможно! Это убеждение, конечно, позволяет увильнуть от необходимости действительно понять, чего хочет пользователь, однако нет ничего более далекого от истины. В моей компании Cooper поставленные клиентами задачи заставляют проектировщиков постигать премудрости финансов, здравоохранения, фармацевтики, управления персоналом, инструментариев программирования, музейного дела, потребительского кредитования и многих других специфических областей. Наши команды, не располагающие подготовкой в интересующих клиентов областях, а иногда даже и не имеющие представления об оных, к восхищению наших клиентов всего за несколько недель становятся достаточно подкованными в теме. Как нам это удается? Секрет в том, что за отправную точку мы неизменно принимаем человека, а не технологию.

Проектирование взаимодействия – это инструмент, позволяющий «узнавать, чего хочет пользователь». Вооружившись этим знанием, вы можете создавать более качественные, более успешные цифровые продукты, приносящие больше денег. А благодаря лояльности довольных пользователей вы сумеете занять своими решениями рынок. Мы

много раз видели, как перегруженные функциями продукты, первыми успевшие на рынок, оказывались вытесненными более поздними продуктами, которые были тщательнее продуманы в плане взаимодействия с пользователем. Не лучше ли продумать все до того, как самый первый выпуск продукта выведет его на далекую от оптимальности траекторию существования?

Ничто так не вдохновляет, как успех. Успех практического применения принципов и методов, лежащих в основе этой книги, а также родственных им, отчетливо демонстрирует, что программные продукты не настолько гибки, как многим казалось, и что досконального изучение пользователей и детальное планирование в постиндустриальном веке важны как никогда.

Если вы преданы идее улучшения мира через совершенствование поведенческих аспектов цифровых продуктов и услуг, я рад приветствовать вас во вселенной целеориентированного проектирования!

Алан Купер

Благодарности

Мы хотели бы выразить свою глубочайшую признательность следующим людям, без которых не появилось бы на свет новое издание «About Face»: Крису Уэббу (Chris Webb) из Wiley, который понял, что настало время для нового издания, Сью Купер (Sue Cooper), которая согласилась с этой мыслью, и Саре Шлейр (Sara Shlaer) из Wiley, которая терпеливо помогала нам в подготовке всех редакций этой книги.

Мы хотели бы также поблагодарить наших коллег и проектировщиков из Cooper за их вклад в эту работу и в предыдущие ее воплощения: Ким Гудвин (Kim Goodwin) за серьезный вклад в разработку и описание концепций и методов, представленных на этих страницах; Ребекку Бортман (Rebecca Bortman) и Ника Майерса (Nick Myers) за пересмотр оформления книги и обложки, а также за иллюстрации; Хью Дабберли (Hugh Dubberly) за помощь в разработке принципов, приводимых в конце главы 8, и за иллюстрации к целеориентированному процессу в главе 1; Гретхен Андерсон (Gretchen Anderson), Элейн Монтгомери (Elaine Montgomery) и Дага Лемуана (Doug LeMoine) за их вклад в материал об исследованиях пользователей и рынка в главе 4; Рика Бонда (Rick Bond) за его многочисленные идеи в связи с юзабилити-тестированием (глава 7); Эрнеста Кинсолвинга (Ernest Kinsolving) и Йорга Берингера (Joerg Beringer) из SAP за их вклад в описание типов приложений веб-порталов в главе 9; Криса Уилдрайера (Chris Weeldreyer) за идеи по проектированию встроенных систем (глава 9); Уэйна Гринвуда (Wayne Greenwood) за вклад в тему отображения элементов управления (глава 10); Нейта Фортена (Nate Fortin) и Ника Майерса (Nick Myers) за вклад в тему визуального проектирования интерфейсов и брендинга (глава 14). Мы говорим спасибо Элизабет Бейкон (Elizabeth Bacon), Стиву Калду (Steve Calde), Джону Даннингу (John Dunning), Дэвиду Фору (David Fore), Нейту Фортену (Nate Fortin), Ким Гудвин (Kim Goodwin), Уэйну Гринвуду (Wayne Greenwood), Ноа Гийо (Noah Guyot), Лейн Халли (Lane Halley), Эрнесту Кинсолвингу (Ernest Kinsolving), Дэниелу Куо (Daniel Kuo), Берму Ли (Berm Lee), Дагу Лемуану (Doug LeMoine), Тиму Маккою (Tim McCoy), Элейн Монтгомери (Elaine Montgomery), Нику Майерсу (Nick Myers), Крису Нойсселу (Chris Noessel), Райану Ольшавски (Ryan Olshavsky), Анджеле Куэйл (Angela Quail), Сьюзи Томпсон (Suzy Thompson), а также Крису Уилдрайеру (Chris Weeldreyer) за их вклад в работы компании Cooper и иллюстрации, представленные в данной книге.

Мы выражаем признательность нашим клиентам – Дэвиду Уэсту (David West) из Shared Healthcare Systems, Майку Кею (Mike Kay) и Биллу Чангу (Bill Chang) из Fujitsu Softek, Джону Чаффинсу (John Chaffins) из CrossCountry, Крису Тугуду (Chris Twogood) из Teradata и Крису Доллару (Chris Dollar) из McKesson – за разрешение использовать проекты, выполнявшиеся для них компанией Cooper, в качестве примеров в этой книге. Мы хотим также поблагодарить многих других клиентов, у которых хватило прозорливости и воображения, чтобы работать с нами и отстаивать наш подход в своих организациях.

Мы хотели бы также засвидетельствовать свое почтение следующим авторам и коллегам, чьи работы много лет служили для нас источником вдохновения и призмой для рассмотрения идей: Кристофер Александер (Christopher Alexander), Эдвард Тафт (Edward Tufte), Кевин Маллет (Kevin Mullet), Виктор Папанек (Victor Papanek), Дональд Норман (Donald Norman), Ларри Константайн (Larry Constantine), Чаллис Ходж (Challis Hodge), Шелли Ивенсон (Shelley Evenson), Клиффорд Насс (Clifford Nass), Байрон Ривз (Byron Reeves), Стивен Пинкер (Stephen Pinker) и Терри Суок (Terry Swack).

Наконец, следует отметить, что фрагменты главы 5, касающиеся когнитивных процессов обработки, изначально были опубликованы в статье Роберта Реймана на сайте UXMatters.com и использованы с разрешения владельцев авторских и имущественных прав.

Введение к третьему изданию

Эта книга посвящена **проектированию взаимодействия** – практике создания цифровых интерактивных продуктов, сред, систем и служб. Как и многие другие дисциплины проектирования, проектирование взаимодействия работает с формой. Однако в первую очередь оно сосредоточено на аспекте, который нечасто затрагивают традиционные дисциплины проектирования и дизайна, – на проектировании *поведения*.

Почти все виды проектирования и дизайна *влияют* на поведение человека: архитектура имеет дело с тем, как люди используют физическое пространство, а графический дизайн – с вопросами мотивации или получения необходимых реакций от людей. Однако сегодня, когда повсеместное распространение получили устройства с микросхемами внутри – от компьютеров до автомобилей и телефонов, – мы ежедневно создаем продукты, которые *демонстрируют* сложное поведение.

Возьмем для примера такую простую вещь, как духовка. До наступления цифровой эры управление духовкой было весьма простым – достаточно было повернуть единственный регулятор в нужное положение. Выключенной духовке соответствовало строго одно положение, каждому из возможных температурных режимов также отвечало одно положение. Всякий раз при повороте регулятора в конкретное положение *происходило одно и то же*. Это можно назвать «поведением», но это определенно очень простое, механическое поведение. Сравните его с поведением современных микроволновок с микросхемами и жидкокристаллическими экранами. Они изобилуют кнопками с надписями, не имеющими отношения к приготовлению пищи: «Начать», «Отменить», «Программировать», которые соседствуют с более привычными, такими как «Запекать» и «Жарить». Угадать, что случится, если нажать какую-либо из таких кнопок, гораздо сложнее, чем в случае со старым регулятором на газовой плите. Более того, результаты нажатия любой из этих кнопок полностью зависят от текущего состояния микроволновки и того, какие кнопки были нажаты до этого. Вот что мы имеем в виду, когда говорим о *сложном поведении*.

Появление продуктов с таким сложным поведением стало причиной рождения новой дисциплины. Проектирование взаимодействия перенимает теорию и методы из таких областей, как традиционный дизайн, юзабилити и инженерные дисциплины. Но в данном случае целое больше суммы частей и обладает собственными уникальными ме-

тодами и подходами. И, скажем сразу, это в большой степени область именно дизайна, которая сильно отличается от научных и инженерных дисциплин. При неизменной взвешенности и рациональности подхода проектирование взаимодействия не обязательно имеет дело с текущим положением вещей – оно связано с придумыванием и синтезом того, каким это положение может и должно быть.

Помимо прочего, проектирование взаимодействия по сути своей – еще и гуманистическая затея. Эта дисциплина направлена в первую очередь на удовлетворение потребностей и желаний людей, имеющих дело с продуктом или услугой. В данной книге мы описываем конкретный подход к проектированию взаимодействия, который мы назвали целеориентированным методом. Мы обнаружили, что когда проектировщик заостряет свое внимание на целях людей – в первую очередь на побудительных мотивах использования продукта, – а также на их ожиданиях, наклонностях и отношении к окружению, возникают решения, которые люди находят мощными и приятными.

Как может заметить даже самый неискушенный наблюдатель, развитие технологий способно крайне быстро приводить к серьезному усложнению интерактивных продуктов. В то время как у механического устройства ясно различимых состояний может быть десяток, цифровой продукт способен находиться в тысячах (а то и большем числе!) различных состояний. Эта сложность может превратиться в кошмар как для пользователей, так и для проектировщиков. Чтобы обуздать ее, мы полагаемся на системный рациональный подход. Это не означает, что мы не ценим и не поощряем изобретательность и творчество. Напротив, оказывается, что методичность помогает нам замечать благоприятные обстоятельства для инновационных идей и предоставляет способ оценки их эффективности.

Согласно гештальт-психологии, человек воспринимает предмет не как набор отдельных свойств и функций, а как единое целое, неразрывно связанное с окружением. Вследствие этого невозможно эффективно спроектировать интерактивный продукт, разобрав его на атомы требований и создав отдельное решение по каждому из требований. Даже относительно простой продукт следует рассматривать цельно и в контексте окружающего его мира. И здесь мы вновь обнаруживаем, что систематический подход помогает получать целостную перспективу, столь необходимую для создания продуктов, которые люди сочтут полезными и привлекательными.

Краткая история проектирования взаимодействия

В конце семидесятых – начале восьмидесятых годов группа увлеченных исследователей, инженеров и дизайнеров в Сан-Франциско занималась изучением того, как люди могли бы взаимодействовать с компь-

ютерами в будущем. В Xerox PARC SRI, а потом и в Apple Computer вовсю обсуждали, в чем состоит создание полезных и удобных **«гуманных интерфейсов»** в применении к цифровым продуктам. В середине восьмидесятых промышленные дизайнеры Билл Могридж (Bill Moggridge) и Билл Верпланк (Bill Verplank), работавшие над первым ноутбуком (GRiD Compass), предложили для описания своей работы термин **проектирование взаимодействия**, однако понадобилось еще десятилетие, чтобы другие проектировщики заново сформулировали это понятие и сделали его общепринятым.

