

ДЖЕФ РАСКИН

ИНТЕРФЕЙС

НОВЫЕ НАПРАВЛЕНИЯ В ПРОЕКТИРОВАНИИ КОМПЬЮТЕРНЫХ СИСТЕМ



THE HUMANE INTERFACE

NEW DIRECTIONS
FOR DESIGNING INTERACTIVE SYSTEMS

JEF RASKIN



ADDISON-WESLEY

ИНТЕРФЕЙС

НОВЫЕ НАПРАВЛЕНИЯ В ПРОЕКТИРОВАНИИ
КОМПЬЮТЕРНЫХ СИСТЕМ

ДЖЕФ РАСКИН



Санкт-Петербург-Москва
2007

Джеф Раскин

Интерфейс: новые направления в проектировании компьютерных систем

Перевод Ю. Асотова

Главный редактор
Зав. редакцией
Научный редактор
Редактор
Художник
Корректурa
Верстка

*А. Галунов
Н. Макарова
А. Михайлов
М. Буйневич
С. Борин
С. Беляева
Н. Гриценко*

Раскин Д.

Интерфейс: новые направления в проектировании компьютерных систем. – Пер. с англ. – СПб: Символ-Плюс, 2007. – 272 с., ил.
ISBN 5-93286-030-8

Это уникальное руководство по разработке интерактивных систем, отражающее опыт и взгляды Джефа Раскина, создателя проекта Apple Macintosh, должен прочитать каждый, кто задумывается об интерфейсе.

Сейчас много говорят об эффективности современных подходов к разработке интерфейсов. Раскин же демонстрирует, что многие из них ведут в тупик, и для создания компьютеров, с которыми было бы проще работать, требуются совершенно новые принципы разработки. Он объясняет, как осуществить эти необходимые сегодня изменения, и высказывает нестандартные идеи, демонстрируя дальновидность и способность к практическому взгляду на вещи. Эта книга, рассказывающая о научном подходе к разработке интерфейсов, может быть полезна как для создателей программного обеспечения, так и для руководителей проектов.

ISBN-13: 978-5-93286-030-4

ISBN-10: 5-93286-030-8

ISBN 0-201-37937-6 (англ)

© Издательство Символ-Плюс, 2003

The Humane Interface. New Directions for Designing Interactive Systems by Jef Raskin, Copyright © 2000, All Rights Reserved. Published by arrangement with the original publisher, Pearson Education, Inc., publishing as ADDISON WESLEY LONGMAN.

Интерфейс: новые направления в проектировании компьютерных систем. Джеф Раскин © 2000, все права защищены. Публикуется по соглашению с оригинальным издателем Pearson Education, Inc. (ADDISON WESLEY LONGMAN).

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 22.05.2007. Формат 70х100¹/₁₆. Печать офсетная.

Объем 17 печ. л. Доп. тираж 1000 экз. Заказ N

Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

*Мы угнетены нашими электронными рабами.
Эта книга посвящается нашему освободителю.*

Оглавление

Предисловие	9
Благодарности	11
Введение. Важность основ	15
1. Предпосылки	19
1.1. Определение интерфейса	20
1.2. Простое должно оставаться простым	20
1.3. Ориентация на человека и на пользователя	21
1.4. Инструменты, которые препятствуют новым идеям	22
1.5. Разработка интерфейса как часть общего цикла разработки	23
1.6. Определение человекоориентированного интерфейса	25
2. Когнетика и локус внимания	27
2.1. Эргономика и когнетика: что мы можем и чего не можем	27
2.2. Когнитивное сознательное и когнитивное бессознательное	29
2.3. Локус внимания	35
3. Значения, режимы, монотонность и мифы	53
3.1. Терминология и условные обозначения	53
3.2. Режимы	57
3.3. Модели «существительное-глагол» и «глагол-существительное»	81
3.4. Видимость и состоятельность	85
3.5. Монотонность	89
3.6. Миф о дихотомии «новичок—эксперт»	92
4. Квантификация	97
4.1. Количественный анализ интерфейса	97
4.2. Модель скорости печати GOMS	98
4.3. Измерение эффективности интерфейса	109
4.4. Закон Фитса и закон Хика	119

5. Унификация	125
5.1. Унификация и элементарные действия	127
5.2. Каталог элементарных действий	129
5.3. Имена файлов и файловые структуры	145
5.4. Поиск строк и механизмы поиска	152
5.5. Форма курсора и методы выделения	162
5.6. Позиция курсора и клавиша <LEAP>	166
5.7. Ликвидация приложений	169
5.8. Команды и трансформаторы	173
6. Навигация и другие аспекты человекоориентированных интерфейсов	181
6.1. Интуитивные и естественные интерфейсы	182
6.2. Улучшенная навигация: ZoomWorld	184
6.3. Пиктограммы	202
6.4. Способы и средства помощи в человекоориентированных интерфейсах	209
6.5. Письмо от одного пользователя	224
7. Проблемы за пределами пользовательского интерфейса	229
7.1. Более человекоориентированные среды программирования	230
7.2. Режимы и кабели	234
7.3. Этика и управление разработкой интерфейсов	237
Заключение	244
А. Однокнопочная мышь: история и будущее	246
В. Теория работы интерфейса для SwyftCard	250
Библиография	253
Алфавитный указатель	257

Предисловие

В любом деле, где мы используем компьютеры, какую-то долю времени мы всегда тратим на то, чтобы наладить их работу. Но если бы мой садовник тратил столько же времени на починку садового совка, сколько у нас уходит на мороку с нашими компьютерами, то я лучше купил бы ему другой, хороший совок. По крайней мере, вы можете купить хороший совок.

Эразмус Смамс

Создание интерфейса во многом напоминает строительство дома. Никакая внутренняя или внешняя отделка не спасет сооружение, если его фундамент не положен надлежащим образом. В книге, которую вы держите в руках, рассматриваются когнитивные принципы, лежащие в основе процесса взаимодействия между машиной и человеком и позволяющие ответить на главный вопрос: почему один интерфейс является удачным, а другой — нет. Существующие сегодня графические пользовательские интерфейсы, например те, которые применяются в операционных системах Windows и Macintosh и архитектура которых основана на самой операционной системе и прикладных программах, являются изначально порочными. Для того чтобы сделать компьютеры более удобными и эффективными в использовании, требуется совершенно иной подход. В этой книге описываются некоторые принципиальные недостатки пользовательских интерфейсов и предлагаются пути их преодоления.

Приемы, описанные здесь, могут быть применены к широкому кругу продуктов, включая веб-сайты, прикладное программное обеспечение, ноутбуки и другие устройства для работы с информацией, а также операционные системы. Однако эту книгу нельзя рассматривать как обзор вопросов, связанных с разработкой интерфейсов «человек-машина». Скорее, она открывает новые направления в этой сфере и в то же время затрагивает уже существующие области разработки интерфейсов, на основе которых можно создавать что-то новое.

Если мы собираемся преодолевать проблемы, присущие сегодняшним «человеко-машинным» интерфейсам, нам необходимо усвоить все то, что здесь изложено; однако этого *недостаточно*. Многие важные аспекты разработки интерфейсов не упоминаются в этой книге, поскольку они хорошо освещены в других работах. Данная книга была задумана как дополнение к существующим (или как введение к будущим) подходам к проблемам разработки интерфейсов.

Книга рассчитана на следующий круг читателей:

- Веб-дизайнеров и менеджеров, которые хотят, чтобы их сайты обладали особой легкостью в использовании, что привлекает аудиторию и помогает пользователям находить требуемую им информацию и покупать необходимые товары
- Разработчиков новых программных продуктов и менеджеров, ответственных за их выпуск, которые желают научиться создавать сайты или продукты, отличающиеся простотой в изучении и использовании, а также первоклассным набором дополнительных компонентов, что в совокупности делает эти продукты привлекательными для потребителей
- Менеджеров компаний, которые настаивают на том, чтобы их продукты не требовали больших затрат на обслуживание и не вынуждали пользователей часто обращаться за помощью к службам поддержки
- Программистов, которые разрабатывают интерфейсы (а кто из них сегодня не занимается этим?) и хотят узнать, как сделать свой труд более ценным
- Менеджеров по информационным технологиям, которым необходимо иметь представление о том, какие свойства интерфейса могут снизить затраты на его изучение и обеспечить его продуктивность
- Потребителей, которые желают знать, чего ждать от разработчиков в будущем и что является неверным в том подходе, который используется сегодня при разработке программного обеспечения
- Студентов, изучающих вычислительную технику и когнитивную психологию, которые хотят узнать, что лежит в основе опыта разработки интерфейсов

И наконец, эта книга также окажется полезной для исследователей, занимающихся проблемами разработки «человеко-машинных» интерфейсов. Возможно, что после прочтения этой книги они уже никогда не смогут относиться к интерфейсам так же, как относились к ним прежде.

Благодарности

Так дружеский совет врагов сметет.

*Уильям Шекспир
«Генрих VI», акт III, сцена I¹*

Перечислить всех, кто помог мне написать эту книгу, трудно, потому что число этих людей огромно, а мой долг им так велик. Многие мои друзья, коллеги, родственники, критики и некоторые щедрые люди, знакомство с которыми ограничивается только Интернетом, внесли свой вклад в виде идей, критики, предложений и редакторской работы. Прошу извинить меня, если вы помогли мне, но я забыл вас упомянуть или написал ваше имя или звание неверно, и прошу сообщить обо всех таких случаях.

Благодарю сотрудников из Addison Wesley Longman, а именно редакторов, дизайнеров, сотрудников отделов по связям с общественностью, маркетингу и других, которые все, несомненно, отличаются не только особой компетентностью, но и доброжелательностью и терпением. С другой стороны, выбранные ими рецензенты, имена которых мне неизвестны, были особенно безжалостны, за что я им также благодарен.

Среди тех, кто перечислен ниже, – мои друзья, знакомые, коллеги, мой брат, преподаватель игры на трубе, который учит моего сына, приятель, который, как и я, увлекается авиамоделированием, – в общем, список на первый взгляд невероятно обширный. Только некоторые из упомянутых людей являются экспертами в области разработки интерфейсов, но все они прочитали рукопись этой книги и либо сделали к ней важные добавления, либо в течение многих лет помогали мне в размышлениях над нею. Это Дэвид Альзофон (David Alzofon) (тот самый, который нарисовал Квазимодо), Билл Эткинсон (Bill Atkinson), Томас Этвуд (Thomas Atwood), Пол Бейкер (Paul Baker), Джерри Баренхолтс (Jerry Barenholtz), Джон Бумгарнер (John Bumgarner), Дэвид Кулкинз (David Caulkins), Ph.D. Уильям Бакстон (William Bux-

¹ Перевод Е. Бируковой, ПСС в восьми томах. Изд-во «Искусство», 1958, т. 1. – *Примеч. пер.*

ton), Ph.D. Ренуик Карри (Renwick Curry), Роберт Фаулз (Robert Fowles), Ph.D. Джош Гаррет (Josh Garrett), Жан-Франсуа Грофф (Jean-Franzois Groff), Ph.D. Скот Ким (Scott Kim), Катлин Мандис (Kathleen Mandis), Памела Мартин (Pamela Martin), Трой Мей (Troy May), Ph.D. Мириам Мейслер (Miriam Meisler), Дуглас Маккенна (Douglas McKenna), Майкл С. Миллер (Michael S. Miller), доктор медицины (M.D.) Дэвид Мошэл (David Moshal), Эндрю Нильсен (Andrew Nielsen), Якоб Нильсен (Jakob Nielsen), Джули Ососк (Julie Ososke), Иан Паттерсон (Ian Patterson), Ph.D. Майкл Раскин (Michael Raskin), Эразмус Смамс (Erasmus Smums), Спайдер Робинсон (Spider Robinson), Минору Таояма (Minoru Taoyama), Шэй Телфер (Shay Telfer), Йессо Текериан (Yesso Tekerian), Брюс Тоньяццини (Bruce Tognazzini), Дэвид Уинг (David Wing), Ph.D. Терри Уиноград (Terry Winograd), участники местного отделения (BayCHI) исследовательской группы по проблемам взаимодействия человека и компьютера Ассоциации по вычислительной технике (ACM), которые давали мне возможность высказывать свои идеи и обсуждали их, а также студенты Центра компьютерных исследований музыки и акустики Стенфордского университета во главе с его директором Джоном Чоунингом (John Chowning).

Я счастлив, что у меня есть грамотная и в то же время любящая меня жена Линда Блум, дипломированная медсестра. Она никогда не переживала, что написание технических книг – не самый подходящий способ содержания семьи, если только это дело стоило того, чтобы я им занимался. Ее внимание к идеям, целям и отдельным деталям этой книги позволило сделать лучше многие ее страницы. Хотя не в моих силах было выбирать родителей, но они заслуживают особого уважения, ибо научили ценить людей больше, чем вещи, а также помогли мне полюбить искусство так же, как и науку, что и привело в итоге к этой работе. Мой сын Аза, несмотря на свой юный возраст, внес в эту книгу намного больший вклад, чем можно было бы ожидать, включая идеи, редактирование и кропотливую работу над иллюстрациями. Кроме того, необходимо отметить, что он и его сестры проявляли поразительное терпение в то время, как я писал эту книгу. Особенно важную роль в моей жизни играет Л. Роланд Дженайз (L. Roland Genise), мой лучший учитель. В университетские годы он подарил мне два таких качества, как уверенность в своих силах и любовь к математике. Также я счастлив упомянуть имена Брайена Хауэрда (Brian Howard) и Дугласа Уайета (Douglas Wyatt), которые разделили со мной дружбу, философию и музыку и были беспощадными редакторами моих первых работ. В этой книге я позволил себе не согласиться с некоторыми идеями доктора Дональда Нормана (Donald Norman). Несмотря на эти небольшие разногласия, я считаю его работы весьма ценными в данной области. Кроме того, без его критических замечаний и подсказок эта книга вряд ли смогла бы появиться. Я благодарен Биллу Верпланку (Bill Verplank), спокойному и приятному человеку, который способен столь мягко и тактично делать замечания, что только оказавшись но-

гами на полу, вы поймете, что на самом деле из-под вас вытащили ковер. Именно он убедил меня совершенно изменить тон и ориентацию книги, о чем я нисколько не пожалел. Лин Дюпре (Lyn Dupré), жесткий и весьма придирчивый редактор, также среди тех, чье мнение повлияло на окончательную форму этой книги. Советую вам обязательно прочитать ее работу «Ошибки в письменной речи». Многие идеи, часть которых изложена в этой книге, рождались и оттачивались в разговорах и работе с моим другом Джеймсом Уинтером (James Winter), доктором медицины (M.D.) и философии (Ph.D.). Специалист по вычислительной технике Дик Карпински (Dick Karpinski), весьма ироничный человек, который сам говорит, что ему палец в рот не клади, помогал мне в самых разных аспектах: объяснял непонятные технические детали, знакомил меня с важными людьми и ценными книгами или просто заскакивал с какой-нибудь хорошей идеей. Напоследок хочу еще вспомнить Питера Гордона (Peter Gordon), мудрого, настойчивого и, что особенно важно, терпеливого человека, который представлял меня в издательстве Addison-Wesley. Переписка с ним, будучи опубликована, открыла бы всему миру нашу склонность к постоянной игре слов и ужасным каламбурам, что могло бы навеки запятнать наши имена, но пролило бы свет на бесчисленные подробности, сопровождавшие создание такой с виду небольшой книжки.

Благодарю сотрудников корпорации Agfa за цифровую камеру, с помощью которой сделаны некоторые иллюстрации.

Кроме того, благодарю тех читателей, которые первыми предложили сделать отдельные исправления или изменения, принятые впоследствии. Это Эрик Блоссом (Eric Blossom), Джон Бонди (Jon Bondy), Поль Каббидж (Paul Cabbage), Питер Джонс (Peter Jones), Дж. А. Митчел (G. A. Michael), Кэм Митчнер (Cam Mitchner), Рич Морин (Rich Morin), Мартин Портмэн (Martin Portman) и Элизабет Рибба (Elisabeth Riba). Автор также благодарен Рич Морин за поддержку сайта *www.jefraskin.com*.

Введение. Важность основ

Один человек, один компьютер.

*Слоган компании
Apple Computer*

Представьте себе, что вы поднялись на борт сияющего шикарнейшей отделкой авиалайнера, оснащенного просторными, комфортабельными кожаными креслами с целым набором встроенной аудио- и видеотехники; в буфете вас ожидают отличная еда и напитки. Вы садитесь в свое кресло и смотрите в большой, чисто вымытый иллюминатор. Со вздохом предвкушения особенно приятного полета вы протягиваете руку к шкафчику впереди вас, чтобы поглядеть, что там. Сначала вы достаете весьма объемистую бутылку любимого напитка, а затем буклет с описанием этого замечательного воздушного лайнера.

В то время как двери закрываются и идут приготовления к взлету, вы усаживаетесь поудобнее и начинаете читать. Из буклета вы узнаете, что интерьер самолета создан трудами самых лучших в мире дизайнеров, что повара из пятизвездочных отелей лично составляли меню и готовили блюда и что в группу разработчиков самолета не были включены инженеры-авиаконструкторы, поскольку всемирно признанные дизайнеры сделали внешний вид самолета таким, что и без того создается впечатление авиалайнера, способного летать во много раз быстрее, чем любой другой.

Еще в буклете мелким шрифтом сообщается, что путешествие на этом самолете нередко даже в хорошую погоду сопровождается болтанкой и что достаточно регулярно с ним случаются катастрофы. Если же перелет обойдется без этих инцидентов, то в целом, как обещают авторы, ваше путешествие будет комфортным и интересным.

Теперь звук закрывающихся дверей внезапно принимает угрожающее значение, вы теряете спокойствие и чувствуете, что попали в ловушку. Вы начинаете думать, что именно этот рейс обречен и что вы предпочли бы сейчас сидеть в более жестком кресле, без любимого напитка и даже без бокового иллюминатора, лишь бы только самолет был оборудован хорошей и надежной техникой.

Представленная абсурдная ситуация довольно точно описывает суть большинства существующих сегодня «человеко-машинных» интерфейсов. Наши компьютеры и сотовые телефоны оснащены самыми последними моделями чипов и другой электронной начинкой. Современные операционные системы способны радовать глаз великолепными цветными заставками и стремительными трехмерными эффектами. Вы щелкаете по кнопке, и вот! – вы видите, как она движется самым реалистичным образом, слышите, как звук щелчка мыши весьма точно передается с помощью цифрового стереофонического воспроизведения, а затем, как только на экране открывается панель, до ваших ушей доносится чарующее глиссандо арфы.

Но когда вы начинаете пользоваться этой системой, выясняется, что в некоторых случаях она неприятно ограничивает вас своим непредсказуемым поведением. Из тысяч команд, предусмотренных в системе, вам не удастся найти ту, которая нужна в данный момент, а простые стандартные процедуры выполняются бесконечно долго. Программа, приобретенная в прошлом году, вдруг перестает запускаться под улучшенной версией той же самой операционной системы, и вам приходится покупать новую версию программы. Ко всему прочему оказывается, что операционная система имеет свойство время от времени зависать.

В основе разработки хороших интерфейсов лежат некоторые основные принципы, которые на сегодня не являются общеизвестными. И вопрос о необходимости изучения этих принципов не возникает, поскольку кажется, что уже определено, как должны выглядеть и работать интерфейсы: ведь они непрерывно совершенствовались в течение двух десятилетий, основные разработчики программного обеспечения опубликовали руководства по созданию интерфейсов, чтобы обеспечить совместимость между ними, а существующие средства разработки позволяют быстро создавать любые интерфейсы, которые выглядят по-современному – подобно тому, как и упомянутый авиалайнер создавался, чтобы быть *похожим* на безопасный и комфортабельный летательный аппарат.

Все эти интерфейсы неспособны выполнять многие важные для нас задачи. Например, чтобы записать какую-то мысль, вы хотели бы просто подойти к компьютеру или другому устройству для обработки информации и начать набирать ее – без всякой загрузки, без необходимости открывать текстовый процессор, создавать файл, вообще без использования операционной системы. (Мое определение операционной системы звучит следующим образом: «То, с чем приходится возиться перед тем, как начать возиться с программой».) Чтобы добавить к репертуару системы несколько средств для выполнения простых операций, вы не обязаны изучать целую прикладную программу. К сожалению, в разработке интерфейсов изначально было взято неверное направление, и это привело к тому, что уровень их сложности стал неоправдан-

но высоким с точки зрения как технологической, так и логической необходимости.

Миллионы из нас имеют противоречивые отношения с информационными технологиями. Мы не можем жить без них, и в то же время нам трудно жить с ними. Тем не менее, проблема создания удобных и простых технологий имеет свои решения, хотя мы и не можем ими воспользоваться – они станут доступными, только если мы оставим груз прошлого. Привычный вариант интерфейса в виде рабочего стола, ориентированный на работу с прикладными программами, является частью этой проблемы. В этой книге предлагаются некоторые альтернативные варианты. В конце концов, компьютерные проблемы – это не погода, и мы все-таки *можем* что-то сделать для их разрешения.

С учетом широкого распространения Интернета, а также очевидной важности компьютерных продуктов, предназначенных для группового взаимодействия, может показаться странным, что содержание этой книги касается, главным образом, разработки однопользовательских интерфейсов. Одной из причин этого является тот факт, что проблема разработки однопользовательских интерфейсов еще не решена. Но главная причина состоит в том, что качество любого интерфейса в конечном итоге определяется качеством взаимодействия между одним человеком и одной системой – между ней и вами. *Если индивидуальное взаимодействие с некоторой системой не проходит для пользователя легко и комфортно, то в результате этот недостаток негативным образом отражается на качестве работы всей системы, независимо от того, насколько она хороша в других своих проявлениях.*

1

Предпосылки

Нет ничего более невозможного, чем написать книгу, которая бы получила одобрение каждого читателя.

Мигель де Сервантес

В этой главе говорится о распространенном непонимании сущности таких систем, как интерфейсы, а также методов их разработки. Интерфейс – это нечто большее, чем окна, пиктограммы, выпадающие меню и мышь. Необходимость проектирования интерфейса уже на ранних стадиях разработки продукта иногда упускается из виду. Другой фактор, который часто недооценивается, состоит в том, что все мы наделены познавательными аппаратами, имеющими между собой много общего. При разработке интерфейсов следует сперва учесть общие факторы, а потом уже рассматривать индивидуальные различия. Но, к сожалению, существующие на сегодня средства конструирования интерфейсов не позволяют подойти к задаче именно таким образом.

Я не согласен с мнением, что пользоваться компьютерами сложно потому, что с их помощью мы пытаемся делать безнадежно сложные вещи. В действительности независимо от того, насколько сложной является задача, выполняемая тем или иным продуктом, составные части этой задачи все равно должны оставаться простыми. Эта глава заканчивается определением человекоориентированного интерфейса.

1.1. Определение интерфейса

Позвоните по вышеуказанному номеру и испытайте невероятное разочарование от нашей системы голосовой почты.

Надпись под рекламным объявлением одной из марок обуви

В этой книге выражения *интерфейс «человек-машина»* или *интерфейс «человек-компьютер»* я обычно буду сокращать до *пользовательского интерфейса* или просто *интерфейса*. Многие считают, что термин *пользовательский интерфейс* относится только к современным графическим пользовательским интерфейсам (graphical user interface, GUI), основанным на окнах и меню, управляемых с помощью мыши. Например, в одной из статей в журнале «Mobile Office» было сказано: «Уже недалеко то время, когда вам совсем не нужно будет задумываться об интерфейсе, вы будете просто разговаривать со своим компьютером». В ответ на это я мог бы заметить, что системы, управляемые голосом, действительно могут обходиться без окон, но телефонные автоответчики их также не имеют, и, тем не менее, их интерфейсы зачастую оказываются чрезвычайно плохими. Итак, способ, которым вы выполняете какую-либо задачу с помощью какого-либо продукта, а именно совершаемые вами действия и то, что вы получаете в ответ, и является интерфейсом. (См. также Raskin, 1993.)

1.2. Простое должно оставаться простым

Технология – странная вещь. Одной рукой она дает вам великие дары, а другой – наносит удар в спину.

С. П. Сноу (цитата из Jarman, 1992)

Несмотря на рост количества специалистов по разработке интерфейсов, мало кто из потребителей заявляет, что новые продукты, например электронные четырехкнопочные наручные часы, стали проще в использовании, чем несколько десятилетий назад. Если вы скажете, что наручные часы, так же как и компьютеры, сегодня имеют намного большую функциональность (с чем можно согласиться) и что, следовательно, интерфейсы этих устройств должны стать более сложными (что сомнительно), то я позволю себе заметить, что эта сложность неоправданно возникает в отношении даже тех задач, которые раньше удавалось выполнять без усилий. Сложные задачи могут требовать сложных интерфейсов, но это не оправдывает усложнения простых задач. Сравните, например, насколько труднее установить время на электронных наручных часах с четырьмя кнопками, чем выполнить *то же самое* действие на механической модели часов. Простые задачи должны оставаться простыми независимо от уровня сложности всей системы.

Из всех нелепостей, создаваемых абсурдными конструкциями интерфейсов, именно усложнение простого чаще всего оказывается поводом для высмеивания в комиксах или комедийных сценах. Например,

в фильме «Городские жулики» (City Slickers) три товарища гонят стадо коров. Один из героев, его играет Билли Кристал (Billy Crystal), безуспешно пытается, видимо, уже не один час, объяснить друзьям, как с помощью видеомагнитофона записать какую-нибудь программу на одном канале во время просмотра другого. Когда, в конце концов, друзья выходят из себя от длинного и непонятного объяснения, персонаж Кристала с радостью соглашается сменить тему и предлагает вместо этого рассказать, как устанавливать время на часах в том же видеомагнитофоне. Это предложение приводит друзей в ярость, что вызывает смех у зрителя. Комический эффект порождается несоответствием между очевидной простотой задачи и сложностью интерфейса. Если бы лицевая панель видеомагнитофона была снабжена специальными кнопками, расположенными над и под цифрами часов, как это показано на рис. 1.1, тогда мало у кого возникали бы трудности при установке времени.

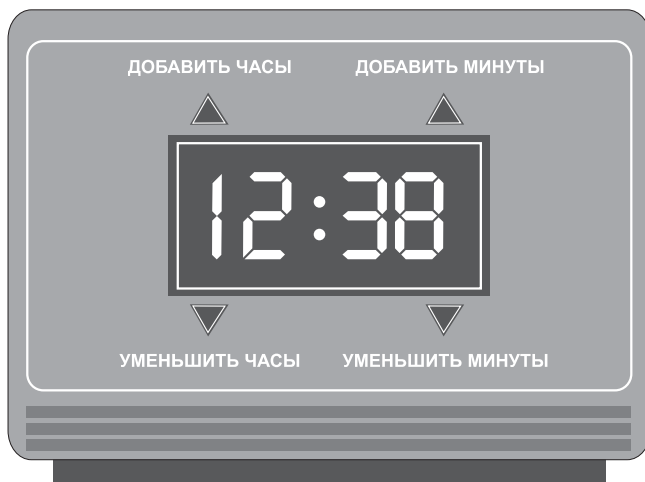


Рис. 1.1. Легко настраиваемые электронные часы для видеомагнитофона. Еще лучшим вариантом были бы часы, в которых время автоматически устанавливается по сигналам точного времени, передаваемым по радио

1.3. Ориентация на человека и на пользователя

Мы слишком усложнили программное обеспечение и забыли главную цель.

Джим и Сандра Сандфорс

Не только разработчики интерфейсов, но и руководители предприятий электронной и компьютерной промышленности понимают необходимость ориентации разработок на нужды пользователей и покупателей. И первым шагом в этом направлении является стремление узнать своего пользователя, что на практике обычно означает обращение за

помощью к специалистам в той или иной области. Специалисты действительно могут хорошо разбираться в особенностях и деталях решаемой проблемы, но их экспертные знания, как правило, не касаются вопросов человеческой психологии. Хотя у пользователей могут быть разные потребности в зависимости от конкретной задачи, тем не менее, в целом они проявляют много общих ментальных характеристик. Прежде чем приступать к разработке самой программы или пытаться учесть различия между отдельными пользователями, разработчики интерфейса могут облегчить свой труд, сосредоточив внимание на том, что является общим для всех людей с точки зрения требований к интерфейсу. По завершении этой стадии разработчики интерфейса уже могут приступить к согласованию различий между отдельными пользователями и группами пользователей и, в конечном итоге, к поиску оптимального варианта, удовлетворяющего широкому диапазону требований пользовательских задач. Однако этот первый важный шаг, во время которого проект интерфейса приводится в соответствие с общими законами психологии, в процессе разработки обычно пропускается. Разработчики интерфейсов предпочитают не задумываться об этом и больше полагаются на так называемые «промышленные стандарты». В результате все широко используемые сегодня модели интерфейсов построены без учета закономерностей мышления и поведения человека. Например, почти во всех компьютерных системах файлы должны иметь собственные имена. Между тем часто возникают ситуации, когда нам трудно вспомнить, под каким именем мы сохранили файл полгода назад. (Одно из возможных решений этой проблемы обсуждается в разделе 5.3.) Таким образом, мы хотим, чтобы программное обеспечение было простым и понятным, своим безупречным поведением показывая нам, что его создатели больше работали над удобством использования, нежели над привлекательным внешним видом своего продукта.

1.4. Инструменты, которые препятствуют новым идеям

Создание хороших интерфейсов требует большой и напряженной работы. Считается, что такие известные на рынке инструменты для построения интерфейсов, как Visual Basic и Visual C++, позволяют снизить стоимость разработки и ускорить ее внедрение. Несмотря на все свои полезные свойства, эти инструменты нечасто будут упоминаться в этой книге. Причина состоит в том, что они основаны на традиционных парадигмах и, следовательно, слишком ограничивают ваши возможности. Аналогичным образом принципы создания интерфейсов в таких системах, как Macintosh или Windows, а также часть подходов, предлагаемых в различных книжных изданиях, посвященных разработке интерфейсов, иногда оказываются явно ошибочными – зачастую из-за корпоративной необходимости поддерживать совместимость с ранними версиями интерфейса, а также из предубеждения, что поль-

зователи непременно отнесутся с неодобрением к попыткам отойти от старых, привычных принципов построения интерфейсов. Действительное усовершенствование интерфейсов возможно, если подходы к их разработке будут серьезно пересмотрены. При этом разработчику необходимо найти компромисс между оправданным применением уже устоявшихся парадигм, которые облегчают изучение интерфейса пользователем, и новыми подходами, которые позволяют сделать интерфейс более удобным и практичным. Конечно, в ситуации, когда часто меняется состав группы разработчиков или круг потребителей продукта, стремление придерживаться известных подходов, возможно, было бы лучшим решением. *Но в тех случаях, когда известно, что большая часть времени у пользователей будет уходить на рутинные, повторяющиеся операции, а обучение в то же время не потребует больших затрат, верным решением является разработка интерфейса с максимальной продуктивностью, даже если впоследствии от пользователя потребуются некоторые усилия по его изучению.*

1.5. Разработка интерфейса как часть общего цикла разработки

Применяемые сегодня методы разработки проектов зачастую не считаются с необходимостью разработки интерфейса. Это упущение может быть следствием того, что специалисты по разработке интерфейсов привлекаются к проекту слишком поздно, когда возможности улучшения качества взаимодействия между пользователем и продуктом большей частью уже потеряны. Интерфейсом удобнее всего заниматься именно на начальных стадиях разработки. И если специалисты по интерфейсам привлекаются уже после того, как программное обеспечение спроектировано и определены его инструменты или когда разработка программы уже почти завершена, то их рекомендации могут потребовать переделки всей выполненной работы, что, естественно, является неприемлемым. Когда бюджет проекта уже исчерпан и рабочий план почти завершен, перспектива отказа от большей части или даже всего дизайна и готового кода, конечно, не может вызвать энтузиазма у менеджеров проекта. Так что даже в такой современной книге по управлению проектами, как «UML Toolkit» (Eriksson and Magnus, 1998), не говорится о необходимости рассматривать интерфейс уже на стадии анализа требований к проекту, которую авторы обозначают как первую фазу его разработки. Однако в действительности разработка интерфейса не должна откладываться до стадии технической реализации, которая в плане Эриксона и Магнуса является третьей фазой. *Определив задачу, для которой продукт предназначен, сначала спроектируйте интерфейс, после чего приступайте к его реализации.* Это повторяющийся процесс. Определение задачи будет меняться во время разработки интерфейса. Поэтому весь процесс разработки продукта будет проходить в соответствии с изменениями в задаче продукта и его интерфейсе. Здесь необходимо стремиться к максимальной

гибкости. На первом этапе разработки следует определить, что именно должен сделать пользователь для получения того или иного результата и как система должна отвечать на каждое его действие.

Пользователи не задумываются над тем, как устроена машина, пока она справляется со своими задачами. При этом не имеет значения, какой именно процессор используется и является ли язык программирования объектно-ориентированным, многопоточным или, быть может, называется какими-то другими умными словами. Для пользователей важнее всего удобство и результаты. Но все, что они видят, – это интерфейс. Другими словами, *с точки зрения потребителя именно интерфейс является конечным продуктом.*

Ваше время бесценно, ваша работа священна

Я приучился часто сохранять проделанную работу, чтобы даже в случае системного сбоя не потерять большую часть своего труда. В конце каждого абзаца или даже после нескольких предложений я при помощи сочетания клавиш вызываю команду сохранения. Эта команда создает копию текста на диске, где он может оставаться относительно защищенным от потери в случае сбоя. Приблизительно каждый час я создаю резервную копию своей работы с помощью энергонезависимого запоминающего устройства, которое может быть физически извлечено из компьютера и таким образом защищено от любых неожиданностей в его работе. Кроме того, каждую неделю я сохраняю резервную копию всей системы на внешнем диске. Это не значит, что я параноик, – я всего лишь считаю, что такой подход практичен. Однако необходимости во всех этих сложных процедурах не должно возникать. *Система должна рассматривать все данные, вводимые пользователем, как бесценные.* И если перефразировать Первый закон робототехники Азимова: «Робот не может причинить вред человеку или своим бездействием допустить, чтобы человеку был причинен вред», то первый закон проектирования интерфейсов должен звучать примерно так: «Компьютер не может причинить вред данным пользователя или своим бездействием допустить, чтобы данным был причинен вред».

Работая над этой книгой, я по совету своих редакторов стал использовать опцию, позволяющую либо принять, либо отклонить изменения, внесенные в документ. Каждый раз, сделав несколько изменений, я запускал команду сохранения. Когда произошел сбой системы, я не стал беспокоиться, полагаясь на сделанные мной периодические сохранения. Однако когда я попытался найти файлы с самыми последними изменениями, это не удалось, и мне пришлось делать ту же работу заново. Немного поэкспериментировав, я выяснил, что при включенной опции «принять или отклонить» команда сохранения, подаваемая с клавиатуры, перестает действовать. Однако пользователю никакого предупреждения об этом не дается. В ре-

зультате пропало больше трех часов моего труда, и мне пришлось тратить время на эксперименты и выяснять, что же произошло и как это предотвратить в будущем. Если не считать излишней сложности сегодняшних компьютерных систем, именно такие досадные мелочи говорят о необходимости усовершенствования подходов к разработке интерфейсов.

Наилучшей формулировкой второго закона интерфейса может быть следующее утверждение: *«Компьютер не должен тратить впустую ваше время или вынуждать вас выполнять действия сверх необходимых»*. В разделе 4.3 будет рассматриваться измерение объема работы, необходимого для выполнения той или иной задачи.

1.6. Определение человекоориентированного интерфейса

Можно создать самолет с любыми техническими характеристиками, которые только пожелает Министерство военно-воздушных сил, если при этом не требуется, чтобы он мог летать.

Вилли Мессершмидт (выдающийся немецкий авиаконструктор времен второй мировой войны)

Интерфейс является ориентированным на человека, если он отвечает нуждам человека и учитывает его слабости. Чтобы создать такой интерфейс, необходимо иметь представление о том, как действуют люди и машины. Кроме того, следует развить в себе способность чувствовать те трудности, с которыми сталкиваются люди. И это не всегда просто. Мы настолько привыкли к тому, как работают программы, что соглашаемся принять их методы работы как данность, – даже в тех случаях, когда их интерфейсы неоправданно сложны, запутанны, неэкономны и побуждают людей к ошибкам.

Многие из нас испытывают раздражение, например, от того, что для запуска (иначе говоря, загрузки) компьютера требуется какое-то время. В 1999 году была реклама одного автомобильного радиоприемника со встроенным компьютером, в которой утверждалось, что «в отличие от домашнего компьютера, эта система не заставит вас долго ждать, пока она загрузится». Внимательное изучение шести наиболее серьезных работ в области разработки интерфейсов показывает, что даже в этих книгах, написанных в основном в то время, когда разработке интерфейсов стали придавать важное значение, проблема загрузки не упоминается (Shneiderman, 1987; Norman, 1988; Laurel, 1990; Tognazzini, 1992; Mayhew, 1992; Cooper, 1995). Тем не менее, я уверен, что каждый из названных авторов всецело согласился бы с тем, что сокращение или устранение задержки при запуске компьютера улучшило бы эффективность его использования, тем более что я еще не встречал пользователя, у которого такая задержка не вызывала бы раздраже-

ние. Однако никогда не существовало технической необходимости в том, чтобы компьютер после включения начинал работать более чем через несколько секунд. Наши компьютеры долго загружаются только лишь потому, что многие дизайнеры и разработчики не потрудились сделать интерфейс в этом отношении ориентированным на человека. Кроме того, некоторые люди думают, что если компьютеры с медленной загрузкой продаются миллионами, то это якобы свидетельствует об их высокой производительности.

Нельзя сказать, что проблема долгой загрузки машины всегда игнорировалась. Уже вышедший из употребления Apple Newton, Palm Pilot и другие карманные компьютеры могут запускаться мгновенно, а появление на некоторых компьютерах «спящего режима» – состояния, в котором компьютер потребляет меньше энергии, чем в обычном режиме, и из которого он может быть быстро переведен в рабочее состояние, – это шаг в правильном направлении.

Инженерам удавалось с успехом решать и более сложные проблемы. Например, в ранних моделях телевизоров необходимо было ждать около минуты, пока разогревалась катодная трубка кинескопа. В некоторых моделях инженеры добавили специальную схему, которая поддерживала катодную трубку в теплом состоянии, что позволило сократить время достижения рабочей температуры. (Поддержание катодной трубки в разогретом состоянии потребовало бы большого расхода электричества и уменьшило бы срок ее службы.) В другом варианте был разработан кинескоп с катодной трубкой, которая разогревалась в течение нескольких секунд. И в том и в другом случае интересы пользователя были удовлетворены. В начале двадцатого столетия был создан автомобиль на паровой тяге, называвшийся Стенли Стимер (Stanley Steamer). Несмотря на все свои очевидные достоинства, этот механизм не имел успеха из-за одного недостатка: чтобы тронуться с места, от момента зажигания до достижения необходимого давления в котле требовалось подождать 20 минут.

Принцип разработки, согласно которому программные продукты не должны вынуждать пользователя ждать без необходимости, можно считать очевидным и ориентированным на человека. Таким же является и стремление не подгонять пользователя. В общем виде этот принцип можно было бы сформулировать следующим образом: *«Ритм взаимодействия должен устанавливаться самим пользователем»*.

Не требуется обладать большими техническими знаниями, чтобы понять, что большая пропускная способность коммуникационных линий может ускорить передачу веб-страниц. Однако другие взаимосвязи иногда бывают не столь очевидны. Поэтому для разработчиков интерфейсов «человек-машина» важно знать внутренние механизмы технологии. В противном случае у них не будет возможности оценивать достоверность утверждений, высказанных, например, программистами или специалистами по аппаратной разработке относительно осуществимости тех или иных элементов интерфейса.

2

Когнетика и локус внимания

Он плакал и раздражался, но все было тщетно.

Доминик Манчини, говоря не о «зависшем» компьютере, а об Эдуарде V, короле Англии. «Occupatione Regni Anglie per Riccardum Tercium (1483)». Цитата приводится в книге Алисон Уир «Принцы в башне» (1992)

При всей сложности компьютеров и других продуктов современной технологии «машинная» часть интерфейса «человек-машина» легче поддается пониманию, чем человеческая – намного более сложная и изменчивая. Тем не менее, многие (возможно, даже очень многие) факторы человеческой производительности не зависят от возраста, пола, культурного происхождения или уровня компетентности пользователя. Эти свойства человеческой производительности и способности к обучению имеют непосредственное отношение к основам разработки любого интерфейса. В частности, тот факт, что мы имеем один локус¹ внимания, оказывает влияние на многие аспекты разработки интерфейсов «человек-машина».

2.1. Эргономика и когнетика: что мы можем и чего не можем

Познай себя!

*Надпись возле Дельфийского Оракула.
Плутарх «Морали»*

Используйте машину или инструмент в соответствии с их возможностями и ограничениями, и они сослужат вам хорошую службу. Разра-

¹ Локус – место, положение (лат.) – *Примеч. науч. ред.*

батывайте интерфейс «человек-машина» в соответствии с возможностями и слабостями человека, и вы поможете пользователю не только справиться с работой, но и сделать его более счастливым, более продуктивным человеком.

Руководства по разработке продуктов, взаимодействующих с нами физически, обычно содержат конкретную информацию, основанную на свойствах и возможностях человеческого скелета и органов чувств. Совокупность сведений в этой области составляет науку **эргономику**. На основе этих знаний можно проектировать стулья, столы, клавиатуры или дисплеи, которые с высокой степенью вероятности будут удобны для своих пользователей. Тем не менее, нельзя пренебрегать тщательным тестированием разрабатываемых продуктов. Вы не станете проектировать машину, обслуживание которой предусматривает, чтобы один человек оперировал двумя переключателями, расположенными в трех метрах друг от друга. Очевидно, что людей с такими физическими размерами не бывает. Тема эргономики в компьютерной технике, выходящая за пределы данного изложения, рассматривается в обзоре разработок интерфейсов, представленном в книге Мэйхью (Mayhew, 1992, гл. 12). В эргономике учитывается статистическая волатильность параметров человеческого тела. Можно спроектировать автомобильное кресло, подходящее для 95% населения, тогда как остальные 5% потенциальных покупателей автомобиля такие кресла покажутся неудобными. Создание кресла, которое можно было бы регулировать в широком диапазоне, в том числе для редких пользователей с ростом 1 м или еще более редких с ростом 2,5 м, было бы механически невыполнимым или же потребовало бы значительных затрат.

Большая часть машин, созданных нашей цивилизацией, были механическими и взаимодействовали с нами главным образом физически. Соответственно, наши физические ограничения сравнительно легко учесть. Постепенно человеческие изобретения стали иметь все большее отношение к области интеллектуальных задач, нежели физических. *Мы должны овладеть эргономикой сознания, если мы хотим создавать интерфейсы, которые могли бы хорошо работать.* Удивительно, но мы часто не замечаем собственные ментальные ограничения, поэтому для определения границ возможностей нашего сознания мы должны прибегнуть к тщательному наблюдению и экспериментированию.

Изучение прикладной сферы наших ментальных способностей называется когнитивным проектированием, или **когнетикой**. Некоторые когнитивные ограничения очевидны: например, нельзя ожидать от обычного пользователя способности перемножать в уме 30-значные числа за 5 секунд, поэтому нет смысла разрабатывать интерфейс, который требовал бы от пользователя такой способности. Однако мы часто не учитываем другие ментальные ограничения, которые оказывают неблагоприятное влияние на нашу продуктивность при работе с интер-

фейсами «человек-машина», хотя эти ограничения присущи каждому человеку. Интересно отметить, что все известные компьютерные интерфейсы, а также многие некомпьютерные интерфейсы «человек-машина» разработаны с расчетом на некие когнитивные способности, которыми, как показывают эксперименты, мы на самом деле не обладаем. Большая часть трудностей, связанных с использованием компьютеров и подобных устройств, возникает скорее из-за низкого качества интерфейса, чем из-за сложности самой задачи или же недостатка старания или умственных способностей у пользователя.

Когнетика, так же как и эргономика, учитывает статистическую природу различий между людьми. Тем не менее, следует прежде всего рассмотреть сами ограничения, присущие нашим когнитивным способностям, поскольку знания об этих ограничениях пока мало находили практическое применение.

К счастью, нам не придется рассматривать физическую структуру мозга хотя бы потому, что наши сегодняшние знания об этом органе весьма неопределенны. Успешный интерфейс вполне может быть разработан на основе прагматического и эмпирического взгляда на то, что может и чего не может человеческий ум, сколько времени требуется человеческому сознанию и телу на выполнение тех или иных задач и какие условия повышают вероятность совершения ошибки.

2.2. Когнитивное сознательное и когнитивное бессознательное

О, доктор Фрейд, доктор Фрейд, если бы только вы занимались чем-нибудь другим!

Давид Лазар «Доктор Фрейд», 1951

Использование таких терминов, как *сознательное* и *бессознательное*, которые имеют вполне определенное значение в психологии, философии и истории и применяются для описания аспектов функционирования нашего мышления, может вызывать некоторые затруднения. В контексте технического проектирования имеет смысл пользоваться более ограниченными понятиями **когнитивного сознательного** и **когнитивного бессознательного**. Еще более точными были бы термины *эмпирическое сознательное* и *эмпирическое бессознательное*, однако более благозвучные варианты, предложенные Килстромом (Kihlstrom), уже стали общепринятыми (Cohen and Schooler, 1997, с. 137). Понимание того, что мы обладаем этими двумя отдельными наборами ограниченных ментальных способностей, а также того, как они работают во взаимодействии с интерфейсами «человек-машина», важно для разработки интерфейсов в той же степени, в какой знание о размере и силе человеческой руки важно для разработки клавиатуры.

Дадим краткое определение: бессознательными называются те ментальные процессы, которые вы не осознаете в тот момент, когда они

происходят. Когнитивное бессознательное – это не грандиозное мифическое порождение фрейдистской психологии, а явление, которое можно продемонстрировать с помощью простого эксперимента, о котором мы скажем далее. Несмотря на обилие книг, посвященных вопросам и парадоксам сознания, изложенный в этой главе подход, взятый из книги Бернарда Баарса «Когнитивная теория сознания» (Bernard J. Baars. *A Cognitive Theory of Consciousness*, 1988), позволяет избежать дилемм, свойственных данному вопросу, и обращает нас только к тому, что мы можем непосредственно наблюдать и о чем можем делать конкретные выводы. Как сказано в предисловии к этой книге, «в науке применяется одна испытанная временем стратегия: оставить на время философские вопросы и сосредоточиться на тех, которые являются эмпирически разрешимыми». Когнетика – это практическая дисциплина. Хотя теоретические исследования могут разъяснять неясное и, в конце концов, приводить к надежным практическим результатам, мы все же стремимся избегать их до тех пор, пока не почувствуем эти результаты. (Аналогичным образом изучение роста человеческих костей, которое может дать полезные сведения с точки зрения эргономики, все же скорее относится к области физиологии, чем собственно к эргономике.)

Сознание и модели человеческого разума

Хотя трактовка когнитивных свойств сознания, предложенная Баарсом, является полезной, его теория разума, в которой моделями служат современные цифровые компьютерные структуры, имеет уже меньшую ценность. Такой подход, на мой взгляд, сомнителен, особенно по той причине, что мыслители издавна использовали последние достижения в области технологии в качестве моделей понимания человеческого существа, а со следующим шагом в технологическом развитии отказывались от них или же ограничивали их применение.

Дон Норман, когнитивный психолог, хорошо знакомый с компьютерами, говорил о человеческом разуме в терминах вычислительных устройств (Norman, 1981). Дж. Р. Андерсон (Anderson, 1993) еще в 1976 году построил модель ментальных операций на основе так называемых продукций – инструмента, который широко использовался в свое время для описания синтаксиса компьютерных языков. Хотя такие аналогии и могут быть полезны, некоторые ученые склонны придавать чрезмерное значение этой метафоре. Я предполагаю, что в ближайшие несколько лет появится большое количество психологических теорий, использующих архитектуру «клиент-сервер», Интернет и Всемирную сеть с ее гипертекстовыми ссылками и браузерами в качестве моделей работы отдельных аспектов сознания.

Следует с осторожностью относиться к этой тенденции, в противном случае мы можем создать аналогии, которые не будут иметь никакой нейрофизиологической основы. В XVII столетии Вселенная и

ее обитатели часто описывались в терминах часового механизма (Dijksterhuis, 1961, с. 495). Однако на фоне нашего сегодняшнего понимания как часовых механизмов, так и организмов, эта метафора утратила свою яркость. В XIX столетии метафора парового двигателя встречалась во многих философских трактатах о физиологии человека. Сегодня мы знаем, что ценность этой аналогии ограничивается главным образом объяснением метаболизма, поскольку метаболическую систему можно назвать тепловым двигателем организма.

Тема сознания рассматривается также в двух других известных книгах: «Новый разум императора» Роджера Пенроуза (Penrose, *The Emperor's New Mind*, 1989) и «Объясненное сознание» Даниела Денетта (Dennett, *Consciousness Explained*, 1991). Несмотря на то что чтение этих произведений довольно захватывающее, они, к сожалению, не представляют ценности с точки зрения разработки интерфейсов «человек-машина». Обобщая, можно сказать, что без непосредственного подтверждения того или иного примера функционального параллелизма следует избегать слишком буквального понимания метафор, в которых человеческий мозг уподобляется компьютеру.

Поскольку рассуждения о том, что является сознательным или бессознательным, кажутся весьма далекими от наших повседневных забот, попытаемся наглядно продемонстрировать их значение в обычной жизни. Попробуйте ответить на следующий вопрос: какая последняя буква в вашем имени? До тех пор пока вы не прочитали предыдущее предложение, вы, вероятно, не думали об этой букве и ее связи с вашим именем. Вы знаете (и уже давно знали), что это за буква и какое место она занимает в вашем имени, но вы не обращали на это своего внимания. Вы не думали, не размышляли об этом. Или, если пользоваться нашей терминологией, вы не осознавали это. Данная информация не запрашивалась, однако вы смогли ее получить, когда в этом возникла необходимость. То место, откуда была извлечена буква, мы будем называть когнитивным бессознательным. Когнитивное бессознательное – это неизбежно какое-то физическое место, хотя оно должно быть представлено определенными физическими процессами в мозге. Представление о том, что бессознательное и сознательное *должны* соответствовать определенным областям мозга, не совсем верно. Скорее, они *могут* им соответствовать. Изменение состояния в момент осознания буквы – это один из возможных механизмов процесса. Другой возможный вариант – это наличие в мозге определенного указывающего устройства, и перемещение воспоминания или мысли из одной области мозга в другую вызывает также и перемещение этого указателя. Вполне возможно, что мысли и воспоминания распределены по всему мозгу, наподобие того, что происходит при голографической записи изображений.

Мы могли бы рассмотреть и другие возможные механизмы и объяснения, но в этом нет необходимости. Для нас важно сейчас только отме-

тить, что в один момент вы не осознавали, какой буквой заканчивается ваше имя, а в другой момент стали осознавать. Хотя иногда я использую здесь метафору перемещения в пространстве, говоря о мысли как о чем-то перемещающемся из области сознательного в область бессознательного или наоборот, все же не следует понимать это образное выражение как указание на некую модель функционирования мозга. (Возможно, исследования физической структуры мозга смогут подтвердить обоснованность такой модели, но для нас это сейчас не имеет значения.)

Когда я пользуюсь в этой книге терминами *сознательное* и *бессознательное*, то, как правило, подразумеваю под ними понятия когнитивного сознательного и когнитивного бессознательного. Когда вы подумали о последней букве в вашем имени, то эта мысль, вызванная прочитанным предложением, стала частью вашего сознания. Это изменение состояния вашей мысли – от бессознательного к сознательному – показывает, что у вас есть, по крайней мере, две формы знания. Для того чтобы построить науку когнетике, мы должны отказаться от солипсизма¹ в нашем подходе и принять, что другие люди смогли бы сделать те же наблюдения относительно своих ментальных процессов, какие только что сделали и вы.

Такой стимул, как, например, прочтение некоторой части этой книги, может вызвать перемещение какой-то информации или ощущения, или любого другого содержимого вашей памяти или знаний из области бессознательного, где оно хранится, в область сознательного, где вы его осознаете. Обращаете ли вы внимание на ощущения от одежды, в которую вы сейчас одеты? В каких местах она стянута, а в каких сидит свободно? До тех пор пока вы не прочитали эти предложения, направившие ваше внимание на одежду, вы, вероятно, не осознавали того напряжения, которое она оказывает на части вашего тела. Вы также можете вспомнить, например, какое-нибудь приятное недавнее событие, и при этом у вас, возможно, возникнет (перейдя из когнитивного бессознательного в когнитивное сознательное) то эмоциональное состояние, которое сопровождало это событие. Теперь обратите внимание на ощущение, связанное с тем, что вы держите в руках эту книгу (если вы читаете ее в печатном варианте), или с тем устройством, посредством которого вы управляете компьютером (если вы читаете книгу в электронном варианте). Думаю, что проведенные эксперименты помогли вам убедиться в существовании когнитивного бессознательного и когнитивного сознательного, а также в том, что с помощью некоторого стимула можно вызвать переход какой-либо ментальной конструкции из одной области в другую.

Сознательно переживать бессознательные процессы вы не можете по определению. Такой бессознательный процесс, как, например, отсле-

¹ Солипсизм – теория, согласно которой считается, что единственное доступное знание – это знание о себе. – *Примеч. науч. ред.*

живание давления в мочевом пузыре, служит стимулом к тому, чтобы осознанной стала необходимость опорожнить его.

Однако в связи с этим возникает один трудный вопрос: кто здесь подразумевается под местоимением «вы»? Можно ли провести различие между вами и вашим сознанием? Поскольку нам предстоит говорить о техническом проектировании, я обойду этот вопрос стороной и просто скажу, что в нашем случае под словом «вы» будет подразумеваться сочетание вашего физического существа и всех присущих ему физических и ментальных феноменов. Для понимания принципов построения интерфейсов нет необходимости рассматривать вопрос возможного различия между вами, вашим сознательным и вашим бессознательным «я».

Вполне вероятно, что воспоминания или ключи к ним физически хранятся в каких-то областях мозга, поскольку в некоторых случаях непосредственная электрическая стимуляция отделов мозга, которая иногда проводится при хирургических операциях, способна вызвать воспоминание, т. е. сделать это воспоминание осознанным. С помощью стимуляции одной области мозга можно многократно вызывать определенное воспоминание, состояние или ощущение, тогда как с помощью стимуляции другой области можно многократно вызывать другие переживания. Исследования мозга, предпринятые с использованием методики магнитно-резонансного представления (magnetic-resonance imaging, MRI), а также позитронно-эмиссионной томографии (positron-emission tomography, PET), позволяют выявить физическую основу различных ментальных процессов. Эти методики упомянуты в связи с тем, что в будущем они могут оказать непосредственную помощь в разработке и особенно в тестировании интерфейсов. Например, существует обратная корреляция между уровнем поглощения глюкозы определенной зоной мозга испытуемого, что является показателем потребления энергии в данной физической структуре его мозга, и степенью легкости, с которой испытуемый использует тестируемый элемент интерфейса. В будущем измерение параметров мозговой активности может найти более широкое применение в качестве средства тестирования интерфейсов. Тем не менее, рассмотрение указанных методов выходит за рамки предмета этой книги.

Приведенные примеры описывали ситуации, когда ментальные конструкции перемещались из области бессознательного в область сознательного. Следующий пример демонстрирует обратную ситуацию. Внезапный звук или другое неожиданное событие может отвлечь ваше внимание от того, чем вы были заняты в этот момент – например, от чтения книги, – к вопросу о причине этого звука. Скажем, услышав звук падающей с полки лампы, вы подумаете: наверное, это кот туда забрался. После того как вы возвратитесь к чтению, знание о происшедшем событии переместится из вашего когнитивного сознательного в когнитивное бессознательное.

Кроме того, существуют пограничные случаи, а также есть многое, что нам еще не известно о сознательном и бессознательном. Например, каждый сталкивался с ситуацией, когда никак не вспомнить чье-либо имя, хотя оно буквально вертится на языке. Иногда вы все же вспоминаете это имя, и тогда можно сказать, что данное воспоминание становится полностью осознанным. В других случаях воспоминание так и остается ускользающим. Существует ли промежуточное состояние между сознательным и бессознательным? Можно ли зафиксировать состояние сознания в момент, когда информация об имени «движется» из одной области мозга в другую? Можно ли считать «почти вспомненное» имя свидетельством частичной, прерываемой связи, наподобие неплотного, искрящего электроконтакта? Эти вопросы довольно интересны, тем не менее, нам необязательно нужны ответы на них; так же как в космологии – нам известно, что вселенная расширяется, хотя мы и не знаем ответа на вопрос, что послужило причиной этого расширения, т. е. что было до так называемого «большого взрыва», с которого, по мнению большинства космологов, и началось существование нашей вселенной.

Как я уже сказал, мы не станем углубляться в какую-либо из метафор, описывающих работу человеческого мозга. В то же время, мы не можем избежать построения его ментальных моделей. (Кстати, поразительно, что мы строим модели нашего мозга внутри нашего мозга.) Пока же я предлагаю читателю представлять сознательное и бессознательное как некие два отдела. Эти отделы являются не просто разными областями или состояниями, в которых хранятся мысли или воспоминания, но они имеют разные способы взаимодействия с окружающим миром и с воспринимаемыми понятиями. Как было установлено когнитивными психологами за последние сто лет, когнитивное сознательное и когнитивное бессознательное обладают свойствами, которые выходят за пределы нашего знания или незнания о них.

В табл. 2.1 обобщены различия между когнитивным сознательным и когнитивным бессознательным. Из нее видно, что когнитивное сознательное включается в тех случаях, когда вы сталкиваетесь с ситуацией, которая кажется новой или представляет угрозу, или когда вам требуется принять нешаблонное решение, т. е. такое, которое основано на происходящем именно здесь и сейчас. Понять логическое содержание проблемы возможно только в том случае, если вы осознаете наличие этой проблемы. Когнитивное сознательное работает последовательно и может оперировать только одним вопросом или контролировать только одно действие в течение некоторого промежутка времени. Человек может осознавать одновременно от 4 до 8 отдельных мыслей или объектов. Как правило, каждые несколько секунд сознательная память очищается.

Сознательное проявляется при решении ветвящихся задач. Необходимо сказать, что иногда трудно отличить ветвящуюся задачу от невет-

вящейся. Например, торможение по сигналу светофора может относиться и к тому и к другому типу задачи. С одной стороны, если вы просто реагируете на красный свет нажатием педали тормоза, данная задача является неветвящейся и, следовательно, обрабатывается когнитивным бессознательным. С другой стороны, если в момент, когда вы приближаетесь к светофору, на нем загорается желтый сигнал, и поэтому вам требуется принять решение, пересекать ли перекресток без остановки или остановиться, то здесь уже вступает в действие когнитивное сознательное. Пока вы изучаете некоторую задачу, вы можете воспринимать ее как ветвящееся событие, требующее сознательного внимания. По мере повторения задачи ее выполнение может стать неветвящимся и автоматическим. В разделе 2.3 мы начнем рассматривать эти свойства и те следствия из них, которые имеют значение с точки зрения разработки интерфейсов.

Таблица 2.1. Свойства когнитивного сознательного и когнитивного бессознательного

Свойство	Сознательное	Бессознательное
Иницируется	Чем-то новым Нестандартными ситуациями Опасностью	Повторением Ожидаемыми событиями Безопасностью
Используется	В новых обстоятельствах	В привычных ситуациях
Решает задачи	Принятие решений	Работа с неветвящимися задачами
Принимает	Логические утверждения	Логические или противоречивые утверждения
Функционирует	Последовательно	Одновременно
Управляется	Волей	Привычными действиями
Производительность	Небольшая	Огромная
Период функционирования	Десятки секунд	Десятилетия (всю жизнь)

2.3. Локус внимания

Вы можете до некоторой степени контролировать превращение бессознательных мыслей в сознательные, в чем вы убедились, переместив знание последней буквы вашего имени в сознательную область. Однако вы не можете намеренно перевести сознательные мысли в бессознательную область. «Не думай о белом слоне», – шепчет девочка, зная, что мальчик не сможет не думать об этом слоне. Но через некоторое время, если разговор не останавливается на слонах, мысли мальчика об этом животном перейдут в бессознательное. Когда это произойдет, он больше не будет обращать внимание на мысль о слоне – слон перестанет быть локусом внимания.

Здесь я использую термин *локус*, поскольку он обозначает некоторое *место* или *область*. Термин *фокус*, который иногда используется в этом контексте, может вызвать неправильное представление о том, как работает внимание, потому что может быть понят как действие.¹ Когда вы находитесь в бодрствующем и сознательном состоянии, вашим локусом внимания является какая-то деталь или объект окружающего мира или идея, о которой вы целенаправленно и активно думаете. Различие между фокусом и локусом внимания можно понять на примере следующего предложения: «Мы можем целенаправленно сфокусировать наше внимание на каком-либо локусе». Тогда как *фокусировать* означает волевое действие, мы, тем не менее, не можем полностью управлять содержанием локуса нашего внимания. Если вы слышите, как позади вас внезапно взорвалась петарда, ваше внимание будет направлено на источник звука. Слово *фокус* также используется при обозначении объекта, который в данный момент выбран на экране. Ваше внимание может быть – или не быть – направлено на такого рода фокус, когда вы пользуетесь тем или иным интерфейсом. Из всех объектов или явлений окружающего мира, которые вы воспринимаете с помощью своих чувств или воображения, в каждый момент времени вы можете сконцентрироваться только на одном. Чем бы ни был этот объект, деталь, воспоминание, мысль или понятие, он становится локусом вашего внимания. В данном случае имеется в виду не только то внимание, которое можно активно обращать на что-либо, но также и пассивное восприятие потока поступающей информации или простое переживание происходящего.

Вы слышите и видите намного больше того, что становится локусом вашего внимания. Когда вы входите в комнату, чтобы найти какой-то потерянный предмет, он может лежать прямо перед вами и, тем не менее, остаться незамеченным. С помощью оптических средств мы можем установить, что изображение нужного предмета попадало на вашу сетчатку. Возможно даже, что это изображение оказывалось в центральной ямке сетчатки, где четкость зрения наибольшая. Из экспериментов по нейрофизиологии мы знаем, что сигнал, представляющий объект, был сформирован и передан с помощью зрительного нерва. И все же вы не заметили искомый предмет, поскольку он не стал локусом вашего внимания. Прислушавшись, я могу заметить, что лампы дневного света в коридоре рядом с моей комнатой раздражающе жужжат. Но если этого не делать, я не услышу этого звука. Магнитная звукозапись покажет, что звук сохраняется, даже когда я не осознаю его. Чаще всего я замечаю этот звук в тот момент, когда свет включается или выключается. Внезапное начало жужжания обращает на него мое внимание. Внезапное прекращение звука заставляет меня осознать, что я слышал его, причем уже после того, как это происходило. В этом

¹ Английское слово *focus* может быть прочитано и как существительное (фокус), и как глагол (фокусировать, сосредотачивать). – *Примеч. пер.*

случае то, что кажется очень точным воспоминанием о звуке, который я только что слышал, внезапно становится локусом моего внимания. Эксперименты показывают, что образы непосредственного восприятия – то, что психологи называют перцептивной памятью – хранятся в течение небольшого периода времени. Известный феномен инертности зрительного восприятия лежит в основе того эффекта, что отдельные кадры кинофильма превращаются в сплошной визуальный поток. Зрительные образы обычно затухают через 200 мс. Эта величина может варьироваться в пределах от 90 до 1000 мс. Слуховые образы затухают в среднем через 1500 мс (в диапазоне от 900 до 3500 мс) (Card, Moran and Newell, 1983, с. 29–31). Сейчас, сидя за столом, я не могу восстановить жужжание осветителей в том же ярком и непосредственном звучании, как я мог это сделать сразу же после того, как оно прекратилось, и внезапная тишина направила мое внимание на тот факт, что этот звук был. Прошло несколько часов, и впечатление рассеялось, оставив только бледное воспоминание о том раздражающем звуке ламп, похожее скорее на описание, чем на ощущение.

Восприятия не всегда откладываются в памяти. Большинство восприятий утрачиваются после того, как затухают. С точки зрения разработки интерфейсов из быстрого затухания сенсорных восприятий следует, что человек, прочитавший или услышавший 5 секунд назад некоторое сообщение, необязательно сможет вспомнить его содержание. Если такое сообщение важно само по себе или содержит важную деталь, например номер в сообщении «Ошибка 39-152», то оно должно оставаться на экране до тех пор, пока не перестанет быть актуальным (такой подход можно назвать наилучшим), или же необходимо предоставить пользователю возможность немедленно обработать эту информацию, прежде чем она исчезнет из его памяти. Когда некоторая информация становится локусом внимания, она перемещается в кратковременную память, определение которой мы дадим в разделе 2.3.4. Там она будет храниться в течение 10 секунд.

2.3.1. Формирование привычек

Все, что стоит сделать хорошо, сначала стоит сделать плохо.

Дик Карпински

Когда вы выполняете какую-то задачу многократно, то с каждым разом делать это становится все проще. Бег трусцой, настольный теннис или игра на фортепиано – это мои каждодневные занятия. С первой попытки все это казалось мне совершенно невозможным. Ходьба является более распространенным примером. По мере повторения – или *с практикой* – выполнение того или иного действия становится для вас *привычным*, и вы можете выполнять его не задумываясь. У Томаса Льюиса (1974), чьи работы по биологии всегда читаешь с радостью, по этому поводу можно найти следующий отрывок:

Печатать на машинке слепым методом, так же как и ездить на велосипеде или ходить пешком по тропинке, лучше всего получается, если об этом не задумываться. Как только вы задумаетесь, вы можете сбиться. Чтобы совершать известные вам действия, требуется всего лишь расслабить мышцы и нервы, которые отвечают за выполнение каждого отдельного шага, предоставить их самим себе и не вмешиваться в их работу. Конечно, это не означает, что вы отказываетесь от собственной воли, потому что решение о совершении действия остается за вами, и вы можете в любой момент вмешаться, чтобы, например, изменить технику исполнения. Если вы захотите, то можете научиться ездить на велосипеде задом наперед или ходить экстравагантной хромающей походкой, подпрыгивая на каждом четвертом шаге и одновременно насвистывая какую-нибудь мелодию. Но если вы станете концентрировать свое внимание на деталях, на движении каждой мышцы, чуть ли не падая на каждом шаге и в последний момент вовремя выставляя ногу, чтобы все-таки не свалиться, то, в конце концов, вы вообще не сможете двигаться и будете только дрожать от напряжения (с. 64).

Как-то раз один наблюдатель сказал, что бейсболист в момент удара должен думать о своей технике, на что звезда бейсбола Йоджи Берра, в продолжение приведенной мысли Льюиса, ответил с характерной краткостью: «Как можно думать и бить одновременно?» (Kaplan, 1992, с. 754).

Любая привычка означает отказ от внимания к деталям. Тем не менее, привычки необходимы всем высшим формам жизни, представленным на Земле. С другой стороны, жизнь возможна даже при отсутствии какого бы то ни было сознания, как, например, жизнь микробов – по крайней мере, насколько мы знаем или хотя бы можем предполагать. Кроме того, термин *привычка* используется и в отрицательном смысле. Вопреки утверждению Льюиса, что в потере своей воли нет ничего плохого, все же случается так, что развиваются дурные привычки. Привычки бывают настолько сильными, что могут даже превратиться в страсть, иногда достигая того предела, когда сознательный контроль полностью утрачивается. (В данном случае я говорю не о физиологической склонности к чему-либо, например к никотину или опиатам, но скорее о нежелательных приобретенных привычках, таких как кусание ногтей.) Поскольку наше сознание есть то, чем мы, по сути, являемся, то в этой связи я вспоминаю наблюдение, сделанное Унамуно: «Приобрести привычку значит перестать быть» (Unamuno, 1913). Возможно, этим высказыванием Унамуно хотел предупредить нас об опасности пагубных привычек. Что же касается рутинных сторон повседневной жизни, то здесь мы как раз желаем, чтобы наше сознательное внимание «перестало быть».

Вы легко можете себе представить, насколько трудно было бы вести машину, если бы вам пришлось задумываться: «Так, я хочу остановиться. Надо подумать. Чтобы сбросить обороты двигателя, необходимо снять ногу с педали акселератора. Теперь нужно преобразовать кинетическую энергию машины в тепло с помощью нажатия на педаль

тормоза...» К счастью, если вы опытный водитель, все эти операции вы проделываете автоматически. Подобным же образом вы развили много маленьких привычек, которые помогают вам пользоваться компьютером, наручными часами, будильником, телефоном и разными другими вещами, имеющими интерфейсы.

При постоянном использовании какого-либо интерфейса у вас формируются определенные привычки, которые впоследствии трудно преодолеть. В этом смысле задача дизайнеров заключается в том, чтобы создавать интерфейсы, которые не позволяют привычкам вызывать проблемы у пользователей. Мы должны создавать интерфейсы, которые, во-первых, целенаправленно опираются на человеческую способность формировать привычки и, во-вторых, развивают у пользователей такие привычки, которые позволяют упростить ход работы. *В случае идеального человекоориентированного интерфейса доля участия самого интерфейса в работе пользователя должна сводиться к формированию полезных привычек. Многие проблемы, которые делают программные продукты сложными и неудобными в использовании, происходят из-за того, что в используемом интерфейсе «человек-машина» не учитываются полезные и вредные свойства человеческой способности формировать привычки.* Хорошим примером служит тенденция предусматривать сразу несколько путей решения одной и той же задачи. В этом случае множество вариантов приводит к смещению локуса внимания пользователя с самой задачи на выбор пути.

Зачастую невозможно изменить привычку волевым действием. Как бы часто или настойчиво вы не говорили себе не делать то или иное привычное действие, вы не всегда можете остановить себя. Предположим, к примеру, что в следующее воскресенье педали тормоза и газа на вашей машине поменяются местами. Специальная красная лампочка на приборной доске будет сигнализировать вам об этом изменении. Хотя, возможно, вам и удастся проехать несколько кварталов без аварии, тем не менее, большинство из нас в такой ситуации не смогло бы избежать ошибок. Как только ваш локус внимания будет отвлекаться от нововведения в конструкции машины, например, в случае если на дороге окажется ребенок, ваша реакция, обусловленная привычкой, заставит вас нажать не на ту педаль. И даже специальная красная лампочка будет здесь бесполезна. Причина в том, что привычку нельзя изменить однократным волевым действием. Для этого требуется тренировка в течение некоторого периода времени. Разработчик может устроить, в том числе и ненамеренно, ловушку для пользователя, если сделает так, что на одном компьютере будут интенсивно использоваться два или более приложения, интерфейсы которых отличаются только несколькими часто применяемыми деталями. В таких обстоятельствах у пользователя, скорее всего, сформируются привычки, которые будут приводить к ошибкам при попытках применить в одном приложении команды, свойственные другому.

2.3.2. Одновременное выполнение задач

На языке когнитивной психологии любая задача, которую вы научились выполнять без участия сознания, становится *автоматичной*. Автоматизм позволяет выполнять сразу несколько действий одновременно. Все одновременно выполняемые задачи, за исключением не более чем одной, являются автоматичными. Та задача, которая не является автоматичной, естественно, находится непосредственно в локусе вашего внимания. Когда вы выполняете одновременно две задачи, ни одна из которых не является автоматичной, эффективность выполнения каждой из них снижается в результате конкуренции за область внимания. Этот феномен психологи называют *интерференцией*. Чем более предсказуемой, автоматичной и бессознательной становится задача, тем больше становится эффективность ее выполнения одновременно с другими задачами, и, тем менее, она конкурирует с ними (Baars, 1988, с. 33).

Человек, по-видимому, имитирует одновременное выполнение нескольких задач, требующих сознательного контроля, через последовательное переключение внимания с одной задачи на другую (Card, Moran, Newell, 1983, с. 42). Действительная одновременность достигается, когда все задачи, кроме разве что одной, становятся автоматичными. Например, вы можете одновременно не спеша идти, что-нибудь есть и при этом решать какую-нибудь математическую задачу. (В это же время можно бессознательно обдумывать и еще одну математическую задачу, но по определению когнитивного бессознательного вы не заметите этого процесса. Здесь я обращаю ваше внимание только на то, что вы не можете *сознательно* работать над двумя разными математическими задачами одновременно.) Для большинства людей все эти действия, за исключением поиска решения математической задачи, настолько знакомы, что могут выполняться «на автопилоте». Однако если при одновременном выполнении всех этих действий вы внезапно почувствуете какой-нибудь неприятный на вкус кусочек вашей походной еды, вы станете думать только о том, что вы такое съели, тогда как математическая задача перестанет быть вами осознаваемой.

Не менее важным, чем понимание того, что в каждый отдельный момент времени нельзя осознавать более одной задачи, является тот факт, что человек не может избежать формирования автоматических реакций. Эта невозможность не зависит от повторения: никаким количеством повторений нельзя научиться *не* формировать привычки при регулярном использовании того или иного интерфейса. Формирование привычек является неотъемлемой частью нашего ментального аппарата. Его невозможно остановить волевым действием. Наверное, когда-нибудь в субботу утром вы нечаянно приезжали туда, где находится ваша работа, хотя собирались поехать в какое-то другое место. Сделали вы это по привычке, которая сформировалась через повторение определенной последовательности действий. Когда вы учились читать,

то поначалу проговаривали по отдельности каждый слог и обращали внимание на произношение каждой буквы. Теперь же (я надеюсь) вы можете читать без необходимости сознательного контроля над процессом составления слов из букв.

Любая последовательность действий, которую вы регулярно выполняете, становится, в конце концов, автоматичной. Набор действий, составляющих последовательность, становится как бы одним действием. Как только вы начнете выполнять некоторую последовательность, требующую не более 1 или 2 секунд времени, вы не сможете остановиться и проделаете все действия вплоть до завершения последовательности. Вы также не сможете прервать последовательность, выполнение которой занимает больше нескольких секунд, если она не стала локусом внимания. Если возвратиться к примеру с субботним утром, то после того как вы сделали неверный поворот, вы могли внезапно осознать, что собирались ехать совершенно в другом направлении. Это осознание помещает текущую задачу управления автомобилем в локус вашего внимания и позволяет вам прервать автоматичную последовательность действий, которая направила вас к месту вашей работы.

Когда вы повторяете какую-то последовательность действий, единственный способ предотвратить формирование привычки – это *удерживать* то, что вы делаете, в локусе внимания. Это очень сложно. Как обычно говорят, наше внимание «гуляет».

Неизбежность формирования привычек имеет свои следствия и с точки зрения разработки интерфейсов. Например, многие из нас пользовались компьютерными системами, которые перед тем как выполнить необратимое действие, например удаление файла, задают вопрос: «Вы уверены?» После этого вам требуется ввести либо *Y* («Да»), либо *N* («Нет») в качестве ответа. В основе этого лежит идея, что, запрашивая подтверждение вашего решения, система оставляет вам шанс исправить ошибку, которая могла бы быть неисправимой. Эта идея считается общепринятой. В этой связи можно привести, например, строки из книги Смита и Дьюэла (Smith and Duell, 1992), адресованные медицинскому персоналу: «Если вы случайно удалите какую-то часть постоянной записи – что трудно сделать, поскольку компьютер всегда спросит вас о подтверждении операции...» (с. 86). К сожалению, Смит и Дьюэл дают нереалистичную оценку ситуации – ведь вы вполне можете случайно удалить запись, даже если упомянутое подтверждение было получено. Так как ошибки случаются редко, вы обычно будете отвечать *Y* («Да») на любую команду, которая требует подтверждения. Из-за постоянного повторения ввод *Y* после команды удаления вскоре становится привычным действием и, вместо того чтобы остаться отдельной ментальной операцией, превращается в часть действия по удалению файлов. В результате вы, не останавливаясь и не проверяя собственное намерение, вводите *Y*. Таким образом, запрос компьютерной системы, предназначенный служить в качестве меры безопас-

ности, из-за привычки становится бесполезным и только усложняет обычный процесс удаления файлов. Все дело в том, что *любой запрос о подтверждении, требующий установленного ответа, вскоре становится бесполезным*. Разработчики, которые используют такого рода подтверждения, и администраторы, которые думают, что запросы о подтверждении обеспечивают безопасность, на самом деле не учитывают силу свойства формирования привычек, присущего когнитивному бессознательному (см. также раздел 6.4.2).

Более эффективный подход заключается в том, чтобы дать пользователю возможность отменить ошибочную команду, даже если после нее были совершены какие-то другие действия. Формирование привычки всегда подтверждать команду без принятия сознательного решения о ее выполнении в локусе внимания происходит неизбежно, даже если ответ на запрос о подтверждении является не заранее установленным, а непредсказуемым. К примеру, компьютер может потребовать, чтобы пользователь ввел слово, случайно выбранное из диалогового окна, дважды или в обратном порядке (этот выбор также может быть случайным):

Запрашиваемое вами действие не может быть отменено. Оно может привести к полной потере информации в данном файле. Если вы уверены, что хотите полностью удалить эту информацию, введите в обратном порядке десятое слово в этом диалоговом окне.

Такой вид запроса о подтверждении операции можно назвать просто драконовским, и при этом он столь же бесполезен. Любой эффективный способ подтверждения непременно будет раздражать, потому что он препятствует формированию у пользователя привычного ответа на запрос и вообще привыканию к этой процедуре. Если по тем или иным причинам, скажем, связанным с правами доступа, какой-либо файл никогда не должен быть удален пользователем, следует предотвратить саму возможность такого действия. Кроме того, подобные меры каждый раз создают у пользователя новый локус внимания, и поэтому он может забыть подумать о правильности своего решения, тем самым сводя на нет назначение запроса о подтверждении и свои собственные старания.

Не существует идеального способа подтверждения операции. Даже если пользователь будет вводить обоснование удаления, – такой метод особенно подходит для ситуаций, связанных с соблюдением правомочности действий, – это, в конце концов, приведет к тому, что он станет каждый раз выбирать один и тот же стандартный ответ. Если основание для выполнения того или иного необратимого действия было с самого начала неверным, никакое предупреждение или запрос о подтверждении этого действия не поможет пользователю избежать ошибки.

В ловушке автоматизма

Я попался в ловушку собственного автоматизма, когда писал эту главу. Я выделил курсивом слово, потом попытался убрать курсивное начертание. В большинстве текстовых процессоров Macintosh, чтобы привести текст к нормальному стилю, требуется, удерживая клавишу со значком яблока (называемую еще клавишей Command), нажать и отпустить клавишу с буквой T (Command-T). Однако в Microsoft Word команда Command-T изменяет формат абзаца. Если бы вы спросили меня, с какой программой я в тот момент работал – или, другими словами, если бы вы сделали это локусом моего внимания, – я бы ответил, что с Word. Тем не менее, чтобы применить нормальный стиль, я автоматическим (!) выполнил команду Command-T и в результате по ошибке получил изменение формата абзаца. Единственный способ избежать подобных ошибок – это предусмотреть в интерфейсе неизбежность формирования привычек.

2.3.3. Сингулярность локуса внимания

Я не могу думать об X, если я думаю об Y.

*Крис, персонаж из телевизионного шоу
«Northern Exposure», 31 октября 1994 г.*

Важно отметить, что локус внимания может быть только один. Этот факт дает возможность решить многие проблемы разработки интерфейсов. Многие не верят, что у них или у других людей только один локус внимания, но эксперименты, описанные в упоминаемой мной литературе, подтверждают гипотезу о том, что мы не можем обрабатывать несколько раздражителей одновременно. Этот довольно неожиданный факт согласуется с нашими утверждениями об ограничениях когнитивного сознательного и заслуживает более подробного обсуждения.

Как отметил Роджер Пенроуз (1989), «характерной чертой сознательной мысли является ее «единственность» – в противоположность большому множеству разрозненных процессов, происходящих одновременно» (с. 398). Бернард Баарс (1988), общепризнанный лидер в исследованиях когнитивного сознательного, объясняет, что когда людей «просят отслеживать какой-то интенсивный поток информации, они, как правило, не осознают никаких других потоков, присутствующих одновременно с ним, даже если все восприятие производится одним и тем же сенсорным органом. Аналогичным образом в состояниях глубокого раздумья, когда человек поглощен какой-то одной информационной цепью, альтернативные стимулы исключаются из сознания» (с. 33). То есть альтернативные стимулы не становятся локусами внимания.

Это наблюдение подтверждается даже нашей повседневной речью. Например, у нас может быть как одна мысль, так и несколько мыслей, но о внимании мы всегда говорим только в единственном числе. Мы ни-

когда не говорим о *вниманиях* (*attentions*), за исключением случаев, не имеющих отношения к основному значению этого слова, как, например, в выражении *unwanted attentions* – ненужные хлопоты (внимания).¹ Хотя вы и не осознаете другие мыслительные линии, кроме одной – той единственной концептуальной нити, которой вы в данный момент следуете, – какое-то неожиданное событие может отвлечь ваше внимание от этой линии. Я уже описывал случаи, как неожиданные события запускали механизмы сознательного внимания. Здесь необходимо только отметить, что после приобретения нового локуса внимания прежний локус теряется, т. е. второй локус внимания не возникает.

Прерывающее событие не обязательно может быть внешним. Внезапная боль или мысль о том, что пришло время назначенной встречи, может ворваться в ваше когнитивное сознательное и расстроить текущий ход мысли, направив его по новому пути.² Но если внешние и внутренние события обычны и не являются экстренными, ваше бессознательное перехватывает этот процесс, и вы начинаете игнорировать эти события – даже не осознавая, что игнорируете их. Другими словами, при обычных, ординарных обстоятельствах ваше внимание не отвлекается. Вы можете научиться время от времени сознательно отслеживать происходящее в окружающем пространстве, чтобы замечать те события, на которые в противном случае вы не обратили бы своего внимания. Для примера можно упомянуть, что пилотов специально обучают регулярно просматривать приборную доску даже без особой внешней причины. Это действие позволяет пилотам заметить, например, небольшие аномальные отклонения в показаниях приборов (не все инструменты самолета снабжены аварийной сигнализацией). Тем не менее, пилоты часто забывают проводить такой просмотр приборов в тех случаях, когда события во время полета вынуждают их переключить свое внимание на какой-то конкретный локус.

Поглощенное внимание привело к гибели 101 человека

Крайним примером такого случая является катастрофа, происшедшая в декабре 1972 года, в результате которой погиб 101 человек. Обычно в кабине пилотов загорается зеленый индикатор, сигнализирующий о том, что шасси выпущено и готово к посадке. Во время того полета индикатор выпуска шасси не зажегся, поэтому командир самолета решил подняться на высоту 2000 футов, чтобы сделать круг, а второй пилот переключил на этой высоте управление самолетом в режим автопилота. После этого все три члена экипажа

¹ Здесь слово *attentions* означает «ухаживания» или «забота». – *Примеч. ред.*

² Значительное внимание уделяется изучению биологических механизмов, которые позволяют животным синхронизироваться с внешними временными циклами. Однако мне не известно ни одной работы, посвященной исследованию того, как мы настраиваемся и реагируем на свои «внутренние часы».

стали пытаться заменить лампочку индикатора, но она застряла, и ее никак нельзя было вытащить. Наверное, из-за всех этих манипуляций с лампочкой кто-то случайно выключил автопилот. Во всяком случае каким-то образом он оказался выключенным. Вскоре, как впоследствии показала запись бортового самописца, зазвучала автоматическая сигнализация – полусекундный сигнал предупредил пилотов о том, что самолет снизился на 250 футов ниже установленной высоты. Также зажегся желтый индикатор предупреждения. Члены экипажа, поглощенные проблемой с зеленой лампочкой, не заметили все эти сигналы. Немного позже, все еще продолжая возиться с лампочкой, второй пилот заметил, что альтиметр показывает угрожающе малую высоту 150 футов. После этого он спросил командира: «Мы еще на высоте 2000, да?» В ответ командир воскликнул: «Эй, что происходит?»

В этот момент зазвучала сирена, предупреждающая о малой высоте. «Несмотря на почти нулевые показания альтиметра, желтый сигнал, предупреждающий об отклонении от назначенной высоты, почти нулевые показания радиоальтиметра и его звучащую сирену, каждый член экипажа был настолько уверен, что самолет находится на высоте 2000 футов, что никто из них не предпринял никаких действий, и уже спустя 8 секунд после того, как командир экипажа посмотрел на показания альтиметра, самолет упал в болота Флориды». (Цитата из Garrison, 1995.)

Вы можете быть в различной степени поглощены задачей, которая в данный момент находится в локусе вашего внимания. Чем больше вы сосредоточены на задаче, тем труднее вам перейти на другой локус внимания и тем более сильный стимул требуется для того, чтобы такой переход произошел. В самом крайнем случае, когда человек полностью поглощен какой-то задачей, он перестает следить за окружающим пространством. Вероятно, вы испытывали такое поглощенное состояние, когда, например, читали книгу или глубоко размышляли о какой-то проблеме, или пребывали в кризисе, который, как говорится, поглотил все ваше внимание. Работа с компьютером часто бывает настолько сложной и требует такого напряжения, что пользователь оказывается целиком поглощенным компьютерной системой и поэтому отвлекается от выполнения стоящей перед ним задачи. Наша цель состоит в том, чтобы оставить саму задачу в качестве локуса внимания пользователя.