Когда в августе 1995 года увидело свет первое издание книги «About Face», ландшафт проектирования взаимодействия своей неорганизованностью напоминал «дикие прерии». Маленький отряд храбрецов, отважившихся носить звание «Проектировщик пользовательских интерфейсов» (user interface designer), предпринимал вылазки под прикрытием разработчиков программного обеспечения, словно кучка крошечных сообразительных млекопитающих, суетящихся в тени неповоротливых динозавров. Мало кто понимал, что представляет собой и какое имеет значение то «проектирование программного обеспечения», о котором мы говорили в первом издании; если им кто-либо и занимался, то это обычно оказывались программисты. Лишь горстка обеспокоенных технических писателей, преподавателей, специалистов служб технической поддержки, а также постепенно растущая группа профессионалов из другой зарождающейся отрасли – юзабилити – осознавали необходимость перемен.

Эти перемены произошли молниеносно благодаря всплеску популярности и лавинообразному росту Интернета. Вдруг у всех на устах оказался термин **«простота использования»**. Традиционные дизайнеры, ваявшие цифровые продукты в краткую эпоху популярности мультимедиа в начале девяностых, дружно перебрались во Всемирную паутину. Как грибы после дождя стали появляться новые, как казалось, названия профессий, связанных с дизайном и проектированием: информационные дизайнеры (information designer), информационные архитекторы (information architect), специалисты по проектированию опыта взаимодействия (user experience strategist) и проектировщики взаимодействия (interaction designer). Впервые возникли руководящие должности, связанные с созданием ориентированных на пользователей продуктов и услуг, например директор по опыту взаимодействия (chief experience officer). Университеты быстро подхватили идею и начали предлагать программы подготовки специалистов по этим дисциплинам. Тем временем юзабилити-специалисты и профессионалы, имеющие дело с человеческим фактором, также получили право голоса и стали признанными борцами за качественное проектирование продуктов.

Хотя Всемирная паутина отшвырнула инструментарий проектирования взаимодействия более чем на десятилетие в прошлое, она, бесспор-

но, совершила благое дело, поместив требования пользователей в поле зрения корпораций. Со времени выхода второго издания «About Face» в 2003 году *опыт взаимодействия* при общении с цифровыми продуктами стал темой первых полос таких изданий, как журналы *Time* и *BusinessWeek*, а такие организации, как Harvard Business School и Стэнфордский университет, признали, что следующее поколение управленцев и технологов должно находить проектированию взаимодействия место в своих бизнес-планах и графиках разработок – и это следует учитывать в программах их подготовки. Люди устали от «технологии ради технологии». Потребители посылают недвусмысленное сообщение: им нужна *хорошая* технология, то есть такая, опыт взаимодействия с которой будет подкупающе простым и приятным.

В августе 2003 года, через пять месяцев после того, как во втором издании «About Face» было провозглашено существование новой дисциплины проектирования – *проектирования взаимодействия*, – Брюс «Тог» Тогнаццини (Bruce «Tog» Tognazzini) обратился к зарождающемуся сообществу со страстным призывом, предложив создать некоммерческую профессиональную организацию. Короткое время спустя Чаллис Ходж (Challis Hodge), Дэвид Хеллер (David Heller), Рик Сесил (Rick Cecil) и Джим Джаррет (Jim Jarrett) создали список рассылки и организовали руководящий комитет. В сентябре 2005 года официально родилась организация IxDA – Interaction Design Association (ассоциация проектирования взаимодействия, *www.ixda.org*). На момент подготовки этой книги в организации уже состояло более 2000 членов из 20 с лишним стран. Нам приятно заявить, что проектирование взаимодействия наконец становится самостоятельной дисциплиной и профессией.

Почему «проектирование взаимодействия»?

В первом издании книги мы описывали дисциплину дизайна программного обеспечения и отождествляли ее с другой дисциплиной – проектированием пользовательских интерфейсов. Из двух названий более живучим оказалось, конечно, «проектирование пользовательских интерфейсов». Мы по-прежнему будем использовать его здесь в тех случаях, когда это уместно, в частности для обозначения расположения элементов интерфейса на экране. Однако в этой книге говорится о дисциплине более широкой, нежели проектирование пользовательских интерфейсов. В мире цифровых технологий столь тесно переплетены форма, функции, содержание и поведение, что, отвечая на вызовы, которые бросает нам проектирование интерактивных продуктов, мы сплошь и рядом затрагиваем самую суть того, чем продукт *является* и что он *делает*.

Как мы уже говорили, проектировщики взаимодействия, заимствуя подходы из других, уже сложившихся дисциплин проектирования,

выходят за пределы этих подходов. Одно время к проектированию цифровых продуктов пытались обратиться промышленные дизайнеры, однако, как и их коллеги из области графического дизайна, они обычно фокусировались на проектировании статической формы вместо интерактивной, то есть такой, которая реагировала бы на внешние факторы и менялась со временем. В этих дисциплинах отсутствует язык, который позволил бы обсуждать проектирование динамичных, развитых моделей поведения и изменяющихся пользовательских интерфейсов.

В последние годы для этой разновидности проектирования предлагались различные названия. С распространением Всемирной паутины появилась *информационная архитектура* (*information architecture*) – дисциплина, посвященная решению вопросов, связанных с навигацией и «находимостью» содержимого – преимущественно (хотя и не исключительно) в контексте веб-сайтов. Будучи очевидно близкой к проектированию взаимодействия, информационная архитектура по большей части сохранила свой узкий, ориентированный на веб-окружение подход к организации содержимого и навигации, использующий страницы, ссылки и в минимальной степени интерактивные интерфейсные элементы. Однако свежие веяния в этой области, такие как Web 2.0 и полнофункциональные интернет-приложения, вывели веб-дизайнеров из оцепенения, подтолкнув их к поиску за пределами архаичных идиом взаимодействия с броузером. Мы считаем, что это пробуждение еще сильнее сближает информационных архитекторов с проектировщиками взаимодействия.

Еще один термин, который приобрел популярность, – *опыт взаимодействия* (*user experience*, UX). Многие люди выступают за использование этого термина в качестве «зонтика», под которым объединяется множество различных дисциплин, связанных с проектированием и удобством использования продуктов, систем и услуг. Достойная и заманчивая цель, которая, однако, слабо связана с ключевой проблемой проектирования взаимодействия, обсуждаемой в этой книге: как конкретно следует проектировать поведение сложных интерактивных систем? И хотя поиск сходств и путей взаимного обогащения подходов в проектировании опыта взаимодействия для покупателя в реальном магазине и для пользователя интерактивного продукта – полезное занятие, мы считаем, что существуют особые методы, уместные именно для проектирования в цифровом мире.

А возможно ли действительно *спроектировать* опыт взаимодействия? Проектировщики всех мастей лелеют надежду управлять опытом людей и *влиять* на него, однако в качестве инструмента выступает всего лишь аккуратное манипулирование свойствами, присущими используемой среде. Дизайнер, работающий над плакатом, определяет опыт взаимодействия при помощи сочетания текста, фотографий и иллюстраций; дизайнер мебели, работающий над креслом, создает опыт по-

средством материалов и инженерных методов; дизайнер интерьеров влияет на опыт расстановкой, освещением, подбором материалов и даже звуком.

При распространении этого подхода на цифровые продукты оказывается полезным считать, что мы воздействуем на опыт пользователей, проектируя механизмы взаимодействия с продуктом. Поэтому мы предпочли термин Могриджа **проектирование взаимодействия** (многими сокращаемое до IxD – от interaction design) для обозначения вида проектирования, описываемого в настоящей книге.

Разумеется, во многих случаях при создании продукта требуется слаженное применение целого ряда дисциплин проектирования, чтобы дать пользователю позитивный опыт (рис. 1). Мы считаем, что именно для таких ситуаций уместен термин *проектирование опыта взаимодействия*.

Рис. 1. *Можно считать, что в проектировании опыта взаимодействия (UX) для цифровых продуктов сочетаются три пересекающиеся темы: форма, поведение, наполнение. Проектирование взаимодействия сосредоточено на проектировании поведения, но помимо этого ставит вопрос о том, как это поведение связано с формой и наполнением. В свою очередь, информационная архитектура занимается структурированием наполнения, но ставит вопрос о том, какие поведенческие модели обеспечивают доступ к этому наполнению и каким образом представлено наполнение. Промышленный дизайн и графический дизайн отвечают за форму продуктов и услуг, но помимо этого должны гарантировать, что форма поддерживает модель использования, а это требует внимания к поведению и наполнению*

Сотрудничество с командой, создающей продукт

Помимо определения проектирования взаимодействия в терминах решаемых задач и отношений с другими дисциплинами проектирования, часто оказывается необходимым найти этой дисциплине подходящее место внутри организации. Мы считаем, что построение четкого процесса разработки, в котором проектирование имеет равные права с разработкой, маркетингом и управлением бизнесом, а зоны ответственности каждой из групп строго определены, значительно увеличивает те выгоды, которые бизнес может извлечь из проектирования. Описанное ниже распределение ответственности, уравновешенное соответствующим распределением полномочий, позволяет значительно повысить шансы продукта на успех и обеспечить организационную поддержку продукта как на протяжении всего цикла разработки, так и по его окончании.

- Команда **проектировщиков** отвечает за степень удовлетворенности пользователей процессом взаимодействия с продуктом. На текущий момент в большинстве организаций за это никто не отвечает. Чтобы нести такую ответственность, проектировщики должны иметь возможность определять внешний вид продукта, его поведение и создаваемое им впечатление. Кроме того, им понадобится доступ к информации: они должны наблюдать за потенциальными пользователями продукта, обсуждать с пользователями их потребности, с инженерами – технологические возможности и ограничения, с маркетологами – возможности и требования, а с руководством – что из себя должен представлять конечный продукт.
- Команда **инженеров** отвечает за реализацию и производство продукта. Чтобы проектирование принесло пользу, инженеры должны нести ответственность за воспроизведение формы и поведения продукта в том виде, как это *задано проектировщиками*, уложившись при этом в бюджет и выдержав сроки. Следовательно, инженерам требуется ясное описание формы и поведения продукта, которое будет направлять их работу и послужит точкой отсчета при оценке затрат времени и средств. Такое описание должно поступать от команды проектировщиков. Инженеры должны также участвовать в обсуждениях технических возможностей и ограничений и оценке выполнимости предложенных проектировщиками решений.
- Команда **маркетологов** несет ответственность за желание клиентов заказать продукт, а потому в их ведении должны находиться все аспекты общения с клиентами. У них должна быть также возможность повлиять на характеристики и дизайн продукта. Для этого участники маркетинговой команды должны иметь доступ к информации, включая результаты исследований проектировщиков, и располагать возможностью проводить собственные исследования. (Следует отметить, что клиенты и пользователи – зачастую совсем разные люди с разными потребностями. Мы обсудим эту тему более подробно в главах 4 и 5.)