Поглощенность задачей или проблемой уменьшает возможность изменения локуса внимания. С другой стороны, такая поглощенность очень важна с точки зрения продуктивности, если она ограничена только текущей задачей и если компьютерная система не отвлекает внимание пользователя на себя. Системы должны разрабатываться таким образом, чтобы пользователь имел возможность сосредоточиться на своей работе. Интерфейсы следует разрабатывать с расчетом на то, что поль-

зователь, поглощенный своей задачей, не станет даже реагировать на ваши попытки пообщаться с ним. Интерфейс должен хорошо работать независимо от степени поглощенности пользователя. К примеру, разработчики интерфейсов иногда считают, что локус внимания пользователя связан с курсором и что посредством изменения формы курсора можно управлять вниманием пользователя. Местоположение курсора – это хорошее место для размещения указателей, но даже там они могут остаться незамеченными. Форма курсора не является локусом внимания пользователя – скорее им может быть место или объект, на который курсор указывает. (Пример приводится в разделе 3.2.)

Многие примеры поглощенности кажутся невероятными до тех пор, пока вы сами не окажетесь в подобной ситуации или пока не услышите так много свидетельств, что убедитесь в силе этого состояния. Аварии и катастрофы самолетов служат убедительными примерами, поскольку они, как правило, хорошо изучены и тщательно документированы. Поэтому приведем еще один случай. Известный пилот управлял самолетом незнакомой для него модели, в которой перед посадкой требовалось выпустить шасси. В этой модели используется специальная звуковая сигнализация, которая срабатывает, когда самолет находится на определенной высоте от земли при не выпущенном шасси. «Я посадил самолет при убранном шасси, посчитав, что звуковой сигнал, который постоянно звучал, пока я приближался к посадочной полосе, имел отношение к пневматическому тормозу. (Это был один из моих первых уроков, связанных со странной ментальной путаницей, приводящей к авариям.)» (Garrison, 1994). Но в том случае не было никакой «странной ментальной путаницы». Гаррисон целиком сосредоточился на том, чтобы совершить хорошую посадку – это одна из самых трудных задач в пилотировании, требующая очень большой концентрации.¹

Способность человека «отстраиваться» от того, что отвлекает, обязательно проявляется в виде реакций типа «все или ничего», как это было в предыдущих примерах. Реакция может быть пропорциональной уровню поглощенности или интенсивности помехи. По мере роста напряжения «люди все больше и больше концентрируются только на нескольких характеристиках окружающей обстановки, обращая все меньше и меньше внимания на другие» (Loftus, 1979, с. 35). Таким образом, *если во время использования какого-либо интерфейса компьютер начинает работать неожиданным образом, то по мере того как ваше волнение, вызванное возникшей проблемой, возрастает, вы с*

¹ Разработчик интерфейсов мог бы удивиться: если в самолете предусмотрена специальная звуковая сигнализация, почему же тогда не предусматривается возможность автоматического выпуска шасси? Хотя в этой книге не могут обсуждаться все подобные тонкости, но здесь нужно сказать, что автоматический выпуск шасси может представлять опасность для тех, кто находится внутри самолета. Поэтому решение о выпуске шасси всегда остается на усмотрение пилота.

меньшей вероятностью сможете заметить подсказки, сообщения и другие средства помощи пользователю.

Чем более критической является задача, тем меньше вероятность того, что пользователь заметит предупреждения относительно тех или иных потенциально опасных действий. Предупреждающее сообщение с наибольшей вероятностью может остаться незамеченным именно в тот момент, когда информация, содержащаяся в нем, имеет наибольшую ценность. Это напоминает курьезное следствие из третьего закона Мерфи¹, но на самом деле не имеет к нему отношения. Единственное, чем мы можем здесь помочь, – это сделать так, *чтобы пользователь не смог совершить ошибок, связанных с работой интерфейса, или, другими словами, чтобы пользователь имел возможность сразу отменить результаты любого действия, а не просто получал предупреждения о потенциальных его последствиях.* Большинство ситуаций в интерфейсе могут быть спланированы таким образом, что сообщения об ошибках становятся ненужными. Сильная критика применения сообщений об ошибках имеется в книге Купера «About Face» (Cooper, 1995, с. 421–440).

2.3.4. Истоки локуса внимания

То, что у нас может быть только один локус внимания, может показаться странным. Попробуем рассмотреть причины этого. Баарс в своей работе (1988) красноречиво отвечает на этот вопрос, пытаясь найти биологическое объяснение тому, что мы развились таким ограниченным образом, и утверждает, что

«сознание и связанные с ним механизмы ставят под сомнение функциональные объяснения проблемы, поскольку возможности сознания парадоксальным образом ограничены. Почему мы не можем переживать две разные «вещи» одновременно? Почему кратковременная память (STM)² способна вместить не более полудюжины несвязанных между собой элементов? Каким образом такие ограниченные возможности оказались приемлемыми? Как было бы замечательно, если бы мы могли читать одну книгу, а писать другую, говорить с приятелем и еще наслаждаться

¹ «Если неприятность может случиться, она случается». Первое следствие из этого закона формулируется следующим образом: «Если никаких неприятностей не ожидается, они произойдут обязательно».

² Кратковременная память (short-term memory, STM) – это такой режим работы памяти, в котором фиксируются только что увиденные, услышанные или другим образом воспринятые раздражители. Если мы не используем эту память или не делаем ее локусом нашего внимания, информация, содержащаяся в ней, исчезает через 10–20 секунд или даже быстрее, если наше внимание обращено на какие-то новые события. Как отмечает Баарс, кратковременная память не только кратковременна, но и очень ограничена по объему, поэтому новая информация вытесняет содержание этой памяти безвозвратно. В книге Лофтуса (Loftus, 1980) представлено описание структуры человеческой памяти, которое рассчитано на неспециалистов и чрезвычайно легко читается.

какой-нибудь вкусной едой, и все это одновременно. Определенно, возможности нашей нервной системы кажутся достаточными для того, чтобы совершать все эти действия в одно и то же время. Стандартный ответ про некие «физиологические» ограничения – что у нас только две руки и один рот – кажется неубедительным, поскольку он приводит к другому вопросу, который еще более усложняет проблему: почему организмы, наделенные самым совершенным мозгом в животном царстве, не развили у себя руки и рты, чтобы нормально управляться с несколькими параллельными процессами? А также – почему наша способность к параллельной обработке информации возрастает с автоматизмом и уменьшается по мере того, как в процесс вовлекается сознание? (с. 348)»

Баарс предполагает, что ответ на эти вопросы связан с тем, что в каждом из нас существует только одно «Я», что есть только одна «целостная система». Но если сказать, что в человеке есть только одна личность, то это вызывает очередной вопрос, а именно: почему ансамбль «сознание-тело» не содержит в себе множества личностей? Здесь я говорю не об изменениях, которые происходят постоянно¹, но о действительно одновременном наличии нескольких независимых друг от друга сознаний в едином физическом организме. Возможно, что наличие только одной личности является формой биологического приспособления к линейности времени либо обусловлено скорее эволюционной случайностью, чем функциональной адаптацией. Тем не менее, причина того, что личность только одна, вероятнее всего, состоит именно в адаптации – содержать в одном теле одновременно множество личностей, наверное, было бы просто физически невозможно. При свойственной нам организации тела две личности не могли бы одновременно разговаривать или поворачивать голову в разных направлениях. Даже если бы наши глаза могли двигаться независимо друг от друга, как у геккона, разве они смогли бы удовлетворить два разных любопытства? Как мне представляется, человек, у которого в результате мутации возникло множество сознаний, стал бы жертвой какого-нибудь хищника, попытавшись убежать от него в разные стороны одновременно, – и это только один из возможных трагических вариантов.²

¹ Постоянное изменение личности – это черта, присущая человеку. Мы все время растем и меняемся. Изменения этого рода, а также изменения, связанные с расстройством человеческой психики, здесь не рассматриваются.

² Мне часто приходилось видеть, как мой кот разрывался между любопытством и страхом, когда все его органы чувств устремлялись на изучение какого-то незнакомого объекта, но в то же время его тело напрягалось в готовности немедленно убежать. В некоторых ситуациях мне и самому приходилось так действовать. Иногда, если не убежать, полученная информация оказывается ценной, иногда задержка становится фатальной – отсюда выражение «любопытство убило кота». Такое двойное состояние, или, как говорят, «раздвоение личности», объясняется скорее внутренним последовательным переключением между двумя процессами, чем их одновременным присутствием независимо друг от друга.

Иногда рождаются сиамские близнецы и двухголовые животные, но они имеют два разных сознания, и эти случаи обусловлены неправильной записью или прочтением генетического кода. Их нельзя назвать успешными с точки зрения эволюции, и они не являются результатом естественного отбора. В естественных условиях такие капризы природы вряд ли смогли бы выжить и дать потомство.

2.3.5. Эксплуатация единого локуса внимания

Мы рассмотрели следствия и возможные истоки существования одного-единственного локуса внимания. Следующий шаг – воспользоваться этим свойством. Конечно, мы не можем перестроить внутренние механизмы сознания людей, но мы создаем продукты, а их интерфейсы можно согласовать с нашими когнитивными способностями.

Наличие у человека только одного локуса внимания имеет и положительные стороны. Фокусники эксплуатируют это свойство нашей психики самым бесстыдным образом. Хороший фокусник может так зафиксировать внимание всей аудитории на одной руке, что ни один зритель не заметит того, что в это время делает другая его рука, хотя она никак и не скрыта. Если нам известно, где в данный момент внимание пользователя зафиксировано, мы можем производить изменения во всех остальных частях системы, зная, что они его не отвлекут. Этот эффект был использован при разработке компьютера Canon Cat (рис. 2.1). Когда пользователь прекращал работу, компьютер сохранял на первой дорожке диска побитовое изображение экрана именно в том виде, который был на момент остановки работы. Когда пользователь возвращался к работе, на экран за долю секунды помещалась та самая картинка. Человеку необходимо около 10 секунд для того, чтобы переключиться с одного контекста на другой или мысленно подготовиться к предстоящей задаче (Card, Moran и Newell, 1983, с. 390), в то время как компьютеру Canon Cat требовалось всего лишь 7 секунд, чтобы считать с диска остальную рабочую часть информации. Поэтому, пока пользователь смотрел на статичное изображение на экране, вспоминая, чем он занимался, и решая, что он собирается делать далее, в его локусе внимания находились приготовления к предстоящей задаче, а система тем временем завершала загрузку. Только после этого экран становился активным, хотя и не меняя своего вида – за исключением того, что курсор начинал мигать. Только отдельные пользователи смогли заметить весь этот фокус. Большинство же владельцев просто думали, что их компьютер умеет магическим образом считывать всю дискету за ту долю секунды, когда появляется первое изображение. Блестяще!

Многие не верят, что человеку требуется в среднем 10 секунд для того, чтобы переключить свое внимание с одного контекста на другой. Это время отсчитывается от последней команды, выполненной в предыдущем контексте, до первой команды, выполняемой в новом контексте.

Этот промежуток времени обычно не замечается, поскольку сознание пользователя в это время занято. Тем не менее, при разработке интерфейсов этот феномен должен использоваться с осторожностью. Если во время работы пользователю необходимо постоянно переходить в один из контекстов, то этот переход может стать для него привычным и поэтому происходить за гораздо меньшее время.

Разного рода запаздывания можно скрыть. Например, карточная игра, в которой на генерацию каждой новой раздачи требуется несколько секунд, кажется более быстрой, если в это время воспроизводить шелестящий звук тасуемых карт. Ценность такой маскировки обнаруживается, если этот звук внезапно отключить. Тогда задержка сразу станет казаться раздражающей (Dick Karpinski, личное сообщение, 1999).



Рис. 2.1. Компьютер Canon Cat. Обратите внимание на две клавиши LEAP, расположенные под клавишей пробела

2.3.6. Возобновление прерванной работы

Обычно, прервав некоторую работу, вы затем к ней возвращаетесь. Если перерыв продолжается всего несколько секунд – в пределах периода затухания кратковременной памяти, – дополнительных стимулов, для того чтобы вы могли вернуться к выполнению текущей задачи, не требуется. Если период более длительный, то возвращение к выполнению прерванной задачи должно быть чем-то инициировано – например, видом лежащей перед вами незаконченной работы. Такие подсказки в обычной жизни встречаются так же часто, как и при рабо-

те с компьютерами: банановая кожура, оставленная вашим 4-летним ребенком на кухонном столе, становится подсказкой о том, что кожуру необходимо выбросить.

Одной из метафор, пронизывающих как персональные компьютеры, так и производные от них технологии, является та центральная, нейтральная отправная область, называемая еще рабочим столом, из которой пользователь может запускать различные приложения. После включения большинство компьютеров отображают рабочий стол, хотя некоторые из них можно настроить на запуск определенного набора приложений. Закрыв приложение, вы обычно возвращаетесь на рабочий стол. Такой подход к интерфейсу не является ориентированным на человека и эффективным. Причина здесь очевидна: когда вы закрываете какое-либо приложение, вы хотите либо (1) возвратиться к предыдущей задаче, с которой вы работали, либо (2) начать новую задачу.

В нынешних системах, основанных на рабочем столе, вы *всегда* должны самостоятельно осуществлять переход к необходимой задаче. *Для интерфейса, который всегда возвращает вас туда, где вы остановились в последний раз, это был бы худший возможный случай, потому что если вы захотели бы вернуться к предыдущей задаче, вам вообще не пришлось бы совершать никаких действий.*

Аналогичным образом, когда вы возвращаетесь на какой-либо сайт, или веб-страницу, лучше всего было бы вернуть вас на то место, где вы были последний раз, а не на начальную страницу, тем более что если сайт хорошо разработан, то переход на начальную страницу осуществляется одним щелчком мыши. По этим же самым причинам очевидно, что *когда вы открываете документ в каком-либо приложении, например в текстовом процессоре, вы должны вернуться к тому месту, где вы с ним работали в тот момент, когда закрыли или сохранили его в последний раз.*

Компьютер Canon Cat имел свойство при запуске всегда возвращаться к последней задаче. Более того, на экране появлялось именно то изображение, включая позицию курсора, которое было на момент последнего его использования. Многие пользователи говорили, что когда они видели на экране ту же самую картинку, это помогало им быстрее вспомнить, чем они занимались, когда последний раз пользовались машиной, поэтому возвращаться к работе с Canon Cat было для них более приятно, чем к работе с компьютером, который при запуске отображал рабочий стол. В последнее время подобный подход используется в компьютере под названием Apple iBook, который сохраняет текущее состояние на диске и при включении машины выводит его обратно на экран.

Разработчики радиоприемников и телевизоров с цифровой настройкой всегда предусматривают возможность сохранения параметров звука и каналов, на которые эти приборы были настроены в последний

раз, что, конечно, усложняет эти приборы и повышает их стоимость, так как для этого требуются элементы энергонезависимой памяти, не обязательные для всего остального механизма. Для разработчиков компьютеров задача упрощается, потому что в компьютерах уже имеется значительный объем такой памяти в виде жесткого диска. Поэтому нет оправдания тем разработчикам, которые не предусматривают подобную возможность также и в компьютерах, имея все необходимые для этого аппаратные средства.

3

Значения, режимы, монотонность и мифы

Нет прогресса без борьбы.

Фредерик Дуглас

Чтобы быть более точными при дальнейшем обсуждении интерфейсов, мы введем здесь некоторые определения и условные обозначения. Основанные на понятии локуса внимания, данного в главе 2, они необходимы нам для того, чтобы понять, что такое режимы и каково их негативное воздействие на интерфейсы. Также в этой главе мы познакомимся с положительным свойством интерфейсов, которое называется *монотонностью* и приводит нас к критике интерфейсов, предусматривающих разные режимы для начинающих и опытных пользователей.

3.1. Терминология и условные обозначения

Мир делится на людей, которые думают, что они правы.

Дидр Мак-Грат

Содержание (или контент) – это информация, которая находится в компьютере или другом устройстве, предназначенном для обработки информации, и которая является для вас осмысленной и полезной. Создание или изменение содержания – это та задача, которую вы предполагаете выполнить с помощью упомянутого устройства. Если вы писатель, содержание – это те ваши произведения, которые хранятся в системе. Если вы художник, ваши рисунки представляют собой содержание системы. Сама система – компьютер, а также меню, пиктограммы и другие его принадлежности – не является содержанием, если только

вы не программист и не разработчик интерфейсов. Теперь мы можем перефразировать первый закон робототехники Азимова (см. главу 1) в терминах содержания: «Никакая система не может причинить вред содержанию или своим бездействием допустить, чтобы содержанию был причинен вред».

Графическое устройство ввода (ГУВ) – это механизм для передачи системе информации об определенном местоположении или выборе объекта на экране монитора. В качестве примера типичного ГУВ можно привести мышь, трэкбол, световое перо, планшетный карандаш (tablet pen), джойстик или тачпад. Под **кнопкой ГУВ** будет подразумеваться основная кнопка любого ГУВ, например левая кнопка двухкнопочной мыши. Как правило, графическое устройство ввода используется для управления местоположением курсора, который представлен на экране монитора в виде стрелки или другого значка, являющегося системной интерпретацией указываемого вами места. Поскольку в конкретный момент мы можем направить свое внимание только на один курсор, система не должна отображать более одного курсора для каждого имеющегося графического устройства ввода. В приложении А приведено обоснование тому, почему у мыши должна быть только одна кнопка.

Одно нажатие и отпускание клавиши или переключателя без каких-либо промежуточных действий мы будем называть **нажатием (tap)**. Этот термин относится только к клавишам на клавиатуре или другим контактными переключателям без фиксации, которые при отпускании автоматически возвращаются в исходное положение и электрическое состояние.

Щелкнуть (click) означает расположить курсор в определенном месте, а затем нажать и отпустить кнопку ГУВ. Таким образом, выражение «щелкните на слове *аллигатор*» означает, что вам требуется навести курсор на слово *аллигатор* и затем, не меняя позицию курсора, нажать кнопку ГУВ и отпустить ее. **Перетащить (drag)** означает нажать кнопку ГУВ в одном местоположении курсора, переместить его в другое место и уже затем отпустить кнопку. Это действие еще иногда называется *щелкнуть и перетащить (click and drag)*. **Дважды щелкнуть (double click)** означает расположить курсор в определенном месте, а затем два раза быстро нажать кнопку ГУВ без промежуточного движения или какого-либо другого действия. (На практике небольшое движение допустимо – обычно ГУВ слегка смещается в момент нажатия кнопки.) В некоторых интерфейсах применялись тройные щелчки, а также щелчки с еще большим количеством нажатий.

Существуют различные варианты обозначений для комбинаций клавиш, используемых при управлении большей частью программного обеспечения. Например, нажатие клавиши <Ctrl> и, при ее удерживании, нажатие и удерживание клавиши <Shift> и затем, при нажатом положении обеих этих клавиш, нажатие клавиши с буквой *t* часто

обозначается в руководствах как $\langle \text{Ctrl} \rangle - \langle \text{Shift} \rangle - \langle t \rangle$ или $\langle \text{Ctrl} \rangle + \langle \text{Shift} \rangle + \langle t \rangle$. Однако эти обозначения операций с клавишами нельзя отличить от условных обозначений последовательностей действий *Control*, *дефис*, *Shift*, *дефис*, *t* или *Control*, *плюс*, *Shift*, *плюс*, *t* соответственно.

Такая двусмысленность может приводить к ошибкам. Например, когда я работал над рукописью этой книги, мне понадобилось посмотреть, какая комбинация клавиш применяется для выполнения необходимой мне команды. В руководстве было указано (в конце предложения).

Control +.

Поэтому я нажал $\langle \text{Ctrl} \rangle$ и затем, удерживая эту клавишу, ввел знак «плюс». Из-за двусмысленного обозначения в руководстве я сделал ошибку. На самом деле требовалось при нажатой клавише $\langle \text{Ctrl} \rangle$ нажать клавишу со знаком точки.

Кроме того, эти обозначения не дают возможности отразить такие действия, как, например, одновременное нажатие и удерживание клавиш $\langle \text{Ctrl} \rangle$ и $\langle \text{Shift} \rangle$ с последующим вводом нескольких алфавитных символов, причем клавиша $\langle \text{Shift} \rangle$ отпускается после ввода первых двух символов, а $\langle \text{Ctrl} \rangle$ остается нажатой. Альтернативный вариант, который я только что использовал, — выражение операций обычным языком, что хотя и понятно, но довольно громоздко.

Чтобы обозначать операции с клавишами точно и кратко, я использую символ стрелки, указывающей вниз, сразу после названия клавиши, на которую требуется нажать и удерживать в нажатом состоянии, — например, *Shift*↓ означает, что необходимо нажать и удерживать клавишу $\langle \text{Shift} \rangle$. Символ стрелки, указывающей вверх, после названия клавиши показывает, что эту клавишу следует отпустить, — например *Shift*↑. Нажатие одной клавиши, скажем, клавиши $\langle t \rangle$, может быть точно обозначено с помощью записи *t*↓ *t*↑. Запись нажатия одной клавиши можно сократить до *t*↓↑ или просто до *t*, если это не вызывает двусмысленности и не может привести к путанице.

Между отдельными действиями, которые следуют друг за другом, ставится пробел. Нажатие клавиши пробела обозначается словом

Пробел

Это обозначение невозможно перепутать с набором слова *пробел* (space), потому что последовательность символов этого слова, вводимая с клавиатуры, будет записываться буквами, разделенными пробелами, а именно:

п р о б е л

Когда есть риск неверного истолкования или необходимо сделать акцент на каждом отдельном действии, я применяю полную запись:

n↓ *n*↑ *p*↓ *p*↑ *o*↓ *o*↑ *b*↓ *b*↑ *e*↓ *e*↑ *l*↓ *l*↑

Любые комбинации клавиш могут быть линейно представлены в этой системе записи. Например, нажатие и удерживание клавиши <Shift>, затем нажатие клавиши <n>, после этого нажатие и удерживание клавиши <Ctrl> при все еще нажатой клавише <Shift>, затем нажатие клавиши <k>, отпускание клавиши <Shift> при все еще нажатой клавише <Ctrl>, затем нажатие клавиши <w> и после этого отпускание клавиши <Ctrl> записывается следующим образом:

$Shift \downarrow n \text{ Control} \downarrow k \text{ Shift} \uparrow w \text{ Control} \uparrow$

При условии, что порядок отпускания клавиш не имеет значения для работы интерфейса, отпускание любой из нажатых клавиш обозначается стрелкой вверх для каждой отдельной клавиши. Таким образом, набор слова *пробел* будет записываться так:

$n \downarrow \uparrow p \downarrow \uparrow o \downarrow \uparrow b \downarrow \uparrow e \downarrow \uparrow l \downarrow \uparrow$

Приведем еще один полезный пример. Чтобы восстановить исходное состояние ОЗУ на компьютере Macintosh, должна быть выполнена следующая последовательность команд:

$Command \downarrow Control \downarrow Power \downarrow \uparrow \uparrow \uparrow Command \downarrow Option \downarrow p \downarrow r \downarrow$

Все клавиши требуется удерживать до тех пор, пока не зазвучит сигнал, после чего:

$\uparrow \uparrow \uparrow \uparrow$

(Выражаясь обычным языком: нажмите и удерживайте клавишу <Command> – отсутствие стрелки вверх означает, что клавишу не следует отпускать сразу же. Затем нажмите и удерживайте клавишу <Ctrl>. Здесь также нет стрелки вверх, поэтому следует, удерживая обе эти клавиши одновременно, нажать клавишу <Power>. Ни одной стрелки вверх еще не было, так что сейчас вы удерживаете сразу три клавиши. Последующие три стрелки вверх означают, что требуется отпустить все три клавиши в любом порядке или одновременно. Затем, как следует из записи, вам необходимо нажать, не отпуская, клавишу <Command>, затем нажать, не отпуская, клавишу <Option>, затем нажать, не отпуская, клавишу <p>, после чего нажать и удерживать клавишу <r>, так что, в конце концов, вы будете удерживать в нажатом положении четыре клавиши одновременно. Это следует делать до звукового сигнала, после чего все четыре клавиши следует отпустить.)

Если при наборе последовательности имеет значение время, то об этом должно быть сказано в примечании. Например, для набора одной буквы *t* запись будет следующей:

$t \downarrow t \uparrow$

Однако при использовании большинства клавиатур, если между $t \downarrow$ и $t \uparrow$ происходит задержка длительностью более 500 мс, на экране приблизительно каждые 100 мс станут появляться еще символы *t*. Эта функ-

ция обычно называется автоповтором. Задержки, в том числе и те, которые запускают автоповтор, часто создают проблемы в работе интерфейсов. Использование задержек в интерфейсах, а также улучшение механизма автоповтора будут обсуждаться далее в разделе 6.4.5.

3.2. Режимы

Поскольку люди более уступчивы, чем компьютеры, бывает легче заставить человека приспособиться к ограничениям компьютера, чем создать компьютер, приспособленный к потребностям человека. И когда это происходит, человек попадает в подчинение к компьютеру, а вовсе не освобождается им.

Карла Дженнингз

Режимы (modes) являются важным источником ошибок, путаницы, ненужных ограничений и сложности в интерфейсе. Многие проблемы, создаваемые режимами, хорошо известны. Тем не менее, практика создания систем без режимов почему-то не находит широкого применения в разработке интерфейсов. Прежде чем приступить к обсуждению методов устранения режимов, следует рассмотреть их подробно, тем более что даже среди специалистов по интерфейсам нет общего мнения относительно того, что представляет собой режим (Johnson and Englebeck, 1989).

Чтобы понимать режимы, нам следует сначала определить понятие жеста. **Жест (gesture)** – это последовательность действий человека, которая выполняется автоматически (после старта). Например, для опытного пользователя набор такого простого слова, как *это*, представляет всего один жест, в то время как для начинающего пользователя, который еще не вполне овладел клавиатурой, набор этого слова по буквам будет состоять из отдельных жестов. Объединение последовательности действий в жест, связанный с определенным психологическим процессом, определяется как **формирование модуля (chunking)**, т. е. соединение отдельных элементов когнитивного процесса в единый ментальный модуль, что позволяет воспринимать множество элементов как целое (Buxton, 1986, с. 475–480; Miller, 1956).

В большинстве интерфейсов допускается несколько интерпретаций конкретного жеста. Например, в одном случае с помощью клавиши <Return> можно вставить в текст символ возврата каретки, тогда как в другом случае нажатие этой клавиши приводит к исполнению предшествующей строки в качестве команды.

Режимы проявляются в том, как реагирует интерфейс на тот или иной жест. *Состояние интерфейса, при котором интерпретация данного конкретного жеста остается неизменной, называется режимом.* Если жест получает другую интерпретацию, это значит, что интерфейс находится в другом режиме. Такое определение дает ценное персона-

чальное представление о том, что составляет режим. Позже мы уточним это определение.

Фонарик, управляемый с помощью кнопочного выключателя, может быть либо включен, либо выключен при условии, что он находится в хорошем рабочем состоянии. При нажатии кнопки свет зажигается, если фонарик находился перед этим в выключенном состоянии, и наоборот, если текущее состояние «включено», то после нажатия кнопки фонарик выключается. Два состояния фонарика соответствуют двум режимам интерфейса. В одном режиме нажатие кнопки включает свет. В другом режиме нажатие кнопки выключает свет. Если вы не знаете текущее состояние фонарика, вы не можете предсказать, к чему приведет нажатие кнопки. Если ваш фонарик лежит глубоко в сумке, и вы его не видите, то не можете узнать, включен свет или нет, при условии, что это нельзя почувствовать по температуре. Чтобы убедиться, что фонарик выключен, вам требуется вынуть его из сумки. Невозможность определить текущее состояние фонарика – это пример классической проблемы, возникающей в интерфейсе, в котором есть режимы. По состоянию управляющего механизма невозможно сказать, какое действие следует выполнить, чтобы достичь поставленной цели. Опирируя с управляющим механизмом без одновременной проверки состояния системы, вы не сможете предсказать, какой результат будет иметь данная операция.

К переключателям трудно подобрать надписи. Например, однажды мне показали интерфейс, в котором кнопка на экране была подписана *Lock* (Блокировать). Когда пользователи в первый раз видели эту кнопку, они считали, что должны нажать на нее для блокировки данных в этом окне. Когда они делали так, подпись кнопки изменялась на *Unlock* (Разблокировать), показывая, что при нажатии на эту кнопку данные разблокировались бы. После этого многие пользователи удивлялись, что данные оказывались разблокированными, поскольку кнопка показывала *Unlock*, что они считали индикатором текущего состояния, как это часто бывает при использовании переключателей на кнопках и в меню. Естественно, это приводило к путанице: на самом деле кнопка показывала *Lock*, когда данные были разблокированы, и *Unlock* – когда они были заблокированы. Очевидно, что проблему нельзя решить, просто поменяв надписи таким образом, чтобы разблокированные данные обозначались как *Unlock*, а заблокированные как *Lock*.

В данном случае следует, во-первых, использовать флажок, а не кнопку, а во-вторых, применить более точные формулировки – *Locked* (Заблокировано) вместо *Lock* (Блокировка), что будет восприниматься более правильно: если флажок установлен, значит, данные заблокированы, если же флажок сброшен – данные не заблокированы. Изменения подписей в этом случае не требуется. Можно использовать более развернутую формулировку, например: «Щелкните по этой кнопке, чтобы разблокировать данные» или даже: «Данные в настоящее время за-

блокированы. Щелкните по этой кнопке, чтобы разблокировать их». Однако на кнопке или около флажка, или в меню трудно разместить полное объяснение, если интерфейс не снабжен функцией увеличения изображения (см. раздел 6.2).

Флажки, тем не менее, оставляют неясным, какие еще есть альтернативы. Например, если флажок отмечен «Сохранить в архиве при закрытии», то при закрытии активного окна данные будут сохранены в архиве. Однако из подписи не понятно, что произойдет, если этот флажок сброшен. Данные будут сохранены где-то в другом месте? Или не сохранены вообще? Или при закрытии окна появится еще какая-то опция? Самым лучшим решением в этой ситуации зачастую оказывается использование набора переключателей (рис. 3.1). Такие переключатели не являются модальными, и пользователь может сразу определить не только текущее состояние, но и альтернативы. Независимо от того, применяются ли флажки или переключатели, важно, чтобы в названиях употреблялись прилагательные, описывающие состояние объекта, а не глаголы, описывающие действие с этим объектом, потому что иначе пользователю не ясно, действие уже произошло или только должно произойти.

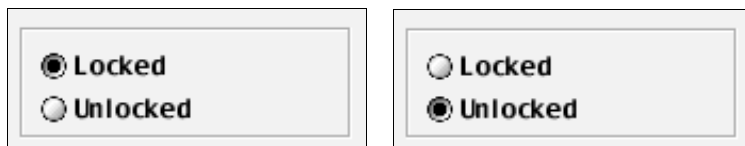


Рис. 3.1. Пара переключателей, обозначенных прилагательными. Выбранная опция отмечена точкой в кружке. Изображение слева отражает заблокированное состояние данных, а справа – разблокированное состояние. Путаница маловероятна, поскольку пользователь видит все возможные варианты

Переключатели стали стандартным средством для выбора одной из нескольких возможностей, и вряд ли следует их заменять какими-то другими механизмами. *Используйте переключатели вместо выключателей. На выключатели стоит полагаться только в том случае, когда можно видеть значение контролируемого состояния и оно находится в локусе внимания пользователя или в кратковременной памяти.*

За исключением случаев, когда текущее состояние находится в локусе внимания, – а обычно это не так, – выключатели могут приводить к ошибкам. Такие ошибки обычно имеют кратковременный характер и легко исправляются. Тем не менее, их нельзя не учитывать при разработке интерфейсов. Заниматься разработкой интерфейсов и при этом не обращать внимания на такие детали – все равно, что пытаться исполнить скрипичный концерт, иногда забывая диезы и бемоли, обозначенные в нотной записи. Такие ошибки раздражают слушателя тем, что отвлекают его внимание от музыки. Точно так же мелкие ошибки в интерфейсе затрудняют ход работы пользователя.



Другую связанную с режимами трудность, которая особенно изматывает пользователей компьютеров, можно проиллюстрировать на примере того, как работает клавиша <Caps Lock>, присутствующая на большинстве типов клавиатур. Часто первым признаком случайного нажатия этой клавиши оказывается то, что вы замечаете, что набранное вами предложение напечатано заглавными буквами, и только после этого вы замечаете также, что включен индикатор <Caps Lock> (если он есть на клавиатуре). «Не случайно ругательства обозначаются в виде цепочки символов, наподобие #&%!#\$&. Именно такая картина возникает, когда кто-то набирает цифры при ошибочно нажатой клавише <Caps Lock>», – утверждает мой коллега доктор Джеймс Уинтер (личное сообщение, 1998).

Несколько десятилетий назад Ларри Кларк заметил, что режимы создают проблемы по той причине, что привычные действия приводят к неожиданным результатам (Clark, 1979).¹ Наиболее часто предлагаемое средство против ошибок, порождаемых режимами, состоит в том, чтобы ясным образом показывать пользователю текущее состояние системы. Доктор Дональд Норман охарактеризовал ошибки, связанные с режимами, как результаты недостаточной обратной связи, осуществляемой индикатором состояния системы (Norman, 1983). Однако, на мой взгляд, истинной причиной является не недостаточная обратная связь, а то, что этот индикатор не находится в локусе внимания пользователя.

Особенно показательный пример неудачного применения индикатора, отражающего текущее состояние системы, можно встретить в интерфейсе системы автоматизированного проектирования (САПР) (computer-aided design, CAD) Vellum (Ashlar, 1995), во всех остальных отношениях превосходном. Всем, кто занимается разработкой графических пакетов, я настоятельно рекомендую изучить программу Drafting Assistant, разработанную Ashlar. Эта программа оснащена чрезвычайно удобным интерфейсом, намного более эффективным и приятным в использовании, чем тот, который применяется в более известном пакете AutoCAD.² Одной из функций, предусмотренных для экономии времени в пакете Vellum, является указатель, который может следовать по внутренней или наружной стороне геометрической фигуры, выде-

¹ Мне не известно, как долго термин *режим* используется при описании интерфейсов. Во внутреннем отчете компании Херох, посвященном системе Gypsy Typescript System, с гордостью заявлялось: «В Gypsy нет *режимов*» (Tesler and Mott, 1975). На самом деле в Gypsy существовали режимы, но наиболее проблемные из тех, что встречались в текстовых процессорах того времени, были устранены. В 1975 году термин *режим* в этом контексте был настолько новым, что авторы даже посчитали нужным заключить его в кавычки.

² Уже во время написания этой книги разработчики AutoCAD лицензировали технологию, которая использовалась в Drafting Assistant.

ляя (трассируя) описываемую границу. Чтобы перейти в режим трассировки, необходимо щелкнуть по соответствующей кнопке палитры инструментов, при этом стандартный курсор  преобразуется в курсор особой, «указательной» формы . По завершении выделения легко забыть переключиться назад, к стандартному курсору. Выделение с помощью мыши является настолько частым действием, что для меня оно уже давно стало автоматичным, и поэтому когда в этой программе я щелкаю, чтобы выделить какой-то объект после операции трассировки, то вместо успешного выделения я с удивлением обнаруживаю, что обведенный указателем контур исчезает. Я совершаю эту ошибку уже не первый год, так же как и другие пользователи, с которыми мне приходилось говорить об этом. Хотя иногда я и ловлю себя на мысли, что делаю неправильно, но все же я никогда *не смогу* научиться всегда замечать, что нахожусь в режиме трассировки, так как выделение с помощью мыши, для меня во всяком случае, является автоматичным действием. В локусе моего внимания находится объект выделения, но не форма курсора. Изменяющийся курсор, используемый в Vellum, — это один из множества примеров, подтверждающих, что средства обратной связи и индикаторы текущего положения недостаточны, чтобы обеспечить устранение ошибок, вызываемых режимами, даже в том случае, когда индикатор расположен в непосредственной близости от локуса вашего внимания. Он может попадать в область высокой зрительной способности, например, когда индикатором служит курсор, и тем не менее, вы не сможете заметить сообщение, передаваемое этим индикатором, поскольку он не находится в локусе вашего внимания. Необходимой темой для будущих исследований должно стать определение частоты ошибок, связанных с режимами, и выяснение условий, влияющих на частоту их возникновения.

Опыт не избавляет от ошибок, порождаемых режимами, поскольку у опытного пользователя уже имеются устойчивые привычки. Отсутствие опыта также не защищает от этих ошибок. В примере с курсорами в программе Vellum начинающий пользователь еще не сформировал привычку переключения в режим обычного курсора после использования указателя трассировки. По мере приобретения опыта в использовании этой функции начинающий пользователь научится делать такое переключение не задумываясь, а значит, теперь возникает проблема привычек. Если текущее состояние интерфейса не находится в локусе внимания пользователя и если интерфейс может работать в разных режимах, то пользователь будет иногда совершать ошибки, поскольку текущий режим не находится в локусе его внимания.

Приведем другой пример. В некоторых популярных моделях компьютеров для создания нового документа применяется следующее сочетание клавиш:

Command ↓ *n* ↓ ↑ ↑

Здесь n обозначает *новый* (new). Но если взять пакет электронной почты America Online, то в нем для получения нового бланка сообщения требуется использовать другую комбинацию клавиш:

$Command \downarrow m \downarrow \uparrow \uparrow$

Я предполагаю, что m здесь обозначает *почта* (mail). Ошибка, которую совершают пользователи при создании нового сообщения, заключается в нажатии неверной комбинации клавиш:

$Command \downarrow n \downarrow \uparrow \uparrow$

Здесь состояние интерфейса, в котором возникает режим, зависит от активного приложения. Проблема возникает в тот момент, когда пользователь применяет по привычке команду $Command \downarrow n \downarrow \uparrow \uparrow$. Начиная с этого момента он делает ту же самую ошибку, но, вероятно, по другим причинам. Он может подумать, что команда $Command \downarrow n \downarrow \uparrow \uparrow$ работает одинаковым образом во всех приложениях, и поэтому совершит ту же ошибку по незнанию.

Норман (1983) указывает три метода предотвращения модальных (т. е. связанных с режимами) ошибок:

1. Не использовать режимы.
2. Обеспечить четкое различие между режимами.
3. Не использовать одинаковые команды в разных режимах, чтобы команда, примененная не в том режиме, не могла привести к неприятностям.

Из приведенных трех методов только первый позволяет полностью избежать модальных ошибок. Что касается второго метода, то, как мы могли уже убедиться, он не всегда работает. Третий метод не сокращает количество ошибок, но позволяет уменьшить их негативные последствия.

В самом крайнем случае можно привлекать внимание пользователя к индикатору текущего режима, но это так же нежелательно, как и сама модальная ошибка, для предотвращения которой индикатор предназначен, поскольку локус внимания пользователя будет переключаться с задачи на текущее состояние системы. Норман определяет модальные ошибки как ошибки, возникающие в результате того, что пользователь неверно классифицирует или анализирует ситуацию (Norman, 1981). Термины *неверно классифицировать* и *анализировать* в данном случае указывают на активное, сознательное участие со стороны пользователя и, таким образом, могут применяться до тех пор, пока он еще не в полной мере знаком с той или иной командой, но не тогда, когда применение этой команды стало для него автоматическим.

3.2.1. Определение режимов

Если бы определение режима основывалось исключительно на конструкции интерфейса (как мы это до сих пор допускали в нашем изложении), тогда все пользователи делали бы одинаковые ошибки, хотя и с разной частотой. Однако это не так. Данный элемент интерфейса может для одного пользователя быть модальным, а для другого – нет. Более полное определение режима должно включать в себя то, как пользователь рассматривает интерфейс: *интерфейс «человек-машина» является модальным по отношению к данному жесту, если, во-первых, текущее состояние интерфейса не находится в локусе внимания пользователя и, во-вторых, если в ответ на некоторый жест интерфейс может выполнить одно из нескольких возможных действий в зависимости от текущего состояния системы.*

Интерфейс может быть модальным по отношению к одному жесту и не быть таковым по отношению к другому. Чтобы можно было классифицировать тот или иной интерфейс как истинно немодальный, он не должен быть модальным для любого жеста.

Степень модальности Q того или иного интерфейса может быть определена через классификацию каждого жеста, допустимого в интерфейсе, как модального или немодального. Затем, с учетом вероятности применения данного немодального жеста N_i , выраженной как $p(N_i)$ и вычисленной для данного пользователя или в среднем для группы пользователей, получаем $Q = \sum p(N_i)$. Диапазон значений Q варьируется от 0 (полностью модальный) до 1 (полностью немодальный).

Обе части определения модального жеста необходимы для решения о том, является ли, например, жест нажатия клавиши <Backspace> модальным или нет. В большинстве компьютерных интерфейсов, если вы набираете какой-то текст, с помощью команды Backspace вы можете удалить символ, который был введен в самую последнюю очередь. Если это был символ e , то нажатие клавиши удалит символ e . Если же это был символ x , то нажатие клавиши удалит символ x . Другими словами, в одном случае клавиша <Backspace> служит для удаления символов e , а в другом случае – для удаления символов x . Если рассматривать только вторую часть определения, использование клавиши <Backspace> является модальным, так как то, какой именно символ она удаляет, зависит от того, какой символ был введен последним, т. е. содержание характеризует состояние системы. Однако в том случае, когда вы осознаете, что удаляемый вами объект находится в локусе вашего внимания, в соответствии с первой частью определения данная операция не является модальной, и поэтому вы не делаете модальных ошибок, используя клавишу <Backspace> для удаления из текста каких-либо символов или любых других объектов, которые можно выделить и удалить.¹

¹ На компьютере Macintosh соответствующая клавиша названа более точно, Delete (удалить).

С другой стороны, команда, которая устанавливает для клавиши <Backspace> направление удаления (вперед или назад), делает интерфейс модальным. Ранее установленное направление удаления обычно уже не находится в локусе вашего внимания, когда впоследствии вы используете клавишу <Backspace>, поэтому иногда удаление будет происходить в неожиданном направлении.

Предотвращение авиакатастроф через устранение режимов

Время от времени из-за простых ошибок в интерфейсах происходят крушения как дистанционно управляемых летательных аппаратов (RPV, remote-piloted vehilces), используемых для военных нужд, так и гражданских радиоуправляемых моделей самолетов. Чтобы понять ошибку и определить метод ее исправления, следует, прежде всего, знать кое-что о том, как эти устройства работают.

При дистанционном управлении летательным аппаратом оператор, находящийся на земле, манипулирует небольшим джойстиком, встроенным в пульт управления (рис. 3.2). Как правило, для того чтобы временно поднять нос аппарата, требуется потянуть джойстик на себя, а для того чтобы опустить нос, требуется толкнуть джойстик от себя. Для наклона вправо и влево джойстик, соответственно, перемещается вправо или влево. Перемещение джойстика вызывает пропорциональное движение устройства, называемого сервомотором. Сервомотор механически соединен с управляющей плоскостью, например с рулем высоты в хвостовом оперении самолета, позволяющим управлять его продольным наклоном.

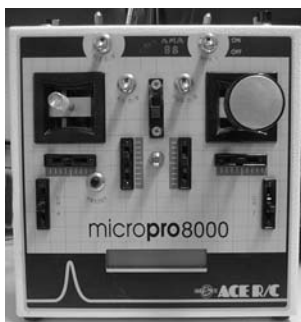


Рис. 3.2. Пульт дистанционного управления гражданского летательного аппарата. Обратите внимание на множество элементов управления, расположенных по всей лицевой панели пульта и в его верхней части

Сервомотор не имеет стандартного направления движения относительно движения джойстика управления, поэтому принято на пульте устанавливать переключатель, который позволяет выбрать, какое направление движения джойстика будет соответствовать тому или иному направлению движения сервомотора. Это явно модальное действие вступает в противоречие с привычными реакци-

ми пользователя, как это обычно и происходит при наличии режимов. Следует сказать, что для управления несколькими разными летательными аппаратами зачастую используется один пульт. В каждом аппарате может быть установлено свое направление движения сервомоторов, от которого зависит поведение самолета в воздухе. При переключении управления между аппаратами оператор должен проверять положение переключателей, регулирующих направление движения сервомоторов, чтобы на перемещение джойстика летательный аппарат отвечал соответствующими действиями.

Почти невозможно научить пилота управлять прибором, когда направление движения сервомоторов заранее не известно. Например, представьте себе, что пилот, который готовится выполнить взлет дистанционно управляемого летательного аппарата, не знает о том, что переключатель направления наклона установлен в неверную позицию. Взлет является очень сложным маневром, требующим большого внимания. Обычно при взлете самолет спонтанно слегка заваливается влево или вправо в момент отрыва от земли, и опытный пилот автоматически подает небольшой сигнал на компенсацию этого заваливания. У хорошего пилота эта реакция настолько быстрая, что, наблюдая со стороны, можно вообще ничего не заметить. Если переключатель направления работы сервомоторов установлен в обратное положение, то заваливание, естественно, станет еще больше. В этом случае хорошо тренированный пилот сделает еще большее компенсирующее отклонение джойстика, тем самым только усугубляя заваливание. В конце концов, полет обычно заканчивается разрушением самолета, как только его крыло касается земли или когда самолет полностью переворачивается на спину. (Все эти события могут произойти за доли секунды.) Я еще не видел ни одного пилота, который смог бы быстро догадаться о возникшей проблеме и вовремя изменить свои реакции на обратные, чтобы безопасно закончить взлет. Такого рода крушения происходили при мне два раза, несмотря на обязательную проверку правильности положения переключателя, регулирующего движение управляющих плоскостей. Проверка положения переключателя, когда она выполняется десятки раз, тоже становится привычкой вплоть до того, что вы можете не заметить, действительно ли управляющие плоскости движутся в правильном направлении. (Если они не будут двигаться вообще, то, естественно, это сразу привлечет ваше внимание к наличию проблемы.) Тем не менее, во время таких проверок я сам довольно часто обнаруживал, что аппарат, который я собирался поднять в воздух, имел обратное направление движения сервомоторов. Однако, испытывая недостаток времени или отвлекаясь на другие дела, я, как и сотни других пилотов, не замечал эту неправильность.

Интересно отметить, что такая ошибка, как правило, не происходит при первом полете на новом аппарате, когда вы особенно внимательны к возможным проблемам. Только после того, как полеты на

каком-то аппарате становятся «второй натурой» или просто привычным делом, она обычно и случается.

Этот пример демонстрирует случай, когда режим невозможно устранить. Режимы изначально присущи сервомоторам и джойстикам управления. Тем не менее, из данной ситуации есть выход. В качестве решения можно предложить следующее: поместить переключатель направления работы сервомоторов внутрь самого летательного аппарата, возможно, сделать его частью самих сервомоторов, и убрать этот переключатель с пульта управления. Обычно внутренняя часть корпуса самолета недоступна, поэтому переключатели могут устанавливаться в нужное положение при сборке аппарата. Правильность настройки будет проверяться сборщиками самолета, затем инспекторами и затем еще раз (как обычно) пилотом перед первым полетом. Хотя этот метод не устраняет полностью возможности крушения, но если первый полет выполнен успешно, то в дальнейшем проблема переключения сервомоторов уже не должна возникать на этом аппарате независимо от того, используется ли данный пульт для управления и другими летательными аппаратами. С точки зрения оператора, режим устранен.

Почти все пульта дистанционного управления моделями летательных аппаратов снабжены переключателями направления работы сервомоторов. Аварии из-за того, что пульт управления находится в неверном режиме, случаются нередко. Еще большие проблемы создает возможность устанавливать на пультах управления и другие режимы с помощью переключателей с длинными ручками, которые можно переставить в неверное положение случайно. В пульте управления, разработанном мною, все эти тумблеры сделаны таким образом, что перед тем как переставить их в другое положение, их ручки требуется сначала вытянуть из пульта (рис. 3.3). Поэтому у меня ни разу не было аварий из-за того, что какой-то тумблер был случайно сдвинут в неверное положение. Таким образом, иногда проблемы с режимами могут быть решены механическим путем.



Рис. 3.3. Такие тумблеры нельзя переключить в другое положение случайно. Перед тем как изменить положение тумблера, его ручку сначала требуется немного вытянуть из пульта

Режимы также ограничивают диапазон действий пользователя. Если жест g вызывает действие a в режиме A , а в режиме B он вызывает дей-

ствие b , то для того чтобы выполнить действие a , необходимо сначала выйти из режима B (если вы в нем находились) и переустановить интерфейс в режим A . И только после этого вы сможете использовать жест g для выполнения действия a . Разделение интерфейса на ограниченные области является неизбежным следствием наличия режимов. Набор состояний, в которых жест g имеет конкретную интерпретацию, можно назвать **диапазоном** (range) жеста g . Программное обеспечение, которое продается в виде прикладных программ, например электронные таблицы, обычно включает один или несколько пересекающихся диапазонов. Некоторые диапазоны относительно велики. Например, следующая последовательность выполняет действие вырезания почти во всех приложениях как на платформе Macintosh, так и в Windows:

Command↓ x ↑

Бывают и небольшие диапазоны. Например, нижеприведенная последовательность позволяет в некоторых компьютерных играх открывать сундук с сокровищами, если только он находится в пределах видимости:

Command↓ h ↑

Группирование команд по разным диапазонам, или, как мы это еще называем, по *приложениям*, позволяет понять и научиться использовать сложные интерфейсы. Тем не менее, существует возможность организовывать интерфейсы, которые могли бы создавать меньше ограничений, чем режимы. *Полностью человекоориентированный интерфейс должен состоять только из одного диапазона.*

В случаях когда интерфейс управляется с помощью другого компьютера, можно было бы подумать, что проблема наличия режимов снимается, поскольку машина может легко запомнить необходимое состояние интерфейса, просто-напросто внутренне синхронизируя его с собственным состоянием. Тем не менее, если интерфейс является модальным, и программа, управляющая этим интерфейсом, изначально не получает сведений о текущем состоянии интерфейса, — а это может произойти, если программа подключается после того, как система была запущена, — то для управления в каком-то из режимов системы программа должна быть оснащена средствами тестирования ее текущего состояния. В этом отношении особые трудности создают такие интерфейсные переключатели, для возвращения которых в исходную позицию требуется сделать подряд несколько переключений, изменив состояние системы (затем цикл повторяется сначала).

Случай, когда интерфейс управляется внешней программой, относится к вопросу разработки человеко-машинных интерфейсов, поскольку набор сохраняемых команд (называемый макросом), который может быть выполнен одним жестом, является рудиментарной формой компьютерной программы. Макрос не может установить такой переключатель в какое-либо из его допустимых состояний, если сам макрос сначала не задаст системе вопрос о ее текущем состоянии. Об этой

проблеме мы уже говорили на примере фонарика в сумке. Одно из возможных решений состоит в том, чтобы обеспечить установку переключателя, имеющего несколько положений, в некое начальное состояние сразу после каждого случая его использования. В результате этого подсчет числа переключений позволит всегда определить текущее состояние переключателя. Если предполагается, что переключатель будет использоваться человеком, то целесообразно предусматривать не более 5 состояний переключателя. Другим решением может быть применение набора переключателей (radio buttons).

Тем не менее, на этом список проблем, к которым могут приводить режимы, не исчерпывается. Ко всему прочему, режимы могут еще лишать пользователя возможности взаимодействия с системой. Это особенно заметно в том случае, когда вы вынуждены прервать свою работу, чтобы ответить на возникшее окно сообщения. Некоторые разработчики считают, что принуждение пользователя остановиться или работать в жестких рамках установленной последовательности действий оказывается полезным, так как позволяет системе самой «направлять» действия пользователя. При некоторых обстоятельствах важно, чтобы пользователь принял то или иное конкретное решение к какому-то моменту времени или перед выполнением следующего шага из последовательности. Если же пользователь не имеет возможности принять свое решение, то и необходимости диалога с ним нет. В первом случае достаточно только поместить на экране часы с обратным отсчетом, но не следует ограничивать пользователя в том, чтобы в это время он мог совершать в системе и другие действия. Во втором случае пусть на экране появляется сообщение, в котором говорилось бы, что перед выполнением очередного шага необходимо принять следующее решение, но система не должна препятствовать пользователю в выполнении других операций, не относящихся к текущей программной последовательности. Необходимо учитывать, например, что для принятия решения пользователю требуется просмотреть какой-то файл или выполнить какие-то вычисления. Другими словами, запросы и подсказки должны даваться не модально, а так, чтобы пользователь мог в максимальной степени сохранять контроль над системой.

3.2.2. Режимы, пользовательские настройки и временные режимы

Возможности установки пользовательских настроек являются одним из примеров режимов и представляют собой большой источник расстройств для пользователя. Как ни странно, но такие возможности обычно подаются как некие преимущества для пользователя. Существующие сегодня интерфейсы зачастую оказываются настолько трудными в применении, что сам пользователь может чувствовать желание как-то перестроить их. Однако Microsoft (1995, с. 4) особенно рекомендует снабжать интерфейсы такими функциями. «Поскольку пользователи обладают довольно разными навыками и предпочтениями, они

должны иметь возможность изменять параметры интерфейса (например, цвет, шрифт и др.) на свое усмотрение.» С другой стороны, одна из пользователей Microsoft Word сравнивала установку личных настроек с установкой часовой бомбы. Речь шла о том, что ей необходимо было сделать список в формате, отличающемся от того, который она обычно использовала. Для этого она прочитала справку о том, как делать такие изменения, и затем выбрала соответствующие настройки. В следующий раз, когда ей понадобилось сделать список, она воспользовалась знакомой ей командой List (Список) и, естественно, получила тот формат, на который она переустановила текстовый процессор, а не тот, который использовала обычно. Ей понадобилось больше часа, чтобы разобраться в том, что произошло, и исправить последствия. (В первый момент она подумала, что возникли неисправности в программе или что она неправильно применила команду, поэтому она повторила ее много раз, прежде чем вспомнить, что до этого изменила настройки.)

Боб Фауэлз (Bob Fowels) из компьютерного центра Пенсильванского государственного университета отмечает:

Пользователи, которые не осознают всей сложности программы Word, могут столкнуться с трудностями, если во время быстрой работы с клавиатурой случайно нажмут клавишу <Command>, <Option> или <Ctrl> в сочетании с какой-то другой клавишей. Вчера, например, моя жена столкнулась с проблемой, с которой мне пришлось повозиться не одну минуту. Всякий раз, когда она нажимала клавишу <Return>, на экране возникал маркированный список. Взглянув на выпадающее меню Правка, я увидел надпись: «Отменить автоформат». После нескольких минут поиска и чтения Справки я-таки нашел место управления функцией автоформатирования и отключил ее. То есть каким-то образом она умудрилась использовать сочетание клавиш, которое включило эту функцию (личное сообщение, 1998).

Пользователь пострадал сразу и от пользовательской настройки, и от режима, и от сочетания клавиш, и от излишней сложности интерфейса.

Пользовательские настройки – это такие изменения в программном обеспечении, которые не отражаются в документации. Например, во время использования Word я попытался отключить функцию, которая была мне неизвестна. В Справке я узнал, что мне следовало всего лишь щелкнуть по соответствующей кнопке на стандартной панели инструментов. Однако человек, который пользовался программой до меня, изменил эту панель инструментов с помощью окна пользовательских установок, и поэтому необходимой кнопки на ней не было, и мне потребовалось много времени, чтобы разобраться, как наладить работу программы нужным мне образом. Этот инцидент указал на еще более важную проблему применения пользовательских настроек, а именно: как можно протестировать качество интерфейса или написать документацию для системы, если ее конфигурация разработчикам не известна заранее? В описанном мною случае изменение режима, произведенное предыдущим пользователем, сделало документацию неверной.

В результате возможность настройки параметров интерфейса зачастую приводит к изменениям, которые не являются оптимальными, поскольку пользователь не всегда может обладать компетентностью специалиста по разработке интерфейсов. Чаще всего пользователь будет выбирать тот метод, который наиболее близок к уже известному ему методу, или настройку, которая требуется ему только на время. Некоторые разработчики утверждают, что для достаточно опытных пользователей следует предусматривать как можно больше настроек, которые они могут использовать для изменения системы под свои нужды. Однако нельзя сказать, что опытные пользователи вместе с тем являются и хорошими разработчиками интерфейсов, и, имея привычки в использовании данного программного продукта, они особенно заинтересованы в стабильной работе системы так, чтобы их привычки не потеряли основу после внесения в систему каких-либо изменений, пусть даже если они вносятся самими пользователями.

Снабжая программу настройками, мы обременяем пользователя задачей, которая не относится к его рабочим функциям. Пользователю, скажем, электронных таблиц приходится изучать не только эти электронные таблицы, но и опции по их настройке. *Время, которое тратится на изучение и выполнение пользовательских настроек, большей частью является потерянным с точки зрения текущей задачи.* Менеджеры часто жалуются на то, что работники тратят время на «игры с настройками». Большинство пользователей просто хотят выполнить свою работу, и их не волнует, какой шрифт по умолчанию принят в электронных таблицах для цифр: розовый Palatino, зеленый Garamond или полужирный разреженный наклонный Bodoni синего цвета.

Установка персональных настроек в совместно используемой среде чревато серьезными сбоями, так как это означает, что в интерфейс могут быть незаметно внесены изменения. В результате вчера правильным действием было нажатие на красную кнопку, а сегодня – на синюю, так как кому-то показалось, что синий цвет красивее. Ваши навыки и привычки становятся ненужными. К тому же, пользователю системы с личными настройками намного труднее помочь по телефону или по электронной почте.

Персональные настройки кажутся удобными, демократичными, допускающими расширения, полными свободы и радости для пользователя, но, тем не менее, я не знаком ни с одним исследованием, в котором говорилось бы о повышении продуктивности, а также *объективном* улучшении юзабилити и облегчении в изучении системы. Добавление функций пользовательских настроек, безусловно, усложняет систему и делает ее более трудной для изучения. Могу предположить, что если вы проведете опрос пользователей, то большинство из них будет в восторге от большого количества параметров, настраиваемых самим пользователем. Однако здесь следует сказать, что когда только появились первые графические пользовательские интерфейсы, большинство пользователей утверждало, что никогда не захотят ими поль-

зоваться. Также важно отметить, что пользователи настраивают интерфейсы в соответствии со своими субъективными представлениями. Наблюдения, сделанные во многих экспериментах, показывают, что *интерфейс, который оптимизирует продуктивность, не обязательно является интерфейсом, который оптимизирует субъективные оценки* (например, см. Tullis, 1984, с. 137).

Основная мысль этого утверждения заключается в том, что если мы являемся опытными разработчиками интерфейсов и можем в максимальной степени оптимизировать данный интерфейс, то пользовательские настройки не могут только ухудшить работу этого интерфейса. Поэтому следует с осторожностью предоставлять пользователю возможности по установке личных настроек. Если пользователь может действительно улучшить работу интерфейса, внося в него всего лишь несколько полезных изменений, – это значит, что, вероятно, мы плохо сделали свою работу.

С другой стороны, если интерфейс какой-то программы – как было однажды сказано – «такой же унылый, как и интерфейс в Microsoft Word 97/98», то ситуация становится обратной. Почти любое изменение, произведенное пользователем, становится почти без преувеличения улучшением. Тем не менее, интерфейс Word – это не тот пример, на который нам следует равняться.

Режимы, которые исчезают после однократного применения, создают меньше ошибок, чем те, которые работают постоянно, просто потому, что они имеют меньше времени на то, чтобы эти ошибки вызвать. В уже описанном примере о курсорах в пакете Vellum, если бы курсор автоматически возвращался в нормальную форму и к его нормальной функции после выполнения функции трассирования, это приводило бы к меньшим ошибкам. Если пользователь применяет временный режим сразу после его включения, факт включения этого режима еще не вышел из кратковременной памяти пользователя, поэтому, скорее всего, он не сделает какой-нибудь модальной ошибки. Установку режима можно даже сделать частью жеста, который выполняет данную команду, и в этом случае ситуация станет полностью немодальной для вас. Однако если вы установите режим для некоторой команды и до ее выполнения отвлечетесь или по какой-то причине задержитесь, то, вероятнее всего, вы совершите модальную ошибку.

С целью избежать режимы, компьютер Canon Cat был спроектирован без кнопки включения питания.¹ Если продукт реагирует на жесты в зависимости от того, включен он или нет, то, следовательно, кнопка

¹ Я был в шоке, обнаружив кнопку питания с задней стороны системного блока на компьютере Canon Cat, когда его первые экземпляры пришли из Японии. Когда я указывал на то, что в технической документации эта кнопка даже не упоминается, в ответ я многократно получал от инженеров только одно-единственное объяснение: «Мы думали, что в технических требованиях содержалась ошибка».

включения питания вводит режим (модальность). Для экономии энергии Canon Cat автоматически переходит в режим пониженного энергопотребления, если в течение 5 минут компьютер не используется. Во избежание превращения этого состояния в режим, было предусмотрено, чтобы любое действие пользователя или любое входящее сообщение без заметной задержки возвращало компьютер в обычное состояние. Кроме этого, действие пользователя не только «оживляло» компьютер, но и выполнялось так же, как если бы машина не находилась в режиме ожидания.

Во многих системах при нажатии любой клавиши компьютер выходит из режима ожидания, но эти нажатия, а также все последующие, сделанные до того, как система полностью «проснется», не производят действия, для которого они были предназначены. Возможность не терять никакие нажатия клавиш было довольно элегантным решением. Например, если у вас случилось вдохновение или вам потребовалось написать записку во время телефонного разговора, вы могли просто начать печатать, не заботясь о том, в каком состоянии находится Canon Cat, или даже не глядя на монитор. Отсутствие режимов (немодальность) характерно тем, что когда у вас появляются привычки в пользовании интерфейсом, вам не нужно задумываться или планировать выполнение того или иного действия, поэтому внимание целиком сосредоточено только на содержании вашей работы. (Если же случилось так, что записку вы стали набирать прямо в середине какого-нибудь другого текста, вы сможете просто выделить ее и переместить в другое место после того, как увековечите свое вдохновение или закончите телефонный разговор.)

Иногда разработчики говорят, что режимы нужны потому, что число необходимых программных функций превышает число жестов, которые пользователь может выполнить с помощью клавиатуры и графического устройства ввода, поэтому жесты должны использоваться многократно. Однако число команд, исполняемых с помощью монитора (например, меню), и команд, вводимых с клавиатуры в командную строку и содержащих множество символов, может быть неограниченным и потому позволяет избежать упомянутой трудности. (О том, как сделать видимыми различные команды, доступные из командной строки, вместо того чтобы запоминать их, мы поговорим позже.)

Можно сделать следующий вывод относительно режимов: *если вы разрабатываете модальный интерфейс, учитывайте, что пользователи будут всегда совершать модальные ошибки за исключением тех случаев, когда значение состояния, контролируемого данным режимом, находится в локусе внимания пользователя (и он может его видеть) либо в его кратковременной памяти. Задача разработчиков состоит в том, чтобы показать, что данный режим используется правильно или что преимущества данного режима перевешивают его неизбежные недостатки.* Тем не менее, безопаснее всегда избегать применения режимов в разработке интерфейсов.

Кнопки, которые меняются за одну ночь

На некоторых самолетах используются кнопки, на которых имеется небольшой точечный дисплей, отражающий текущее назначение кнопки. Изменение назначения кнопки контролируется бортовым компьютером в течение всего полета. Рекламный вариант одной из моделей таких кнопок показал их преимущества с точки зрения так называемого человеческого фактора. Во-первых, при использовании этих кнопок требуется их меньшее количество, что важно, так как площадь кабины самолета ограничена. Во-вторых, эти кнопки позволяют вносить изменения в бортовое электронное оборудование и другие системы самолета без необходимости переделывать электропроводку внутри кабины.

С точки зрения когнетики, чем понятнее надписи, тем лучше их видно. На первый взгляд кажется, что человек, который отбирает необходимые надписи, должен провести тестирование словесных формулировок, которые в них используются, и убедиться в том, что отобранные символы являются эффективными и понятными.

Однако при более тщательном размышлении выясняется, что эти кнопки могут представлять некоторые потенциальные проблемы. Следует учитывать, что надпись на кнопке закрывается пальцем как раз в момент нажатия на нее, тем самым не давая в самую последнюю секунду проверить, что это кнопка – правильная. Но это еще не самая большая проблема, поскольку перед тем как нажать кнопку, вы обычно смотрите на надпись на ней, – если вообще куда-нибудь смотрите. (Конечно, я не смотрю на обозначения на кнопках, когда сейчас набираю текст на клавиатуре.)

Есть еще и другая, более глубокая проблема, касающаяся «исчезновения» кнопок. Например, вы хотите включить ручное управление кондиционированием в салоне. Вы вспоминаете, что есть кнопка с надписью **MANUAL AIR** (Ручное управление кондиционером), но нигде не можете ее найти. Дело в том, что на ней стоит другое обозначение – **COMM BACKUP** (Резервная связь). Поэтому вам необходимо подумать, как вернуть то обозначение, которое было раньше на этой кнопке. Возможно, где-то установлен еще один тумблер или кнопка, с помощью которых обозначения на кнопках можно изменить. Или, вероятно, обозначения являются контекстными и поэтому не позволят вам включить кондиционирование прямо сейчас. Как бы то ни было, когда кнопки могут иметь изменяемые назначения, то нужная вам кнопка может исчезнуть, если система окажется в другом режиме.

Но самая серьезная проблема связана с привычками. Представьте, что опытный пилот протягивает руку, чтобы нажать какую-то кнопку. Радиосвязь не работает из-за каких-то неполадок. Необходимо включить резервную связь. Пилот сразу нажимает кнопку

СОММ ВАСКУР. Щелк! Но ничего не выходит. Дело в том, что незадолго до этого второй пилот по просьбе пассажира регулировал температуру в салоне, и поэтому кнопки находятся в режиме регулирования климата. То есть нажатием этой кнопки командир корабля просто включил ручное управление системы кондиционирования корабля.

Кнопки с изменяемыми названиями могут быть полезны, например, когда консоль используется одновременно несколькими людьми таким образом, что кнопки сохраняют свои обозначения только для каждого отдельного пользователя. Но такая ситуация встречается редко. Программируемые клавиши, которые могут иметь разные назначения, влияющие на экранные кнопки, или кнопки настройки дисплея имеют те же слабые места, что и кнопки с изменяемыми значениями. Сходная проблема возникает при использовании функциональных клавиш, которые на многих компьютерах обозначены символами от F1 до F12. Если бы их функции оставались неизменными, то такое название, конечно, являлось бы неудобным. Если же их функции изменяются, то вы не можете пользоваться этими клавишами автоматически. В любом случае их можно назвать неудачным решением.

Чем меньше кнопок, тем лучше?

Переносной осциллограф Флюка¹, который я недавно тестировал (рис. 3.4), кажется невероятно мощным и надежным устройством. 35 кнопок, которыми он оснащен, контролируют несчетное количество режимов и функций. Мне было трудно научиться пользоваться этим прибором, и оказалось, что с ним невозможно работать быстро, если пытаться работать со многими его функциями.



Рис. 3.4. Измерительный прибор Флюка. Мощный прибор с простым дизайном, с бесчисленным множеством функций и малым количеством кнопок, которым трудно научиться пользоваться и в котором иногда бывает трудно добраться до необходимой вам функции. С другой стороны, прибор снабжен щупами разных цветов, что исключает традиционный источник путаницы относительно того, какой щуп для чего предназначен

¹ Благодарю компанию John Fluke Manufacturing Co. за предоставленный экземпляр прибора.

Осциллограф Tektronix (рис. 3.5) с мириадами настроек, которые на первый взгляд выглядят отпугивающими, все же легче и изучить, и использовать. Количество переключателей и ручек почти такое же, как и у цифрового прибора, но многие из управляющих ручек имеют множество настроек, которые доступны сразу. Кроме того, настройки на передней панели обозначены таким образом, что сразу заметны, а не спрятаны в меню, которые требуется еще вызвать, чтобы увидеть. Прибор Флюка предлагает гораздо большую функциональность, но за счет усложнения некоторых задач, которые раньше были довольно простыми. Поворачивающиеся ручки, конечно, являются намного более простыми в использовании, чем любая из кнопок в приборе Флюка. Следует еще добавить, что Флюк перестарался в создании слишком ветвящейся структуры подменю.

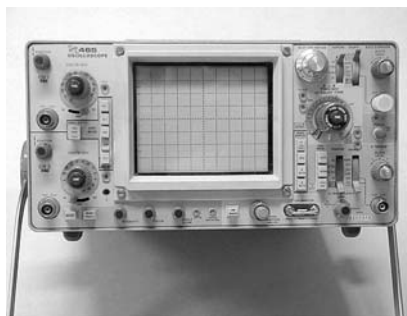


Рис. 3.5. Осциллограф Tektronix снабжен мириадами настроек, и хотя в первый момент и кажется очень сложным, с ним, тем не менее, можно легко и быстро работать. Все ручки, которые могут изменить настройку прибора, обозначены красным цветом

По относительной легкости использования эти осциллографы очень напоминают мне два радиоприемника. В моей авторadioле (рис. 3.6) имеются предварительные настройки для 18 радиостанций, сгруппированные в 3 группы по шесть станций каждая. С помощью специальной кнопки можно переходить от одной группы к другой, а ряд из 6 кнопок соответствует станциям в каждой группе. На жидкокристаллическом экране отображается номер группы (от 1 до 3). Таким образом, семи кнопок оказывается достаточно для выбора 24 станций.



Рис. 3.6. Радиоприемник в моей машине довольно трудно использовать по назначению. Кроме того, почти невозможно запомнить, как в нем устанавливаются часы

Другой мой радиоприемник — это превосходный электронный прибор Sony 2010 (рис. 3.7), в котором можно иметь 32 предварительные настройки, закрепленные за 32 одинаковыми кнопками, выстроенными по 8 кнопок в 4 ряда. На первый взгляд этот радиоприемник кажется более сложным, но, как показали мои опросы вла-

дельцев таких радиоприемников и мой продолжительный личный опыт использования, такая конструкция позволяет с большей легкостью найти настройку на необходимую станцию, особенно в темноте. Для этого следует всего лишь отсчитать вниз ряд, в котором находится нужная кнопка, и затем, так же отсчитав ее номер в ряду, нажать на нее. В устройстве запоминаются все настройки, связанные с каждой станцией, в том числе и то, в каком диапазоне она находится (FM, AM, короткие волны, авиадиапазон (aircraft band)) и другие параметры, которые вы установили. При нажатии одной из 32 кнопок все настройки, требуемые для воспроизведения станции, устанавливаются сразу и автоматически. Те станции, которые я слушаю чаще всего, я закрепил за кнопками, находящимися ближе к левому верхнему углу.



Рис. 3.7. В коротковолновом радиоприемнике Sony 2010 имеется много кнопок, но можно легко и быстро научиться пользоваться этим прибором

В автомобильном радиоприемнике удобство использования жертвуется в пользу меньшего числа кнопок и небольшого размера лицевой панели. Худшим свойством конструкции этого приемника является то, что для того чтобы узнать, какой из рядов предварительных настроек сейчас активен, вы вынуждены отрывать взгляд от дороги, иначе вы не найдете нужной вам станции. (Чтобы переключиться с AM на FM или наоборот, требуется найти и нажать две кнопки, тогда как в приемнике Sony достаточно нажать только одну кнопку, чтобы перейти на заранее настроенную волну.)

Не будет большим преувеличением сказать, что использование модальных интерфейсов в автомобилях должно быть запрещено. Очевидно, что мой автомобильный радиоприемник имеет неудовлетворительный интерфейс с точки зрения поиска и выбора радиостанций. Еще более неудовлетворительным является способ настройки часов, который предусмотрен в этом радиоприемнике. Чтобы настроить часы, мне требуется постоянно смотреть на них: сначала необходимо нажать и удерживать кнопку DISP в течение 2 секунд, чтобы переключиться в режим настройки. Для выбора часа требуется повернуть ручку настройки радиоприемника против часовой стрелки, а для выбора минут – по часовой стрелке. Чтобы закончить настройку часов, следует нажать кнопку DISP. (Эта же кнопка служит для регулировки громкости.) Плюсом здесь является то, что

в моих радиоприемниках по крайней мере есть нормальные ручки для регулировки звука, а не те кнопки, которые предназначены для этого в некоторых более современных приборах, потому что с помощью этих кнопок можно довольно долго настраивать звучание на требуемую громкость, не говоря уже о том, что текущее значение громкости не видно на глаз и его нельзя определить на ощупь. В пользовательском тестировании, которое я недавно провел, каждый из 55 испытуемых предпочел для этой роли именно ручки.

Меньшее число кнопок и более простой вид лицевой панели не всегда являются лучшим решением. Это справедливо не только для графических интерфейсов, но и для инструментов и приборов.

3.2.3. Режимы и квазирежимы

Использование клавиши <Caps Lock> для набора заглавных букв существенно отличается от удерживания клавиши <Shift> для той же цели. Первый случай устанавливает режим, второй – нет. Ряд экспериментов, проведенных в университете г. Торонто, подтвердил, что удерживание кнопки в нажатом состоянии, нажатие на ножную педаль или любая другая форма физического удерживания интерфейса в определенном состоянии не приводит к возникновению модальных ошибок (Sellen, Kurtenbach и Buxton, 1992). Другие исследования показали, что в основе этого феномена лежат нейрофизиологические причины. Большая часть нашей нервной системы функционирует таким образом, что постоянный стимул порождает сигналы, которые со временем снижают свою способность привлекать наше внимание. Это снижение продолжается до тех пор, пока наша когнитивная система совсем не перестает получать никакие сигналы. Однако сигналы, которые сообщают нам о том, производят ли в данный момент наши мышцы усилие, не снижают уровня своего воздействия.

Включение и физическое удерживание того или иного элемента управления во время выполнения другого действия было названо *пружинным режимом* (spring-loaded mode) или *режимом с запертой пружиной* (spring-locked mode) (Johnson и Englebeck, 1989). Но такая терминология здесь не подходит, поскольку никакая физическая пружина может вообще не использоваться, клавиша или кнопка не запирается, и их удерживание не вызывает модальных ошибок. Выражение *режим, удерживаемый пользователем* (user-maintained mode), является более точным, но от него трудно образовать прилагательное. Для обозначения режимов, которые удерживаются пользователем кинестетически, я стал использовать термин *квазирежим* и его прилагательное *квазимодальный* (рис. 3.8).

Квазирежимы являются весьма эффективными с точки зрения устранимости режимов (Raskin, 1989). Однако излишнее использование квазирежимов может привести к абсурдным условиям, требующим запо-



Рис. 3.8. Нотр-Дамский горбун

минания десятков команд, например *Control*↓*Alt*↓*Shift*↓*Esc*↓*q*↑↑↑↑. Для сохранения эффективности число квазирежимов, скорее всего, должно быть от 4 до 7. Тем не менее, необходимо сказать, что один квазирежим может решить сразу множество различных проблем (см. раздел 5.4).

Типичная проблема, которая легко решается при помощи квазирежимов, возникает в тех случаях, когда интерфейс предлагает пользователю некий набор вариантов, например выпадающее меню в Macintosh. При таком применении квазирежимов вы нажимаете кнопку графического устройства ввода, и, пока вы удерживаете кнопку в нажатом положении, под названием меню появляется список вариантов. Вы перемещаете курсор на требуемый вариант и отпускаете кнопку, чтобы выбрать соответствующий пункт меню.

Другим случаем применения квазирежимов является циклический переход по некоторому списку опций. Если цикл начинается с одного и того же элемента списка и переход происходит в одном и том же направлении и если в списке содержится небольшое число элементов, то для выбора конкретной опции потребуется только определенное число нажатий. Например, в компьютере Canon Cat при форматировании выделенного текста пользователь может изменить стиль абзаца в пределах четырех вариантов: по левому краю, по центру, по правому краю или по ширине. Стиль абзаца определялся с помощью повторного нажатия на клавишу с надписью на фронтальной поверхности

¶ STYLE

Для того чтобы задействовать функцию клавиши, обозначенную на ее фронтальной поверхности, в компьютере Canon Cat была предусмотрена специальная клавиша <Use Front> (рис. 3.9).

В общем смысле клавиша <Use Front> позволяет вам активизировать функции, обозначенные не сверху, а на фронтальной поверхности клавиш. Стиль абзаца назначался в квазирежиме (вы нажали и удерживали клавишу <Use Front>, пока нажимали кнопку ¶Style), и поэтому компьютеру было ясно, сколько раз клавиша была нажата. В результате вы выучили, что одно нажатие выравнивало выделенный текст по левой стороне, два нажатия – по правой стороне, и т. д. Если бы цикл всегда начинался с той опции, которая использовалась последний раз, как это часто делается в интерфейсах, применение команды назначения стиля абзаца не могло бы стать привычным, и вам всегда приходилось бы смотреть на монитор, чтобы увидеть результат команды. Заметьте, что если бы нажатие на кнопку назначения стиля абзаца не требовало также и удерживания кнопки <Use Front>, это бы не работало правильно, потому что именно удерживание кнопки <Use Front> устанавливало квазирежим и позволяло системе определить начало отсчета.

Элементы интерфейса часто можно называть привычными в том случае, что они могут легко использоваться «слепым» пользователем. Интерфейсы, построенные на основе принципов, изложенных в этой книге, часто могут использоваться даже слепыми пользователями, – а по отношению к тому, что находится вне нашего локуса внимания, мы все являемся в самом прямом смысле слепыми.



Рис. 3.9. Клавиша <Use Front> и некоторые другие командные клавиши, используемые в компьютере Canon Cat. Надпись USE FRONT имеет светло-голубой цвет, так же как и надписи на фронтальной поверхности тех клавиш, которые могут быть задействованы с помощью клавиши <Use Front>

В некоторых меню или палитрах самый используемый элемент или тот, который был использован последний раз, помещается наверх, что является простым примером *адаптивного меню* или *адаптивной палитры*. Меню или палитра делаются адаптивными исходя из того предположения, что если оставлять часто используемый элемент на виду, без необходимости поиска его в меню, то это может ускорить работу пользователя. Здесь полезно сравнить два метода. В первом методе выбранный элемент *убирается* из общего списка и помещается в основную палитру или меню. Во втором методе выбранный элемент *копируется* в основную палитру или меню (рис. 3.10).

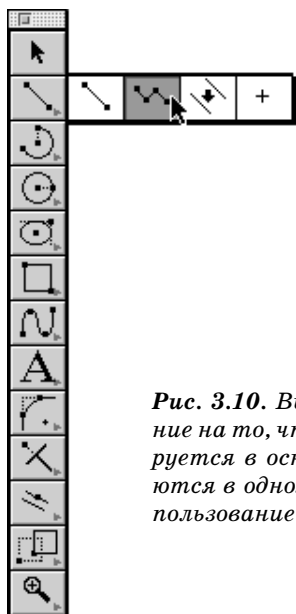


Рис. 3.10. Выдвигающаяся палитра Ашлара. Обратите внимание на то, что первый элемент из выдвигающейся группы дублируется в основном меню, поэтому все элементы группы остаются в одном и том же порядке, что позволяет сделать их использование автоматическим

Поначалу первый метод кажется лучше, потому что в списке остается на один элемент меньше, следовательно, пользователю придется сделать на один выбор меньше, а места на экране становится больше. Однако в этом случае пользователю придется останавливаться, чтобы проверить, где именно находится нужный ему элемент: в основном меню или в выдвигающемся, особенно если с момента последнего его использования прошло больше нескольких секунд. Но второй метод, при котором пользователь всегда знает, каким по счету является тот или иной элемент в списке, обычно оказывается эффективнее, поскольку с когнитивной точки зрения, если вы помните, что и где находится в основном меню, вы экономите время и не станете просматривать весь список элементов. Но если вы не помните, вы сможете воспользоваться списком как обычно.

Пакет Vellum (Ashlar, 1995) является примером продукта, в интерфейсе которого грамотно используются адаптивные палитры (рис. 3.10).

В этом пакете палитры действительно являются адаптивными, и каждый инструмент можно найти в ожидаемом месте. Таким образом, опытный пользователь может решить не пользоваться адаптивным меню и автоматически отправится к тому месту, где привычно располагается необходимый ему инструмент. В разделе 3.5 будет рассмотрен еще один аспект использования нескольких способов для выбора какого-либо инструмента.

Существует идея, по которой интерфейс должен адаптироваться в соответствии с эмоциональным состоянием пользователя, и в некоторых продуктах (например, Bob от компании Microsoft) действительно делается попытка приспособить интерфейс под характеристики личности пользователя. Однако пока непонятно, каким образом интерфейс может подстраиваться под пользователя, не нарушая процесса привыкания. И даже если когда-нибудь мы и сможем уверенно определять текущее эмоциональное состояние пользователя и использовать эту информацию для улучшения характеристик интерфейса, не нарушая системы автоматических реакций и навыков пользователя, интерфейсы все же должны будут обладать способностью удовлетворять те когнитивные потребности пользователя, которые являются неизменными. Поэтому принципы, о которых идет речь в этой книге, могут быть применены и, вероятно, должны быть применены еще до настройки интерфейса в соответствии с эмоциональными потребностями пользователя.

В основном существует только два типа сигналов, которые вы вводите в компьютер или другое устройство для обработки информации: те, которые служат для создания содержания, и те, которые используются для управления системой. В этом отношении предлагается следующий практический подход: *квазирежимы должны использоваться для управленческих функций, тогда как для создания содержания должны применяться операции без задействования квазирежимов*. Именно этот подход можно считать наиболее верным, поскольку управлять системой, удерживая кнопку для включения квазирежима, труднее. Однако важнее, чтобы пользователь мог больше времени уделить созданию содержания и работе с ним, а не применению различных команд для управления системой.

3.3. Модели «существительное-глагол» и «глагол-существительное»

Большой класс команд предусматривает применение некоторого действия к некоторому объекту. Например, при использовании текстового процессора вы можете выбрать какой-то абзац и изменить его шрифт. В этом случае объектом является абзац, а действием – выбор нового шрифта, при этом в интерфейсе могут использоваться две последовательности: либо (1) сначала выбор глагола (изменить шрифт), а затем выделение существительного (абзац); либо (2) сначала существительное, а потом глагол. На первый взгляд может показаться,

что оба варианта являются симметричными, и порядок не важен. Однако для большинства интерфейсов ситуация не является симметричной, и порядок (существительное-глагол или глагол-существительное)¹ имеет большое значение с точки зрения юзабилити интерфейса.

В большинстве руководств по разработке интерфейсов рекомендуется именно модель взаимодействия существительное-глагол (Apple, 1987; Hewlett Packard, 1987; IBM, 1988; Microsoft. 1995). Анализ, проведенный с точки зрения локуса внимания пользователя, показывает преимущества такой модели:

- *Уменьшение количества ошибок.* Последовательность глагол-существительное устанавливает режим. Если вы выбрали команду изменения стиля, то она будет сразу применена к тексту, который вы выделите. Если между назначением команды и выделением текста произойдет задержка или внимание пользователя отвлечется, но после того как текст будет выделен, результат может оказаться для него неожиданным. При использовании модели существительное-глагол команды выполняются сразу, пока еще находятся в локусе внимания пользователя.
- *Скорость.* Пользователю не требуется переключать свое внимание с содержания (которое и вызвало необходимость выполнения операции) к самой команде и затем опять к содержанию, чтобы можно было выделить необходимый участок текста. В модели существительное-глагол вы сначала выделяете текст, находящийся в локусе вашего внимания, и затем переключаете внимание на команду. Таким образом, число переключений локуса внимания уменьшается на единицу.
- *Простота и обратимость.* При использовании модели глагол-существительное должна быть предусмотрена возможность отмены или отката команды. Если пользователь назначает команду и затем решает изменить ее, следует учитывать, что он находится в этот момент в режиме, и система ожидает, что будет сделано выделение текста для применения уже назначенной команды. Поэтому должен быть предусмотрен механизм подачи системе сигнала о том, что вы не хотите выделять текст и хотите назначить другую команду. В модели существительное-глагол, для того чтобы выбрать другой текст, не требуется нажимать кнопку отката или применять какой-либо другой способ отмены действия.

Тем не менее, в каждом из увиденных мной руководств по разработке интерфейсов, в которых рекомендуется модель существительное-глагол, использование модели глагол-существительное также допускается. В руководстве Microsoft говорится, что модель глагол-существительное необходимо использовать для палитр, например, в выборе стиля

¹ Также используются другие термины: *объект-действие* и *действие-объект*.

кисти в графических редакторах (Microsoft, 1995). Вообще, это не совсем правильно. В данном случае также можно использовать чистую модель существительное-глагол. Вы можете рисовать с использованием набора параметров, принятого по умолчанию (например, тонкая линия черного цвета), и затем с помощью команд изменять цвет, ширину, текстуру и другие параметры линии. Однако на самом деле мы хотим сразу видеть, как выглядит каждый мазок, который мы делаем, со всеми его параметрами.

Общепринятый способ, при котором пользователь сначала выбирает атрибуты из одной или нескольких палитр, – аналогично тому, как вы можете перед началом рисования макать кисть в ту или другую банку с краской, – приводит к модальным ошибкам, которые, как мы уже видели, неизбежно должны здесь возникать. Имеет смысл делать так, чтобы выбранный режим сохранялся до тех пор, пока пользователь специально не сменит его. В этом случае вы можете начать рисование и неожиданно обнаружить, что для этого используются другие атрибуты, но, к счастью, результат рисования в данный момент находится в локусе вашего внимания. Программное обеспечение, которое в нашей терминологии мы можем назвать человекоориентированным, в этом случае позволит вам сразу отменить не устраивающий вас результат, изменить атрибуты и продолжить работу. Модальные ошибки всегда вызывают раздражение, поэтому модель существительное-глагол или другой немодальный метод был бы более эффективным в этой ситуации, но пока, насколько я знаю, никто еще не нашел удачного решения этой проблемы.

В целом, подход «существительное-глагол» является более предпочтительным. Применение методов «глагол-существительное» должно ограничиваться только выбором из палитр, если они предназначены для непосредственного использования.

Пример приведения проблемной модели «глагол-существительное» к модели «существительное-глагол»

Работники разных отделов одной транснациональной корпорации использовали специальную компьютерную систему для размещения заказов на товары. На ее примере мы рассмотрим один из способов перехода с естественной, на первый взгляд, модели «глагол-существительное» на модель «существительное-глагол». Такое изменение помогло на практике устранить те ошибки, которые возникали при использовании первого варианта интерфейса, и ускорило процесс выполнения заказа на товары.