- **Руководство** отвечает за прибыльность полученного продукта, а потому имеет полномочия принимать решения о том, над чем следует работать перечисленным выше группам. Чтобы принимать такие решения, руководство должно получать внятные сведения от других групп: результаты исследований проектировщиков и характеристики продукта, результаты маркетинговых исследований и прогнозы продаж, оценки инженеров относительно времени и затрат, необходимых для создания продукта.

Что есть и чего нет в этой книге

Мы попытались дать читателям эффективные, практичные инструменты для проектирования взаимодействия. Наш набор инструментов состоит из *принципов*, *шаблонов* и *процессов*. В *принципах* проектирования сформулированы общие идеи о практике проектирования, а также правила и советы относительно наилучшего применения тех или иных идиом взаимодействия и пользовательского интерфейса. *Шаблоны* проектирования описывают такие наборы идиом взаимодействия, которые регулярно применяются для реализации определенных пользовательских требований и решения типичных проблем проектирования. *Процессы* проектирования определяют схему, позволяющую понять и описать требования пользователей, преобразовать эти требования в общую структуру проекта и, наконец, найти лучший способ применения принципов и шаблонов проектирования в конкретных ситуациях.

Существуют книги, посвященные принципам и шаблонам проектирования; книг, посвященных процессам проектирования, гораздо меньше; и совсем мало книг, повествующих одновременно обо всех этих инструментах и их совместном применении для целей качественного проектирования. При написании этой книги нашей целью было сведение трех инструментов воедино. Помогая вам проектировать более эффективные и полезные диалоговые окна и меню, эта книга одновременно поможет понять, как пользователи воспринимают ваш цифровой продукт и взаимодействуют с ним и как использовать это понимание в качестве отправной точки проектирования.

Интеграция принципов, процессов и шаблонов проектирования – ключ к созданию эффективного взаимодействия и эффективных интерфейсов цифровых продуктов. Объективно хорошего пользовательского интерфейса попросту не существует – качество зависит от контекста: кем является пользователь, что он делает, каковы его мотивы. Подход «универсальный интерфейс для всех» *упрощает* создание пользовательских интерфейсов, однако не обязательно *улучшает* качество результата. Если вы хотите добиться качественных решений в проектировании, вам не избежать тяжелой работы по изучению людей, которым адресован продукт. И вот тогда оказывается полезным иметь в распоряжении набор принципов и шаблонов, применимых в конкретных

ситуациях. Мы надеемся, что эта книга не только побудит вас лучше понять пользователей своего продукта, но и научит создавать высококлассные продукты, опираясь на это понимание.

Эта книга *не* предназначена для того, чтобы быть руководством по стилю или стандартом по интерфейсам. Более того, в главе 14 вы узнаете, почему применимость подобных инструментов ограничена весьма узким набором ситуаций. Тем не менее мы надеемся, что процессы и принципы, описанные в этой книге, станут полезным дополнением к используемым вами руководствам по стилю. Такие руководства хорошо отвечают на вопрос *«что?»*, однако редко способны ответить на вопрос *«почему?»*. Эта книга пытается дать ответы на вопросы второго типа.

Мы рассмотрим четыре основные стадии проектирования интерактивных систем: исследование предметной области, анализ пользователей и их требований, определение структуры системы, детализация интерфейсных решений. Многие практики добавили бы пятую стадию – *проверку* эффективности решений на реальных пользователях. Это часть дисциплины, широко известной как *юзабилити*. Будучи важной и результативной составляющей многих начинаний в области проектирования взаимодействия, эта дисциплина, однако, вполне самостоятельна. Проверку проектирования и юзабилити-тестирование мы бегло обсудим в главе 7, однако настоятельно рекомендуем читателям обратиться к постоянно растущему списку книг по юзабилити за более подробной информацией о проведении юзабилити-тестов и анализе их результатов.

Новое в третьем издании

Многое изменилось в мире проектирования интерфейсов с тех пор, как в 1995 году увидело свет первое издание этой книги. В то же время многое осталось прежним. В третьем издании мы сохранили все, что не потеряло актуальности, обновили то, что с тех пор изменилось, и включили новые материалы, которые отражают не только перемены в отрасли за последние одиннадцать лет, но и появление новых концепций, созданных авторами с учетом меняющихся требований времени. Вот некоторые серьезные изменения в «About Face 3.0»:

- Структура книги полностью переработана с тем, чтобы подать идеи в более удобной справочной форме. Книга состоит из трех разделов: первый посвящен процессу и общим идеям о пользователях и проектировании, второй – общим принципам проектирования взаимодействия, третий описывает низкоуровневые принципы проектирования.
- Первая часть гораздо более подробно, чем во втором издании, описывает процесс целеориентированного проектирования, и теперь книга более точно отражает сложившуюся в компании Cooper прак-

тику, включая методики исследований, создание персонажей и применение персонажей и сценариев для синтезирования решений в области проектирования взаимодействия.

- На протяжении всей книги мы старались более явно и наглядно рассказывать о концепциях, методах и проблемах проектирования визуальной части пользовательских интерфейсов, а также о задачах, возникающих за пределами настольных компьютеров.
- Мы обновили терминологию и примеры сообразно состоянию дел в отрасли, а текст в целом подвергся значительным правкам, стал более понятным и легкочитаемым.

Надеемся, что благодаря этим дополнениям и изменениям читатели смогут по-новому взглянуть на рассмотренные темы.

О примерах

Вы держите в руках книгу о проектировании самых разнообразных цифровых интерактивных продуктов. Но поскольку проектирование взаимодействия уходит корнями в программное обеспечение для настольных компьютеров, а подавляющее большинство современных персональных компьютеров работают под управлением Microsoft Windows, наш текст определенно имеет перекос в сторону платформы, на которой потребность в эффективных целеориентированных пользовательских интерфейсах особенно высока.

Сделав эту оговорку, отметим, что основной материал книги не зависит от используемой платформы и в равной степени полезен на всех платформах для персональных компьютеров – Mac OS, Linux и прочих, а большая его часть остается применимой и на таких специфических платформах, как киоски, наладонные устройства, встроенные системы и т. п.

Большинство примеров в этой книге взято из программ пакета Microsoft Office – Word, Excel, PowerPoint, Outlook, а также из Internet Explorer, Adobe Photoshop и Illustrator. Есть две причины, по которым мы старались использовать в качестве источника примеров эти популярные программы. Во-первых, в таком случае примеры будут, вероятно, хотя бы в минимальной степени знакомы читателям. Во-вторых, важно было продемонстрировать, что целеориентированное проектирование позволяет улучшить дизайн пользовательского интерфейса даже самых отточенных продуктов. Мы включили в книгу также некоторое количество примеров из менее распространенных приложений – там, где они были особенно показательны.

Ряд примеров позаимствован из ныне устаревших программ или операционных систем – мы решили сохранить их в новой версии книги, поскольку они хорошо иллюстрируют определенные особенности проектирования. Однако подавляющее большинство примеров взяты из современных программных продуктов и операционных систем.

Для кого эта книга

Хотя содержание книги в основном ориентировано на тех, кто занимается проектированием взаимодействия на практике либо обучается этой дисциплине, пользу от чтения книги получат все, кого заботит взаимодействие пользователей с цифровыми технологиями. Программисты, дизайнеры и проектировщики, работающие над созданием цифровых продуктов, юзабилити-специалисты, а также руководители проектов – все они найдут что-либо полезное для себя. Читатели предыдущих изданий книги «About Face» и «The Inmates Are Running the Asylum»[1] найдут свежую и обновленную информацию о методах и принципах проектирования.

Надеемся, эта книга заинтригует вас и обогатит знаниями, но самое главное – надеемся, что она заставит вас по-новому думать о проектировании цифровых продуктов. Практика проектирования взаимодействия активно развивается, а сама дисциплина настолько молода и изменчива, что порождает огромное разнообразие мнений и мыслей. Если у вас есть интересное мнение или вы просто хотите пообщаться с нами, мы будем рады получить ваши письма по адресам *alan@cooper.com*, *rmreimann@gmail.com* и *dave@cooper.com*.

От редакторов перевода

Вы держите в руках особенную книгу. Для проектировщиков интерфейсов она является тем же, чем для венецианских кораблестроителей XIV века была «Arte de far vasselli» («Искусство судостроения»), с той лишь разницей, что Алан Купер и его команда не делают из своих открытий государственного секрета, и поэтому вы можете спокойно листать эти страницы при дневном свете, не пугаясь собственной тени.

Проектирование взаимодействия – бурно развивающаяся дисциплина. Перемены в ней происходят порой быстрее, чем в земной атмосфере. Терминология (особенно русская) не только не устоялась, но иногда просто-напросто отсутствует. При работе над переводом мы особенно остро ощущали и ураганную скорость перемен, и неизбежную на старте любой науки бедность лексикона, и колоссальную значимость этой книги. Все это привело нас к идее параллельно с подготовкой бумажного издания создать интернет-ресурс, где можно было бы разме-

[1] А. Купер «Психбольница в руках пациентов. Почему высокие технологии сводят нас с ума и как восстановить душевное равновесие», дополненное издание. – Пер. с англ. – СПб: Символ-Плюс, 2009.

щать уточнения к книге и новости мира проектировщиков, обсуждать термины, до хрипоты спорить о принципах... Мы сделали это и приглашаем вас в гости. Наш адрес *http://aboutface.gui.ru*.

Вселенная проектирования взаимодействия еще так молода, что вы можете успеть в ряды ее демиургов!

I

Введение в целеориентированное проектирование

3

Новички, эксперты и середняки

Высвобождая из целлофана новенькую коробку с программой или мобильным телефоном, пользователи компьютеров в большинстве своем прекрасно отдают себе отчет в том, что впереди у них – несколько дней разочарования и раздражения в связи с освоением нового интерфейса. С другой стороны, многие опытные пользователи цифровых продуктов нередко испытывают постоянное раздражение из-за того, что программа продолжает считать их зелеными новичками. И найти точку равновесия между потребностями новичков и потребностями экспертов кажется невозможным.

Это одна из вечных головоломок в проектировании взаимодействия и интерфейсов: как отыскать единое интерфейсное решение, отвечающее потребностям и начинающих пользователей, и экспертов? Некоторые программисты и проектировщики решили вовсе отказаться от этой затеи; вместо этого они выделяют режим для начинающих с помощью набора мастеров, а функциональность, необходимую экспертам, прячут глубоко в меню. Естественно, делать лишнюю работу, связанную с продвижением по шагам мастера, не хочется никому, однако переключение в режим для экспертов требует от пользователя сакрального знания о назначении загадочных команд в длинной последовательности меню и обычно представляет собой прыжок с довольно высокой скалы в акулье логово интерфейса, спроектированного исходя из модели реализации. Как же выйти из этого положения? Решение загадки связано с иным пониманием того, как пользователи овладевают новыми понятиями и задачами.

Вечные середняки

Большинство пользователей – не начинающие и не эксперты; они *середняки*.