В исходном варианте процесс состоял из трех шагов:

1. Необходимо выбрать отдел, для которого заказ был предназначен. Для этого следовало поставить флажок рядом с названием отдела. Один из флажков, соответствующий отделу, в котором находился данный компьютер, был установлен по умолчанию.
2. Требовалось выбрать необходимые товары из прокручиваемого списка. Рядом с каждым товаром находилось текстовое окно ввода, в которое можно было занести заказываемое количество каждого товара.
3. Необходимо щелкнуть по одной из двух кнопок внизу дисплея: либо «Отменить заказ», либо «Подтвердить заказ».

Первая проблема состояла в том, что пользователи не могли выбрать свой отдел, если они делали заказ не со своего рабочего места. Разработчики компании предложили отменить установку отдела по умолчанию, вынуждая пользователей каждый раз сначала выбирать отдел. Они заметили, что в этом случае единственно возможная ошибка связана только с выбором неправильного отдела. Тем не менее, такое решение раздражало тех пользователей, которые обычно работали за своим компьютером, и необходимость каждый раз вводить информацию, которая уже была доступна системе, вызвала раздражение.

По сути дела, проблемой являлась все та же ситуация «глагол-существительное». Сначала необходимо было ввести, что вы хотите сделать (доставить эти товары в этот отдел), и затем выбрать, что именно вы хотите доставить. Однако при открытии окна заказа в локусе вашего внимания находятся (или только что находились) товар или товары, которые вам нужны.

Часть решения заключалась в том, чтобы разместить список товаров наверху окна заказа. Таким образом, пользователь мог начать с выделения нужных товаров. А список отделов, который раньше не имел никаких надписей, получил заголовок в форме вопроса: «В какой отдел следует доставить заказанные товары?»

Если система могла определить собственный отдел пользователя (на основе журнала работы пользователя), в первую строку списка отделов автоматически добавлялся элемент под названием «Мой отдел», который так же, как и другие элементы списка, имел флажок.

Все другие отделы, расположенные в алфавитном порядке, составляли остальную часть списка. Рядом с каждым отделом находился флажок. При установке флажка появлялось сообщение: «Отмеченные товары заказаны», которое исчезало сразу после любого действия пользователя.

В измененном варианте интерфейса выбор отдела являлся действием, т. е. глаголом, после которого информация об отмеченных това-

рах отправлялась для обработки. Психологически это стало действием, завершающим всю операцию заказа, а не предваряющим его. Кроме того, следует отметить, что к этому моменту бланк заказа уже заполнен, и внимание пользователя сместилось с самих товаров на необходимость их доставки. Таким образом, ход работы стал соответствовать тому, куда обычно следует внимание пользователя.

Отпала необходимость в установке места доставки по умолчанию, так как интерфейс предоставлял четкое психологическое окончание процессов выбора товаров и отправки заказа. В начальном варианте интерфейса требовалось подтверждение заказа, тогда как в новом варианте количество щелчков мыши сократилось, ибо от пользователя не требовалось делать еще одного лишнего щелчка. С введением нового варианта интерфейса у пользователей появилась возможность сформировать привычку выбирать из списка отделов обычное место доставки, но оставалась вероятность того, что пользователь в редких случаях мог забыть отметить при необходимости и другие отделы, в которые товары должны быть доставлены. Тем не менее, даже с наличием этой потенциальной ошибки новый вариант интерфейса был существенно улучшен по сравнению с предыдущим.

3.4. Видимость и состоятельность

На пустом диске ты можешь искать бесконечно долго.

Источник неизвестен

Независимо от того, чем является продукт – переносным двухполосным радиоприемником или рабочим столом экрана компьютера, не всегда ясно, какие именно функции сейчас доступны, что они выполняют и как получить к ним доступ. Пользователь должен применять собственные органы чувств для того, чтобы определять, какие функции сейчас доступны и каким образом их можно использовать.

Элемент интерфейса можно считать **видимым**, если он либо в данный момент доступен для органов восприятия человека (обычно это глаза, но здесь также рассматриваются и другие сенсорные модальности), либо он был настолько недавно воспринят, что еще не успел выйти из кратковременной памяти пользователя. Если элемент интерфейса не видим, то мы говорим, что он **невидимый**. Для нормальной работы интерфейса «должны быть видимы только необходимые вещи – те, что идентифицируют части работающих систем, и те, что отображают способ, которым пользователь может взаимодействовать с устройством. Видимость отображает связь между предпринимаемыми действиями и их реальной отдачей» (Norman, 1988, с. 8).¹ Если интерфейс вынуждает вас запоминать существование того или иного его элемента, это означает, что этот элемент является невидимым. Если вы вынуждены углубляться в недра интерфейса, чтобы случайно или в результате собственной настойчивости натолкнуться на последовательность дей-

ствий, которая активизирует какой-то его элемент, то этот элемент является невидимым. Если вам приходится обращаться за помощью к справочной системе для того, чтобы узнать, как выполнить то или иное действие, это означает, что методы выполнения этого действия невидимы. Многие компьютерные игры являются, по сути дела, недокументированными интерфейсами, в которых способы управления или пути достижения желаемых результатов невидимы. Стоит добавить к этим играм документацию, и они станут неинтересными. Однако *большинство людей не хотят играть в игры, когда им требуется выполнить свою работу, поэтому перед разработчиком интерфейсов стоит задача сделать каждый элемент своего продукта видимым.*

Если в интерфейсе соблюдается принцип видимости, то каждая функция и способ ее использования для большинства людей из той культуры, на которую этот интерфейс ориентирован, становятся очевидными только лишь по одному виду. Элемент управления, который имеет такое свойство, стали называть **состоятельным** (affordance) (Norman, 1988, с. 123). «Состоятельность является хорошим средством для связи элемента интерфейса с его целью. Ручки используются для настройки; гнезда – для вставки чего-то; мячи – для их бросания или пинания» (Norman, 1988, с. 9). Если вы как разработчик помещаете в какой-то продукт ручку, подобную тем, с помощью которых регулируется громкость, пользователи будут пытаться ее крутить. Если вы поместите что-нибудь, наподобие кнопки, пользователи будут пытаться на нее нажимать. То, какую состоятельность будет иметь элемент интерфейса, зависит от опыта и знаний пользователя, а также от контекста, в котором он этот элемент встречает.

При анализе интерфейсов следует всегда спрашивать себя, каким образом пользователь может узнать, что то или иное действие возможно; также следует всегда стремиться к тому, чтобы каждый видимый элемент был состоятельным. Пиктограммы часто рассматриваются как символические обозначения состоятельности, но, как мы выясним в разделе 6.3, они не всегда выполняют эту функцию.

Видимость означает не просто возможность заметить наличие. Любой объект может быть видимым в том смысле, что он может быть обнаружен в определенном месте, но при этом он может быть очень небольшим по размеру или иметь небольшой контраст с фоном, чтобы быть легко замеченным. Оптимизация качества восприятия интерфейса является важным аспектом с точки зрения эргономики, и основное значение здесь имеют когнитивные свойства интерфейса.

¹ Книгу «*The Psychology of Everyday Things*» (Психология каждодневных вещей) можно назвать классическим трудом, с которым следует ознакомиться каждому.

Система BART и отсутствие состоятельности

Упоминание в этой книге системы BART (Bay Area Rapid Transit)¹ связано с интерфейсом, который используется автоматами по продаже билетов. Эти автоматы являются результатом игнорирования администрацией 25-летнего опыта работы старых роботов по продаже билетов. Пока я стою в очереди к одному из двух еще не сломавшихся автоматов, я замечаю, как подходящие люди приходят в замешательство при виде этого устройства. Хотя у меня довольно большой опыт пользования разными устройствами, но мне самому не удастся справиться с той неестественной последовательностью действий, которая требуется для применения этих билетных автоматов.

Очевидно, что для того чтобы начать взаимодействие с автоматом, сначала требуется опустить в него деньги или старый билет, который является денежным эквивалентом необходимой суммы, — что и пытался в первую очередь сделать почти каждый во время моего наблюдения. Обычно автоматы по продаже работают так: опускаешь деньги, выбираешь, получаешь. Такой подход стал парадигмой.

Автоматы по продаже билетов системы BART, наверное, были изобретены на другой планете. Сначала вам необходимо нажать на одну из двух кнопок: либо BART, либо BART-MUNI. (В инструкции, которую трудно найти где-то поблизости, объясняется весьма тонкое различие между этими двумя кнопками.) Эти кнопки расположены в верхней части передней панели автомата. Место, конечно, вполне разумное для начала взаимодействия, в отличие от самого действия, которое для этого используется. Щели для старого билета (слева), для монет (посередине) и для бумажных купюр (справа) находятся на большом расстоянии друг от друга вместо того, чтобы быть расположенными в одном месте. Жидкокристаллический дисплей находится по центру высоко вверх, поэтому люди невысокого роста не могут легко прочитать возникающие сообщения. В любом случае он находится настолько далеко от основной части интерфейса, что большинство людей его просто не замечает. Возможно, это даже и хорошо, потому что время его отклика настолько большое, что люди, которые все-таки смотрят на него, или пытаются повторить операцию, или стоят в недоумении, предполагая, что автомат сломался, что, кстати, часто бывает.

Кроме этого, есть кнопка «поправка» (correction). Ее название наводит на мысль, что использовать ее следует в случае, если вы сделаете какую-то ошибку. Но на самом деле она предназначена совсем не для исправления ошибок, а для того чтобы получить билет на сумму меньшую, чем та, которую вы опустили в автомат. Другими сло-

¹ Система обеспечения автобусного и железнодорожного сообщения в Сан-Франциско. — *Примеч. науч. ред.*

вами, с помощью этой кнопки вы можете сказать машине, что хотите получить сдачу.

Другим признаком того, что автомат разработан плохо, являются яркие крупные цветные числа 1, 2, 3, 4 и большие стрелки, которые, как кажется, должны как-то инструктировать вас. Если бы лицевая панель автомата была спроектирована грамотно, то вам бы они не понадобились. Числа в любом случае нисколько не помогают, поскольку во время опускания в автомат денег (\$5) вам все равно надо помнить, какой шаг вы сейчас выполняете (например, шаг 2). Если же вам нужен билет за \$3, то нужно еще не забыть нажать кнопку «поправка», чтобы получить сдачу в виде четвертаков (8 шт.), и тут вы слышите, что прибывает поезд на 7:06. Так, а на каком, вы сказали, мы сейчас шаге? Если вы встречаете компьютерный интерфейс, в котором есть много ярких красок и мириады объяснительных надписей, можете быть уверены, что он представляет собой сплошную путаницу.

Автоматы BART могли бы работать, например, следующим образом. (Однако имейте в виду, что если бы вы консультировали компанию BART, то потребовалось бы провести пользовательское тестирование (юзер-тестинг) вашего варианта, чтобы убедиться, что люди пользуются им именно так, как вы и предполагали.)

Слева (т. к. английский и испанский языки читаются слева направо) располагается щель для старых билетов, горизонтальная щель для банкнот и небольшая вертикальная щель для монет. Все они сгруппированы в одном месте и имеют понятные надписи. На дисплее, размещенном на уровне глаз человека среднего роста, отображается сообщение: «Опустите в автомат деньги ИЛИ старый билет системы BART ИЛИ и то и другое». После этого на экране сразу же появляется общая сумма в виде следующего сообщения:

Вы опустили \$4.55.

Если вам нужен билет стоимостью больше, чем \$4.55, опустите дополнительную сумму денег или еще один использованный билет.

Если вам нужен билет стоимостью меньше, чем \$4.55, нажмите клавишу:

Получить сдачу с билета

Если вам нужен билет стоимостью \$4.55, нажмите одну из следующих клавиш:

Выдать билет BART

Выдать билет BART-MUNI

Если пользователь нажмет кнопку «Получить сдачу», то ему будет задан вопрос о том, на какую сумму он хочет купить билет, после чего потребуются нажать кнопку ENTER, находящуюся под числовой

клавиатурой. После этого на экране отобразится текущая сумма, которую следует либо увеличить, либо уменьшить, либо подтвердить. После выдачи билета пользователю должна быть возвращена сдача. Если опытный пассажир уже имеет точную сумму денег, то для него количество действий сокращается с шести до следующих двух: опустить деньги и нажать кнопку «Выдать билет».

3.5. Монотонность

Человек – слишком сложный организм. Если уж ему суждено исчезнуть, то вымрет он от желания простоты.

Эзра Паунд

Разработчики интерфейсов часто предлагают пользователям сразу несколько методов достижения того или иного результата. Например, одна и та же команда может выполняться как с помощью меню, так и с помощью сочетания клавиш. В большинстве текстовых процессоров вы можете переместить фрагмент текста либо за три шага (выделение, вырезание, вставка), либо за два (выделение и перетаскивание). При этом сам процесс выделения может выполняться более чем одним способом. Таким образом, для пользователя обычно имеется целый «шведский стол» методов.

Одним из оправданий такого разнообразия методов для выполнения задачи может быть то, что одни пользователи могут предпочесть один метод, а другие пользователи – другой. Например, для новичка научиться пользоваться меню может быть легче, тогда как опытный пользователь, скорее всего, не будет отрывать рук от клавиатуры и отдаст команду с ее помощью (см. раздел 3.6). Другим оправданием может быть то, что один метод (например, выделение, вырезание и вставка) может быть полезным для работы с разнесенными частями документа, а другой метод (выделение и перетаскивание) может быть эффективным, только когда на дисплее видны и исходная и конечная позиции одновременно. Другим обоснованием множества методов может быть то, что каждый из них порожден традицией, и поэтому разработчики считают разумным использовать как можно больше навыков из уже существующих у пользователей.

Этот последний довод, называемый еще *обратной совместимостью* (backward compatibility)¹, является самым слабым и может привести к абсурдным интерфейсам в виде собрания несовместимых методов. Однажды во время одного из длинных авиаперелетов нам пришлось приземлиться из-за плохой погоды. Пока мы ждали ее улучшения, я прошел в кабину пилотов, где смог изучить устройство автопилота. В нем

¹ Я всегда говорю, что если из этого выражения убрать слово «совместимость» (compatibility), его истинное значение сразу становится понятным.

имелось не менее пяти способов для ввода координат и столько же способов для включения большинства его функций. Когда я спросил пилота о причинах такой избыточности, она ответила, что автопилот сделан таким образом, чтобы функционально быть как можно более подобным автопилотам на других самолетах, на которых пилоты могли изучать данный прибор и тем самым избежать необходимости переучивания. Однако возникал вопрос относительно того, насколько такой подход можно считать правильным и насколько точно выполняется дублирование других автопилотов. Как она мне еще объяснила, пилоты должны изучать не только те небольшие, но, тем не менее, раздражающие различия между старой системой автопилота и ее эмуляцией на новой модели, но также и остальные четыре способа использования автопилота. То есть пилот обязан знать *каждый* аспект работы *каждого* элемента оборудования пилотской кабины. Более того, следует отметить, что многие новые функции автопилота имеются только в некоторых эмуляциях, и ранние модели автопилотов не содержат этих функций, т. к. и копируемые автопилоты их не имели.

Тактика гибридизации различных вариантов интерфейса, а вернее, сбрасывания всех методов подряд в одну кучу увеличивает время обучения, создает сложный в использовании автопилот и, как отметила упомянутый пилот, «создает путаницу в кабине и увеличивает вероятность ошибки». Хотя она и не сказала этого, но, вероятно, это также увеличивает стоимость и сложность приборов, руководств по их использованию и стоимость их обслуживания. То же касается любого интерфейса, который представляет собой совокупность разрозненных подходов, накопленных с течением времени, включая также Macintosh и Windows.

Для описания интерфейса, имеющего только один способ выполнения той или иной задачи, я использую термин *монотонный*, который все же не совсем удачен (см. приложение В, а также Alzofon и Raskin, 1985, с. 95). **Монотонность** – это вторая сторона отсутствия модальности. В интерфейсе, который не имеет режимов, данный жест пользователя может иметь один и только один результат: жест g всегда приводит к действию a . Тем не менее, вполне возможно, что другой жест h тоже может приводить к результату в виде действия a . Однако монотонный интерфейс допускает только один способ выполнения того или иного действия, т. е. в этом случае действие a может вызываться только лишь жестом g и более никаким другим. Интерфейс, который является полностью немодальным и монотонным, допускает только одно соответствие между причиной (командами) и следствием (действиями). Чем более монотонным является интерфейс с точки зрения пространства данной задачи, тем легче для пользователя сформировать автоматичность ее выполнения, которая, в конце концов, создается тем, что снимается необходимость выбирать способ достижения результата.

Другой причиной отсутствия монотонности в интерфейсе может быть нерешительность разработчиков. Я встречал случаи, когда проблема наличия каких-то двух и более вариантов, ни один из которых не имеет преимуществ по сравнению с другими, предусмотрительно «решалась» с помощью внедрения сразу *всех* этих вариантов. В пользу такого подхода обычно приводится довод о том, что пользователь в этом случае получает выбор, как будто бы пользователь является экспертом по интерфейсам и может сам выбрать самый эффективный вариант.

Общераспространенным мифом (более подробно о нем см. в разделе 3.6) является то, что интерфейсы для начинающих и опытных пользователей системы должны быть совершенно различными. Иногда разработчики говорят о том, что «необходимо найти компромисс между легкостью изучения интерфейса и общей скоростью работы системы» (Card, Moran, и Newell, 1983, с. 419). Это действительно может касаться каких-то особых моделей интерфейсов, но это необязательно имеет отношение ко всем интерфейсам вообще и в частности к тем видам интерфейса, которые обсуждаются в этой книге. *Современные графические пользовательские интерфейсы, имеющие рабочий стол, являются сочетанием, по крайней мере, двух разных интерфейсов: с одной стороны, сравнительно видимой и быстроизучаемой системы на основе меню, а с другой – неполного собрания трудноизучаемых и труднозапоминаемых комбинаций клавиш. Сочетание двух ошибок не может дать правильного ответа.*

Когда требуется сделать выбор между несколькими методами, ваш locus внимания временно смещается с текущей задачи на принятие решения о выборе. Это является главным обоснованием монотонности системы. Если условия для принятия решения остаются достаточно простыми и ясными, то в каждом случае вы можете поступать неким привычным способом, тем самым делая ситуацию «монотонной». Таким образом, перед разработчиками интерфейсов стоит задача по поиску монотонного решения для того, чтобы обеспечить такие преимущества, как легкость изучения, простоту внедрения, минимум документации и небольшой размер расходов на обслуживание. Эти преимущества могут быть достигнуты либо без затрат для разработчика, либо за счет относительно небольшой единовременной затраты на тщательную разработку и тестирование интерфейса. Монотонность означает не то, что с каким-то содержанием нельзя работать разными способами, а то, что для вызова одной и той же команды не должно использоваться множество жестов.

Монотонность проявляется спонтанно. Многие пользователи сами делают интерфейс монотонным через предпочтение какого-то одного метода и игнорирование его альтернатив независимо от ситуации. Опытные пользователи, которые гордятся знанием каждого закоулка какой-то системы, часто называют таких пользователей «чайниками». Тем не менее, такие «чайники» могут использовать интерфейсы более эффективно, чем «профессионалы». С точки зрения разработчика та-

кие пользователи попусту теряют преимущества других способов, но с точки зрения пользователя оказывается, что это именно разработчики впустую тратят свои ресурсы.

На мой взгляд, интерфейс, который не имеет режимов и является – насколько это возможно – монотонным, был бы чрезвычайно удобным в использовании при условии, что все другие характеристики имеют, по крайней мере, нормальное качество, принятое для современных интерфейсов. Применяя такой интерфейс, пользователь смог бы сформировать необычайно высокий уровень доверия к собственным привычкам. Только в результате этих двух свойств мог бы возникнуть эффект исчезновения интерфейса из сознания пользователя, предоставляя ему возможность направить все свое внимание на текущую задачу. Чтобы изучить все психологические свойства системы, являющейся монотонной и полностью (или почти полностью) лишенной режимов, требуется провести еще множество экспериментальных исследований.

Если я прав, то использование продукта, интерфейс которого основан на немодальности и монотонности, могло бы быстро вызывать привыкание (близкое к зависимости) у пользователей, приводя к тому, что все они могут приобрести преданность этому продукту и предпочитать его всем другим. Для пользователей такого продукта оказалось бы психологически трудным перейти к применению конкурирующего продукта. В отличие от продажи наркотиков, продажа интерфейсов, вызывающих привыкание у их пользователей, не является противозаконной, а сам интерфейс приносит пользователям облегчение. С другой стороны, это все же имеет сходство с продажей наркотиков в том, что может приносить такие же сверхвысокие доходы.

3.6. Миф о дихотомии «новичок–эксперт»

В первую очередь мы люди, а потом уже либо эксперты, либо новички.

Клиффорд Насс, программа радио CBC «Quirks and Quarks» (Выходки и кварки) от 23 января 1994 г.

Мнение психолога Клиффорда Насса аналогично мнению, которое имеет автор этой книги: разработка интерфейса должна начинаться с учета слабых и сильных сторон человека. Каждый элемент интерфейса должен соответствовать как нашим когнитивным способностям, так и требованиям задачи, стоящей перед пользователем, хотя этим требованиям к разработке не исчерпываются. Мнение Насса также отражает распространенный взгляд, суть которого заключается в том, что всех пользователей можно разделить на две группы: экспертов и новичков (и, вероятно, еще тех, кто временно находится на переходной стадии от одной группы к другой). Эта дихотомия неверна. Как пользователь сложной системы вы не являетесь ни новичком, ни экспертом, и вас нельзя отнести к какой-то точке на одномерном диапазо-

не между этими двумя полюсами. Вы можете знать или не знать каждый элемент интерфейса или каждый набор связанных элементов, которые работают одинаковым образом. Вы можете знать, как использовать многие команды и опции какого-то программного пакета, – вы даже можете работать с этим пакетом профессионально, и поэтому другие люди могут обращаться к вам за советом по его использованию. Однако вы можете не знать, как пользоваться некоторыми другими командами, или вообще не знать о существовании этих команд или даже целых категорий команд в данном пакете. Например, пользователю программы для обработки фотоизображений, который применяет ее только для создания онлайн-овых изображений, может никогда не понадобиться возможность выполнения цветоделения, которая используется, главным образом, для коммерческой печати (или он может даже и не знать об этой возможности).

Разработчики интерфейсов делали разные попытки учесть допущение о том, что пользователей можно разделить на новичков и экспертов. Но поскольку это допущение неверно, все эти попытки, естественно, провалились. Хорошим примером могут послужить адаптивные системы, способные автоматически переключаться с режима для начинающих пользователей в режим для опытных пользователей, когда они определяют, что ваше умение владеть системой достигло необходимого уровня. Если во время использования этой системы в режиме для начинающих пользователей она внезапно переключается в «экспертный режим», для пользователя происходит неожиданное изменение рабочей среды, по крайней мере, ее части. Не лучшим вариантом является и то, если система будет переключаться постепенно, часть за частью. В этом случае она будет проявлять себя неустойчиво и беспорядочно, поскольку привычки и навыки, которые вы успели вчера сформировать как новичок, становятся бесполезными, если сегодня данный элемент переключился в экспертный режим.

Однажды я встретил работающую в сети программу, в которой была предусмотрена возможность перехода на экспертный режим, если вам удалось хотя бы раз успешно ею воспользоваться. Если вы не работали с этой программой более шести месяцев, то автоматически возвращались обратно к статусу новичка. Конечно, любая такая схема может не соответствовать реальной способности пользователя учиться и запоминать. Если программа, которая продвинула вас на более высокий уровень, после слишком короткого периода времени переключилась обратно в режим для новичков, необходимость опять пользоваться методами, предусмотренными для начинающих, вызовет у вас раздражение. Если же программа не переключится в режим для начинающих вовремя, то перед вами могут оказаться элементы, использование которых вами уже прочно забыто. При современной технологии система не может точно определить, когда вы забыли, как пользоваться данной возможностью, поэтому она не может точно переключиться обратно в режим для новичков. Если же программа будет постоянно время

от времени оценивать ваш уровень владения системой, это будет незойливым для пользователя.

Большинство попыток сделать интерфейс адаптивным оказываются неблагоприятными. Всякий раз, когда система автоматически как-то изменяется, даже если эти изменения настолько же незначительны, как и, скажем, измененный порядок элементов в меню, ваши ожидания от работы системы не оправдываются, и накопленные привычки и навыки теряют смысл. (Кстати говоря, в своей новой операционной системе Windows 2000 компания Microsoft представила адаптивные меню.¹) С другой стороны, не существует теории о том, что один и тот же неизменный интерфейс не может служить хорошо в течение всего времени использования, начиная от начального уровня и заканчивая профессиональным. Лучшим вариантом было не менять ничего в течение всего времени использования продукта, и, наверное, не следует проводить специальных исследований для того, чтобы убедиться, что для выполнения какой-то задачи лучше изучать только один интерфейс, а не несколько.

Очень легко попасть в ловушку идеи разработки разных интерфейсов для разных классов пользователей, потому что она чревата слишком большими обобщениями, которые могут привести к ложному упрощению процесса разработки. Немногие из этих обобщений окажутся верными для каждого пользователя в любом (значительно большем) классе пользователей, который можно выделить. Средством против такого заблуждения может быть взгляд на интерфейс не с точки зрения класса пользователей, а с точки зрения индивида. Каждый человек, который использует какое-то программное обеспечение достаточно долгое время, проходит через сравнительно короткий этап изучения каждой команды и каждого элемента интерфейса, а также через намного более длительный период обычного (и, надеюсь, автоматичного) его использования. Нам действительно необходимо разрабатывать такие системы, которые легко изучить и понять, но еще более важным является то, чтобы эти системы могли эффективно использоваться в течение длительного времени. Исключение могут составить только программы, которые предполагается использовать недолго, поэтому каждый ее пользователь будет начинающим, и вопрос приобретения навыков и привычек здесь не имеет значения. В качестве примера такого интерфейса можно привести выставочный киоск, оснащенный компьютером.

Фаза изучения работы элемента интерфейса требует сознательного внимания. Поэтому простота и ясность функции, а также ее види-

¹ Когда я писал этот раздел, операционная система Windows 2000 была новым продуктом, и поэтому я смог поговорить только с несколькими ее пользователями. Типичным отзывом было высказывание: «Сначала адаптивные меню казались неплохой идеей, но как только я столкнулся с тем, что какое-то меню изменилось, это меня выбило из колеи. Больше мне эта идея не нравится.»

мость имеют особую важность. Фаза профессионального владения интерфейсом главным образом характеризуется бессознательным использованием его возможностей. Такое использование подкрепляется следующими качествами интерфейса: соответствие задаче, немодальность и монотонность. Эти требования ни в коей мере не противоречат друг другу. Таким образом, *хорошо разработанный, человекоориентированный интерфейс совсем не требует разделения на подсистемы для новичков и экспертов.*

Конечно, это не значит, что интерфейс не может быть разделен на такие части. Однако если вы разрабатываете интерфейс и у вас возник соблазн создать комбинации клавиш специально для опытных пользователей, подумайте, не следует ли вместо этого переработать существующий метод так, чтобы его механизм мог подойти под требования всех пользователей.

4

Квантификация

Гармония мира проявляется Формой и Числом, и потому и сердце, и душа, и вся поэзия Натуральной Философии воплощаются в понятиях математической красоты.

Д'Арки Уентуорф Томпсон «О росте и форме» (1917)

Существует множество методов количественного анализа элементов интерфейса. Однако ясное руководство о том, как использовать эти методы, можно встретить редко. В этой главе будет дано простое описание и подробные примеры модели GOMS, разработанной Кардом, Мораном и Ньюэллом, а также критерий эффективности Раскина, закон Хика и закон Фитса.

4.1. Количественный анализ интерфейса

Он все тыкал и тыкал пальцами в компьютер, а Мелроуз просто изумлялся тому, что машина, которая предназначена для избавления человека от всей мелкой, рутинной работы, выполняла такую простую задачу настолько долго, что Буб, наверное, уже десять раз успел бы все сделать руками.

*Марта Граймс «The Stargazer»
(детективный роман)*

Многие количественные и эвристические методы используются для анализа и изучения интерфейсов. Эти методы составляют значительную часть содержания большинства книг, посвященных этой теме, включая и те, которые указаны в библиографическом списке за такими авторами, как Шнейдерман (Shneiderman), Норман (Norman) и Мэйхью (Mayhew). Например, с помощью пассивного наблюдения за тестированием нового интерфейса с участием нескольких доброволь-

цев опытный разработчик интерфейсов может узнать столько же ценной информации, сколько можно получить с помощью любого метода количественного анализа. Здесь я хочу сосредоточиться на количественных методах не для того, чтобы принизить значение качественных методов, но скорее для того, чтобы найти между ними баланс и показать ценность численных и эмпирических методов, которые не являются широко известными. Количественные методы часто могут свести спорные вопросы к простым вычислениям. Еще одним, более важным преимуществом этих методов является то, что они *помогают нам понять важнейшие аспекты взаимодействия человека с машиной*.

Одним из лучших подходов к количественному анализу моделей интерфейсов является классическая модель GOMS – «правила для целей, объектов, методов и выделения» (the model of goals, objects, methods, and selection rules), которая впервые привлекла к себе внимание в 80-х годах (Card, Moran and Newell, 1983). Моделирование GOMS позволяет предсказать, сколько времени потребуется опытному пользователю на выполнение конкретной операции при использовании данной модели интерфейса. После обсуждения модели GOMS мы рассмотрим количественные методы оценки производительности интерфейсов, скорости движения курсора и времени, необходимого для принятия решения.

4.2. Модель скорости печати GOMS

Цель точной науки заключается в том, чтобы свести проблемы природы к установлению количеств посредством операций с числами.

Джеймс Клерк Максвелл
«К вопросу о фарадеевых силовых линиях» (1856)

Здесь я хочу обсудить только один простейший, но довольно ценный аспект метода GOMS – модель, основанная на оценке скорости печати. Разработчики, которые знакомы с методом GOMS, редко проводят детальный и формальный анализ модели интерфейса. Отчасти это происходит из-за того, что основы GOMS и других количественных методов известны им настолько, что они изначально руководствуются этими методами в процессе разработки. К формальному анализу, конечно, прибегают в случаях, когда необходимо выбрать один из двух вариантов разработки, когда даже небольшие различия в скорости могут давать большой экономический и психологический эффект. Иногда разработчики пользуются поражающими своей точностью расширенными моделями GOMS, как, например, анализ с использованием метода критического пути GOMS (critical-path method GOMS, CPM-GOMS) или версия, называемая естественным языком GOMS (natural GOMS language, NGOMSL), в которой учитывается поведение неопытного пользователя, например время, необходимое ему для обучения. С помощью этих методов можно, например, предсказать, сколько времени понадобится пользователю для выполнения некоторого набора дейст-

вий при использовании данного интерфейса с абсолютной погрешностью менее 5%. В расширенных моделях почти все оценки не выходят за пределы стандартного отклонения, принятого для измеренных значений времени (Gray, John и Atwood, 1993, с. 278). Для вопросов, которые вызывают жаркие споры и по поводу которых авторитетные разработчики зачастую высказывают совершенно разные мнения, полезно вооружиться количественными методами, имеющими теоретическое обоснование и получившими экспериментальную апробацию. Более полный обзор и библиографию, посвященные различным моделям GOMS, можно найти у Джона (John, 1995); там же можно найти и модель CPM-GOMS, разработанную самим Джоном.

4.2.1. Временные интервалы в интерфейсе

Точность цифр есть истинная душа науки.

*Д'Арки Уентуорф Томпсон
«О росте и форме» (1917)*

Разработчики модели GOMS во время ее создания заметили, что время, требующееся для выполнения какой-то задачи системой «пользователь–компьютер», является суммой всех временных интервалов, которые потребовались системе на выполнение последовательности элементарных жестов, составляющих данную задачу. Хотя для разных пользователей время выполнения того или иного жеста может сильно отличаться, исследователи обнаружили, что для большей части *сравнительного* анализа задач, включающих использование клавиатуры и графического устройства ввода, вместо проведения измерений для каждого отдельного пользователя можно применить набор стандартных интервалов. С помощью тщательных лабораторных исследований был получен набор временных интервалов, требуемых для выполнения различных жестов. Ниже приводится оригинальная нomenclатура, в которой каждый интервал обозначен одной буквой (Card, Moran и Newell, 1983).

- | | |
|--------------|--|
| $K = 0.2$ с | Нажатие клавиши. Время, необходимое для того, чтобы нажать клавишу. |
| $P = 1.1$ с | Указание. Время, необходимое пользователю для того, чтобы указать на какую-то позицию на экране монитора. |
| $H = 0.4$ с | Перемещение. Время, необходимое пользователю для того, чтобы переместить руку с клавиатуры на ГУВ или с ГУВ на клавиатуру. |
| $M = 1.35$ с | Ментальная подготовка. Время, необходимое пользователю для того, чтобы умственно подготовиться к следующему шагу. |
| R | Ответ. Время, в течение которого пользователь должен ожидать ответ компьютера. |

На практике указанные значения могут варьироваться в широких пределах. Для опытного пользователя, способного печатать со скоростью 135 слов/мин., значение K может составлять 0.08 с, для обычного пользователя, имеющего скорость 55 слов/мин., – 0.2 с, для среднего неопытного пользователя, имеющего скорость 40 слов/мин., – 0.28 с, а для начинающего – 1.2 с. Нельзя сказать, что скорость набора не зависит от того, что именно набирается. Для того чтобы набрать одну букву из группы случайно взятых букв, большинству людей требуется около 0.5 с. Если же это какой-то запутанный код (например, адрес электронной почты), то у большинства людей скорость набора составит около 0.75 символов в секунду. Значение K включает в себя и то время, которое необходимо пользователю для исправления сразу замеченных ошибок. Клавиша <Shift> считается за отдельное нажатие.

Широкая изменяемость каждой из представленных мер объясняет, почему эта упрощенная модель не может использоваться для получения абсолютных временных значений с какой-либо степенью точности. Тем не менее, с помощью типичных значений мы можем сделать правильную *сравнительную оценку* между какими-то двумя интерфейсами по уровню эффективности их использования. Если оцениваются сложные интерфейсы, включающие пересекающиеся временные зависимости, или если должны быть с точностью достигнуты определенные временные интервалы, то следует применять более сложные модели (например, CPM-GOMS), которые не рассматриваются в этой книге.

Двойная «дискликсия»¹

Интерфейсная техника, называемая «двойным кликом», т. е. двойное нажатие кнопки ГУВ за короткий временной промежуток и без какого-либо значительного перемещения курсора между двумя нажатиями, имеет некоторые недостатки. Вы не можете точно сказать, какие объекты на экране ответят на двойной клик, а какие нет. Кроме того, не всегда ясно, какой именно может быть ответ. Отображаемые на экране монитора элементы не имеют каких-либо признаков, означающих, что двойной клик может вызвать какой-то результат, – т. е. эта функциональность является невидимой. То, каким образом двойной клик используется во многих современных интерфейсах, вынуждает пользователей запоминать не только то, по *каким именно* элементам можно дважды щелкнуть мышью, но и то, какой результат возникает в ответ на это действие по отношению к разным классам элементов интерфейса.

Первые две проблемы можно, по крайней мере, частично решить с помощью использования новых условностей. Что касается двойно-

¹ Дискликсия (dysclicksia) – от англ. слова click (щелкнуть мышью) и приставки dis- (нарушение функции). – *Примеч. пер.*

го клика, то он сам по себе представляет проблему. Двойной клик требует использования кнопки мыши дважды в одном и том же месте или при небольшом перемещении и, в большинстве случаев, в течение небольшого промежутка времени, приблизительно за 500 мс. Если пользователь щелкает слишком медленно, машина воспринимает это как два отдельных клика, а не как один двойной клик. Если пользователь слишком сильно двигает мышью в промежутке между двумя кликами, может произойти та же самая ошибка. Если пользователь слишком быстро нажимает кнопку ГУВ, например, как при работе с некоторыми текстовыми процессорами при попытке выделить буквы внутри слова, система может воспринять два нажатия как двойной клик и в результате будет выделено все слово.

Другая проблема возникает, когда пользователь пытается выделить графический элемент, который можно переместить с помощью ГУВ. Поскольку при быстром нажатии на кнопку ГУВ это устройство очень легко сдвинуть с места, графические программы могут воспринять это действие не как двойной клик, а как попытку переноса объекта в другое место (drag-and-drop). Аналогичным образом, чтобы изменить текст в окне, пользователь может посчитать необходимым изменить местоположение случайно сдвинутого окна и внести в текст поправки, которые изначально предполагались.

Некоторые из нас не страдают от дискликсии. Эти счастливые люди никогда не промахиваются мышью. Они могут вполне беззаботно и с особым своеобразием делать и один, и два клика по мышью и никоим образом не страдать от каких-то побочных эффектов. Они всегда помнят, что может система ответить на двойной щелчок, а что не может. Такие люди способны попасть из револьвера калибра 0,357 в летящую птицу, двигаясь при этом на автомобиле по извилистой горной дороге. Однако нельзя рассчитывать на то, что все пользователи настолько удачливы. Поэтому интерфейс необходимо разрабатывать с учетом пользователей, страдающих от дискликсии, а также с учетом проблем, присущих использованию двойного клика.¹

Длительность ответа, поступающего от компьютера, R , может оказывать неожиданный эффект на действия пользователя. Если при использовании какого-то управляющего элемента на экране монитора в течение приблизительно 250 мс ничего не возникает, пользователь, скорее всего, может почувствовать беспокойство, решит сделать еще одну попытку или подумает, что система неисправна.

Нельзя сделать такой продукт, который мог бы завершать все операции за время, не превышающее времени реакции пользователя, но

¹ Термин *дискликсия* (dysclicksia), означающий заболевание, единственным лекарством от которого является хорошо разработанный интерфейс, был предложен Памом Мартином (Pam Martin) (переписка, 1997).

можно сделать такие интерфейсы, в которых в течение этого времени всегда бы выдавалось сообщение о том, что информация от пользователя принята и правильно распознана. В противном случае действия пользователя во время задержки – чаще всего просто молотья по клавиатуре с целью получить хоть какой-то ответ – могут привести к нежелательным реакциям со стороны системы, приводя тем самым к еще большей задержке или даже повреждению содержания. Например, при попытке скачать какой-нибудь файл с America Online с помощью броузера (например, Netscape) очень часто возникает большая задержка. При этом не появляется никакого признака, что действительно что-то происходит. Возникает только небольшое статическое сообщение о том, что компьютер ждет ответа, но оно находится далеко за пределами локуса внимания пользователя. Через несколько секунд неопытный пользователь начинает кликать по кнопкам на экране, что приводит к остановке загрузки файла – опять же без всякого сообщения об этом.

Если задержки неизбежны, важно, чтобы в интерфейсе была предусмотрена сообщающая о них обратная связь, – например, можно использовать индикатор хода выполнения задачи (status bar) (рис. 4.1), отражающий время, оставшееся до конца операции. Если неизвестно, сколько именно времени займет операция, так и скажите об этом пользователю! Нельзя лгать пользователю или вводить его в заблуждение.

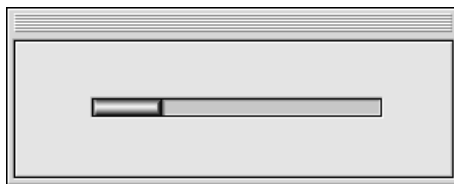


Рис. 4.1. Индикатор хода выполнения задачи. Важно, чтобы отображение времени было линейным. Текстовое сообщение об оставшемся времени, если оно точное, также можно считать человекоориентированным решением в тех случаях, когда задержки неизбежны

4.2.2. Расчеты по модели GOMS

Вычисления времени, необходимого на выполнение того или иного действия (например, «переместить руку с графического устройства ввода на клавиатуру и набрать букву»), с помощью модели GOMS начинаются с перечисления операций из списка жестов модели GOMS (см. раздел 4.2.1.), которые составляют это действие (в приведенном примере это H и K). Перечисление движений (K , P и H) – это довольно простая часть модели GOMS. Более сложным, например, в модели скорости печати GOMS, является определение точек, в которых пользователь остановится, чтобы выполнить бессознательную ментальную операцию, – интервалы ментальной подготовки, которые обозначаются символом M . Основные правила (по Card, Moran и Newell, 1983,

с. 265), позволяющие определить, в какие моменты будут проходить ментальные операции, представлены в табл. 4.1. В разделе 4.2.3 мы рассмотрим, как эти правила применяются на практике.

Таблица 4.1. Расстановка ментальных операций

Правило 0 Начальная расстановка операторов <i>M</i>	Операторы <i>M</i> следует устанавливать перед всеми операторами <i>K</i> (нажатие клавиши), а также перед всеми операторами <i>P</i> (указание с помощью ГУВ), предназначенными для выбора команд; но перед операторами <i>P</i> , предназначенными для указания на аргументы этих команд, ставить оператор <i>M</i> не следует.
Правило 1 Удаление ожидаемых операторов <i>M</i>	Если оператор, следующий за оператором <i>M</i> , является полностью ожидаемым с точки зрения оператора, предшествующего <i>M</i> , то этот оператор <i>M</i> может быть удален. Например, если вы перемещаете ГУВ с намерением нажать его кнопку по достижении цели движения, то в соответствии с этим правилом следует удалить оператор <i>M</i> , устанавливаемый по правилу 0. В этом случае последовательность <i>P M K</i> превращается в <i>P K</i> .
Правило 2 Удаление операторов <i>M</i> внутри когнитивных единиц	Если строка вида <i>M K M K M K...</i> принадлежит когнитивной единице, то следует удалить все операторы <i>M</i> , кроме первого. Когнитивной единицей является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент. Например <i>У, перемещать, Елена Троянская</i> или <i>4564.23</i> являются примерами когнитивных единиц.
Правило 3 Удаление операторов <i>M</i> перед последовательными разделителями	Если оператор <i>K</i> означает лишний разделитель, стоящий в конце когнитивной единицы (например, разделитель команды, следующий сразу за разделителем аргумента этой команды), то следует удалить оператор <i>M</i> , стоящий перед ним.
Правило 4 Удаление операторов <i>M</i> , которые являются прерывателями команд	Если оператор <i>K</i> является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор <i>M</i> , стоящий перед ним. (Добавление разделителя станет привычным действием, и поэтому разделитель станет частью строки и не будет требовать специального оператора <i>M</i> .) Но если оператор <i>K</i> является разделителем для строки аргументов или любой другой изменяемой строки, то оператор <i>M</i> следует сохранить перед ним.
Правило 5 Удаление перекрывающих операторов <i>M</i>	Любую часть оператора <i>M</i> , которая перекрывает оператор <i>R</i> , означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует.

Кроме того, отметим, что в этих правилах под **строкой** будет пониматься некоторая последовательность символов. **Разделителем** будет считаться символ, которым обозначено начало или конец значимого фрагмента текста, такого как, например, слово естественного языка или телефонный номер. Например, пробелы являются разделителями для большинства слов. Точка является наиболее распространенным разделителем, который используется в конце предложений. Скобки используются для ограничения пояснений и замечаний и т. д. Операторами являются *K*, *P* и *H*. Если для выполнения команды требуется дополнительная информация (как, например, в случае когда для установки будильника пользователю требуется указать время его включения), эта информация называется **аргументом** данной команды.

4.2.3. Примеры расчетов по модели GOMS

Разработка интерфейса обычно начинается с определения задачи или набора задач, для которых продукт предназначен. Суть задачи, а также средства, имеющиеся для реализации ее решения, часто формулируют в виде требования или спецификации. В нижеприведенном примере в качестве пользователя выступает лаборант Хол.

Требования

Хол работает на компьютере – печатает отчеты. Иногда его отвлекают экспериментаторы, находящиеся в этой же комнате, чтобы попросить перевести температурные показания из шкалы Фаренгейта в шкалу Цельсия или наоборот. Например, Холу могут сказать: «Переведи, пожалуйста, 302.25 градуса по шкале Фаренгейта в градусы по шкале Цельсия». Значение температуры Хол может ввести только с помощью клавиатуры или ГУВ. Голосовые или другие средства ввода отсутствуют. Просьбы о переводе из одной шкалы в другую поступают приблизительно с равной вероятностью. Приблизительно 25% значений – отрицательные. 10% значений являются целочисленными (например, 37°). Результат перевода из одной шкалы в другую должен отражаться на экране монитора. Другие средства вывода результатов не используются. Хол читает вслух экспериментатору полученное значение. Вводимые и выводимые числовые значения температур могут иметь до десяти цифр с каждой стороны от десятичного разделителя.

При разработке интерфейса для системы, с помощью которой Хол сможет выполнять такие просьбы, следует минимизировать время, необходимое для перевода из одной шкалы в другую. Скорость и точность операций должны быть максимальными. Рабочая площадь экрана не ограничена. Окно или область экрана, предназначенная для перевода температурных значений, является постоянно активным и готово к вводу данных с помощью клавиатуры или ГУВ. То, каким образом Хол сможет вернуться к выполнению его

основной работы, не учитывается. Задача считается выполненной с получением результата перевода.

Для оценки требуемого Холу времени исходите из среднего временного значения на введение четырех символов, включая десятичную запятую. Также, из соображений простоты, будем считать, что Хол вводит все символы без ошибок, и поэтому средства выявления ошибок и сообщения о них не требуются.

Сейчас я предлагаю читателям прервать чтение и попытаться разработать интерфейс по этим простым условиям. Для записи решения, включая зарисовки изображений, которые будет наблюдать Хол на экране монитора, не потребуется много времени; поэтому постарайтесь не просто подумать об этой задаче, но и записать ее решение. (Возможно, вы захотите продолжить чтение, проигнорировав мою просьбу, однако я все же прошу вас подумать. Чтение последующих разделов будет намного более интересным, если вы сделаете попытку решить предложенную задачу самостоятельно.) После того как вы закончите разработку интерфейса, ознакомьтесь с анализами, проведенными по методу GOMS, которые будут представлены далее. После этого вы сможете проанализировать свой собственный вариант.

4.2.3.1. Интерфейс для Хола: вариант 1. Диалоговое окно

Инструкции в диалоговом окне (рис. 4.2) довольно просты. На их основе можно описать метод действий, который должен использовать Хол в терминах жестов модели GOMS. Запись по модели GOMS будет представлена последовательно по мере того, как будут добавляться новые жесты.

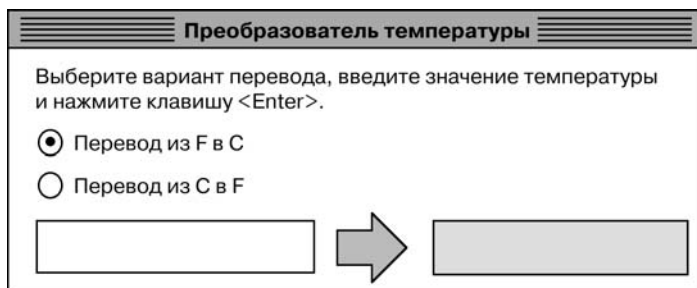


Рис. 4.2. Вариант диалогового окна с использованием группы переключателей

- Перемещение руки к графическому устройству ввода данных:
H
- Перемещение курсора к необходимому переключателю в группе:
H P
- Нажатие на необходимый переключатель:
H P K

В половине случаев в интерфейсе уже будет выбрано требуемое направление перевода, и поэтому Холу не придется кликать на переключатель. Сейчас мы рассматриваем случай, когда переключатель не установлен в требуемое положение.

- Перемещение рук снова к клавиатуре:

$H P K H$

- Ввод четырех символов:

$H P K H K K K K$

- Нажатие клавиши <Enter>:

$H P K H K K K K K$

Нажатие клавиши <Enter> завершает часть анализа, касающуюся метода. В соответствии с правилом 0 мы ставим оператор M перед всеми операторами K и P за исключением операторов P , указывающих на аргументы, которых в нижеследующем примере нет:

$H M P M K H M K M K M K M K M K$

Правило 1 предписывает заменить $P M K$ на $P K$ и удалить все другие операторы M , которые являются ожидаемыми (в указанном примере таких нет). Кроме того, правило 2 предписывает удалять операторы M в середине цепочек. После применения этих двух правил остается следующая запись:

$H M P K H M K K K K M K$

В соответствии с правилом 4 следует оставить оператор M перед конечным K . Правила 3 и 5 в данном примере не применяются.

Следующий шаг – это заменить символы операторов на соответствующие временные интервалы (напомним, что $K = 0.2$; $P = 1.1$; $H = 0.4$; $M = 1.35$).

$$H + M + P + K + H + M + K + K + K + K + M + K = \\ 0.4 + 1.35 + 1.1 + 0.2 + 0.4 + 1.35 + 4 * (0.2) + 1.35 + 0.2 = 7.15 \text{ с}$$

В случае когда переключатель уже установлен в требуемое положение, метод действий становится следующим:

$M K K K K M K$

$$M + K + K + K + K + M + K = 3.7 \text{ с}$$

По условиям задачи оба случая являются равновероятными. Таким образом, среднее время, которое потребуется Холу на использование интерфейса для перевода из одной шкалы в другую, составит $(7.15 + 3.7)/2 \approx 5.4$ с. Но поскольку описанные два метода являются разными, Холу будет трудно использовать их автоматически. Нерешенной проблемой количественных методов анализа остается оценка процента появления ошибок при использовании данной модели интерфейса.

Далее мы рассмотрим графический интерфейс, в котором используется известная всем метафора.

4.2.3.3. Интерфейс для Хола: вариант 2. ГИП (GUI, graphical user interface)

В интерфейсе, показанном на рис. 4.3, используется наглядное отображение термометров. Хол может поднять или опустить указатель на каждом термометре методом перетаскивания с помощью ГУВ. Хол определяет, какой ему необходимо сделать пересчет, перемещая стрелку указателя либо по шкале Цельсия, либо по шкале Фаренгейта. Холу не требуется вводить символы посредством клавиатуры – он просто выбирает значение температуры на одном из термометров. При перемещении указателя на одном термометре указатель на другом перемещается на соответствующее значение. Точность устанавливается с помощью регуляторов масштабирования шкал. Также возможно изменить текущий диапазон значений. Изменение шкалы или диапазона

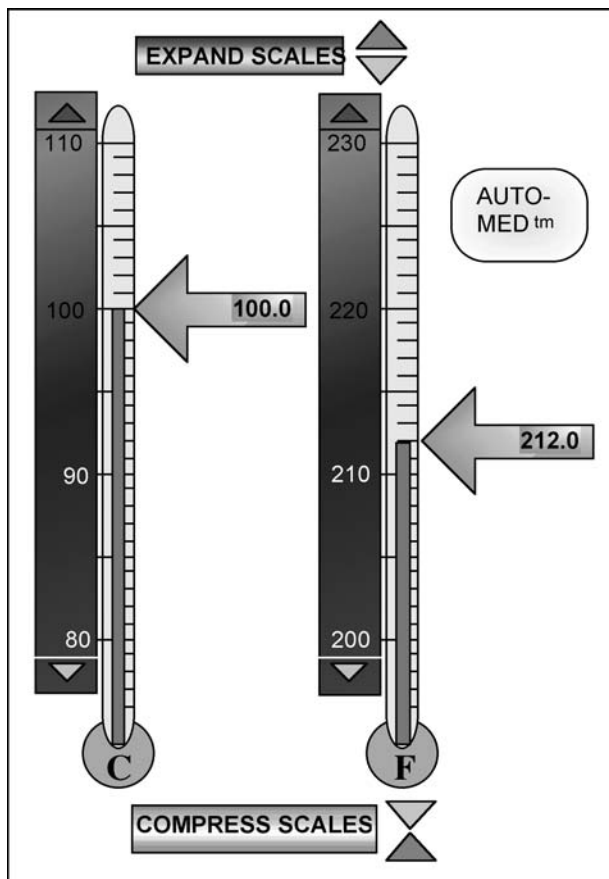


Рис. 4.3. ГИП для задачи Хола

на одном термометре автоматически приводит к соответствующему изменению на другом. Точное числовое значение отображается на перемещаемой стрелке. Температура показывается как в числовом виде, так и с помощью уровня градусника, поэтому Хол может, на свое усмотрение, пользоваться либо графическим вариантом представления данных, либо символьным. Сервис «Автомед» позволяет установить диапазоны термометров с центром в районе 37 градусов шкалы Цельсия и 98.6 градусов Фаренгейта на случай, если кто-то из сотрудников работает со значениями температуры тела человека. Эта опция служит для экономии времени.

С помощью нажатия кнопок «Расширить шкалу» (Expand Scales) и «Сжать шкалу» (Compress Scales) можно уменьшить или увеличить цену деления шкал в 10 раз. Для перехода к значению, которое в данный момент не видно на экране, Хол расширяет шкалу, затем прокручивает до нужного места на шкале, устанавливает стрелку на необходимое температурное значение и потом сжимает шкалу до получения требуемой точности, при необходимости подстраивая стрелку указателя.

Провести анализ этого графического интерфейса с помощью модели скорости печати GOMS довольно сложно, поскольку способ, которым Хол может его использовать, зависит от того, где в данный момент установлена стрелка указателя, какой необходим диапазон температур и какая требуется точность. Рассмотрим сначала простой случай, при котором диапазон температурных шкал и точность перевода уже находятся в желаемом положении. Анализ позволит определить минимальное время, необходимое для использования этого интерфейса.

- Запишем, какие жесты использует Хол, когда перемещает руку к ГУВ, щелкает по кнопке и удерживает ее, указывая на стрелку одного из термометров:

H P K

- Продолжим записывать те жесты, которые использует Хол для перемещения стрелки к необходимому температурному значению и отпускает кнопку ГУВ

H P K P K

- Поставим операторы *M* в соответствии с правилом 0:

H M P M K M K

- Удалим два оператора *M* в соответствии с правилом 1:

H M P K K

Когнитивные единицы, разделители последовательностей и т. д. здесь не используются, поэтому правила 2–5 не применяем. Складывая значения операторов, получаем общее время:

H M P K K

$$0.4 + 1.35 + 1.1 + 0.2 + 0.2 = 3.25 \text{ с}$$

Результат вычисления относится к удачному случаю, когда исходный термометр уже предустановлен на требуемый диапазон и точность. Теперь рассмотрим случай, при котором Хол расширяет шкалу, чтобы увидеть необходимое температурное значение, изменяет диапазон, сжимает шкалу, чтобы получить требуемую точность, и затем перемещает стрелку указателя. Далее я привожу общую запись метода, который использует Хол, без промежуточных шагов. (Я исхожу из того, что Хол является опытным пользователем и не прокручивает шкалу туда и обратно, чтобы найти на ней нужный участок.) Холу приходится несколько раз пользоваться стрелками для прокрутки температурной шкалы. На каждую операцию прокручивания экрана может потребоваться нескольких жестов. Кроме того, требуется время на то, чтобы отобразить изменения на экране, связанные с его прокруткой. Чтобы оценить время прокручивания, я построил такой интерфейс и измерил эти значения. Все они были равны 3 с и более. Обозначая время прокручивания шкал через S , запишем последовательность жестов, которые применяет Хол.

$H P K S K P K S K P K S K P K K$

В соответствии с правилами расставляем операторы M :

$H + 3(M + P + K + S + K) + M + P + K + K$

$0.4 + 3 * (1.35 + 0.2 + 3.0 + 0.2) + 1.35 + 0.4 + 0.2 + 0.2 = 16.8 \text{ с}$

За исключением редких случаев, когда шкалы уже с самого начала установлены правильно, идеальному пользователю понадобится более 16 с на то, чтобы выполнить перевод из одной шкалы в другую, тогда как реальный, т. е. не идеальный пользователь, может сбивать шкалы и стрелки указателей, и поэтому ему понадобится даже больше времени.

4.3. Измерение эффективности интерфейса

Каждый инструмент несет с собой тот дух, в котором он был создан.

Вернер Карл Гейзенберг

Мы рассмотрели два интерфейса: в одном из которых требуется около 5 с на выполнение задачи, а в другом – более 15 с. Отсюда ясно, какой из интерфейсов лучше удовлетворяет поставленным условиям. Следующий вопрос – это определить, насколько быстро работает тот интерфейс, который отвечает поставленным требованиям.

Если имеется модель интерфейса, то с помощью GOMS и его расширений можно определить время, необходимое пользователю на выполнение любой, четко сформулированной задачи, для которой данный интерфейс предусмотрен. Однако модели анализа не могут дать ответ на вопрос о том, насколько быстро должен работать интерфейс. Чтобы ответить на него, мы можем воспользоваться мерой, применяемой в теории информации. Далее мы будем рассматривать термин *информа-*

ция в техническом смысле, т. е. как квантификацию некоторого объема данных, передаваемых с помощью средства коммуникации, как, например, при разговоре двух людей по телефону, или если человек подает некоторый сигнал машине, например с помощью нажатия кнопки ГУВ, когда курсор находится в определенной области экрана. Перед тем как углубиться в детали техники измерения того, какой объем информации нужен для выполнения поставленной задачи, обоснуем необходимость такого измерения.

Чтобы сделать правильную оценку времени, необходимого на выполнение задачи с помощью самого быстрого интерфейса, прежде всего следует определить минимальное количество информации, которое пользователь должен ввести, чтобы выполнить задачу. Это минимальное количество не зависит от модели интерфейса. Если методы работы, используемые в предполагаемом интерфейсе, требуют введения такого количества информации, которое превышает минимальное, это означает, что пользователь делает лишнюю работу, и поэтому интерфейс можно усовершенствовать. С другой стороны, если от пользователя требуется ввести именно то количество информации, которое необходимо для выполнения задачи, то для этой задачи интерфейс нельзя сделать более производительным путем изменения количества информации. В этом случае пути улучшения интерфейса (а также много путей для ухудшения) все же остаются, но по крайней мере данная цель повышения производительности будет уже достигнута.

Информационно-теоретическая производительность определяется так же, как понятие производительности определяется в термодинамике – отношением мощности на выходе к мощности на входе процесса. Если в течение какого-то периода времени электрогенератор, работающий от двигателя производительностью в 1000 ватт, производит 820 ватт, то он имеет производительность $820/1000=0.82$. Производительность также часто обозначается через проценты. В этом случае производительность электрогенератора будет составлять 82%. Идеальный генератор (который не может существовать с точки зрения второго закона термодинамики) должен иметь производительность 100%.

Информационная производительность интерфейса E определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь. Так же как и в отношении физической производительности, параметр E может изменяться в пределах от 0 до 1. Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет 1. (Это формальное положение вводится для того, чтобы избежать деления на 0, как в случае ответа на выводимое прозрачное сообщение об ошибке (см. раздел 5.5).)

Производительность E может равняться и 0 в случаях, когда пользователь должен ввести информацию, которая совершенно бесполезна (рис. 4.4). Следует отметить, что в интерфейсах можно встретить не-

мало деталей, которые имеют сомнительную ценность из-за параметра $E = 0$. Примером такого бесполезного элемента может быть диалоговое окно, в котором есть только одна-единственная возможность для действия пользователя, например кнопка OK. (В JavaScript есть даже специальная команда `Alert`, предназначенная только для того, чтобы делать такие ненужные диалоговые окна. Разработчики языка JavaScript были достаточно разумны, чтобы убрать из него команду `goto` и сделать программирование на этом языке структурным, но они упустили из виду аспект интерфейса.)



Рис. 4.4. Диалоговое окно с информационной теоретической эффективностью 0

В параметре E учитывается только информация, необходимая для задачи, и информация, вводимая пользователем. Два или более методов действия могут иметь одинаковую производительность E , но иметь разное время выполнения. Возможно даже, что один метод имеет более высокий показатель E , но действует медленнее, чем другой метод, — например $M K M K$ и $M K K K$. В этом примере при использовании первого метода должно быть введено только два символа. При использовании второго метода требуется ввести три символа, но времени на все действие тратится меньше. Трудно привести другие примеры из обычной жизни, в которых происходит аналогичная перестановка скорости и информационной производительности.¹ Как правило, чем более производительным является интерфейс, тем более продуктивным и более человекоориентированным он является.

Информация измеряется в битах. Один бит, который представляет собой один из двух альтернативных вариантов (таких как 0 или 1, да или нет), является единицей информации.² Например, чтобы выбрать один из каких-либо четырех объектов, потребуется 2 бита информации. Если объекты обозначить как A, B, C и D, первый бит информа-

¹ Можно разработать и более сложные критерии производительности. Например, как в нашем случае, оператор M не используется в вычислениях. Тем не менее, простой критерий, который здесь описан, вполне достаточен для нашего рассмотрения.

² Слово бит (bit) было предложено математиком Джоном У. Тюки (John W. Tukey) как сокращение от слов «двоичная цифра» (Binary digiT) (Shannon и Weaver, 1963, стр. 9).

ции определит выбор между А и В или С и D. Когда первый выбор сделан (например, С и D), второй бит определит выбор между следующими двумя элементами (либо С, либо D). Двух двоичных выборов, или двух битов, достаточно для выбора одного элемента из четырех. Чтобы сделать выбор из группы восьми элементов, потребуется 3 бита. Из шестнадцати элементов – 4 бита, и т. д. В общем случае при количестве n равновероятных вариантов суммарное количество передаваемой информации определяется как степень 2, равная n :

$$\log_2 n$$

Количество информации для каждого варианта определяется как

$$(1/n) \log_2 n \quad (1)$$

Если вероятности для каждой альтернативы не являются равными и i -я альтернатива имеет вероятность $p(i)$, то информация, передаваемая этой альтернативой, определяется как

$$p(i) \log_2 (1/p(i)) \quad (2)$$

Количество информации является суммой (по всем вариантам) выражения (2), которое при равновероятных вариантах сводится к выражению (1). Отсюда следует, что информационное содержание интерфейса, в котором возможно сделать только нажатие единственной клавиши (а ненажатие клавиши не допускается), составляет 0 бит:

$$1 \log_2 (1) = 0 \quad (3)$$

Однако может показаться, что нажатие единственной клавиши способно, например, вызвать подрыв динамита для разрушения здания. Таким образом, передает ли это нажатие какую-нибудь информацию? На самом деле нет, потому что ненажатие кнопки не было предусмотрено как альтернатива – интерфейс допускает «только нажатие единственной клавиши». Если же нажатие клавиши не производится в течение 5-минутного периода, когда подрыв возможен, то здание не будет разрушено, и поэтому нажатие или ненажатие передает до 1 бита информации, так как в этом случае имеется альтернатива из двух вариантов. Из выражения (2) следует, что в вычислениях используется вероятность (p) того, что здание будет разрушено. Таким образом, вероятность того, что оно не будет разрушено, составляет $1 - p$. С помощью выражения (2) мы можем вычислить информационное содержание данного интерфейса:

$$p \log_2 (1/p) + (1 - p) \log_2 (1/(1 - p)) \quad (4)$$

При $p = S$ результат выражения (4) составит:

$$S \times 1 + S \times 1 = S + S = 1$$

Значение выражения (4) будет меньше 1, если $p \neq S$. В частности при $p = 0$ или $p = 1$ оно составит 0, как это видно из выражения (3).

Этот пример показывает важный момент, который заключается в том, что мы можем оценить объем информации, содержащейся в сообщении, только в контексте всего набора возможных сообщений. Чтобы подсчитать количество информации, передаваемой некоторым полученным сообщением, необходимо знать в частности вероятность, с которой это сообщение может быть отправлено. Количество информации в любом сообщении не зависит от других сообщений, которые были в прошлом или могут быть в будущем, не связано со временем или продолжительностью и не зависит от каких-либо иных событий, так же как результат подбрасывания симметричной монеты не зависит от результата предыдущих подбрасываний или от времени дня, когда это подбрасывание производится.

Кроме того, важно учитывать, что:

«нельзя путать понятие информации с понятием смысла ...информация является мерой свободы выбора сообщения... Следует отметить, что при наличии только двух возможных сообщений утверждать, что какое-то сообщение передает какой-то объем [1 бит] информации, неправильно. Понятие информации не применимо к отдельным сообщениям (в отличие от понятия смысла), но применимо к ситуации в целом; при этом единица информации показывает, что в данной ситуации имеется некоторый объем свободы в выборе сообщения, который удобно обозначать как стандартный или единичный объем информации» (Shannon и Weaver, 1963, с. 9).

Однако действия, которые совершает пользователь при выполнении задачи, можно с большей точностью смоделировать в виде процесса Маркова, в котором вероятность последующих действий зависит от уже совершенных пользователем действий. Тем не менее, для данного рассмотрения достаточно использовать упомянутые вероятности отдельных, единичных событий, при этом будем исходить из того, что все сообщения являются независимыми друг от друга и равновероятными.

Также можно вычислить количество информации, которое передается с помощью устройств, отличающихся от клавиатуры. Если экран дисплея разделен на две области – со словом «Да» в одной области и словом «Нет» – в другой, то один клик, совершенный в одной из областей, будет передавать 1 бит информации. Если имеется n равновероятных объектов, то нажатием на один из них сообщается $\log_2 n$ бит информации. Если объекты имеют разные размеры, то количество информации, сообщаемой каждым из них, не изменяется, но увеличивается время перемещения ГУВ к более мелким объектам (далее мы покажем способ вычисления этого времени). Если объекты имеют разные вероятности, формула остается аналогичной той, которая была дана для случая ввода с клавиатуры равновероятных данных. Различие состоит только в том, что для нажатия клавиши может потребоваться 0.2 с, тогда как для нажатия кнопки, изображенной на экране, в среднем может потребоваться около 1.3 с (без учета времени перемещения руки пользователя с клавиатуры на ГУВ).

В случае голосового ввода информации его информационное содержание можно вычислить, если рассматривать речь как последовательность вводимых символов, а не как непрерывный поток определенного диапазона и продолжительности.

Данный подход к теории информации и ее связи с разработкой интерфейсов является упрощенным. Но даже в такой упрощенной форме, которую мы также использовали при рассмотрении модели GOMS, теория информации может дать нам общий критерий оценки качества интерфейса.

4.3.1. Производительность интерфейса для Хола

Аккуратный подсчет есть путь к знаниям всех существующих вещей и тайных секретов.

Папирусы Рхинда, 1650 г. до н. э.

Полезно подробно рассмотреть пример вычисления среднего количества информации, требуемого для некоего интерфейса. Для этого я снова использую пример интерфейса для перевода температур из одной шкалы в другую. В соответствии с условиями требуется, чтобы количество символов, вводимых в температурный преобразователь, равнялось в среднем 4. Кроме того, по условиям задачи десятичная точка используется однократно в 90% вводимых данных, а в 10% – вообще не встречается; знак минус появляется один раз в 25% данных и совсем не встречается в остальных 75% данных. Из соображений простоты, а также потому, что не требуется ответ с точностью до 1%, я буду исходить из того, что все остальные цифры встречаются с одинаковой частотой, и не буду учитывать те 10% данных, которые не содержат десятичной точки.

Требуется определить множество возможных вариантов ввода и их вероятности. Возможны 5 вариантов (*d* означает цифру):

1. *-.dd*
2. *-d.d*
3. *.ddd*
4. *d.dd*
5. *dd.d*

Первые два варианта встречаются в 12.5% случаев, и количество каждого из них составляет 100. Каждый из последних трех вариантов встречается в 25% случаев, и количество каждого из них составляет почти 1000.¹ Вероятность каждого из первых двух вариантов ввода составляет $(0.125/100)=0.00125$. Вероятность любого из последних

¹ Слово «почти» здесь используется потому, что температура 0 градусов не будет вводиться как 0.00 или 00.0.

трех вариантов ввода составляет $(0.75/3000)=0.00025$. Сумма вероятностей, как это и должно быть, составляет 1.

Количество информации (в битах), передаваемое каждым вариантом, определяется выражением (2)¹:

$$p(i) \log_2 (1 / p(i))$$

Значение этого выражения составляет приблизительно 0.012 для отрицательных значений ввода и 0.003 – для положительных. $200 \times 0.0067 + 3000 \times 0.003$ дает в сумме 11.4 бита для каждого варианта ввода.

Важно учесть вероятности вариантов. Если использовать простой подход, в котором все 12 символов (минус, десятичная точка и 10 цифр) принять как равновероятные, то вероятность каждого символа составит $1/12$, а количество информации, содержащейся в 4-значном варианте ввода, составит приблизительно

$$4 \log_2 (12) \approx 14 \text{ бит}$$

В теории информации есть теорема, в которой утверждается, что максимум информации передается при условии, что все символы равновероятны. Поэтому если принять все варианты как равновероятные, то общее значение будет равно количеству информации в каждом отдельном варианте или превышать его. Очевидно, что такое допущение позволяет упростить вычисление информационного содержания. Если же результирующее значение приближенного вычисления меньше количества информации, которое пользователь должен ввести в интерфейс, то проводить еще более точные вычисления уже нет необходимости.

Мы только что выяснили, что каждый раз, когда Холу требуется провести преобразование температурных значений, он должен ввести в среднем около 11 бит информации. Мы можем разделить это количество на то количество информации, которое требуется ввести в интерфейс, что мы сейчас и сделаем. В результате мы получим производительность (эффективность) данного интерфейса.

Другим упрощением, позволяющим провести быстрый анализ интерфейса, является вычисление различных жестов на основе количества информации, передаваемого одним нажатием клавиши или одной операцией ГУВ. При передаче информации нажатием клавиши ее количество зависит от общего количества клавиш и относительной частоты использования каждой из них. Таким образом, нажатия клавиши могут использоваться как приблизительная мера информации. Если на клавиатуре имеется 128 клавиш, и каждая из них используется с одинаковой частотой, то нажатие любой из них будет передавать 7 бит ин-

¹ Для извлечения логарифма по основанию 2 с помощью калькулятора или компьютера, в котором возможно вычислять только натуральные логарифмы, используйте следующее выражение: $\log_2 (x) = \ln (x) / \ln (2)$.

формации. В действительности частота использования клавиш существенно различается. Например, пробел или буква *e* используются чаще, чем *y* или **, поэтому в большинстве приложений на каждое нажатие клавиши приходится в среднем около 5 битов. По условиям нашей задачи среднее число символов вводимых температурных значений не должно превышать 4.

Для данного анализа удобнее использовать более простую меру, чем теоретическая информационная производительность. **Символьная эффективность** часто имеет такую же практическую ценность, что и информационная производительность. Она определяется как минимальное количество символов, необходимое для выполнения задачи, отнесенное к количеству символов, которое в данном интерфейсе требуется ввести пользователем.

Если в нашем интерфейсе потребуется вводить в среднем 4 символа, то символьная эффективность такого интерфейса составит 100%. При добавлении еще одной клавиши, обозначающей шкалу перевода температурного значения, а также еще одной для разделения, средняя длина ввода возрастет до 6 символов, а символьная эффективность снизится до 67%. Если в качестве устройства ввода Хол будет использовать числовую клавиатуру, состоящую из 16 клавиш, то каждой отдельной клавишей будет передаваться 4 бита информации, и поэтому производительность интерфейса возрастет. (Однако по условиям задачи такой возможности не предусмотрено.)

Поскольку любая задача в соответствии с анализом GOMS требует как минимум одного ментального оператора, наиболее производительный интерфейс с использованием клавиатуры для перевода температурных значений из одной шкалы в другую будет теоретически иметь следующее среднее время использования:

$$M + K + K + K + K = 2.15 \text{ с}$$

Таким образом, он будет значительно быстрее, чем любой из двух уже рассмотренных вариантов. Однако введение 4 символов с помощью стандартной клавиатуры дает, по крайней мере, 20 бит информации, тогда как требуется только 10. Следовательно, теоретическая информационная производительность составляет 55%, а значит, существует возможность улучшения. Как мы уже видели, использование стандартной числовой клавиатуры вместо полной клавиатуры снижает объем информации, вводимой на каждые 4 символа, до 16 бит, что повышает производительность до 60%. Желаемая числовая клавиатура, содержащая только цифры, знак минус и десятичную точку, позволит немного повысить производительность – до 70%. Дальнейшее повышение производительности возможно через использование особых кодировок обозначений температуры и изобретение новых устройств ввода, но здесь возникают трудности, связанные с обучением и лишними расходами, поэтому остановимся на варианте с 70% теоретической информационной производительности. Независимо от того, могут ли

теоретические границы быть достигнуты на практике или нет, они дают нам направление в разработке интерфейса.

4.3.2. Другие решения интерфейса для Хола

В разделе 4.3.1 мы приостановили дальнейшие попытки улучшения интерфейса, достигнув 70% теоретической информационной производимости. Данная производительность определена для пока еще неизвестного, теоретического интерфейса, в котором каким-то образом можно получить 100% эффективность использования клавиш. Давайте посмотрим, насколько мы можем приблизиться к этому идеалу с помощью стандартной клавиатуры и ГУВ.

Рассмотрим интерфейс, в котором используется клавиатура со всеми символами. В таком интерфейсе на экране появляется следующее сообщение:

Для перевода температуры из одной шкалы в другую укажите нужную шкалу с помощью символа С (шкала Цельсия) или F (шкала Фаренгейта). Введите числовое значение температуры, затем нажмите клавишу Enter. Результат преобразования будет отображен на экране.

GOMS-анализ показывает, что пользователь должен сделать 6 нажатий клавиш. По правилам расстановки операторов *M* получаем следующую запись:

M K K K K K M K

Среднее время составит 3.9 с.

Мы можем уменьшить это время, если сами символы С и F будем использовать в качестве разделителей. Рассмотрим интерфейс, в котором появляется следующая инструкция:

Для перевода температуры из одной шкалы в другую введите числовое значение температуры и следом поставьте символ С, если оно в шкале Цельсия, или F, если оно в шкале Фаренгейта. Результат преобразования будет отображен на экране.

В данном примере нажимать на клавишу <Enter> не требуется. Некоторые примитивные средства разработки интерфейсов требуют, чтобы пользователь обязательно использовал клавишу <Enter>, и поэтому в них невозможно использовать символы С и F в качестве разделителей. Такие инструменты не подходят для разработки человекоориентированных интерфейсов.

GOMS-анализ показывает, что для интерфейса с символами С и F в качестве разделителей запись будет следующей:

М К К К М К

Среднее время составит 3.7 с. Если бы мы не знали, что теоретически минимальное время составляет 2.15 с, то это решение могло бы показаться удачным. Оно является значительно более эффективным, чем ранее рассмотренные, поэтому мы могли бы на нем остановиться. Однако теоретический минимум подстегивает нас к поиску еще более быстрой интерфейсной модели. Рассмотрим интерфейс, изображенный на рис. 4.5. Такой интерфейс можно назвать *разветвленным*. В нем один ввод дает в результате два вывода.

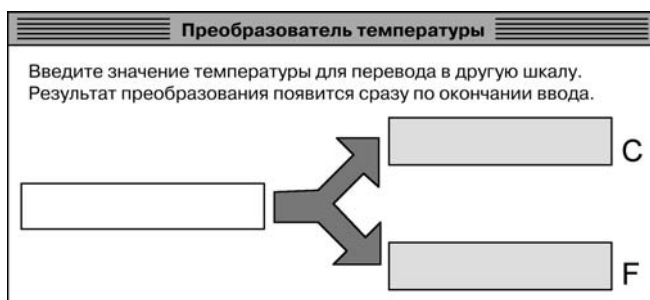


Рис. 4.5. Интерфейс, в котором не используется разделитель. Более эффективным является вариант, в котором выполняется посимвольная обработка вводимых данных и одновременное преобразование в обеих шкалах

В разветвленном интерфейсе нет необходимости в разделителе. Кроме того, пользователю не нужно указывать, какое именно преобразование требуется провести. GOMS-анализ показывает, что для 4 символов, которые в среднем будут вводиться, запись будет следующей:

М К К К К

В разветвленном интерфейсе достигается минимальное время 2.15 с, и его символьная эффективность составляет 100%.

Если, как в нашем примере, в месте вывода происходит изменение результата в тот момент, когда символы вводятся, это колебание цифр не отвлекает пользователя, потому что в локусе его внимания находится именно ввод данных. Непрерывно изменяемые значения на выводе могут быть даже полезными – после нескольких применений пользователь будет замечать эти колебания только краем глаза, что будет служить подсказкой о том, что система отвечает на вводимые данные. Если используется посимвольный режим работы интерфейса, то для большей эффективности такая система должна реагировать довольно быстро. В частности скорость реакции не должна быть меньше скорости ввода. Такая проблема может возникать только при сетевом использовании интерфейса.

Хотя это и не входит в условие задачи, но может возникнуть вопрос о том, как происходит очистка полей в преобразователе для выполнения следующего преобразования. Добавляет ли операция очистки еще одно нажатие клавиши? Необязательно. Например, мы можем разработать интерфейс таким образом, что каждый раз, когда оператор возвращается к своему основному занятию или переходит к другой задаче, значения, указанные в преобразователе, могут автоматически затеняться, а сам температурный преобразователь становится неактивным; причем в это время все значения могут оставаться в своих полях так, что при необходимости их можно было бы опять увидеть, но уже при следующем вводе все они будут стерты.

Разветвленный интерфейс не обязательно является самым лучшим из тех, что уже были рассмотрены – или из тех, что возможны, – только потому, что он имеет оптимальную скорость работы и является весьма эффективным. Кроме скорости есть и другие параметры, которые также являются важными: частота появления ошибок; время, необходимое для изучения интерфейса; возможность длительного запоминания способа использования интерфейса. Особенно следует обратить внимание на частоту появления ошибок в разветвленном интерфейсе, поскольку есть вероятность того, что Хол может прочесть результат не из того поля. Это важно особенно потому, что когда он услышал, например слово *Цельсий*, ему необходимо прочесть значение из поля шкалы Фаренгейта. Тем не менее, разветвленный температурный преобразователь определенно входит в небольшое число тех интерфейсов, которые можно рассматривать как рабочие варианты для программы преобразования температурных значений из одной шкалы в другую. Другие рассмотренные нами примеры, которые могли бы показаться удачными, если бы мы не проводили с ними GOMS-анализ, на самом деле не являются таковыми.

Используемая как в простом анализе временных затрат на нажатия клавиш, так и в полном информационно-теоретическом исследовании, квантификация теоретического интерфейса с минимальным временем использования или с минимальным количеством используемых символов, или с минимальным количеством используемой информации может быть полезна с точки зрения разработки интерфейса. *Без количественных ориентиров мы можем только догадываться о том, насколько хорошо мы разработали интерфейс и есть ли возможность его улучшения.*

4.4. Закон Фитса и закон Хика

Основы знаний надлежит искать в математике.

Роджер Бэкон «Opus Majus», XIII в.

Различные количественные законы, имеющие отношение к разработке интерфейсов, имеют хорошее когнитивное обоснование, и их правильность была неоднократно проверена. Эти законы часто дают нам

дополнительные данные, на основе которых можно принимать те или иные решения, связанные с разработкой интерфейсов. Закон Фитса (Fitts' Law) позволяет определить количественно тот факт, что чем дальше находится объект от текущей позиции курсора или чем меньше размеры этого объекта, тем больше времени потребуется пользователю для перемещения к нему курсора. Закон Хика (Hick's Law) позволяет количественно определить наблюдение, заключающееся в том, что чем больше количество вариантов заданного типа вы предоставляете, тем больше времени требуется на выбор.

4.4.1. Закон Фитса

Представим, что вы перемещаете курсор к кнопке, изображенной на экране. Кнопка является *целью* данного перемещения. Длина прямой линии, соединяющей начальную позицию курсора и ближайшую точку целевого объекта, определяется в законе Фитса как *дистанция*. На основе данных о размерах объекта и дистанции **закон Фитса** позволяет найти среднее время, за которое пользователь сможет переместить курсор к кнопке.

В одномерном случае, при котором размер объекта вдоль линии перемещения курсора обозначается как S , а дистанция от начальной позиции курсора до объекта – как D (рис. 4.6), закон Фитса формулируется следующим образом:

$$\text{Время (мс)} = a + b \log_2 (D / S + 1)$$

(Константы a и b устанавливаются опытным путем по параметрам производительности человека.)¹

Вычисляемое время отсчитывается от момента, когда курсор начинает движение по прямой линии, до момента, когда пользователь щелкает мышью по целевому объекту. Логарифм по основанию 2 является мерой трудности задачи в количестве бит информации, которое требуется для описания (одномерного) пути перемещения курсора.

¹ В математике, которая считается образцом точности и ясности, все еще применяется старый стиль написания формул, в котором неопределенные переменные пишутся в начале, еще до того, как вы можете узнать, что они обозначают. Например, можно встретить выражение наподобие следующего:

$$A = \pi r^2,$$

где r является радиусом окружности, а A – ее площадью.

Такой порядок может приводить к путанице, поскольку читателю придется перечитывать выражение еще раз сначала, особенно если оно довольно длинное и содержит множество других неопределенных переменных. С точки зрения читателя намного удобнее следовать естественному порядку, в котором термины определяются до их использования:

Окружность с радиусом r имеет площадь A , вычисляемую как:

$$A = \pi r^2$$

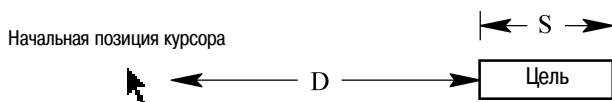


Рис. 4.6. Расстояния, которые используются в законе Фитса для определения времени, необходимого для перемещения курсора к цели

Для вычисления времени можно использовать любые единицы измерения дистанции, т. к. D/S является отношением двух дистанций и поэтому не зависит от единицы измерения. Отсюда следует, что хотя указательное устройство может переместиться на расстояние большее или меньшее, чем то расстояние, на которое переместится на экране курсор, закон все равно работает, при условии, что соотношение между движением ГУВ и курсора является линейным. Закон Фитса может применяться только к тем типам перемещения, которые совершаются при использовании большинства человеко-машинных интерфейсов, т. е. к таким перемещениям, которые невелики относительно размеров человеческого тела и которые являются непрерывными (совершаемыми одним движением). Для приближенных вычислений я использую следующие значения констант в уравнении закона Фитса: $a = 50$, $b = 150$.

Также проводились тестирования с законом Фитса, дополненным некоторыми более сложными параметрами, такими, например, как перемещение курсора между прямыми или искривленными границами (Assot и Zhai, 1997). Для двумерных целей обычно можно получить корректное приближенное значение времени, необходимого для перемещения курсора к объекту, используя в качестве параметра S наименьшее из значений размеров объекта по горизонтали или по вертикали (Mackenzie, 1995).

Закон Фитса позволяет объяснить, например, почему переместить курсор к меню в стиле Apple Macintosh, расположенному на границе экрана (рис. 4.7), намного быстрее, чем переместить курсор к меню в стиле Microsoft Windows, которое всплывает из-за границы экрана (рис. 4.8). Размер S меню в Windows на моем экране составляет 5 мм. Эффективный размер меню в Macintosh является довольно большим,



Рис. 4.7. Меню в Macintosh, расположенное у верхней границы экрана, увеличивается в размере, что является более эффективным по сравнению с меню, которое всплывает из-за границы экрана

потому что пользователю не требуется останавливаться в пределах полосы меню; он может просто передвинуть ГУВ на любое расстояние, превышающее расстояние, необходимое для размещения курсора в пределах меню, – курсор в любом случае остановится на границе экрана.

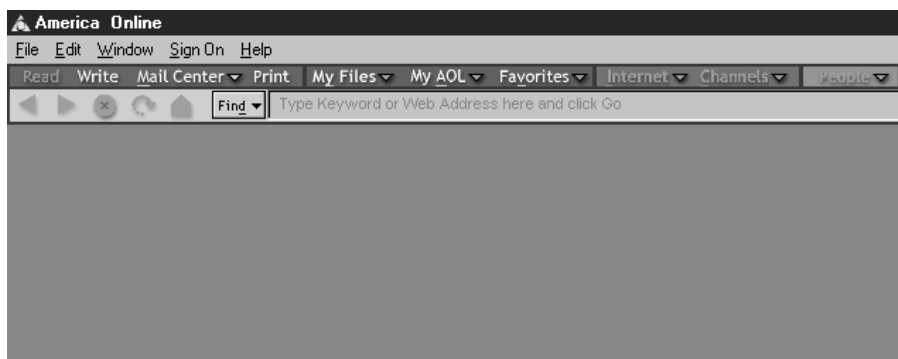


Рис. 4.8. Меню в Windows находится под верхней границей экрана, поэтому пользователю нужно более точно подводить курсор к меню, чтобы можно было вызвать подменю

Ряд выполненных мной тестов помог установить, что пользователи Macintosh обычно останавливаются в пределах 50 мм от границы экрана, поэтому для Macintosh мы можем принять 50 мм как S . При использовании 14-дюймового плоского монитора среднее расстояние, на которое требуется переместить курсор, чтобы достичь панели меню, составляет 80 мм. Таким образом, время перемещения курсора к какому-либо из элементов меню в Macintosh будет:

$$50 + 150 \log_2 (80 / 50 + 1) = 256 \text{ мс}$$

Это время является намного меньшим, чем то, которое требуется на перемещение курсора к необходимому элементу меню в Windows:

$$50 + 150 \log_2 (80 / 5 + 1) = 663 \text{ мс}$$

Полученные значения времени относятся только лишь к перемещению курсора. Щелчок по целевому объекту, к которому перемещался курсор, добавляет в среднем еще 0.1 с. (Значение оператора $K = 0.2$, принятое в модели GOMS, учитывает и нажатие кнопки, и ее отпускание, в то время как указанное значение учитывает только нажатие кнопки.) При проведении обычного эксперимента вам придется добавить еще 0.25 с, чтобы учесть время человеческой реакции в начале движения курсора. С учетом всех этих факторов мы получим именно те значения, которые я получил экспериментальным путем: в среднем пользователю требуется около 0.6 с, чтобы открыть Apple-меню, в то время как для открытия Windows-меню требуется более 1 с. Таким образом, этот анализ объясняет, почему при разработке интерфейса Macintosh меню были намеренно помещены на границе экрана.