Распределение уровня опыта людей в определенной области деятельности, как и многие другие распределения генеральной совокупности, тяготеет к классической форме статистической колоколообразной кривой (рис. 3.1). Практически для любой сферы деятельности, требующей знаний или навыков, на графике, который связывает уровень навыков с числом обладателей навыков такого уровня, в левой части соберутся сравнительно немногочисленные начинающие, в правой окажется горстка экспертов, а большинство – середняки – попадет в центр.

Статистика, правда, кое о чем умалчивает. Колоколообразное распределение – это мгновенный снимок ситуации. Хотя середняки в большинстве своем остаются середняками, начинающие не слишком долго задерживаются в новичках. Сложность поддержания навыков на высоком уровне ведет к тому, что экспертами становятся и перестают быть довольно быстро, однако новички сменяются еще быстрее. Как начинающие, так и эксперты имеют тенденцию с течением времени переходить в разряд людей со средним уровнем навыков.

Хотя *все люди* какое-то минимальное время пребывают в ранге начинающих, *никто* не задерживается в этой группе надолго. Люди не любят быть некомпетентными, новички же некомпетентны по определению. Напротив, обучение и совершенствование приносят удовлетворение, поэтому начинающие очень быстро становятся середняками – или же вообще выпадают из процесса. Скажем, все лыжники некоторое время являются начинающими, но те, кто в течение заметного времени больше падает, чем катается, быстро покидают спорт. Остальные вскоре переходят от детских горок к нормальным склонам. И лишь очень немногие дорастают до самых сложных трасс «для смертников».

Рис. 3.1. *Требования, предъявляемые пользователями к цифровым продуктам, очень сильно зависят от их опыта*

Никто не желает оставаться начинающим.

Те, кто находится в левой части кривой распределения, либо смещаются к центральной части «колокола», либо полностью исчезают с графика и ищут себе такой продукт или род деятельности, который *позволит* им стать середняками. Таким образом, большинство пользователей постоянно обладают адекватными навыками и стремятся к их совершенствованию, а их навыки уходят и вновь возвращаются подобно приливу и отливу, в зависимости от того, как часто они работают с программой. Первым важность проектирования для середняков отметил Ларри Константайн. В своей книге «Software for Use»[1] (Constantine, 2004) он называет таких пользователей **совершенствующимися середняками** (improving intermediates). Авторы этой книги предпочитают термин **вечные середняки**, ибо хотя начинающие быстро переходят в разряд середняков, они редко продвигаются дальше и становятся экспертами.

На хорошем горнолыжном курорте есть пологий спуск для начинающих и несколько сложных склонов, бросающих серьезный вызов умелому лыжнику. Однако чтобы продолжать работать и приносить прибыль, курорт будет делать основную ставку на вечно среднего лыжника, при этом не отпугивая начинающих и не оскорбляя чувства экспертов. Начинающему должно быть легко перейти в мир средних лыжников, а профессионал не должен на своем отвесном спуске спотыкаться о средства помощи для боязливых и осторожных вечных середняков.

Хорошо сбалансированный пользовательский интерфейс во многом строится по тому же принципу. Вместо того чтобы потакать потребностям новичков или экспертов, он направлен главным образом на удовлетворение нужд вечных середняков. В то же время он обеспечивает механизмы, достаточные для эффективной работы «крайних» составляющих аудитории.

Зачастую пользователи-середняки были бы рады больше узнать о программе – просто у них нет на это времени. Однако порой у них появляется возможность плотно поработать с продуктом на протяжении нескольких недель – чтобы, скажем, закончить крупный проект. В этот период они узнают что-то новое о программе, и тогда их знание выходит за прежние границы.

А иногда они месяцами не запускают программу и забывают существенную часть того, что когда-то знали. Они не станут из-за этого начинающими, но по возвращении к программе им потребуются подсказки, чтобы освежить в памяти прежние знания.

---

[1] Л. Константайн, Л. Локвуд «Разработка программного обеспечения». – Пер. с англ. – СПб: Питер, 2004.

Для некоторых специализированных продуктов имеет смысл оптимизировать взаимодействие исходя из потребностей экспертов. Особенно это касается инструментов, поддерживающих профессиональную деятельность технически ориентированных людей, в которой важнее всего обеспечить высокий уровень эффективности. В эту категорию часто попадают инструменты для разработчиков, а также специальные измерительные и медицинские приборы. Мы ожидаем, что пользователи этих продуктов, приступая к работе, уже обладают необходимыми техническими познаниями и готовы потратить значительное время и силы на то, чтобы в совершенстве овладеть приложением.

Подобно этому есть и такие продукты, которые требуют оптимизации для начинающих. В частности, это продукты, которые используются редко или кратковременно, а также продукты для людей с определенными физическими ограничениями. В нашей практике примерами таких продуктов могут служить информационные киоски для общественных мест вроде музеев, а также устройства, помогающие пожилым пациентам с ограниченной подвижностью измерять кровяное давление.

Нас часто спрашивают, для какой части аудитории следует оптимизировать потребительские веб-сайты – для начинающих или для середняков? Мы полагаем, что здесь применимы те же соображения, что и в случае других цифровых продуктов. Качественно спроектированный интерфейс веб-сайта должен помогать пользователям быстро осваивать навигацию и функциональность. Здесь стоит обратить внимание на тот факт, что даже клиент, уже посетивший ваш сайт несколько раз и знакомый с вашими предложениями, а также в целом с идиомами взаимодействия, характерными для интернет-пространств, может заходить к вам недостаточно часто, чтобы запомнить, как сайт организован. Поэтому взаимодействие с сайтом важно делать как можно более прозрачным и очевидным. Кроме того, в последнее время набирает популярность идея отслеживать действия пользователя на сайте и на этой основе адаптировать сайт к потребностям пользователя. В этой связи полезно использовать cookie-файлы, чтобы идентифицировать новых посетителей и предлагать им ненавязчивую помощь при первом обращении к сайту.

Проектирование для пользователей с различной подготовкой

Теперь соотнесем наш «колокол» середняков с тем, как происходит разработка программного обеспечения. Программисты неизбежно становятся экспертами в создаваемых ими программах, поскольку вынуждены изучить все возможные варианты использования, вплоть до наиболее невероятных и сомнительных, чтобы реализовать их в программном коде. Они естественным образом склонны проектировать программы на основе модели реализации, назначив всем вариантам

взаимодействия *равные* приоритеты. Будучи экспертами, они могут легко разобраться в таком интерфейсе.

В то же время сотрудники отдела продаж, маркетологи, руководство часто демонстрируют продукт покупателям, журналистам, партнерам, инвесторам, которые совершенно не знакомы с продуктом. Из-за постоянного взаимодействия с начинающими у этих профессионалов возникает сильно искаженный взгляд на сообщество пользователей. Поэтому не удивительно, что отдел продаж и маркетологи отстаивают перекраивание интерфейса под нужды начинающих. Образно говоря, они требуют к каждому «велосипеду» прикрутить еще «пару маленьких колес», чтобы выручить несчастных новичков.

Программисты организуют взаимодействие способами, подходящими только для экспертов, тогда как маркетологи требуют инструментов взаимодействия, удобных только для начинающих, – и это при том, что, как мы только что видели, самая крупная, самая устойчивая и самая важная группа пользователей – это середняки.

Сложно поверить, что обычной практикой может стать игнорирование потребностей большинства реальных пользователей, однако чаще всего именно так и происходит. Это хорошо видно на примере многих корпоративных и коммерческих программных продуктов. Дизайн продуктов в целом скроен в расчете на пользователей-экспертов, однако, чтобы угодить представлениям маркетологов о новых пользователях, на эти продукты навешены неуклюжие инструменты вроде мастеров (wizards) и помощника Скрепыша. Эксперты вообще редко пользуются такими инструментами, а у начинающих очень скоро возникает желание избавиться от них как от надоедливого напоминания о былой некомпетентности. Однако вечные середняки, составляющие большинство, окружены этими недостатками вечно.

Оптимизируйте для середняков.

Наша цель состоит не в том, чтобы угождать начинающим, – равно как и не в том, чтобы подгонять середняков быстрее становиться экспертами. Наша цель тройственна: быстро и безболезненно переводить начинающих в середняки, не создавать препятствий на пути середняков, желающих стать экспертами, и – самое главное – следить за тем, чтобы вечные середняки, оставаясь в своей части спектра навыков, были счастливы.

Необходимо больше времени уделять тому, чтобы наши программы были мощными и простыми в использовании именно для пользователей из разряда вечных середняков. Нужды начинающих и экспертов, безусловно, следует учитывать – но только не ценой создания неудобств для самого крупного сегмента пользователей. В оставшейся части главы мы опишем ряд базовых стратегий для реализации этой задачи.

Что нужно начинающим

Начинающие, несомненно, – люди чувствительные, и деморализовать новичка очень легко. Однако не следует забывать, что человек *никогда* не ставит перед собой цели оставаться начинающим. Никто не хочет быть новичком. Это лишь обряд инициации, через который должны пройти все. Хорошие программы сокращают этот обряд, не заостряя на нем внимания.

Проектировщику взаимодействия лучше всего представлять себе, что пользователи (особенно начинающие) – люди одновременно очень умные и очень занятые. Им требуется некоторый инструктаж, но не слишком обширный, так что этот процесс должен быть скоротечным и целенаправленным. Если инструктор по лыжам начнет читать лекцию по метеорологии и экологии горных склонов, он растеряет своих слушателей независимо от их способностей к горнолыжному спорту. То, что пользователь желает научиться работать с программой, вовсе не означает, что ему хочется или необходимо знать, как эта программа устроена внутри.

Считайте пользователей людьми очень умными, но очень занятыми.

С другой стороны, умные люди всегда учатся лучше, если видят причины и следствия, так что вы должны дать им некоторое представление о том, почему все работает так, как работает. Чтобы избавиться от противоречия, мы используем ментальные модели. Когда модель представления интерфейса хорошо соответствует пользовательской ментальной модели (как это описано в главе 2), мы обеспечиваем пользователю необходимый уровень понимания, не заставляя его разбираться в модели реализации.

Встречаем новичков

Новый пользователь должен быстро усвоить используемые в программе понятия и уловить ее предназначение, иначе он от нее откажется. Поэтому первоочередная задача проектировщика – убедиться, что программа адекватным образом отражает ментальные модели задач пользователя. Возможно, между сеансами работы с программой пользователь не будет помнить, какая в точности команда требовалась для работы с определенным объектом, но, если концептуальная структура интерфейса соответствует его ментальной модели, он определенно запомнит такие важные понятия, как отношения между объектами и действиями.

Переход начинающих в разряд середняков требует дополнительной помощи со стороны программы, но эта дополнительная помощь станет для них обузой, как только они достигнут цели. Отсюда следует, что любая предлагаемая вами дополнительная поддержка не должна быть

фиксированной частью интерфейса. Она должна уметь исчезать тогда, когда необходимость в ней отпала.

Стандартная встроенная справка – неподходящий способ поддержки начинающих. Более подробно о справочной системе мы поговорим в главе 26, но сейчас стоит сказать, что ее основное назначение – быть источником справочной информации, а начинающим нужна не справочная информация, им нужна обзорная информация, такая как «Знакомство с программой» (guided tour).