4.4.2. Закон Хика

Перед тем как переместить курсор к цели или совершить любое другое действие из набора множества вариантов, пользователь должен выбрать этот объект или действие. В законе Хика утверждается, что когда необходимо сделать выбор из n вариантов, время на выбор одного из них будет пропорционально логарифму по основанию 2 от числа вариантов плюс 1, при условии, что все варианты являются равновероятными. В этом виде закон Хика очень похож на закон Фитса:

$$\text{Время (мс)} = a + b \log_2 (n + 1)$$

Если вероятность i -го варианта равна $p(i)$, то вместо логарифмического коэффициента используется

$$\sum_i p(i) \log_2 (1/p(i) + 1)$$

Коэффициенты, используемые в выражении закона Хика, в большой степени зависят от многих условий, включая то, как представлены возможные варианты, и то, насколько хорошо пользователь знаком с системой. (Если варианты представлены непонятным образом, значения a и b возрастают. Наличие навыков и привычек в использовании системы снижает значение b .) Мы не будем рассматривать эти зависимости — для нас важно, что для принятия того или иного решения требуется время; что для принятия сложных решений требуется больше времени, чем для принятия простых решений; и что взаимосвязь является логарифмической. При отсутствии более точных данных для проведения быстрых и приблизительных вычислений мы можем воспользоваться теми же значениями a и b , которые использовали для закона Фитса.

При использовании любых положительных и ненулевых значений a и b из закона Хика следует, что предоставление пользователю сразу нескольких вариантов одновременно обычно является более эффективным, чем организация тех же вариантов в иерархические группы. Выбор из одного меню, состоящего из 8 элементов, производится быстрее, чем из двух меню, состоящих из 4 элементов каждое. Если все элементы могут быть выбраны с равной вероятностью и если не учитывать время, необходимое для открытия второго меню (которое, конечно, еще более увеличило бы время для интерфейса, состоящего из двух меню), то сравнение времени для выбора одного элемента из восьми ($a + b \log_2 8$) с удвоенным временем для выбора одного элемента из четырех $2(a + b \log_2 4)$ покажет, что

$$a + 3b < 2(a + 2b)$$

поскольку $\log_2 8 = 3$, а $\log_2 4 = 2$, а также поскольку $a < 2a$ и $3b < 4b$.

Это согласуется с данными, полученными в экспериментах со структурами меню (см. например, Norman и Chin, 1988).

Наше рассмотрение законов Фитса и Хика нельзя считать полным. Например, следует обратить внимание на то, что эти законы не случайно принимают ту же форму, что и теорема Шэннона-Хартли (Shannon-Hartley). Тем не менее, этого короткого рассмотрения вполне достаточно для того, чтобы отметить их ценность с точки зрения разработки интерфейсов. Они могут быть полезными даже в том случае, когда эмпирические значения коэффициентов a и b не известны (как это было в нашем примере). (Более подробные сведения см. в Card, Moran и Newell, 1983, с. 72–74.)

5

Унификация

Это чрезвычайно хитроумно, чрезвычайно сложно и крайне эффективно, но в то же время грубо, неэкономно и топорно, и чувствуется, что есть лучший способ.

К. Стрэтчи (говоря не о Windows, а о компьютере IBM Stretch в 1962 г.)

Если пытаться создать универсальный интерфейс, в котором были бы учтены те требования, о которых шла речь в предыдущих главах, то выяснится, что для этого нужно радикально изменить нашу обычную практику. Здесь возможно много направлений. Одно из них заключается в том, чтобы посмотреть, что мы можем сделать в условиях существования Интернета и сотен миллионов компьютеров, а также других устройств обработки информации, которые уже существуют или которые только разрабатываются сегодня.

В настоящее время аппаратная конфигурация обычного персонального компьютера является почти универсальной. Если принять точку зрения, что внутри почти всех приложений, использующих общие аппаратные средства, акцент делается на унификацию физических действий, у нас появляется возможность разработать всеобъемлющий и в то же время простой интерфейс.

Набор действий, которыми пользователь влияет на содержание – будь оно текстовым, графическим или мультимедийным, – можно организовать в простую таксономию, с помощью которой мы сможем описать любой интерфейс в некой унифицированной форме. Такая организация позволила бы упростить разработку интерфейсов. Например, внедрение универсального средства «отменить/повторить» (undo/re-do) тоже позволяет создавать единообразные интерфейсы, тем самым

избавляя от необходимости придумывать средство обработки ошибок специально для каждой программы.

Разные приложения имеют разные наборы команд, и пользователь обычно не может в целом использовать команды приложения А при работе с приложением В или наоборот. Если же сделать команды независимыми от приложений, то тем самым мы сможем устранить модальность, которая изначально им присуща. При такой унификации общее количество команд, которое пользователю придется запоминать, существенно сократится, – главным образом потому, что унификация позволит избавиться от огромного числа повторений команд. Например, в компьютере Canon Cat с помощью всего 20 команд можно было управлять текстовым процессором, электронными таблицами, созданием, сортировкой и обработкой баз данных, вычислениями и т. д. В современных системах аналогичной мощности используется более 100 команд для выполнения того же самого набора задач. Тысячи команд, которые используются в современных средах, можно было бы сократить до сотни. Так как не все команды могут применяться ко всем типам данных, возникнет необходимость применять к объектам преобразователи типов данных, чтобы создать новые объекты, к которым при определенных условиях уже можно будет применить выбранную команду.

Кроме того, может быть снято и другое разделение, которое имеется сегодня между теми средствами, которые содержатся в коммерческих программных продуктах, и теми, которые могут быть созданы самим пользователем. Например, сегодня меню являются объектами операционной системы, которые устанавливаются в каждом приложении. Однако меню представляют собой всего лишь какой-то текст. Почему бы тогда не дать возможность пользователю самому составлять список часто используемых команд, защитить его от случайного изменения, прикрепить наверху экрана и использовать его как обычное системное меню? Для упрощения создания таких меню в интерфейсе можно было бы предусмотреть возможность блокировать и разблокировать какой-то текст, а также возможность его перемещения вместе с другим содержанием или закрепления в каком-то месте на экране. Текст может быть в разных состояниях.

Хотя Eudora и Microsoft Word являются программами, в которых можно изменять меню, тем не менее, для изменения его содержания вы должны использовать только специально предназначенные для этого средства. В данном случае мы как раз и говорим о том, что должна быть возможность создавать меню обычными средствами создания и редактирования текстов. В этом смысле меню можно рассматривать как содержание.

Еще одним шагом к упрощению интерфейса является устранение трудно запоминаемых и неудобных файловых имен, а также системных файловых структур. При наличии хороших механизмов поиска

использование имен файлов и файловых структур перестает быть необходимым.

5.1. Унификация и элементарные действия

Сущности не должны множиться без необходимости.

Уильям Оккамский

Набор аппаратного оборудования, из которого состоит интерфейс компьютера, стал стандартным – одно или несколько устройств для ввода текста (клавиатура, планшет для письма, устройство распознавания речи), ГУВ и двухмерный цветной дисплей. Эта, в общем неплохая, формула может иметь некоторые различия. Например, сенсорный экран может использоваться одновременно в качестве устройства ввода текста, ГУВ и дисплея. Микрофоны, устройства ввода видеоданных и другие устройства обычно не входят (кроме случаев экспериментирования) в состав обычного человеко-машинного интерфейса. На самом деле мы используем интерфейс для того, чтобы контролировать функционирование этих устройств.

Если вы посмотрите на человека, который работает с каким-нибудь существующим сегодня стандартным компьютерным оборудованием, но не будете видеть, что отображается на экране монитора, и знать, какую задачу оператор выполняет, и, в общем, не сможете предположить, что он делает. Конечно, здесь возможны исключения. Если вы видите, что пользователь пристально смотрит на экран и маниакальным образом вращает ручку джойстика под ритмичные и повторяющиеся звуки, то сможете догадаться, что, скорее всего, он играет в какую-то компьютерную игру. Но, в целом, действия пользователя при использовании одного приложения, например текстового процессора, в большой степени похожи на действия, которые он выполняет при использовании других приложений, например баз данных или электронных таблиц.

Такое однообразие действий пользователя в разных приложениях подсказывает нам, что интерфейсы для различных приложений не так уж и отличаются друг от друга, как вам самим это может показаться при использовании этих приложений. Приложения отличаются друг от друга больше потому, что вы обращаете внимание на содержание того, что выполняется, т. е. на различные изменения смысла каждого действия. В частности, вы не обращаете внимание на физические действия, которые выполняете при работе на компьютере.

Другой аспект, который является общим почти для всех приложений, заключается в том, что при их использовании требуется вводить какой-то текст. (Даже в играх вам иногда приходится вводить собственное имя в случае выигрыша.) Поэтому стоит подумать над тем, чтобы обработка текста – будь это небольшой текст, как, например, цепочка символов в строке поиска, или, наоборот, большой текст, как,

например, текст романа – была обеспечена набором удобных и эффективных команд.

И люди, и наше программное обеспечение не являются совершенными. Не все нажатия клавиш, движения пером или речевые действия приводят к отображению именно тех символов, которые нужны. Поэтому в интерфейсе должна быть предусмотрена возможность сразу стирать символы на экране с помощью клавиши Backspace или Delete и применять эти средства и для других форм введенных данных. Для внесения еще больших изменений, как, например, добавление абзаца, требуется предусмотреть возможность выделения и удаления целых областей. В отношении больших участков текста также важно, чтобы пользователь имел возможность переместить курсор в любое место текста, чтобы вставить туда символы. Другими словами, всякий раз, когда вводится текст, пользователь ожидает, что в его распоряжении должны быть многие возможности текстового процессора.

Когда вы вводите текст, вы помещаете его в какой-то документ или поле, как, например, область формы, предназначенная для ввода своего имени. В существующих сегодня системах допустимые функции редактирования различаются в зависимости от поля или типа документа, – для документа текстового процессора это могут быть одни правила редактирования, для электронных таблиц – другие. Правила редактирования могут различаться даже внутри одного документа, который содержит элементы, созданные с помощью других приложений (в разделе 5.7 мы рассмотрим одно из решений этой проблемы).

Два различных, но внешне схожих сегмента программного обеспечения какой-либо системы могут быть для пользователя большим источником ошибок и негативных эмоций. Однако именно такая ситуация наблюдается почти во всех персональных компьютерах. На моем компьютере установлено 11 текстовых редакторов, в каждом из которых имеется свой набор правил редактирования. Возможно даже, что в нем имеются и какие-то другие редакторы, которые я упустил. Таким образом, все это приводит к бесполезной путанице.

Для создания человекоориентированного интерфейса для компьютеров или компьютерных систем (таких, например, как Palm Pilot) важным шагом является обеспечение одинаковых правил для всех случаев, в которых вводится или редактируется текст. Например, в Macintosh или Windows вы не можете при вводе имени файла сделать его орфографическую проверку, поэтому, если вы не уверены в правильности написания слова *randevu* (*randezvous*), которое вы хотите использовать в качестве имени файла,¹ то вам придется вводить его наугад или открыть текстовый процессор и набрать в нем (или скопировать) это слово, чтобы проверить правильность написания. Могу пред-

¹ Конечно, если в вашей операционной системе недопустимы длинные имена, то здесь появляется еще одна проблема.

положить, что если бы я подал разработчикам программного обеспечения идею о том, чтобы пользователь мог проверять орфографию файловых имен, то они, весьма вероятно, и добавили бы такую возможность, но такое особое добавление, – которое, вероятно, было бы решено в виде какого-нибудь нового элемента в одном из системных меню (скорее всего, меню Правка), – только бы увеличило и без того абсурдную сложность программного обеспечения. Наилучшим решением здесь было бы упрощение на основе уже описанной идеи, заключающейся в том, что одна команда орфографической проверки должна применяться к любому тексту, независимо от того, какую роль он играет в данный момент.

Разработка интерфейсов должна быть основана на идее, что *любые объекты, которые выглядят одинаково, одинаковы*. Этот принцип может существенно упростить интерфейс с точки зрения как пользователя, так и разработчика и может быть применен не только в отношении текстов. Любой объект в этом случае становится состоятельным. Если пользователь не может по виду объекта на экране определить, что с ним можно и чего нельзя делать, это означает, что ваш интерфейс не удовлетворяет критерию видимости, который мы обсуждали в разделе 3.4. Тем самым вы ставите пользователя в положение, при котором ему необходимо догадываться о том, какие операции допустимы и к каким последствиям они могут привести. Техника создания интерфейсов, при которой пользователю в результате приходится догадываться о возможностях элементов программного обеспечения, является более подходящей для разработки игр, чем прикладных инструментов.

В идеале невозможно достичь того, чтобы по внешнему виду всегда можно было определить функцию. Например, один объект может быть очень похож на какой-то другой. Растровое изображение текста выглядит точно так же, как и сам текст не в графическом формате, однако в современных системах невозможно применять операции текстового редактирования к растровым изображениям. Эта проблема может быть частично решена, если в системе будет предусмотрена возможность конвертирования объекта в тот формат, в котором данная операция может быть к нему применена (об этом см. далее в разделе 5.8).

5.2. Каталог элементарных действий

Если вы разрабатываете интерфейс, то должны знать палитру всех его возможностей, аналогично тому, как художник имеет на своей палитре набор всех возможных красок. Спектр **элементарных действий**, которые пользователь может выполнить, довольно ограничен. Все взаимодействие между пользователем и интерфейсом построено на этом наборе элементарных действий. С помощью клавиатуры вы можете стучать по клавишам или же нажимать и удерживать их, выполняя при этом какие-то другие действия. С помощью ГУВ вы можете перемещать курсор в пределах экрана (или экранов) вашей системы и, та-

ким образом, управлять компьютером, регулируя скорость, направление и ускорение движения ГУВ (хотя обычно скорость и ускорение движения ГУВ используются опять же только с целью указания). С помощью кнопки ГУВ вы можете передавать информацию о том, на какое место на экране монитора вы хотите указать. Все эти элементарные действия могут иметь весьма различный смысл в зависимости от того, в каком приложении они применяются.

Сенсорные графические планшеты могут регистрировать угол наклона пера, что связывает с каждой указанной пользователем позицией еще два числовых значения. Эти значения редко используются за исключением тех случаев, когда пользователь занимается рисованием от руки. Музыкальные клавиатуры позволяют ввести в компьютер как скорость, так и силу, с которой клавиша нажимается. Кроме того, существуют такие устройства, как джойстики и устройства ввода трехмерных данных. Тем не менее, в большинстве случаев используется обычная клавиатура и стандартное, двухмерное ГУВ. В этом разделе будут рассмотрены, главным образом, стандартные устройства ввода и вывода данных. Во многих случаях будет понятно, каким образом излагаемые принципы могут быть распространены и на более необычные физические или даже ментальные интерфейсы. Полагаю, что ясная таксономия и перечень элементарных действий, а также выполняемых с их помощью операций, могут быть весьма полезными для обсуждения и разработки интерфейсов.

Элементарные действия, выполняемые пользователем в различных комбинациях, порождают набор **элементарных операций**, которые применяются к содержанию и используются почти во всех интерфейсах. Перечислим, какие операции могут быть применены к содержанию:

- **Указание.** Пользователь может указать на то или иное содержание.
- **Выделение.** Пользователь может выделить какое-то содержание.
- **Активизация.** С помощью «клика» пользователь может активизировать содержание.
- **Модификация** или использование (с помощью команд):
 - **Генерация.** Модификация из «пустого» в «непустое».
 - **Удаление.** Модификация из «непустого» в «пустое».
 - **Перемещение.** Вставка содержания в одно место и одновременное его удаление из другого.
 - **Трансформация.** Преобразование в другой тип данных.
 - **Копирование.** Содержание может быть отправлено или получено от внешнего устройства или скопировано в другую область внутри системы. Например, содержание можно распечатать, отправить по электронной почте, сохранить на жестком диске, копировать в другой документ и т. д.

Эти элементарные операции могут и должны быть основой компьютера или самой программы, т. е. они должны являться частью аппаратного или базового программного обеспечения, а не входить в состав множества программных пакетов, и каждая элементарная операция должна всегда вызываться одинаковым образом, независимо от того, к каким объектам они применяются. В основном *когнитивные различия между программами заключаются в способах представления выделенного содержания и того, как пользователь может с ним оперировать*. В электронных таблицах значения представляются в табличной форме, а применяемая к ним операция может состоять в том, что столбец без итогового значения внизу преобразуется в столбец, внизу которого указывается сумма значений всех его ячеек. В текстовом процессоре текст и иллюстрации представляются в виде страниц, а типичной операцией, применяемой к ним, является изменение начертания текста с обычного на наклонное. В программе обработки веб-страниц страница из текстового процессора может быть преобразована в HTML-формат. В программе обработки фотоизображений фотография с низким контрастом может быть преобразована в фотографию с высоким контрастом.

Большинство операций, выполняемых с содержанием, можно описать с помощью этих элементарных операций. Например, во многих системах имеется возможность сделать запрос о свойствах какого-нибудь объекта. (Если система оснащена двухкнопочным ГУВ, пользователь обычно может выполнить это действие с помощью нажатия на правую кнопку при условии, что курсор наведен на этот объект и система находится в соответствующем состоянии.) **Запрос** свойств объекта означает, что необходимо получить дальнейшую информацию об элементе или набор связанных с ним опций. Но также его можно рассматривать и как операцию, примененную к одному объекту, чтобы вывести на экран связанный с ним другой объект. С точки зрения пользователя, нет необходимости в том, чтобы операции, выполняемые в операционной системе, отличались от операций, выполняемых в приложениях, и поэтому такого различия не должно быть.

То, что интерфейсы всех приложений основаны на небольшом наборе элементарных операций, подтверждает тот факт, что приложения как таковые не очень отличаются друг от друга с точки зрения интерфейса, независимо от того, насколько они сложны и разнообразны с точки зрения задач, для которых они предназначены. Такое базовое подобие можно использовать для создания мощных компьютерных систем, обладающих беспрецедентной степенью простоты и эффективности.

Для начала нам следует определить несколько методов отбора и выделения содержания, к которому предполагается применить какую-то операцию. Эти методы будут рассмотрены в разделе 5.2.1.

5.2.1. Подсветка, указание и выделение

Подсветка (highlighting) означает, что с помощью каких-либо средств отображенному на экране объекту придается заметное отличие. Функция подсветки заключается в том, чтобы пользователь мог, пассивно наблюдая изображение на экране, определить, что некоторый объект получил от системы особый статус. Семантика этого статуса зависит от типа объекта и от команд, которые пользователь может применить к данному объекту. Для зрячих пользователей выделение обычно визуально. В качестве визуальных методов выделения может использоваться обращение яркости, изменение цвета или контраста, подчеркивание, мигание или любое другое периодическое изменение, добавление к объекту статичной или анимированной рамки. В качестве не визуальных методов выделения может использоваться набор разных голосов или изменение интонации.

Когда пользователь наводит курсор на какие-то объекты, они должны быть подсвечены. Типичным объектом в текстах является символ. Подсветка единичного объекта при перемещении курсора без каких-то других действий со стороны пользователя (как, например, нажатие на кнопку мыши) является **указанием** (indication). С помощью указания пользователь может в любой момент знать, на какие объекты он указывает с точки зрения системы. В очень многих современных системах пользователь должен догадываться о том, что будет выделено или активировано при нажатии на кнопку ГУВ. Если догадка неверна, пользователю придется сделать другую попытку, что приводит к потере времени и сил. Указание может быть особенно полезным, когда объекты, которые пользователь хочет выделить, имеют небольшие размеры или расположены близко друг к другу, или перекрывают друг друга, или их границы неясны. Указание необходимо в тех случаях, когда интерфейс разработан в соответствии с принципом видимости.

Подсвечивание, используемое для указания, не должно быть слишком контрастным или ярким, чтобы движение курсора не вызывало раздражение. В некоторых случаях полезно, чтобы указание объектов не происходило, если скорость перемещения курсора превышает определенное пороговое значение. Следует обратить внимание на то, что чем меньше объект (т. е. чем меньше визуальный угол указанного объекта), тем больший контраст должен использоваться для его указания — однако это вопрос эргономический.

Указание недостаточно используется в современных системах. Активное использование указания в разработке интерфейса позволяет существенно сократить количество щелчков мышью в сравнении с современными интерфейсами. По сути дела, указание часто может заменить клик мышью, и вместо двойного щелчка можно делать только один, как при выборе ссылки в браузере. Допустим, что пользователь хочет убрать неактивное окно с экрана. В каждом окне имеется кнопка **Закрыть**. Для этого как в операционной системе Windows, так в Macin-

тош пользователь должен сначала щелкнуть по окну, чтобы сделать его активным, и только потом нажать на кнопку Закрыть. Этот лишний щелчок, который делается для активизации того окна, которое пользователь, на самом деле, хочет закрыть, вызывает особое раздражение. Но если бы окно можно было активизировать всего лишь перемещением к нему курсора, то для закрытия окна одного нажатия на кнопку мыши было бы достаточно. Конечно, если вы разработаете систему, в которой активизация происходит только в определенных местах или при определенных условиях, то тем самым вы создадите модальное противоречие, которое будет только сбивать пользователей с толку. Активизация должна происходить системно. Поскольку такой подход становится более известным, спрос на него со стороны пользователей увеличит его распространение.

Выделение (selecting) – это процесс, с помощью которого пользователь указывает, что один или несколько объектов имеют особый статус, который может быть воспринят системой. Как результат процесса получается **выборка (selection)**. Обычно пользователь делает выборку с целью применения к ней в ближайшем времени команды. В отличие от менее постоянного указания, выделение, обозначающее выборку, является более устойчивым и сохраняется, даже если пользователь отведет курсор в сторону. Пользователь может выделить объект, щелкнув по кнопке ГУВ, указывающего на него. Кроме того, пользователь может сделать выделение расположенных рядом объектов с помощью вырисовывания прямоугольника или другой фигуры, при этом все объекты, которые окажутся в области фигуры, будут выбраны. Другим удобным способом отбора является создание многоугольника или произвольной фигуры. В этом случае все объекты, оказавшиеся внутри фигуры, будут выбраны после того, как пользователь замкнет ее границу. После того как выбор сделан, предыдущая выборка должна стать **старой выборкой (old selection)**. (В большинстве современных систем старая выборка просто-напросто отменяется (deselect).) Этот процесс может быть многократно повторен, поэтому пользователь может создать дополнительно к первой старой выборке вторую старую выборку, третью и т. д. – вплоть до n -й старой выборки. У математика здесь, скорее всего, возникнет желание назвать текущую выборку нулевой старой выборкой. Выделение, с помощью которого обозначается выборка, должно быть более заметным и отличаться от того, которое используется для указания. Выделенные старые выборки также должны хорошо отличаться друг от друга – возможно, с уменьшением визуального контраста для более старых выборок. Для легкого распознавания старых выборок они могут иметь буквенно-цифровые обозначения.

Выборка может включать как отдельный объект на экране, так и геометрическую область, или же быть **составной** из различных выборок. В большинстве современных интерфейсов пользователь делает составные выборки – в том числе и разрывные – из набора отдельных выборок,

для чего необходимо сделать начальную выборку. Затем, как правило, пользователь может нажать клавишу <Shift> и, удерживая ее и находясь, таким образом, в квазирежиме, щелкнуть по другим объектам, чтобы присоединить их к общей выборке или отсоединить их от нее.

Однако этот способ имеет три недостатка. Во-первых, команда для создания составных выборок является невидимой. Во-вторых, при создании большой составной выборки легко допустить ошибку (например, если пользователь случайно отпустит клавишу <Shift> и щелкнет по следующему объекту, вся сложная выборка, которая была создана к этому моменту, будет потеряна). В-третьих, механизм используется как «переключатель»: один и тот же жест служит как для отмены выделения (если объект был уже выделен), так и для установки выделения (если объект был не выделен).

Первая проблема заключается в отсутствии видимости и может быть легко решена с помощью, например, экранной подсказки. Вторая проблема заключается в том, что при составлении выборки имеется большой риск совершения ошибки. Более удобный способ создания сложных выборок состоит в наличии специальной команды, с помощью которой текущая выборка определяется как объединение старого и текущего выделения. Такая команда позволила бы пользователю сосредоточиться на создании выборки, не заботясь о том, что было выбрано до этого. Только после подтверждения текущего выделения она может быть добавлена к составной выборке. Возможность обращения к старым выборкам и обозначения каждой из них особой подсветкой позволяет применять команды с множеством аргументов, как, например, использование двух аргументов для команды взаимозамены двух выборок. Сравните метод перестановки двух участков текста, который вы используете сейчас, с другим методом: создание двух выборок и затем применение команды перестановки.

В большинстве существующих сегодня систем команды Отменить (Undo) и Повторить (Redo) нельзя применить к процессу создания выборок. Это не совсем оправданно, поскольку ошибки при создании выборок случаются довольно часто. Необходимым элементом любого человеко-ориентированного интерфейса являются универсально применимые команды Отменить и Повторить. Число или уровни допустимой отмены выполненных команд должны ограничиваться только лишь объемом доступной памяти. Команды Отменить и Повторить должны быть всепроникающими и применяться к любой операции, которая логически может быть повторена или отменена. Также эти команды должны быть обратными друг другу (инверсивными) – опять же в той мере, насколько это логически возможно. Это означает, что выполнение команды Повторить после команды Отменить или выполнение команды Отменить после команды Повторить не должно приводить к изменениям в содержании. Очевидно, что эти команды не должны применяться к самим себе. *Операторы Отменить и Повторить являются осново-*

полагающими, и их функция настолько важна, что в будущих системах для них должна быть предусмотрена специальная клавиша. Команда Повторить должна назначаться следующим образом: *Shift*↓ *Undo*↓↑↑. На клавише должны быть ясно обозначены два слова: *Отменить* (Undo) и *Повторить* (Redo) (рис. 5.1). Такая клавиша могла бы с большей пользой заменить собой вызывающую много проблем клавишу <Caps Lock>.



Рис. 5.1. Клавиша Отменить/Повторить (Undo/Redo)

Что касается третьей проблемы, то в разделе 3.2, посвященном переключателям, я уже говорил, что в человекоориентированном интерфейсе переключатели вообще не должны использоваться. В данном случае простым решением проблемы могло бы быть использование одной команды или квазирежима для добавления объекта к выборке, а другой команды или квазирежима – для удаления объекта из выборки. При попытке добавить к выборке объект, который там уже имеется, или удалить из выборки объект, которого в ней нет, сама выборка останется неизменной.

Интерфейс обычно имеет одну точку, в которой, как предусматривают разработчики, должен проходить процесс взаимодействия между пользователем и системой – точку **фокуса**. Например, если вы печатаете слепым методом и набираемый вами текст появляется на экране, то место, где появляется текст, является фокусом и часто совпадает с локусом вашего внимания. Если вы не владеете методом слепого набора, то ваш локус внимания будет перемещаться между клавиатурой и дисплеем. В интерфейсах в каждый момент обычно имеется только один курсор. Его позиция определяется с помощью ГУВ, клавиш управления курсором или команд (например, Найти (Find)).

Локус внимания всегда находится на каком-то физическом, ментальном или отображаемом объекте. То же самое можно сказать и о системном фокусе. Например, в существующих текстовых процессорах при перемещении курсора внутрь документа (действие, которое само по себе не должно быть необходимым) он может быть расположен между

двумя буквами, и, таким образом, может показаться, что никакой объект не является фокусом системы. На самом деле в фокусе находятся две буквы – та, которая слева и может быть удалена командой Delete¹, и та, которая справа и где появится следующая введенная буква.

Когда процессом взаимодействия управляет человек, в фокусе обычно находится текущая выборка. Если же система отвечает на действие пользователя или внешней системы, в фокусе обычно находится результат действия.

5.2.2. Команды

Я как писатель-фантаст уверен, что этот чертов робот должен говорить на человеческом языке, а не наоборот.

Снайдер Робинсон

Некоторые команды (как, например, Отменить) не обязательно могут применяться к выборкам. Другие команды действуют только по отношению к текущей выборке – как, например, команда, которая удаляет текущую выборку. Некоторые из этих команд вводятся с клавиатуры. Однако число клавиш на клавиатуре меньше, чем количество возможных команд. Каждая дополнительная клавиша-модификатор (как, например, <Shift>, <Alt>, <Command>, <Control> или <Option>) позволяет удвоить число клавиатурных команд. Полная клавиатура, в которой любая комбинация клавиш может быть воспринята компьютером, допускает астрономическое число сочетаний клавиш. Например, программное обеспечение, в котором предусмотрено использование любых трехклавишных дваждыквазимодальных сочетаний на 110-клавишной клавиатуре, позволяет передать более одного миллиона команд с помощью только одного жеста. Однако широкое использование клавиш-модификаторов, особенно в сочетаниях клавиш, очень часто приводит к появлению чрезвычайно сложных комбинаций, в которых можно просто «сломать пальцы». Кроме того, комбинации редко бывают запоминаемыми или понятными. (Знаете ли вы, какое действие выполняет на вашем компьютере сочетание <Control>+<Shift>+<Option>+<\\>?) Запомнить различные сочетания клавиш не просто. Такое запоминание является недопустимым требованием к памяти пользователя. Кроме того, такие команды нарушают критерий видимости, если только в системе каким-то образом не отображается то, какой результат будет получен при применении той или иной команды. Конечно, если в отдельных случаях какой-то из этих жестов не может быть выполнен или если жест имеет разные значения в разные моменты, это означает модальность системы по отношению к данному жесту, что приводит к проблемам, которые обсуждались в главе 3.

¹ Левый символ обычно удаляется командой Backspace. – *Примеч. науч. ред.*

Если разделить систему на приложения таким образом, что данная команда может многократно использоваться, но с разными значениями в разных приложениях, появляется возможность увеличить количество команд, которые пользователь может вызвать для заданного множества сочетаний клавиш. Однако использование команд в приложениях, в которых для них устанавливаются разные значения, приводит к модалным ошибкам. Кроме того, различающийся смысл жеста создает ненужные трудности для его запоминания. Частично эта трудность облегчается с помощью меню. Тем не менее, пользователю все равно приходится запоминать, где находится та или иная команда. (Возможно, пользователю даже придется сначала вспомнить, в каком именно приложении используется необходимая ему команда, особенно если используется несколько приложений с подобными функциями.) Процесс просмотра меню иногда становится привычным, но иногда он оказывается утомительным, особенно если искомая команда находится в каком-то из подменю и если способ организации меню, который показался разработчику очевидным, не является таковым для пользователя.

Для назначения команд требуется такой метод, который был бы таким же быстрым и физически простым, как нажатие на пару клавиш, и с помощью которого было бы проще и быстрее найти необходимую команду, чем с помощью системы меню. Нежелательно повторять двойной метод, который используется в большинстве известных графических пользовательских интерфейсах и включает в себя как систему меню, так и набор непонятных горячих клавиш. Например, нет ничего запоминаемого в сочетании

Command ↓ *v* ↓ ↑ ↑

которое используется для вставки, кроме того, что клавиша *v* расположена рядом с клавишей *c*, используемой в сочетании

Command ↓ *c* ↓ ↑ ↑

которое запоминается несколько лучше, поскольку *c* может напоминать слова «вырезать» (*cut*) или «скопировать» (*copy*).

Другой метод позволяет решить многие из этих проблем. Предположим, что на клавиатуре есть клавиша <Вычислить> (*Calculate*). При нажатии на эту клавишу текущая выборка рассматривается как арифметическое выражение и вычисляется. Далее я буду использовать подчеркивание, чтобы показать выборку. Допустим, что текст является следующим:

Я хочу купить 3 + 4 рубашек

При нажатии на клавишу <Вычислить> он будет преобразован в следующий текст:

Я хочу купить 7 рубашек

До нажатия на клавишу <Вычислить> $3 + 4$ было обычным текстом. Он ничем не отличался от остального текста, за исключением того, что он был выделен. Пять символов (включая пробелы), из которых состояла выборка, можно было переместить или удалить, или же к ним могла быть применена любая другая обычная команда текстового процессора. Но в данном случае была использована операция Вычислить. Пользователю не потребовалось открывать окно калькулятора или вызывать специальное приложение.

Теперь рассмотрим случай, когда на клавиатуре нет клавиши <Вычислить>. (Хотя специальная клавиша для вычисления математических выражений – это пока только ценная идея, но она, несомненно, была бы более полезной, чем те клавиши, которые уже существуют, как, например, <F9>.) То, что нам необходимо – это более общий механизм для команд.

Перед обсуждением такого механизма рассмотрим требования, которым должен отвечать новый метод вызова команд:

- Этот механизм не должен быть модалным
- Он должен включать любое число команд (в частности, он не должен ограничиваться количеством клавиш на клавиатуре)
- Пользователь должен иметь возможность вызывать команду непосредственно с клавиатуры
- Пользователь должен иметь возможность вызывать команду с помощью графического устройства ввода
- Использование механизма не должно требовать изобилия специальных клавиш
- Механизм не должен порождать большого количества квазирежимов

Проиллюстрируем один общий метод примером. (Этот несколько тривиальный арифметический пример используется только для того, чтобы продемонстрировать данный метод. Более эффективные способы применения этого метода будут рассмотрены далее.) Предположим, что имеется следующий текст:

Я хочу купить $3 + 4$ рубашек вычислить

Выделим сумму $3 + 4$, и затем выделим слово *вычислить*, при этом сумма станет старой выборкой.

В другом методе, который предоставляет пользователю всю мощь командной строки, при нажатии на клавишу <Command> вызывается выбранная команда. Если для выполнения команды требуется аргумент, то в его качестве используется старая выборка. В данном методе сама команда удаляется, а результат вычисления становится левой выборкой.

Я хочу купить 7 рубашек

Суть заключается в том, что команды не должны ограничиваться только меню, но могут быть частью вашего текста, или же, если это уместно, команда может быть представлена графическим объектом, а не словом или набором слов. Важно также и то, что в этом случае пользователь может назначать команды самым простым способом – всего лишь набирая их с клавиатуры или рисуя их. Такой способ не противоречит методу, при котором команда выбирается из уже имеющегося списка.

Преимуществом меню является то, что список команд, из которых оно состоит, является видимым. Тем не менее, вместо того чтобы выбирать команду из меню, пользователь может с такой же легкостью выбирать команду из небольшого документа, в котором содержится список всех команд. Такой документ может быть составлен как разработчиками, так и самим пользователем. Кроме того, в этом документе может быть не только список команд, но и, например, описания команд и даже заметки, сделанные пользователем. Такой документ, служащий в качестве меню, может использоваться как обычный текстовый документ, а не как нечто, что может быть изменено только программистами или только с помощью специальных средств настройки.

Такой подход имеет ряд преимуществ. Например, онлайн-руководства автоматически содержат примеры использования команд, которые в нем описываются. В современных системах команды, имеющиеся в меню, могут как иметь, так и не иметь своего клавиатурного аналога. Однако при таком подходе каждая команда, описанная в меню последовательностью символов, имеет свой клавиатурный эквивалент. Это обеспечивается не благодаря стараниям разработчиков, а по самой природе такой системы. Как команда из меню, так и клавиатурный эквивалент имеют одинаковое написание. И большинство пользователей будут чаще использовать именно клавиатурный эквивалент. Другим преимуществом этого подхода является то, что вы можете составить меню только из тех команд, которые вы используете, просто набрав их в список с помощью текстового процессора. Конечно, если вы постоянно изменяете список, а не просто, скажем, добавляете к нему новые команды, то теряется преимущество их привычного расположения.

Аналогично тому, как гиперссылки в тексте часто выделяются визуальными методами, например изменением цвета (обычно на синий) и подчеркиванием, команды также могут обозначаться какими-то особыми способами (например, красным цветом или обратным курсивом). При таком выделении пользователь сможет указывать на имя команды (или на какую-то из букв ее имени) и затем вызывать ее нажатием на клавишу <Command>. В этом случае выделение команды для ее вызова перестает быть необходимым.

Если для команд не использовать особый шрифт или цвет, то придется предусматривать другие правила для обозначения того, что какое-то слово или последовательность слов следует рассматривать как единую

команду. Разумнее было бы избежать использования каких-то из существующих сегодня условностей, которые используются для группировки отдельных слов в одну единицу, разделенную пробелами или другими символами. Например, если бы у нас была команда, которую мы бы хотели назвать «преобразовать изображение в формат JPEG», то по существующим правилам нам пришлось бы написать ее в виде преобразовать.изображение.в.формат.JPEG, <преобразовать изображение в формат JPEG> или преобразовать_изображение_в_формат_JPEG. Такие способы написания являются слишком «компьютерными», неудобными и унылыми, особенно для тех, кто только знакомится с компьютерами. *Синтаксис, который мы выбираем для написания команд, не должен исключать пробелы или символы новой строки.* Любые ограничения, накладываемые на набор допустимых символов, используемых для набора команд, в будущем приведут к проблемам. Более того, любые подобные ограничения пользователю придется учитывать при назначении команде имени. Следует также иметь в виду и другой принцип, заключающийся в том, что *использование обычаев, которые не совпадают с традициями обычной речи, приводит к возникновению непонимания между пользователем и компьютером. Следует сделать так, чтобы машина подстраивалась под нас, а не подстраивать обычаи естественной речи под то, что проще решить с точки зрения программирования.*

Еще одно взаимодействие ввода с клавиатуры и создания выборки вызывает проблемы в современных интерфейсах. В человекоориентированном интерфейсе ввод с клавиатуры не замещает выделенный текст и не отменяет выделение текущей выборки. Это отличается от распространенного правила, в соответствии с которым ввод символов замещает содержимое текущей выборки, что время от времени приводит к проблемам в тех случаях, когда новый материал неожиданным для пользователя образом удаляет текст, который он не собирался удалять. Идея, что при вводе символов они должны замещать выборку, стала использоваться для того, чтобы сэкономить на одном нажатии клавиши – в большинстве редакторов, чтобы заместить текстовый блок, вы можете просто выделить его и начать ввод. Без этого правила пользователю придется сначала выделить некоторый текст, затем нажать <Backspace> или <Delete> и потом вводить новый текст. В способе, который используется сегодня, экономится только лишь нажатие на клавишу <Backspace>, при этом при первом нажатии текст исчезает и замещается текстом, который вводится далее. Это происходит независимо от того, находился ли замещаемый текст на экране или нет и (обычно) состоял ли он из нескольких символов или трех четвертей вашего документа. Таким образом, вне локуса вашего внимания вы можете удалить текст в 40 страниц. Если вы вовремя заметите это, то, наверное, сможете отменить действие. Но если же вы не заметите удаление, и если ничего не подскажет вам, что текст был удален, вам может не повезти. Человекоориентированный интерфейс никогда не подвер-

гает работу пользователя риску. Экономия одного нажатия на клавишу куплена за слишком большую цену. Случайная потеря хотя бы одного символа может означать потерю части телефонного номера или адреса электронной почты, которая не может быть восстановлена из оставшейся части. *Удаление текста должно проходить явным образом по желанию пользователя и не быть побочным эффектом другого действия.*

Понятие локуса внимания позволяет точно определить, что мы определяем как побочный эффект. **Побочный эффект** – это следствие применения команды, при котором изменяется содержание или события, которые не находятся в локусе вашего внимания. В только что рассмотренном случае в локусе вашего внимания находится текст, который вставляется, а побочным эффектом является удаление. *Устранение побочных эффектов должно быть одной из целей для разработчика человекоориентированного интерфейса.*

Другой возможностью текстовых процессоров, которая часто считается полезной, является перетаскивание выделенного текста из одного места в другое. Однако при этом у пользователя нет возможности создать другую выборку, которая пересекалась бы с текущей, или создать подвыборку текущей выборки. Если вы попытаетесь сделать какую-нибудь из этих выборок, система воспримет это действие как попытку переместить текущую выборку. Это означает, что сначала вам следует щелкнуть в каком-то месте вне выборки, чтобы снять ее выделение. Таким образом, жесту перетаскивания были даны два разных значения, а именно: выделение и перетаскивание выборки. Это может препятствовать формированию привычки. Ошибки возникают в результате того, что, хотя символы выборки находятся в локусе вашего внимания, текущее состояние выборки не находится в локусе вашего внимания, несмотря на то, что оно визуальное выделено. Во время моих наблюдений я видел, как некоторые пользователи могли случайно перетаскать выборку, хотя собирались создать новую выборку.

Другая проблема возникает при перетаскивании текста (она также может возникать и в графических программах): иногда случается так, что как только вы начинаете перетаскивать выборку, выясняется, что место, куда вы хотите перенести ее, не видно на экране. В этом случае приходится возвращать выборку обратно или ставить ее в другое место и применять метод вырезания и вставки. Принцип монотонности предполагает, что предпочтительным является наличие только одного метода. В некоторых системах при перетаскивании выборки к нижнему или верхнему краю экрана автоматически начинается его прокрутка, но она происходит слишком медленно, если место назначения находится на расстоянии нескольких страниц. Кроме того, прокрутка может происходить и слишком быстро, что не позволяет остановиться или даже заметить необходимое место.

Если бы не маркетинг, я бы не стал оснащать интерфейс возможностью перетаскивания текста, по крайней мере, в том виде, в каком эта опция сейчас используется на персональных компьютерах. В этом случае меньшее число ошибок и проблем было бы хорошей компенсацией для тех пользователей, которые привыкли к возможности перетаскивать текст. Желательно, чтобы для выделения и для перетаскивания использовались отдельные квазирежимы, поскольку тогда не будет возникать «когнитивная путаница» между этими опциями. Например, ГУВ может быть снабжено кнопкой для создания выборок, а также специальным устройством (например, встроенной сбоку кнопкой), с помощью которой можно было бы сжимать ГУВ (при этом для того, чтобы было понятно, что действие произошло, важно наличие тактильной обратной связи, например в виде щелчка кнопки), что означало бы, что вы забрали выборку. В этом случае между этими функциями не возникало бы никакой путаницы (или она была бы минимальной). После нескольких секунд объяснений и одной-двух попыток использования такого устройства уже было бы понятно, как им пользоваться. Более простой метод для разделения функций выделения и перетаскивания заключается в том, чтобы использовать другую кнопку мыши для перетаскивания или же использовать квазирежим (например, удерживание в нажатом положении специальной, ясно обозначенной кнопки во время использования основной кнопки мыши). (См. более подробно в приложении А.)

Функцию перетаскивания, которой может быть оснащен ГУВ, можно также использовать вместо прокручивания. Вы можете захватывать какую-то часть документа и с помощью этого устройства перемещать ее: вверх или вниз – в узких документах, и во всех направлениях – в широких документах. Когда курсор перетаскивания (который в некоторых из сегодня существующих системах разумно отображается в виде руки) достигает границы экрана, прокручивание продолжается в выбранном направлении до тех пор, пока пользователь не отпустит устройство перетаскивания или не вернет курсор обратно внутрь окна. Метод прокручивания с помощью полос прокрутки может приводить к путанице. В частности, при нажатии на кнопку со стрелкой вниз содержимое экрана прокручивается вверх. Если же использовать стрелки, повернутые в другую сторону, это создаст только еще большую путаницу. Кроме того, кнопки со стрелками на полосе прокрутки имеют небольшие размеры и поэтому требуют больше времени при использовании, а, как хорошо показывает анализ по закону Фитса, возможность захватывать элементы в любой части документа является намного более быстрым способом.