«Знакомство с программой» как отдельная функция, представленная в диалоговом окне, – отличное средство для представления обзорной информации о программе, ее назначении и возможностях. Когда пользователь запускает программу, ему демонстрируется диалоговое окно, содержащее сведения об основных целях и возможностях программы и перечисляющее ключевые инструменты. Если этот справочный материал сосредоточен вокруг вопросов, возникающих у новичков, таких как цели и возможности продукта, и обходит стороной темы, актуальные для середняков и экспертов (об этом чуть ниже), он будет адекватной помощью начинающим.

Новички часто полагаются на меню при изучении и исполнении команд (в главе 22 мы подробно обсудим, почему это так). Каким бы медленным и тяжеловесным инструментом ни были меню – они полны и подробны, и это дает чувство уверенности. Открываемые командами меню диалоговые окна (если таковые имеются) также должны содержать (краткие) пояснения и удобную кнопку отмены (Cancel).

Что нужно экспертам

Эксперты – это также важнейшая группа, поскольку они оказывают непропорционально большое влияние на менее опытных пользователей. Потенциальный покупатель, рассматривая ваш продукт, будет больше полагаться на мнение эксперта, чем на мнение середняка. Если эксперт сказал: «Продукт не очень хороший», – то, возможно, имел в виду: «Продукт не очень хороший для экспертов». Однако начинающий не знает об этом и последует совету эксперта – хотя совет, возможно, неадекватен.

Экспертам временами требуются экзотические возможности, причем некоторые из этих возможностей они могут использовать очень часто. В то же время эксперты определенно выступают за более быстрый доступ к регулярно используемым инструментам из рабочего набора, который может быть довольно большим. Иначе говоря, экспертам нужны короткие пути ко всему.

Любой, кто пользуется программным продуктом по нескольку часов в день, очень быстро усваивает особенности интерфейса. Вопрос о том, *хотят* ли пользователи заучивать часто используемые команды, не ставится: это попросту неизбежно. Высокая частота использования программы одновременно оправдывает запоминание и требует его.

Эксперты все время активно ищут новую информацию о связях между своими действиями с одной стороны и поведением и внешним видом программы – с другой. Экспертам нравятся новые мощные функции. Их уровень владения программой позволяет им не испытывать беспокойства по поводу возрастающей сложности.

Что нужно вечным середнякам

Вечным середнякам нужен доступ к инструментам. Им не нужно объяснять назначение и возможности программы – они уже все это знают. Всплывающие подсказки (см. главу 23) – превосходная идиома для вечных середняков. Всплывающие подсказки ничего не говорят о назначении, смысле и возможностях – они лишь наикратчайшим образом обозначают функцию, занимая минимальное экранное пространство.

Вечные середняки знают, как пользоваться справочными материалами. Они обладают мотивацией копнуть глубже и научиться – при условии, что не потребуется осваивать сразу большие объемы информации. Это означает, что встроенная справка – инструмент для вечных середняков. Они пользуются ею посредством предметного указателя, так что эта часть справки должна быть очень хорошо проработана.

Вечные середняки разделяют функции на регулярно используемые и применяемые лишь изредка. Пользователь может экспериментировать с непонятными возможностями, но вскоре он выявит – вероятно, подсознательно – рабочий набор часто используемых инструментов и будет требовать, чтобы инструменты из этого набора были размещены на самом видном месте в пользовательском интерфейсе – там, где их будет легко найти и запомнить.

Вечные середняки обычно знают о существовании дополнительных возможностей – хотя могут не нуждаться в них и не представлять себе, как с ними работать. Однако знание о наличии таких возможностей добавляет вечному середняку уверенности, убеждает его, что он совершил правильный выбор, когда сделал ставку на эту программу. Лыжника со средними навыками может очень вдохновлять знание о том, что уже вот за теми деревьями лежит действительно крутой экстремальный спуск для экспертов, – даже если он не планирует когда-либо им воспользоваться. Это задает идеал, к которому можно стремиться и о котором можно мечтать, а также создает ощущение первоклассности лыжного курорта.

Код вашей программы обязан учитывать как потребности абсолютных новичков, так и все возможные ситуации, с которыми может столкнуться эксперт. Но не позволяйте этому техническому требованию влиять на проектные решения. Да, вы должны обеспечить экспертов такими возможностями. Да, вы должны обеспечить поддержку для начинающих. Но большую часть своих талантов, времени и ресурсов вы обязаны отдать проектированию наилучшего взаимодействия для самой представительной части аудитории – вечных середняков.

17

Новый взгляд на файлы и операцию сохранения

В мире цифровых технологий мышление, основанное на моделях реализации, сильнее всего проявляет свою мерзкую сущность в области управления файлами и в понятии «сохранение». Если вы когда-нибудь пытались научить свою маму обращаться с компьютером, то знаете, что слово *трудно* является недостаточно выразительным для того, чтобы передать всю серьезность проблемы. Начало не сулит ничего плохого: «Запусти текстовый редактор и набери письмо». Это вполне понятное для нее действие – то же самое, что писать на бумаге. Но, закончив писать, вы щелкаете по кнопке Закрыть – и появляется диалоговое окно с вопросом: «Сохранить изменения?» Вы оба раздосадованы. Она смотрит на вас и спрашивает: «Что это означает? Все ли в порядке?»

Эта проблема возникает по вине программного обеспечения, которое заставляет людей думать подобно компьютерам, безо всякой необходимости вынуждая их бороться с внутренними механизмами хранения данных. Это является проблемой не только для вашей мамы – даже опытные пользователи могут путаться и совершать ошибки. Люди тратят тысячи долларов на устройства и программы, а в результате вынуждены отвечать на вопросы вроде этого: «Вы действительно желаете сохранить документ, над которым трудились полдня?»,– и постоянно помнить о команде Сохранить как..., когда требуется всего лишь поработать с копией документа.

По опыту авторов, люди считают файловые системы компьютеров – механизмы хранения файлов приложений и данных на жестких дисках – очень сложными для понимания и применения. Это один из самых важных механизмов в компьютерах, и ошибки в работе с ним имеют малоприятные последствия. Большинство людей не понимают, в чем разница между оперативной памятью и дисковой памятью, но, к сожалению, традиционный способ проектирования программных

продуктов заставляет пользователей – и даже вашу маму – осваивать эту разницу и думать о своих документах исходя из того, как устроен компьютер.

Популяризация веб-приложений и других программ на основе баз данных создала отличную возможность для того, чтобы отказаться от необходимости думать в терминах модели реализации файловой системы компьютера. К сожалению, здесь возникла другая проблема – всякий раз когда пользователь изменяет документ или настройки, он обычно должен щелкнуть по кнопке Отправить или Сохранить изменения, и это снова вынуждает его думать о том, как работает система, – в данном случае в терминах клиент-серверной архитектуры.

В этой главе описан иной способ взаимодействия, относящегося к файлам и сохранению, – способ, лучше гармонирующий с ментальными моделями пользователей ваших продуктов. К счастью, не мы одни думаем в этом направлении.

Что не так с сохранением файлов?

Каждая запущенная программа одновременно существует в двух местах: в памяти и на диске. То же самое справедливо в отношении любого открытого файла. На сегодня это просто обязательное положение вещей – существующая технология применяет различные механизмы для быстрого доступа к данным (оперативная память) и для долгосрочного хранения этих данных (диски). Однако большинство пользователей этого не осознают. Наши ментальные модели (если не говорить о программистах) иные: существует один документ, который мы непосредственно создаем и в который вносим изменения.

Когда открывается диалоговое окно «Сохранить изменения?», изображенное на рис. 17.1, пользователь подавляет приступ страха и смущения и щелкает по кнопке Да в силу привычки. Диалоговое окно, на которое пользователи всегда реагируют одинаково, избыточно. Его следует удалить из интерфейса.

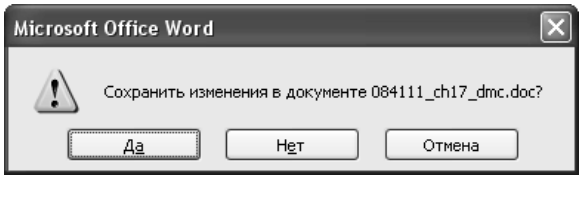


Рис. 17.1. *Вопрос, который редактор Word задает, когда вы закрываете файл с внесенными изменениями. Это диалоговое окно – результат того, что программист навязывает несчастному пользователю модель реализации файловой системы. Оно настолько не соответствует ожиданиям пользователей-новичков, что они зачастую непреднамеренно выбирают вариант «Нет»*

Диалоговое окно сохранения изменений исходит из неверного предположения о том, что сохранение изменений и отказ от сохранения – равновероятные варианты поведения пользователя. Диалоговое окно считает оба варианта одинаково важными, хотя кнопка Да нажимается на несколько порядков чаще кнопки Нет. Как говорилось в главе 10, здесь возможность перепутана с вероятностью. Пользователь может ответить *«нет»*, но почти всегда отвечает *«да»*. Ваша мама думает: «Если бы эти изменения были мне не нужны, разве я оставила бы их в документе?». Вопрос о необходимости сохранить изменения звучит для нее абсурдно.

Еще одна странность этого диалогового окна: почему оно спрашивает вас о необходимости сохранить изменения только тогда, когда вы уже закончили работу? Почему этот вопрос не был задан непосредственно после внесения изменений? Связь между закрытием документа и сохранением изменений отнюдь не естественна, хотя опытные пользователи и привыкли к ней.

Приложение открывает диалоговое окно сохранения изменений, когда пользователь закрывает файл или выходит из приложения, потому что настало время, когда программа должна согласовать копию документа, находящуюся в памяти, с копией, хранящейся на диске. Технология реализации этой функциональности связывает сохранение изменений с закрытием файла или программы, однако пользователь не видит никакой связи. Когда мы выходим из комнаты, мы не рассматриваем вариант отмены всех перестановок, которые в ней сделали. Когда мы ставим книгу на полку, мы не стираем заметки, которые сделали на полях.

Набравшись опыта, мы привыкаем использовать это окно в целях, для которых оно не предназначалось. Когда нет простого метода отменить большое количество изменений, мы пользуемся диалоговым окном сохранения изменений и отвечаем «нет». Если вы вдруг обнаруживаете, что внесли значительные изменения не в тот файл, вы вызываете это диалоговое окно в качестве запасного выхода для возврата к первоначальному состоянию. Это удобно – но все же это трюк. Существуют лучшие способы решения проблемы (например, функция возврата к определенной версии – Revert).

В чем же реальная проблема? Файловые системы в операционных системах современных персональных компьютеров (Windows XP, Windows Vista или Mac OS X) технически превосходны. Источником проблемы вашей мамы является ошибка разработчиков, которые скрупулезно перенесли прекрасную модель реализации на пользовательский интерфейс.

В реальности многим приложениям нет нужды иметь функции управления документами или файлами. Созданные компанией Apple приложения iPhoto и iTunes дают пользователя богатую и простую в применении функциональность, позволяя обычному человеку игнорировать

факт существования файлов. Список проигрывания в iTunes можно создать, изменять, раздавать, переносить на iPod, хранить годами, несмотря на то, что пользователь ни разу не сохранял его явным образом. Фотографии переносятся с фотоаппарата в iPhoto, после чего их можно сортировать, показывать, отправлять по электронной почте, печатать – и при этом пользователям никогда не приходится задумываться о файловой системе.

Проблемы модели реализации

Файловая система компьютера – это инструмент управления данными и программами на диске. Речь идет не только о крупных жестких дисках, где хранится большая часть вашей информации, но также о flash-дисках, компакт-дисках CD-R и DVD-R и сетевых дисках. Программы Finder в Mac OS и Проводник (Explorer) в Windows графически представляют файловую систему во всей ее красе.

Управление дисками и файлами не входит в список целей пользователей.

Хотя файловая система – это внутренний механизм, который не должен затрагивать пользователей, ее влияние повсеместно сказывается на пользовательском интерфейсе большинства приложений. Соскользнуть к мышлению в модели реализации очень легко, потому что в своих поисках более совершенных решений для работы с файловой системой проектировщики взаимодействия сталкиваются со сложными задачами. Особенности реализации файловой системы непосредственно влияют на нашу способность создавать полезные и осмысленные интерфейсы управления версиями документов, отношениями между документами и даже процедурной инфраструктурой наших приложений. Это влияние, вероятно, никуда не исчезнет, если мы не приложим усилия, чтобы изменить положение дел.

В настоящее время большинство программных продуктов обращаются с файловой системой точно так же, как и оболочка операционной системы (Проводник, Finder). Это все равно что заставлять автолюбителя обращаться с машиной так, как с ней обращается автомеханик. Хотя такой подход неудачен с точки зрения взаимодействия, он является стандартом де-факто, и попыткам улучшить ситуацию препятствует значительное сопротивление.

Закрытие документа и отказ от нежелательных изменений

Давнишние пользователи компьютеров привыкли думать, что закрытие документа является подходящим моментом для отказа от нежелательных изменений, внесенных по ошибке или просто для того, чтобы пова-

лять дурака. Это неверно, потому что наилучшим моментом для отказа от изменений является момент, когда они сделаны. Существует проверенная идиома, поддерживающая такой подход: функция отмены.

Сохранение документа

Во многих приложениях при первом сохранении документа или выполнении команды Сохранить как... из меню Файл вы получаете диалоговое окно Сохранение документа. Типичный пример такого окна приведен на рис. 17.2.

Это диалоговое окно выполняет две функции – функцию именования файла и функцию выбора папки для сохранения файла. Обе функции требуют от пользователя глубокого знания файловой системы и умения предсказывать, откуда файл будет удобно открывать впоследствии. Пользователь должен уметь придумывать допустимые и запоминающиеся имена файлов и разбираться в навигации по иерархии папок. Многие пользователи, разобравшиеся с именованием файлов, оставили все попытки разобраться с деревом папок. Они помещают документы на Рабочий стол или в каталог, предлагаемый приложением по умолчанию. Рано или поздно какое-нибудь неаккуратное действие

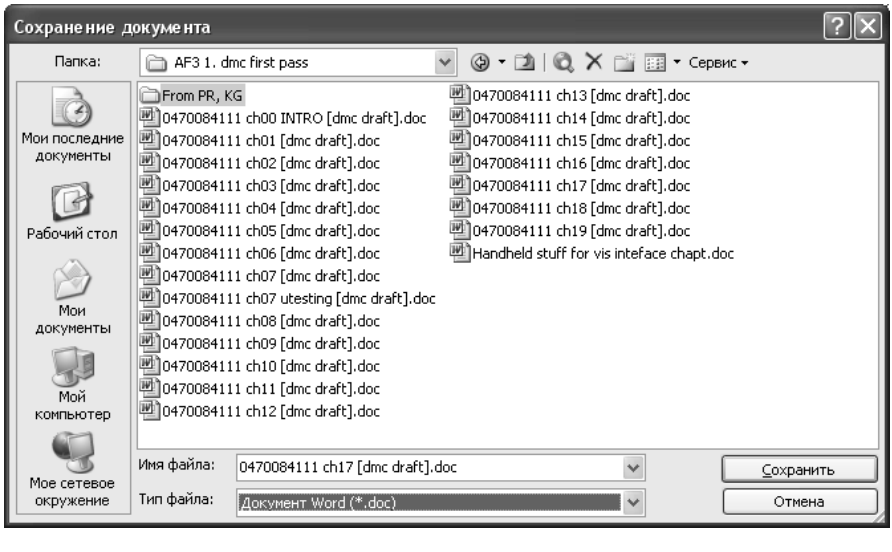


Рис. 17.2. *Диалоговое окно «Сохранение документа» выполняет две функции: оно позволяет дать файлу имя и поместить файл в папку, выбранную пользователем. Однако пользователи не понимают, что такое сохранение, и заголовок диалогового окна не соответствует их ментальной модели этой функции. Более того, если диалоговое окно позволяет вам именовать и размещать документ, вы, возможно, ожидаете, что оно позволит переименовать и переместить документ. К сожалению, наши ожидания разрушаются неудачной реализацией*

заставляет приложение забыть об этом умолчании – и пользователям приходится для поиска файлов вызывать специалиста.

Диалоговому окну «Сохранение документа» следует определиться со своим предназначением. Если смысл его существования – именование и размещение файлов, то оно плохо справляется с этой задачей. Назвав и определив файл, пользователь больше не может поменять его имя и местоположение – по крайней мере, с помощью этого диалогового окна, претендующего на функциональность, связанную с именованием и размещением файлов. Не может пользователь переименовать и переместить файл и с помощью какого-либо другого инструмента программы. В Windows XP это диалоговое окно вообще-то позволяет переименовывать файлы – но только не те, с которыми вы работаете в данный момент. Каково, а? Новичкам не повезло больше всех; опытные пользователи осваивают длинный и трудный путь: закрыть документ, открыть приложение Проводник, переименовать файл, вернуться в основное приложение, из меню Файл воззвать к диалоговому окну Открытие документа и открыть переименованный документ заново.

То, что пользователя вынуждают открыть Проводник, чтобы переименовать документ, не самое большое из зол, но здесь присутствует скрытая ловушка. Все начинается с приманки: Windows без проблем поддерживает одновременную работу нескольких приложений. Привлеченный этой возможностью, пользователь пытается переименовать файл в Проводнике, не закрыв документ в приложении. Этот вполне разумный поступок приводит в действие капкан – и стальные челюсти безжалостно смыкаются на ноге пользователя. Он получает грубый отпор в виде сообщения об ошибке, изображенного на рис. 17.3. Попытка переименовать открытый файл является нарушением правил совместного доступа, и операционная система отвергает ее, выдавая высокомерное сообщение об ошибке.

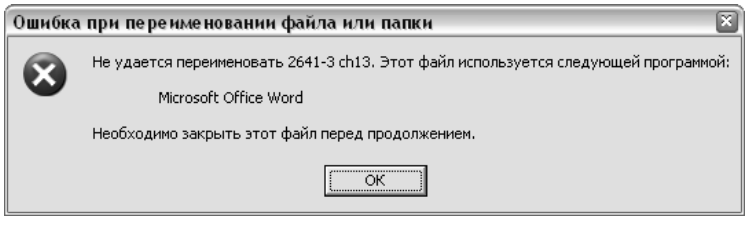


Рис. 17.3. *Если пользователь попытается с помощью Проводника переименовать файл, открытый в редакторе Word, выяснится, что Проводник слишком глуп, чтобы справиться с проблемой. Кроме того, он настолько груб, что выдает высокомерное сообщение об ошибке. Новичок, получивший резкий отпор и от редактора, и от операционной системы, может вообразить, что документ вообще невозможно переименовать*

Невинный пользователь всего лишь пытается переименовать свой документ, но обнаруживает, что заблудился в лабиринте тайн операционной системы. По иронии судьбы единственная сущность, одновременно отвечающая за этот открытый документ и обладающая достаточными полномочиями для его переименования, – то есть само приложение – отказывается даже от попытки сделать это.

Архивирование

Не существует никакой явной функции для создания копии, или архивирования, документа. Пользователь должен делать это с помощью диалогового окна «Сохранение документа», но в этой операции много непонятного. Если пользователь уже сохранил файл под именем «Альфа», то должен явным образом вызвать диалоговое окно «Сохранение документа» и поменять название. Файл «Альфа» закрывается и сохраняется на диске, а «Альфа (новый)» остается открытым и доступным для редактирования. Это действие не имеет смысла с точки зрения единственности реального документа и готовит пользователю весьма неприятную ловушку.

Вот вполне вероятный сценарий, который приводит к неприятностям. Предположим, пользователь работал с документом «Альфа» последние двадцать минут и теперь желает создать архивную копию на диске перед внесением значительных экспериментальных изменений в оригинал. Он вызывает диалоговое окно «Сохранение документа» и меняет имя на «Альфа (новый)». Программа помещает документ «Альфа» на диск, позволяя редактировать документ «Альфа (новый)». Однако пользователь за время работы ни разу не сохранял документ «Альфа», так что документ записан на диск без изменений, внесенных в течение последних двадцати минут! Эти изменения существуют лишь в документе «Альфа (новый)», который сейчас находится в памяти компьютера и с которым работает приложение. Приступив к его модификации в полной уверенности, что результат предыдущей работы хранится в файле «Альфа», пользователь подвергнет изменениям единственную копию этой информации.

Каждый знает, что можно «ввернуть» шуруп в доску при помощи молотка и забить гвоздь плоскогубцами, но опытный мастер понимает, что использование неподходящего инструмента в конце концов «выйдет боком». Либо сломается инструмент, либо будет безнадежно испорчена работа. Диалоговое окно «Сохранение документа» является неподходящим инструментом для создания копий и управления ими – и пострадает в конечном счете именно пользователь.

Модель реализации против ментальной модели

Модель реализации файловой системы входит в противоречие с ментальной моделью, которой придерживаются почти все пользователи.

Большинство пользователей представляют себе файлы похожими на реальные бумажные документы и наделяют их поведением реальных объектов. Говоря простыми словами, пользователи полагают, что в отношении всех документов справедливы два факта: во-первых, документ всегда один, а во-вторых, он принадлежит пользователю. Модель реализации файловой системы нарушает оба этих правила. Когда документ открыт, существуют две копии, и обе принадлежат открывшему приложению.

Как уже говорилось, каждый файл с данными, каждый документ и каждая программа во время работы с ними на компьютере присутствуют в двух местах одновременно – на диске и в оперативной памяти. В то же время пользователь представляет себе документ в виде книги или блокнота на полке. Время от времени блокнот берется с полки, и в него добавляются новые записи. Существует только один экземпляр блокнота: он находится либо на полке, либо в руках пользователя. При работе на компьютере диск служит полкой, а память, где происходит редактирование, эквивалентна рукам пользователя. Но в мире компьютеров блокнот не исчезает с полки – создается его копия, и именно она находится в оперативной памяти. Когда пользователь вносит изменения в документ, он на самом деле изменяет копию в оперативной памяти, в то время как оригинал хранится на диске нетронутым. Когда пользователь заканчивает работу и закрывает документ, программа оказывается перед выбором: либо заменить оригинал измененной копией, либо отбросить копию со всеми изменениями. С позиции программиста, рассматривающего все возможные варианты, события могут развиваться по любому из двух сценариев. С точки зрения модели реализации оба варианта равноправны. Но с точки зрения пользователя вообще не требуется никакого решения: он внес изменения и теперь откладывает документ в сторону. В реальном мире он взял бы блокнот с полки, сделал бы какие-то записи и положил его обратно на полку. Представьте себе полку, которая вдруг стала интересоваться, хочет ли он сохранить добавленные записи!

Прощаемся с моделью реализации

В этот момент серьезные читатели-программисты, вероятно, заерзали на своих местах. Они думаю, что мы замахнулись на святое. Непорочная копия на диске чудесна, и нам лучше воздержаться от призывов избавиться от нее! Спокойно: с реализациями наших файловых систем все в порядке (хотя мы с нетерпением ожидаем прихода более совершенного индексирования в систему Windows). Мы всего лишь должны скрыть от пользователя факт их существования. Мы по-прежнему можем предлагать ему все преимущества этой дополнительной копии, не разрушая при этом его ментальную модель.

Если мы будем визуализировать файловую структуру в соответствии с ментальной моделью пользователей, мы добьемся многого. Во-пер-

вых, работа пользователей станет более продуктивной. Если им не придется тратить силы и умственную энергию на управление файловой системой своих компьютеров, они смогут лучше сосредоточиться на своих задачах. И, разумеется, им не придется повторно выполнять многочасовую работу, потерянную из-за ошибки в сложной шахматной партии, связанной с управлением версиями файлов в современной операционной системе.

Во-вторых, мы научим наших мам действительно хорошо обращаться с компьютерами. Нам больше не придется отвечать на недоуменные вопросы о необъяснимом поведении интерфейса. Мы просто покажем программу и расскажем, как она позволяет работать с документом. И сообщим маме, что по завершении работы она сможет сохранить документ на диске точно так же, как если бы положила блокнот на полку. Наше осмысленное объяснение не будет прервано диалоговым окном сохранения изменений. Между прочим, мамы – типичные представительницы рынка массовой компьютерной продукции. Они владеют компьютерами и используют их, но не любят их, не доверяют им и используют их неэффективно.

Еще одним достоинством такого подхода будет то, что проектировщикам взаимодействия больше не придется учитывать в своих продуктах топорные особенности файловой системы. Мы сможем структурировать команды приложений в соответствии с целями пользователей, а не в соответствии с потребностями операционной системы. Отпадет необходимость называть самый левый пункт в строке меню словом «Файл». Это название – постоянно торчащие из-за фасада приложений уши технологии. Далее в этой главе мы обсудим некоторые альтернативы.

Изменение названия и содержания меню Файл противоречит давно установленному, хотя и неофициальному стандарту. Но выгода от этого изменения перевесит возможные отрицательные последствия. Опытным пользователям определенно придется приложить некоторые усилия, чтобы привыкнуть к новым идиомам, но эти усилия будут гораздо меньше, чем можно предположить. Ведь эти люди уже продемонстрировали свои способности к обучению и свою терпимость, сумев понять и освоить модель реализации. Им не составит труда изучить более удачную модель, и продуктивность их труда не упадет. Новички же немедленно почувствуют значительные преимущества этой модели. Мы, компьютерные специалисты, уже забыли, как высока гора, на которую мы взошли, но новички, которые только подходят к подножию Эвереста компьютерной грамотности, оказываются обескуражены. Если мы хоть что-то сделаем для уменьшения той высоты, на которую они вынуждены взбираться, это совершенно изменит положение вещей, и такой шаг позволит им в дальнейшем покорять гораздо более опасные вершины.

Проектирование с унифицированной файловой моделью

Правильно спроектированный программный продукт должен всегда обращаться с документом как с единственным экземпляром, а не как с двумя копиями, одна из которых хранится на диске, а вторая в памяти. Такой подход мы назовем **унифицированной файловой моделью**. В этой модели пользователям никогда не приходится сталкиваться с внутренними механизмами компьютера: запись данных в память и на диск – задача файловой системы.

Стандартный комплект инструментов управления файлами в большинстве приложений включает в себя команды Открыть, Сохранить и Закрыть, а также связанные с ними диалоговые окна «Сохранение документа», «Сохранить изменения» и «Открытие документа». Как мы убедились, все вместе они создают путаницу при выполнении одних задач и абсолютно непригодны для выполнения других. Ниже мы излагаем альтернативный подход к управлению документами, более точно соответствующий ментальной модели пользователей.

Помимо представления документа как единственного объекта этот подход включает в себя реализацию ряда операций, ориентированных на цели пользователя. Каждая такая операция, если у пользователя возникает в ней необходимость, должна быть представлена в интерфейсе самостоятельной функцией.

- Автоматическое сохранение документа
- Создание копии документа
- Создание версий документа
- Именование и переименование документа
- Размещение и перемещение документа в файловой системе
- Указание формата хранения документа
- Отмена отдельных изменений
- Отказ от всех изменений

Автоматическое сохранение документа

Сохранение документа – одно из самых важных действий, которому должен научиться каждый пользователь компьютера. Вызов этой функции означает принятие всех изменений, внесенных пользователем в копию документа, находящуюся в оперативной памяти, и запись этих изменений в копию документа на диске. В унифицированной модели мы упраздняем различия между этими двумя копиями в пользовательском интерфейсе, так что функция Сохранить должна исчезнуть из основных элементов интерфейса. Сказанное *не означает*, что она должна совсем исчезнуть из программы. Это по-прежнему необходимая операция.

Сохраняйте документы и настройки автоматически.

Приложение должно автоматически сохранять документы. Начать следует с того, что, когда пользователь заканчивает работу с документом и вызывает функцию закрытия, программа должна записать все изменения на диск, не требуя от пользователя подтверждения.

В идеальном мире этого было бы достаточно; однако компьютеры и программы дают сбои, иногда пропадает питание, и разнообразные другие непредсказуемые аварийные события могут приводить к потере результатов вашего труда. Если питание отключится до того, как вы закрыли документ, все изменения, внесенные в него, будут потеряны, поскольку он будет стерт из оперативной памяти. Копия на диске сохранится, но могут потеряться результаты нескольких часов работы. Чтобы этого не случилось, приложение должно также сохранять документ через определенные интервалы времени на протяжении всего сеанса работы. В идеале программа должна сохранять малейшие изменения, как только пользователь внесет их, то есть после любого нажатия на клавишу. В большинстве программ это вполне возможно. Другой вариант – накапливать небольшие изменения в памяти и сохранять их через разумные интервалы времени.

Важно, чтобы эта функция автоматического сохранения не влияла на скорость работы пользовательского интерфейса. Сохранение следует сделать фоновой задачей либо выполнять в те моменты, когда пользователь не взаимодействует с приложением. Никто не набирает текст непрерывно. Каждый время от времени останавливается, чтобы собраться с мыслями, перевернуть страницу или сделать глоток кофе. От приложения требуется лишь дождаться момента, когда пользователь прервется на пару секунд, и сохранить файл.

Автоматическое сохранение устроит практически всех пользователей. Однако некоторые (в том числе и авторы этих строк) настолько боятся сбоев и потери данных, что имеют привычку сохранять документ после каждого абзаца, а иногда и после каждого предложения (для чего пользуются комбинацией Ctrl + S). Все программы обязаны обеспечивать возможность сохранения вручную, но не должны *требовать* этого от пользователей.

Создание копии документа

В приложении должна присутствовать явно обозначенная функция Создать копию. Копия должна быть идентична оригиналу, но никоим образом не привязана к нему. Это означает, что последующие изменения в оригинале не влияют на эту копию. Новая копия должна автоматически получать некоторое стандартное имя, например «Копия Альфа», если исходный документа назывался «Альфа». Если документ

с таким именем уже существует, новая копия должна получить имя «Копия Альфа #2». (Другой разумный вариант – «Копия Альфа» и «Вторая Копия Альфа», однако здесь есть одна тонкость: оригиналы и файлы копий в списке файлов, по умолчанию отсортированном по алфавиту, не будут расположены рядом.) Копии следует помещать в тот же каталог, что и оригинал.

Существует соблазн сопровождать команду копирования диалоговым окном, но работа пользователя не должна прерываться. Приложение должно выполнять свою работу тихо, эффективно и разумно, не досаждая пользователю вопросами вроде: «Вы уверены, что хотите сделать копию?» С точки зрения пользователя это – простая команда. Если возникнут непредвиденные обстоятельства, программа должна самостоятельно принять конструктивное решение.

Именование и переименование документа

Имя документа должно отображаться в заголовке приложения. Если пользователь решит переименовать документ, у него должна быть возможность щелкнуть по имени и отредактировать его по месту. Что может быть проще и естественнее?

Размещение и перемещение документа в файловой системе

Большинство обрабатываемых документов уже существует. Их требуется открывать, а не создавать с нуля. Это означает, что их расположение в файловой системе уже известно. Хотя мы задумываемся о папке для документа в момент его создания или первого сохранения, ни одно из этих событий не является важным вне модели реализации. Новый файл следует поместить в какое-то разумное место, где пользователь сможет впоследствии найти его, например на Рабочий стол.

Помещайте файлы туда, где пользователи смогут их найти.

Конкретное место зависит от пользователей и типа проектируемого продукта. Для сложных монопольных приложений, ежедневно используемых людьми, иногда бывает уместно определять особое место для хранения связанных с этими приложениями документов, но в случае временных приложений и редко используемых монопольных приложений не стоит прятать файлы пользователей в укромных уголках файловой системы.

Если пользователь захочет явным образом указать место документа в файловой иерархии, он может запросить эту возможность через меню. Тогда появится диалоговое окно «Перемещение документа» с выделенным именем текущего документа. В этом диалоговом окне (более

удачно названном варианте окна «Сохранение документа») у пользователя будет возможность переместить документ туда, куда он пожелает. Приложение, таким образом, автоматически распределяет файлы, а диалоговое окно служит лишь для их перемещения.

Указание формата хранения документа

В нижней части существующего диалогового окна «Сохранение документа», изображенного на рис. 17.2, присутствует дополнительный раскрывающийся список, позволяющий пользователю указать формат хранения файла. Эта возможность не должна находиться в данном месте. Привязка формата к действию сохранения файла без нужды усложняет эту операцию. В редакторе Word, если пользователь без задней мысли изменит формат, операция сохранения и все последующие попытки закрыть документ будут сопровождаться пугающим и неожиданным диалоговым окном. Переопределение физического формата файла происходит относительно редко, а сохранение файла – дело обычное. Эти две функции не следует объединять.

С точки зрения пользователя физический формат документа, будь то RTF, ASCII или формат редактора Word, является характеристикой документа, а не файла на диске. Указание формата не должно быть увязано с действием сохранения документа на диск. Оно более уместно в диалоговом окне «Свойства» (документа), доступном через пиктограмму, расположенную рядом с именем файла документа.

Интерфейс этого диалогового окна должен ясно доводить до сознания пользователя тот факт, что выполнение функции чревато потерей важных данных.

В случае с графическими программами, где возможность сохранения изображений в различных форматах весьма желательна, для этой цели подходит диалоговое окно «Экспорт» (уже присутствующее в некоторых графических редакторах).

Отмена отдельных изменений

Если пользователь непреднамеренно внесет в документ изменения, которые должны быть аннулированы, ситуацию легко исправить с помощью уже существующей функции отмены (более подробно поведение отмены мы описывали в главе 16). Не следует прибегать к услугам файловой системы для создания суррогата отмены. Файловая система может служить механизмом реализации этой функции, но отсюда никак не следует, что сама функция должна быть представлена пользователю в терминах файловой системы. Идея непосредственного обращения к файловой системе для отмены изменений противоречит смыслу функции отмены.

Функция создания версий, описанная далее в этой главе, демонстрирует, как ориентированный на файлы подход к функции отмены может быть реализован в согласии с унифицированной файловой моделью.

Отказ от всех изменений

Хотя это редкий сценарий, но нужно оставлять пользователю возможность отказаться от всех изменений, сделанных после открытия или создания документа, и нам определенно не помешает предоставить ему такую возможность. Вместо того чтобы ставить пользователя перед необходимостью изучать файловую систему для достижения этой цели, предоставьте ему в главном меню функцию отказа от всех изменений.

Еще один хороший способ представить данную операцию – предложить функцию возврата к определенной версии, действие которой основано на системе версий, описанной ниже. Поскольку функция отказа от изменений предполагает значительную потерю данных, вы должны защитить пользователя недвусмысленными предупреждениями. Желательно, чтобы сама эта функция могла быть отменена, что сравнительно легко реализовать.

Создание версий документа

Создание **версии** аналогично команде копирования. Разница заключается в том, что версия находится в ведении приложения и для пользователя выглядит как все тот же документ. Пользователям должно быть понятно, что они могут вернуться к тому состоянию, которое документ имел на момент создания версии. Пользователь должен располагать возможностью видеть список версий и сопутствующую статистику, такую как время создания версии и ее размер. Одним щелчком пользователь может выбрать версию и сделать ее активным вариантом документа. Документ, который был активным на момент выбора версии, сам становится одной из версий. Кроме того, раз дисковое пространство сегодня не в дефиците, имеет смысл создавать версии регулярно, даже если пользователям это не приходит в голову.

Новое меню Файл

Наше новое меню Файл теперь выглядит так, как показано на рис. 17.4, и вот его функции:

- Функции Создать и Открыть остались прежними.
- Функция Закрыть закрывает документ без диалоговых окон и прочей суеты, предварительно сохранив все изменения.
- Команды Переименовать и Переместить открывают диалоговое окно, позволяющее пользователю переименовать текущий файл или переместить его в другую папку.
- Команда Создать копию создает новый файл, содержащий копию текущего документа.
- Команда Печать открывает единое диалоговое окно управления печатью.

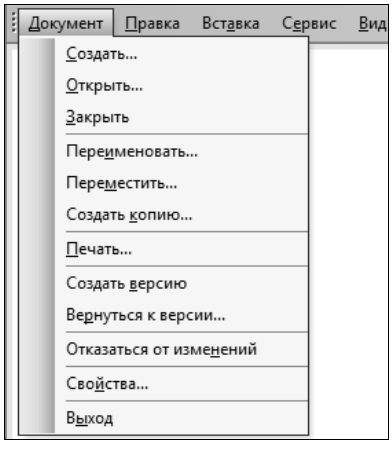


Рис. 17.4. *Пересмотренное меню Файл лучше отражает ментальную модель пользователя и не следует модели реализации, выбранной программистом. Файл существует в единственном экземпляре и принадлежит пользователю. Если пользователь захочет, он сможет создать версию или копию, переименовать файл, отменить все изменения или изменить формат файла. Ему больше не нужно разбираться в тонкостях и беспокоиться о соответствии копии в оперативной памяти и копии на диске*

- Команда Создать версию аналогична команде копирования, но приложение занимается сопровождением версий посредством диалогового окна, открывающегося по команде Вернуться к версии.
- Команда Отказаться от изменений отменяет все изменения, внесенные в документ с момента его открытия.
- Команда Свойства открывает диалоговое окно, позволяющее пользователю изменить физический формат документа.
- Команда Выход работает как обычно, закрывая документ и приложение.

Новое название для меню Файл

Теперь, когда у нас есть унифицированная модель хранения документа, заменяющая собой двойственную модель реализации (хранение на диске и в оперативной памяти), отпадает необходимость давать крайнему левому пункту главного меню приложения название Файл, отражающее модель реализации, а не пользовательскую модель. Существуют две разумных альтернативы.

Можно назвать это меню в соответствии с типом документа, с которым работает приложение. Например, в программе обработки электронных таблиц можно назвать его Таблица, а в программе, генерирующей счета, его можно назвать Счет.

Вторым вариантом является общее название Документ. Это разумный выбор для таких приложений, как текстовые процессоры, программы работы с электронными таблицами или графические редакторы. Однако он вряд ли подойдет для более специализированных нишевых приложений.

В противоположность этому те немногие программы, которые все-таки представляют содержимое дисков в виде файлов, такие как оболочки операционных систем и вспомогательные программы, должны иметь меню Файл, поскольку они обращаются с файлами как с файлами.

Индикация состояния

Если файловая система демонстрирует пользователю файл, который нельзя модифицировать из-за того, что он открыт другим приложением, она должна недвусмысленно указать на это обстоятельство. Достаточно пометки рядом с именем файла или красного цвета шрифта. Новый пользователь по-прежнему может нарваться на сообщение об ошибке (см. рис. 17.3), но у него, как минимум, будет визуальная подсказка о том, почему возникла ошибка.

В существующей модели в двух вариантах существуют не только файлы с данными, но и запущенные приложения. Когда пользователь запускает текстовый редактор через меню Пуск, на панели задач появляется кнопка приложения Word. Но если пользователь возвращается в меню Пуск, приложение Word там по-прежнему доступно! С точки зрения пользователя случилось вот что: он вытащил из ящика инструментов молоток – и обнаружил, что в ящике с инструментами по-прежнему лежит молоток.

Вероятно, такое поведение системы менять не следует: поддержка нескольких копий одного приложения – одна из сильных сторон компьютера. Но программное обеспечение должно облегчить пользователям понимание этого неинтуитивного действия. Скажем, в меню Пуск можно каким-то образом ссылаться на уже запущенное приложение.

Являются ли диски и файловые системы важным конструктивным элементом?

С точки зрения пользователя необходимость дисков ничем не обусловлена. С точки зрения инженера-электронщика есть три причины для использования дисков:

- диски дешевле твердотельной памяти;
- диски не теряют записанную на них информацию после выключения питания;
- диски являются физическим средством переноса информации с одного компьютера на другой.

Вторая и третья причины, конечно, серьезны, но указанные качества не являются исключительными характеристиками дисков: носители, созданные по другим технологиям, справляются с этими задачами не хуже. Существуют разные типы энергонезависимой памяти, которые не теряют записанную на них информацию после выключения питания. Некоторые типы памяти сохраняют данные при очень слабом электрическом питании и даже без него. Компьютерные сети и телефонные линии применяются для передачи данных на другие компьютеры – и часто с меньшими затратами, чем при использовании съемных дисков.

Первая причина, стоимость, является действительным объяснением существования дисков. Энергонезависимая твердотельная память намного дороже дисков.[1] К сожалению, у жестких дисков множество недостатков по сравнению с оперативной памятью. Они намного медленнее, чем твердотельные носители. Они еще и менее надежные, поскольку обладают движущимися частями. Они обычно потребляют больше электроэнергии и занимают больше пространства. Однако самая большая проблема дисков такова: компьютер, а точнее его центральный процессор, не может напрямую читать информацию с диска или записывать ее туда. Вспомогательные программы должны перенести информацию в оперативную память, прежде чем процессор сможет что-либо делать с этой информацией. Когда процессор закончил работу, вспомогательные программы переносят данные обратно на диск. Таким образом, обработка с использованием дисков неизбежно на несколько порядков медленнее и сложнее, чем работа с одной лишь памятью.

Диски – технологический трюк, а не конструктивный элемент.

Плата за использование дисков, проявляющаяся в виде медленной работы и повышенной сложности, настолько высока, что ничто, кроме их сравнительно ничтожной стоимости, не может заставить нас пользоваться ими. Диски не делают компьютеры лучше, мощнее, быстрее или проще в обращении. Наоборот: они делают компьютеры слабее, медленнее и сложнее. Диски – компромисс, так сказать, «ослабление» архитектуры компьютеров, основанной на твердотельной памяти. Если бы проектировщики компьютеров имели финансовую возможность использовать оперативную память вместо дисков, они бы сделали это без колебаний. Фактически они это уже делают в новейших моделях коммуникаторов и портативных компьютеров, в которых применяется память Compact Flash и другие твердотельные технологии памяти.

---

[1] Лавинообразный выход на рынок дешевых компактных ноутбуков с твердотельными дисками (так называемых нетбуков) свидетельствует о том, что эта разница перестала быть значимой. – *Примеч. ред.*

Дисковая технология наложила свой отпечаток на проектирование программного обеспечения, но это было сделано исключительно для удобства реализации, а не для улучшения обслуживания пользователей и не по какой-то другой причине, связанной с целями пользователей.

Время перемен

В пользу программного обеспечения, реализованного в соответствии с моделью файловой системы, можно привести лишь два аргумента: наши программы уже проектируются и создаются подобным образом, и пользователи к ним привыкли.

Ни один из этих аргументов не является весомым. Первый вообще несерьезен, поскольку новые приложения, использующие унифицированную файловую модель, могут свободно сосуществовать со старыми программами, придерживающимися модели реализации. Файловая система, лежащая в основе, вообще не меняется. Подобно тому, как панели инструментов быстро наводнили интерфейсы большинства программ в последние годы, унифицированная модель может быть реализована с успехом и под аплодисменты пользователей.

Второй аргумент – более коварный, поскольку те, кто его выдвигает, прячутся за спины пользователей, как за живой щит. Более того, если вы спросите самих пользователей, то они отвергнут новое решение, поскольку ненавидят перемены, особенно когда эти перемены касаются предмета, на изучение которого ушло много сил, в данном случае – файловой системы. Однако пользователи не всегда хорошо предсказывают успех того или иного решения, особенно если оно отличается от всего, что они уже испытали.

В 80-е годы компания Chrysler продемонстрировала покупателям эскизы автомобиля с принципиально новым дизайном – мини-вэна. Публика его единодушно отвергла. Будучи уверенной, что дизайн превосходен, компания не послушала пользователей и выпустила свой Caravan. И оказалась права: те же люди, которые поначалу отвергли дизайн, не только помогли этой модели стать одним из самых продаваемых мини-вэнов, но и сделали мини-вэн самым популярным автомобильным архетипом со времен появления кабриолета.

Пользователи не являются проектировщиками интерфейса, и от них нельзя ожидать понимания эффектов сдвига интерфейсной парадигмы. Однако рынок показывает, что люди с радостью отказываются от неудобного и плохо спроектированного программного продукта в пользу более простого в обращении, даже если не понимают причин, послуживших источником изменений.