Вышеописанный пример со специальной функцией захватывания, которой может быть оснащена мышь, показывает, что при разработке интерфейсов программного обеспечения часто могут возникать идеи по улучшению аппаратного оборудования, так же как те или иные характеристики аппаратного строения могут приводить к улучшению

программного обеспечения. Необходимо сказать, что всегда лучше разрабатывать аппаратное и программное обеспечение вместе, несмотря на то, что такая возможность редко случается. Попытки «подставить» чистый программный интерфейс в аппаратное оборудование, которое было разработано для другого интерфейса, редко могут дать удовлетворительные результаты. Тем не менее, в большинстве случаев это именно то, что мы должны сделать.

5.2.3. Экранные состояния объектов

Элементы человекоориентированного интерфейса должны быть доступными для начинающего пользователя и эффективными для опытного пользователя, причем переход от одного к другому не должен требовать переучивания. Хороший интерфейс должен давать одну ментальную модель, которая подходит для обоих классов пользователей, с учетом, конечно, того факта, что по отношению к одним частям системы мы можем быть опытными пользователями, а по отношению к другим – начинающими. В предыдущем разделе было предложено, чтобы клавиша, выполняющая некоторый текст как команду, могла быть применена к выделенному тексту независимо от его происхождения. В результате выполняется определенная команда, при условии, что выделенный текст является именем этой команды, – в противном же случае никакого действия следовать не должно. Хорошо, если бы для ввода команд можно было использовать квазирежим с помощью удерживания клавиши <Command>. Однако это удобство в существенной мере зависело бы от эргономичности клавиши <Command>. В целом, такая возможность улучшила бы использование систем с командной строкой, которые многим нравятся за скорость и удобство работы, но, в то же время, многими ненавидятся за трудность в изучении. Улучшения здесь два: вы можете подавать команды в любом месте и в любое время, а т. к. команды аналогичны представленным в меню, то можно легко переходить от меню к непосредственной подаче команд и обратно.

Ввод команд требует специального места, а также времени на то, чтобы его найти, поэтому удобнее, чтобы пользователь мог ввести команду в любом месте, где находится курсор, и в тот момент, когда это требуется. После выполнения команды введенное имя команды должно быть удалено с тем, чтобы имена команд не оказались разбросанными по всему содержанию. С другой стороны, если вы выполняете команду из списка, то удаление имени команды нежелательно, поскольку список, по сути дела, является меню. Создание такого меню не должно требовать ничего, кроме как напечатать список команд, выделить его и затем применить к нему команду (например, Создать Меню (Make Menu)), чтобы назначить командам особый стиль, который обычно используется для написания команд, а также чтобы заблокировать этот список во избежание его случайного изменения.

Приведем некоторые другие команды, изменяющие состояние текста. С помощью команды Заблокировать (Lock) можно просто запира́ть текст или другое содержание. Заблокированное содержание можно просматривать, выделять и копировать, но нельзя изменять или перемещать. Обратная команда – Разблокировать (Unlock) – может быть применена к выделенному содержанию для его разблокировки при условии, что оно было заблокировано (в противном случае команда не дает никакого результата, т. е. она не должна быть переключателем). С помощью другой команды – Заблокировать с паролем (Lock with Password) – можно заблокировать старую выборку, используя текущую выборку в качестве пароля. Эта команда также имеет обратный аналог (Разблокировать по паролю (Unlock with Password)). Возможность заблокировать какое-то содержание может быть довольно полезной. Например, она может использоваться для создания стандартных форм для заполнения. Неизменные элементы формы могут быть заблокированы, в том числе и по паролю, при этом простое блокирование позволит предотвратить случайные изменения, а блокирование по паролю – несанкционированные изменения. Если бы электронная инструкция по использованию некоторой компьютерной системы входила в набор текстов, с которыми компьютер изначально поставлялся, – что в общем является неплохой идеей, – то такое руководство, вероятно, блокировалось бы производителем по паролю еще на заводе.

Команды Экранная блокировка (Screen Lock) и Экранная разблокировка (Screen Unlock) позволяют заблокировать и разблокировать позиции объектов, выделенных на экране. С помощью этих команд пользователь может создавать меню, которые будут сохранять свое место на экране, в то время как другие объекты перемещаются под ними. Можно даже связывать позицию меню с днем недели (насколько такая возможность является полезной – это уже другой вопрос). Чтобы использовать эту команду, вы размещаете объект в требуемой позиции на экране, выделяете его и применяете команду Экранная блокировка. Также должна быть версия этой команды с использованием пароля для тех случаев, когда нежелательно, чтобы пользователь мог изменить позицию того или иного меню.

Другой полезной командой является регулировка прозрачности выборки. В некоторых ситуациях, таких как, например, отображение сообщений об ошибках, полезно сформировать выборку достаточно прозрачной, чтобы лежащий под ней материал можно было видеть и продолжать с ним работать (рис. 5.2). Аналогичным образом с помощью другой команды можно было бы определять, скрывает ли данный объект другой объект либо скрывается им, либо просматривается через него. Прозрачное окно сообщения может исчезать медленно, а не внезапно, давая пользователю время заметить его. Также необходимо, чтобы в некотором документе сохранялись все сообщения для последующего обзора.

beyond building the most primitive of mechanical engines if careful thinkers had not learned to distinguish between energy, force, work, and power. These words are still used loosely in everyday speech, but professional mechanical designers and physicists use them carefully and with well-defined meanings. Professionals in the field of information-related design will have to learn to be equally careful. We must not allow sloppy thinking to muddy the deep waters we find here at the meeting of art, psychology, and electronic technology. Things are difficult enough as it is.

Example of a transparent error message.

Рис. 5.2. Эффективность прозрачного сообщения об ошибке на фоне текста составляет 1

Для удаления прозрачного диалогового окна не требуется нажатие какой-либо клавиши, так как вы можете продолжить работу через него. Такое окно не создает режимов и является высокоэффективным (эффективность составляет 1). Как и любой другой метод, эта идея имеет свои ограничения и может быть использована чрезмерно. Поток ненужных сообщений все равно отвлекает внимание, даже если пользователь может продолжить работу во время того, как они исчезают. В соответствии с принципом видимости, должно быть предусмотрено визуальное отличие для текста, который является заблокированным, экранно заблокированным, заблокированным по паролю и т. д.

Принцип человекоориентированного интерфейса заключается в том, что система сама должна быть построена из тех же самых элементов, что знакомы вам по повседневному использованию этой системы. Такой подход позволяет создавать более понятные продукты.

5.3. Имена файлов и файловые структуры

Люди скорее готовы страдать до последней возможности, чем защищать свои права через уничтожение тех форм правительства, к которым они привыкли.

*Томас Джефферсон, из Декларации независимости
Соединенных Штатов Америки*

Для многих пользователей максимально возможная в Macintosh длина имени файла в 31 символ была просто счастьем в сравнении с ужасным ограничением в 8 символов в более ранних системах. Тем не менее, даже такая модификация являлась всего лишь смиренной рубашкой большего размера. За исключением некоторых реальных ограничений, налагаемых аппаратным оборудованием, интерфейс не должен иметь ограничений, связанных с длиной. Интерфейс должен использовать распределение динамической памяти, ссылки, хеширование или любые другие методы, но никогда не должен ставить для

пользователя ограничений, таких как «вы можете использовать не более 255 категорий» или «объем абзаца не должен быть более 32 000 символов».¹

Чем является имя файла? С точки зрения пользователя, это «ручка», за которую файл можно взять. По своему опыту мы знаем, что имена файлов работают не так, как этого стоило бы ожидать, – они мешают, когда нужно что-то сохранить, и бывают бесполезными, когда нужно что-то найти. Если говорить более конкретно, файловые имена становятся назойливыми, когда вы собираетесь что-то сохранить, так как вам приходится останавливаться в середине самого процесса сохранения, чтобы придумать имя для файла.² Создание имен – это трудное занятие. Требуется на месте, за пару мгновений выдумать уникальное, запоминающееся имя, которое, к тому же, должно отвечать правилам, принятым в данной файловой системе. Кроме того, в этот момент в локусе вашего внимания находится не проблема создания имени файла, а вопрос сохранения вашей работы. Файловые имена приводят к неудобствам и при поиске файлов. Имя, которое вы придумали, может быть не особенно запоминающимся, и поэтому вы, вероятно, можете забыть его через несколько недель (или даже раньше). Я, например, редко когда могу вспомнить имя файла, если только я не пользовался им совсем недавно, и даже простое просматривание списка файлов вызывает смятение. Ну, скажите, что может содержаться в файле с именем «notes ybn 32»? Когда я придумывал его, оно казалось таким умным и запоминающимся. Кроме того, многие файлы очень похожи друг на друга. Сколько разных, оригинальных и запоминающихся имен вы можете выдумать для писем к своему бухгалтеру по поводу оплаты налогов за прошлый год? Их, конечно, можно организовать по дате, но многие ли из нас способны помнить, что, например, письмо о списании служебного грузовика было написано 14 августа?

Необходимость давать имена файлам увеличивает ментальную нагрузку на пользователя. Назначение имени не делает ничего, кроме добавления к самому файлу еще нескольких символов. И вам приходится запоминать этот файл только лишь по тому кусочку символов и больше ни по чему другому. Я считаю это одним из главных бедствий, которым страдают современные компьютерные системы. Этот метод используется также и во многих других информационных средствах.

¹ Совсем недавно мне случилось попользоваться текстовым процессором с таким ограничением на размер абзаца. Я превысил это ограничение уже как только вставил в абзац фотографию. Когда я сказал об этом разработчикам, они признали, что никогда не думали об этом. Таким образом, никогда не следует ставить ограничения только потому, что так писать программу легче. Такие ограничения всегда будут слишком узкими.

² Все это касается современных систем. В будущих, человекоориентированных системах пользователю никогда не придется явным образом делать сохранение информации.

Между именем файла и самим файлом не должно быть различия. Человеческий мозг способен более эффективным образом использовать быструю, полнотекстовую поисковую систему, поэтому любое слово или фраза из файла может служить ключом к нему. (Более того, желательно, чтобы по запросу «письмо о стрекозе» выполнялся поиск чего-либо в форме письма, а также не только слова *стрекоза*, но и связанных с ним терминов и выражений в случаях, если упоминались научные названия этого вида (например, *Odonata*). Если такие письма обнаружены не были, поиск должен продолжиться среди документов, не являющихся письмами, и так далее, расширяясь до сетевых ресурсов и Интернета.) Вы не можете помнить содержание документа по его имени «Письмо Джиму от 21/12/92», но вы можете помнить, что когда-то вы писали Джиму по поводу одной голубой Edsel, которая однажды пролетала мимо вас. Поиск по слову *Edsel* найдет только один или два элемента во всей вашей системе, если только вы не интересуетесь именно видом Edsel – в этом случае вы, вероятно, выбрали бы другую схему поиска. Неограниченное по длине имя файла является самим файлом. *Содержание текстового файла и есть его самое лучшее имя.*

Графические и звуковые файлы зачастую тоже требуют своих имен. В разделе 6.2 будет рассмотрен подход, в котором обходятся те трудности, которые накладывают традиционные файловые структуры на нетекстовые файлы. Если не говорить о нетекстовых файлах, наличие быстрой полнотекстовой системы поиска позволяет отказаться от использования целого вида бесполезных элементов – файловых имен. Вместе с отказом от имен файлов также отпадает необходимость в механизмах их обработки (например, каталогах), правилах их создания и синтаксических ограничениях при создании. С устранением файловых имен исчезает большая ментальная нагрузка на пользователя, а также значительная часть внутреннего программного обеспечения – того, что в настоящее время пользователям приходится изучать, а программистам разрабатывать.

Наилучшей моделью интерфейса для полнотекстовой системы поиска является интерактивная, в которой вы можете видеть каждый найденный элемент в его контексте. В таком интерфейсе вы сразу оказываетесь там, где вам необходимо, как только вы это увидели. В некоторых системах делаются копии найденных элементов в том ряду, в котором они содержатся (Drori, 1998). Однако в сравнении с первым методом поиска этот метод не является эффективным, поскольку вам придется затем выполнить еще вторую операцию по получению самого элемента (например, вам придется дополнительно щелкнуть по копии необходимого документа).

Для тех пользователей, которые хотят иметь систему, напоминающую обычную файловую структуру, может быть предусмотрена специальная команда по созданию «информационного документа» или дополнительной страницы в конце каждого документа, когда документ

выбран и такая команда применена. Информационный документ или дополнительная страница может содержать информацию о дате и времени создания или изменения документа, историю редакций, размер и другую полезную информацию. Программное обеспечение, предназначенное для выполнения такой команды, должно получать и хранить необходимую информацию невидимым для пользователя образом.

Разработчики могут поставлять разные средства в зависимости от потребностей пользователя. Для пользователей, предпочитающих старые методы, можно предусмотреть утилиты для создания документов, которые могли бы выглядеть и действовать так же, как и те неудобные каталоги, которыми мы сейчас пользуемся.

Другой тип организации, который легче изучить и использовать, чем обычные файловые системы, происходит из естественной иерархичности, присущей многим естественным языкам. В них слова разделяются пробелами, предложения (или последовательности слов) разделяются одним из небольшого числа разделителей (в английском языке в число этих разделителей входит точка, вопросительный и восклицательный знаки) и последующим пробелом. Абзацы (или последовательности предложений) разделяются, по крайней мере, одним знаком возврата каретки. Символ раздела или разрыва страниц служит для отделения глав или других уровней организации текста.

В последовательной системе разрыв страницы должен быть символом и, в отличие от большинства современных систем, он должен вести себя как любые другие символы с точки зрения их вставки, удаления и поиска. Как и символ возврата каретки, символы разрыва страниц могут быть скрытыми символами, обозначающими фиксированную длину физической страницы, но они не должны быть частью содержания.¹

Имеет смысл не ограничиваться этими иерархическими уровнями, как это делается во многих современных системах. Документы являются последовательностями страниц, разделенных символами этого документа, каждый из которых может быть набран, найден или удален так же, как и любой другой символ. Также могут быть разделители и более высокого уровня, как, например, символ папки и тома или даже разделитель секции и библиотеки. Тем не менее, число уровней определяется объемом данных. Набор из двух последовательных символов документа является хорошим разделителем для наборов документов. Если требуются другие уровни организации, то три или четыре последовательных символа документа можно использовать в качестве разделителей. Вместо клавиш <Папка> (Folder), <Том> (Volu-

¹ Доктор Джеймс Уинтер (James Winter) из Information Appliance продолжил такого рода унификацию структуры, указав на то, что в английском языке уже имеется аналогичная иерархическая структура с символами-разделителями, которая используется для отражения высших уровней организации текста.

me) и <Библиотека> (Library), которые будут использоваться редко, можно предусмотреть клавишу <Документ> (Document), которую можно нажимать многократно, вплоть до четырех раз соответственно. Такое решение позволит избежать резкого увеличения количества клавиш на клавиатуре. Важно, чтобы все символы разделителей имели специальную клавишу, иначе они не будут работать как все другие набираемые символы. То есть пользователь не должен нажимать на какую-то клавишу (например, <Return>) и потом применять команду Вставить разрыв страницы. Для этого он должен использовать только специальный символ страницы.

Если разнообразные символы разделения работают так же, как и все другие символы, исчезает необходимость в обучении пользователей тому, как с ними управляться. Если пользователь все же настаивает на использовании явных имен документов, он может ставить такие имена сразу после символа документа. Для того чтобы найти документ с именем «Собаки Азии», поиск должен будет производиться по фразе «Собаки Азии», перед которой стоит символ документа. При таком поиске будут игнорироваться все случаи использования последовательности «Собаки Азии», кроме тех, которые используются в качестве имени документа. Для получения каталога может использоваться команда, которая сможет собрать документы, включающие искомую подстроку и предваряющиеся символом документа, а также другими символами, вплоть до и включая ближайший символ конца абзаца или разделитель более высокого уровня.

Отказ от использования иерархической файловой структуры не означает отказ от структурирования сохраняемой информации. Ничто не мешает вам создавать оглавления и указатели или размещать все ваши письма к дяде Альберту и тете Агате на соответствующих страницах. Ничто не мешает вам создавать сразу перед письмами титульную страницу (как отдельный документ) «Письма к дяде Альберту и тете Агате». В этом случае вы, в сущности, создадите файловое имя, но оно не потребует использования специальных механизмов в программном обеспечении. По желанию, если вы действительно любите файловые имена и иерархии, вы можете создать свою собственную иерархическую файловую структуру. Однако тогда, структура будет частью вашего *содержания*, а не *интерфейса*.¹ Поэтому вместо специального механизма поиска файлов вы сможете использовать обычные средства поиска. Перед группой файлов вы можете поместить в виде документа название папки. Также перед группой папок вы можете поместить название тома. (В этом случае за символом тома будет следовать название тома, чтобы избежать совпадений с другими такими названиями, не являющимися именами тома.) Отсутствие встроенной файловой

¹ То же самое касается и файловых имен и меню, которые были предложены в этой книге.

организации не мешает созданию файла, предназначенного для какой-либо цели, но, в то же время, понятного *вам*, т. к. он создан именно вами. Тем не менее, поскольку система при этом никак не изменяется, другой пользователь может искать что-либо в ваших данных и фактически проигнорировать вашу структуру и рассматривать ее как сплошной, неструктурированный файл.

Преимуществом организации произвольной файловой структуры является то, что такая структура не устанавливается разработчиками системы, у которых могут быть совсем другие идеи, отличные от ваших. В этом случае вам не приходится создавать ментальную модель того, что разработчики пытались сделать. У многих пользователей формируются неправильные модели того, как та или иная система работает. Ментальные модели трудно изменяются и поэтому они могут долгое время создавать трудности для таких пользователей (Norman, 1988).

Сказанное не является теорией. Отказ от использования в компьютерах SwyftWare и Canon Cat файловых имен, директорий, а также от механизмов, с помощью которых с ними производятся манипуляции, оказался одним из самых удачных решений. Пользователи, привыкшие к использованию обычных компьютерных систем, признавали, что им было трудно перейти к использованию систем, основанных на принципе организации информации по содержанию. Но когда такой переход происходил, обычные методы вскоре казались им неудобными. А пользователи, которые начинали знакомиться с компьютерной техникой с компьютера Canon Cat, не испытывали радости от необходимости изучать более сложные методы использования обычных файловых систем, когда они переходили на PC или Macintosh.

В сравнении с обычными графическими пользовательскими интерфейсами методы, описываемые в этой книге, могут показаться сложными. Но кажущаяся сложность возникает только лишь из-за того, что новый подход еще не знаком, а также из-за того, что мы привыкли к тому множеству действий, которое мы должны выполнить, и к тем проблемам, которые возникают при использовании существующих сегодня систем. Конечно, новая система потребует от пользователя усилий на изучение. Но если вы сравните успехи начинающих пользователей, работающих с разными системами, или сравните эффективность работы опытных пользователей, преимущества более простой системы станут очевидными.

Представьте, что у вас есть n документов, которые вы должны скопировать на внешний носитель – пусть это будет жесткий диск. Для этого, используя, например, операционную систему Macintosh, вы должны будете перетаскать пиктограмму каждого документа на пиктограмму диска. Новая парадигма, на первый взгляд, делает операцию более сложной. Вам нужно найти начало и конец каждого документа, выбрать документ, переместить курсор к нужному месту на диске и затем переместить каждый документ.

Вспомните, что в обычном графическом пользовательском интерфейсе вы всегда начинаете с открытия некоторого общего приложения. Ваш первый шаг – это добраться до рабочего стола. Вы должны помнить, какие пиктограммы соответствуют нужным документам, и вам (или кому-то другому) всегда приходится выполнять действия по назначению имен этим документам. Вы также должны помнить, в какой папке эти документы находятся. Поэтому кажущаяся простота достигается только после того, как выполнен значительный объем работы, и пользователь разделался с большим количеством обременительных ментальных требований. Более эффективным методом является изобретение под названием LEAP. Предположим, что, как и в случае для обычного графического интерфейса, курсор находится в одном из документов, которые вы хотите переместить. С помощью LEAP пользователь может выбрать документ шестью нажатиями клавиш, причем не глядя на экран и не держа в голове имя документа. Нажатие 6 клавиш отнимает меньше времени, чем перетаскивание пиктограммы.

Функция LEAP работает следующим образом: имеется две клавиши <LEAP>, находящиеся под большими пальцами. <LEAP-Up> производит поиск вперед от позиции курсора, а <LEAP-Down> – назад. При нажатии на клавишу <LEAP> включается квазирежим, после чего все, что вы ввели, воспринимается как шаблон поиска. Для выборки документа вы должны нажать <LEAP-Up> и ввести символ документа (LEAP-Up↓ Doc↑↑↑). Мы перемещаемся к началу документа. Затем мы нажимаем <LEAP-Down> и вводим символ документа (поиск в этом случае найдет символ документа в конце документа). Одновременное нажатие на обе клавиши <LEAP> позволяет выделить весь текст. (Вероятно, удобнее всего это делать с помощью больших пальцев, которые редко используются при работе с клавиатурой. См. рис. 2.1 с изображением обычной клавиатуры, снабженной клавишами <LEAP>.) Альтернативным вариантом является клавиша <Выбрать> (Select.) Для того чтобы сделать эту функцию видимой, рядом с клавишами <LEAP> необходимо поместить какое-то обозначение. Например, можно использовать надпись: «Для создания выделения нажмите одновременно на обе клавиши LEAP». Обратите внимание, что пользователю не требуется смотреть на экран при выборе документа. Когда документ выбран, нажатием клавиши <LEAP> курсор перескакивает на тот объект, куда нам необходимо вставить текст, после чего дается команда Копировать (Copy). При таком выделении документ включает свои разделители. Таким образом, если документ перемещается, он сохраняет в себе свою сущность как документ, поскольку эти разделители перемещаются вместе с ним.

Та же самая техника, которая использовалась для копирования документов (или другой выборки любой длины, начиная от единственного символа и заканчивая набором документов или даже всем содержанием системы (!)) из одного места в другое, может быть использована и для перемещения выборки. Разница состоит только в том, что

подается команда Переместить (Move) вместо команды Копировать (Сору). В функциональном отношении этот процесс является не более сложным, чем аналогичные процессы в обычных пользовательских графических интерфейсах. Зачастую он оказывается более быстрым, а число методов, понятий и структур, которые должен знать пользователь, оказывается меньшим.

Сравним перемещение нескольких выборок из разных документов на диск с помощью клавиши <LEAP> с выполнением этой же задачи в обычном пользовательском графическом интерфейсе. В первом случае метод будет аналогичным тому, который применялся для перемещения документов на диск. Объекты, предназначенные для выборки, сначала *находятся* (при этом необходимости их открывать нет), потом они *выбираются*, как это было описано выше (за исключением того, что в начале и в конце выборки используются символы не документа, а текста), и затем они *копируются* в нужное место. В обычном пользовательском графическом интерфейсе пользователь должен сначала *открыть* документ, в который будет производиться копирование (возможно, это потребует использования команды Создать (New) из меню Файл (File)), потом *найти* документ, который содержит нужную выборку, *открыть* этот документ, *найти* выборку внутри документа, *выделить* ее, *применить* команду Копировать (Сору), *сделать активным* документ назначения, *вставить* выборку, *активировать* рабочий стол, *найти* следующий документ, содержащий нужную выборку, *повторить* всю процедуру до тех пор, пока все выборки не будут вставлены в документ назначения. После этого вы должны *сохранить* результат на диске, используя специальное диалоговое окно.

Даже если бы уровень сложности выполнения задач был одинаковым для обоих подходов, концептуальная простота методов, описанных в этой книге, делала бы их более предпочтительными. В большинстве же случаев объем необходимой работы также оказывается намного меньшим, чем при использовании обычных интерфейсов.

5.4. Поиск строк и механизмы поиска

Маленький шаг большого человечества.

Нейл Армстронг (1969)

Прежде чем продолжить рассмотрение функции LEAP, имеет смысл рассмотреть несколько более подробно вопрос использования интерфейсов для поиска. **Строкой** (string) называется последовательность¹ символов. Обычные английские слова и предложения являются при-

¹ Здесь я использую термин **последовательность** в математическом смысле, чтобы обозначить множество объектов, включающее первый объект, второй и т. д.

мерами строк. При **поиске по строке** (string search) происходит просмотр (обычно длинной) последовательности, называемой **текстом**, с целью обнаружения (обычно короткой) последовательности, указанной пользователем и называемой **шаблоном** (pattern). Каждый случай совпадения между подстрокой текста и заданной комбинацией называется **объектом поиска** (target). Например, при попытке найти в большом письме место, где вы писали о кошке по кличке «маленькая Татсу», наиболее подходящим объектом поиска будет *маленькая Татсу*, а еще более короткая строка *Tatсу* будет хорошим шаблоном. Совпадение может быть полным, может зависеть от регистра символов или от других параметров (например, соответствие может быть по рифме). Наиболее распространенный критерий поиска, по которому легко искать, заключается в том, что строчные буквы в задаваемой комбинации соответствуют как строчным, так и прописным буквам в тексте, тогда как прописные буквы в задаваемой комбинации соответствуют только прописным буквам в тексте. Обычно поиск начинается от текущей позиции курсора и осуществляется вперед. В большинстве систем с помощью модальных пользовательских установок можно произвести поиск в обратном направлении (рис. 5.3).

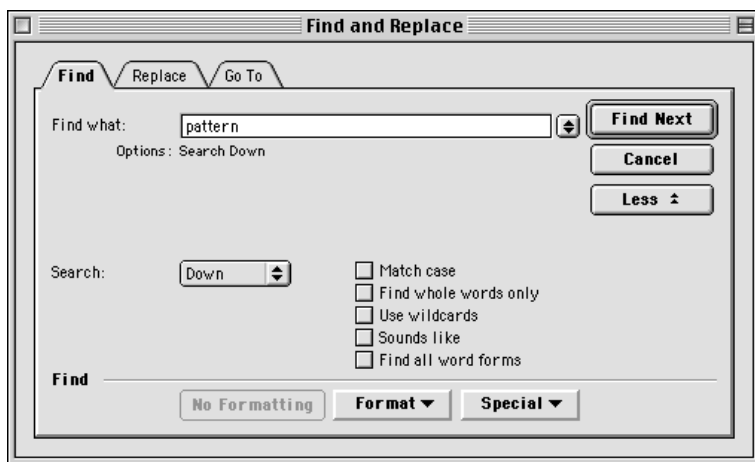


Рис. 5.3. Модальное диалоговое окно с модальными настройками типа и направления поиска

Интерфейсы к средствам поиска обычно строятся на основе двух подходов. Наиболее распространенным является **поиск с разделителями** (delimited search), который встречается в большинстве текстовых процессоров. В типичном поиске с разделителями пользователь включает режим, в котором любой введенный текст рассматривается как шаблон для поиска. Обычно для этого используется диалоговое окно, снабженное полем для ввода символов. После вызова диалогового окна пользователь вводит комбинацию символов и разделитель, в качестве которого обычно используется некий символ, запрещенный к отобра-

жению в шаблоне (например, Return). В большинстве диалоговых окон пользователь также может ограничить шаблон нажатием на кнопку OK, Search, Find или Find Next с помощью ГУВ. Когда объект поиска обнаружен, он выбирается, а курсор располагается сразу в конце выборки.

Этот традиционный метод является довольно злосчастным для пользователя, хотя большинство компьютерщиков настолько привыкли к этому, что уже не замечают никакого неудобства. Например, пользователь может ввести последовательность для поиска с опечаткой, но заметить ее слишком поздно, т. к. он уже нажал по привычке клавишу <Return>. Поэтому ему придется сидеть и ждать, пока закончится поиск, который, как уже заранее известно, не даст результата. Большинство систем поиска являются непрерываемыми, и это является серьезной ошибкой разработчиков. Из-за того, что компьютер ждет, пока пользователь закончит ввод шаблона, по которому начнется поиск, поиск с разделителями часто вынуждает пользователя ждать без необходимости.

Менее распространенным методом является **пошаговый поиск** (incremental search), известный пример которого можно увидеть в текстовом редакторе EMACS, работающем под операционной системой UNIX (Stallman, 1993). В большинстве случаев использования пошагового поиска, так же как и при поиске с разделителями, пользователь должен сначала вызвать диалоговое окно, в котором имеется поле для ввода шаблона поиска. Когда пользователь вводит первый символ шаблона, система использует этот символ как полный шаблон и сразу же начинает поиск первого экземпляра этого символа в выбранном направлении. Если экземпляр этого первого символа обнаруживается до того, как введен следующий символ шаблона, то он выбирается, а курсор помещается сразу в конце выборки. Если же следующий символ шаблона вводится до того, как экземпляр обнаруживается, то этот символ добавляется к шаблону и поиск продолжается теперь уже в отношении экземпляра расширенного шаблона. Процесс повторяется по мере добавления символов к шаблону поиска.

При использовании клавиши <Backspace> или <Delete> для удаления символов из шаблона пошагового поиска поиск возвращается к предыдущему экземпляру, найденному по тому шаблону, который был до добавления к нему следующего символа. Пользователь может затем добавить символы к шаблону, чтобы продолжить поиск без сброса результатов уже выполненного поиска по неполному шаблону. Многие системы поиска не имеют этой полезной характеристики.

Пошаговый поиск имеет ряд других преимуществ в сравнении с поиском с разделителями. Пошаговый поиск требует меньше времени. Поиск начинается, как только первый символ шаблона введен. Система не ожидает того момента, когда шаблон будет введен полностью. При использовании поиска с разделителями компьютер ждет, пока пользо-

ватель полностью введет шаблон и обозначит его разделителем, после чего уже пользователь ждет, пока компьютер производит поиск. При использовании поиска с разделителями пользователь должен сначала предположить, по какому шаблону компьютер сможет отличить нужный объект от других подобных объектов, тогда как при использовании пошагового поиска пользователь сразу же может определить, что шаблон оказался достаточным, чтобы выявить нужный объект, потому что он уже появился на экране. Таким образом, как только пользователь видит, что нужная точка найдена, он может прекратить введение шаблона. Если же он введет слишком много символов, т. е. скорость ввода будет больше, чем скорость поиска, шаблон все равно будет введен, и курсор установится приблизительно в том месте, которое предполагается. Если пользователь ошибается при вводе шаблона в систему поиска с разделителями, для исправления ошибки он должен ждать до тех пор, пока поиск по неверному шаблону не завершится, – в лучшем случае пользователь может воспользоваться механизмом для остановки поиска, если такой предусмотрен. В большом тексте поиск может занимать значительный период времени. В хорошо разработанном пошаговом поиске пользователь может удалить неверно введенный символ в любое время и возвратиться к последнему найденному экземпляру. Поскольку использование клавиши Backspace для исправления ошибки может быть привычным, процесс исправления проходит довольно быстро, и поиск останавливается сразу же. Чтобы возобновить поиск, пользователь может ввести правильный символ.

Еще одним преимуществом пошагового поиска является то, что в нем имеется постоянная обратная связь во время введения символов шаблона, т. е. результаты поиска видны сразу. При использовании поиска с разделителями пользователь не знает, насколько введенный им шаблон является подходящим или даже насколько правильно он был набран, до тех пор пока ввод не закончен и попытка поиска не начата. С точки зрения построения интерфейса, пошаговый поиск имеет так много преимуществ, а поиск с разделителями – так мало, что, на мой взгляд, использование поиска с разделителями редко когда может быть предпочтительным.¹ Несмотря на то что почти все разработчики и пользователи признают, что пошаговый поиск более предпочтителен, почти все инструменты по разработке интерфейсов позволяют создавать средства поиска с разделителями и затрудняют или даже делают невозможным создание средства пошагового поиска. Примерами таких инструментов являются JavaScript и Visual Basic.

Пошаговый ввод шаблона поиска делает возможным изменять шаблон интерактивно прямо во время процесса поиска, а значит, позволяет пользователю оптимизировать поиск по получаемой обратной связи.

¹ Поиск может быть либо с приращением (пошаговый (incremental) поиск), либо с убавлением (excremental).

Даже построение булевой модели поиска делается более эффективным, если результаты поиска отображаются по мере того, как пользователь уточняет шаблон. Найденный экземпляр должен появляться посередине экрана, а не наверху или внизу, так, чтобы материал до и после экземпляра был виден, таким образом, найденный экземпляр всегда отображается в своем контексте. Найденный экземпляр также должен всегда отображаться в одном и том же месте относительно экрана или окна, чтобы пользователь быстро выучил, где искать результаты. В компьютере Canon Cat найденный экземпляр всегда появлялся по вертикальному центру экрана. Он не должен отображаться на каком-то из краев экрана; таким образом, материал вокруг экземпляра всегда был виден.

Если в тексте не содержится экземпляра шаблона, поиск оказывается неудачным. Во многих системах в этом случае поиск прекращается и не может использоваться до тех пор, пока не нажата специальная клавиша (обычно <Enter> или <Return>) или кнопка на экране. На экране появляется модальное сообщение, в котором говорится, что вы должны сделать необходимый поклон, прежде чем вам будет позволено продолжить пользование компьютером. В многоэкранных системах или в случаях, когда экран визуальнo перегружен, такое сообщение может совсем не попасть в локус вашего внимания, и вы можете совсем его не заметить. В результате вам может показаться, что компьютер не отвечает на нажатие клавиш, как будто бы он завис. При использовании же пошагового поиска вы без всякой специальной подсказки сможете заметить, что поиск не удался, потому что курсор в этом случае сразу же возвращается в начальную позицию, и дополнительные нажатия не дают никакого результата. Здесь же может быть полезным короткий звуковой сигнал или мигание на экране, особенно если поиск длился дольше периода действия кратковременной памяти (скажем, дольше 10 секунд), и поэтому пользователь забыл, как выглядел дисплей до начала поиска. Звуковой сигнал еще полезен для пользователей с ухудшенной зрительной способностью.

5.4.1. Разделители в шаблоне поиска

Другим большим недостатком поиска с разделителями является то, что разделитель, используемый для обозначения конца шаблона, не может быть отображен. Во многих случаях и другие разделители тоже не могут отображаться. Я просмотрел четыре популярных текстовых процессора. В одном из них использование Return вообще не допускалось. Во втором текстовом процессоре для того, чтобы вставить Return в шаблон поиска, пользователь должен набрать $\uparrow r$. В третьем текстовом процессоре в этих целях использовалась последовательность $\backslash \backslash$. В четвертом – Return можно было вставить в шаблон поиска с помощью специального диалогового окна с выпадающим меню, содержащим разные разделители (рис. 5.4). Однако было бы легче просто использовать клавишу <Return>. В конце концов, именно таким обра-

зом вы вставляете этот разделитель в текст. Почему же при создании шаблона поиска должен использоваться другой метод? Основной принцип состоит в следующем: *одна и та же последовательность символов должна набираться одинаковым образом. Пользователь не должен в одном случае применять один метод, а в другом – другой. Иными словами, в отношении специальных символов не должно применяться ничего специального.*¹

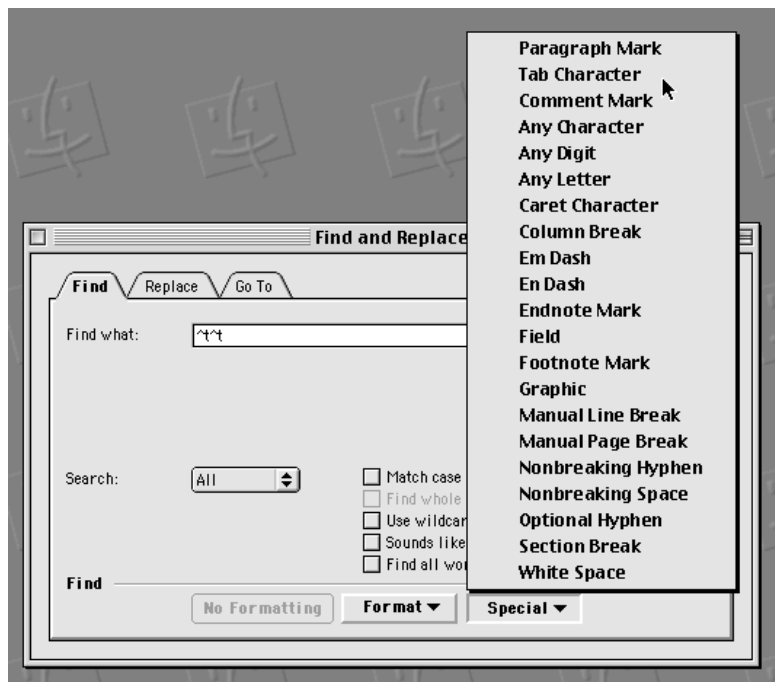


Рис. 5.4. Открытое окно поиска в Word. Показан список символов, которые могут быть использованы. В улучшенном варианте пользователь мог бы помещать, например, символ табуляции с помощью нажатия клавиши <Tab>. Обратите внимание на два обозначения символа табуляции, которые вставлены в поле поиска (Find what)

Хотя пошаговый поиск лучше, чем поиск с разделителями, вариант пошагового поиска, который используется в EMACS, может быть все же улучшен. Например, курсор должен возникать не на последнем символе объекта поиска, а на первом. В общем, вы вряд ли можете управлять тем, каким будет последний символ в шаблоне поиска, поскольку вы вводите только те символы, которых достаточно для поиска нужного объекта. Поэтому вы не знаете точно, где окажется курсор

¹ Аналогичной проблемой является использование зарезервированных слов в языках программирования.

после того, как поиск будет произведен. Если же курсор всегда будет устанавливаться на первом символе шаблона, вы можете знать, как отобразится объект поиска. Кроме того, это может быть полезным для быстрого перемещения курсора внутри текста, поскольку в локусе вашего внимания находится символ, на который вы хотите переместить курсор. Придумать шаблон, в котором этот символ является последним, намного труднее, чем просто ввести требуемый символ и последующие за ним другие символы.

В обычных пользовательских графических интерфейсах как пошаговый поиск, так и поиск с разделителями запускается модально, с помощью диалогового окна, тогда как использование клавиши <LEAP> является безмодальным. Идею использования для поиска квазирежима можно дополнить применением специальной кнопки на микрофоне (или ГУВ), удерживаемой для включения квазирежима, в котором вводимые слова, рисунки или рукописный текст могут включаться в шаблон поиска. Другие методы ввода имеют аналогичные средства для создания квазирежима поиска (Raskin и Winter, 1991).

Скорость пошагового поиска может быть увеличена с помощью некоторых приемов. Например, при вводе первого символа искомой строки компьютер сразу приступает к поиску первого экземпляра этого символа в тексте, после чего найденный экземпляр подсвечивается и переносится вместе со своим контекстом в окно экрана. Обычно этот процесс проходит быстро, поскольку в тексте имеется много потенциальных экземпляров и какой-то из них, с большой вероятностью, оказывается поблизости. Пока пользователь вводит следующий символ, поиск может продолжиться в отношении второго экземпляра первого символа и последующего возможного символа в порядке уменьшения частоты использования. Программа может сохранять ссылки на каждый из обнаруженных экземпляров. Как результат, при вводе второго символа компьютер может быть готов отобразить обнаруженный объект, создавая эффект мгновенного поиска.

Поиск строк также может быть ускорен и такими методами, как алгоритм Бойера-Муура (Moore и Boyer, 1977), в котором время поиска уменьшается по мере увеличения длины последовательности. Если пользователь возвращается в шаблоне поиска на одну позицию назад, сохранение ссылки на последнее обнаруженное место (причем для каждого символа последовательности) сделает возвращение чрезвычайно быстрым. Индексирование всех запоминающих устройств может сократить время поиска в локальных системах и сетях до миллисекунд. Скорость взаимодействия через глобальные сети или Интернет также зависит от применяемых методов индексирования. Пошаговый поиск в различных вариантах использовался в таких коммерческих продуктах, как IDE компании Borland, факсовая программа компании Global Village, компьютеры Canon Cat и SwiftWare.

5.4.2. Единицы взаимодействия

Пошаговый поиск является одним из примеров использования общего принципа разработки человекоориентированных интерфейсов, который заключается в следующем: *программа должна взаимодействовать с пользователем на основе наименьшей значимой единицы ввода*. Взаимодействие с данными, вводимыми с помощью клавиатуры, должно быть познаковым, а не построчным. Взаимодействие с данными, вводимыми с помощью голосовых устройств, должно быть пословным, а для некоторых приложений даже может быть поморфемным и т. д.

Взаимодействие посредством последовательного ввода строки текста, т. е. строки, отделенной знаком Return, – это пережиток времен телеайпа, который должен быть передан в музей вместе с оборудованием того времени.¹ Сегодня мы можем и должны иметь возможность пользоваться интерфейсами, способными реагировать на каждый вводимый символ во всех случаях, когда такая реакция может улучшить качество взаимодействия. Как и всегда, разработчики должны быть внимательны – например, посимвольное взаимодействие не должно приводить к выдаче сообщений об орфографических ошибках в середине набора слова, что может усложнить работу пользователя, владеющего слепым методом набора.

В маленьких текстах, для которых поиск строк был изначально придуман, он обычно проходил от текущей позиции курсора до конца текста. В текстах большего размера в случае, если до конца документа искомый объект так и не был обнаружен, обычно удобнее продолжить поиск с начала документа до позиции курсора, если пользователь забыл, что искомый объект находится выше. Неопубликованное тестирование, проведенное в компании Information Appliance, показало, что если поиск производится быстро, то циклический поиск оказывается особенно удобным. «Быстрый» означает здесь, что между запуском поиска и его успешным либо неуспешным окончанием не остается времени на действия пользователя. Как правило, это время составляет порядка 250 мс. Во многих системах пользователь может выбрать, каким образом будет проходить поиск – либо циклично (по кругу), либо с остановкой в конце документа. Это порождает типичную проблему модальности. Если установлен нециклический поиск, и пользователь не знает об этом, сообщение «строка не обнаружена» может привести к неверному пониманию, что в тексте нет экземпляров по запрошенному шаблону поиска. Часто я наблюдал, что в этом случае пользователи несколько раз запускали поиск повторно, т. к. ясно помнили, что видели такой экземпляр в тексте, и не понимали, почему поиск заканчивается безрезультатно. Может пройти несколько секунд или даже ми-

¹ Наверное, когда-то будет создан музей форм взаимодействия, аналогичный музеям старого программного и аппаратного оборудования. Такой музей мог бы быть создан в Интернете.

нут, прежде чем пользователь догадается, в чем состоит проблема, или же так и останется в недоумении. Если необходимо иметь разные виды поиска, можно избежать использования режимов, предусмотрев для каждого вида поиска соответствующую команду или экранную кнопку для запуска.

Во многих случаях модальности можно избежать с помощью набора кнопок для запуска каждого вида поиска. Такой набор может быть предусмотрен вместо окна установки параметров поиска, снабженного одной-единственной кнопкой запуска поиска по заданным условиям. Такой подход позволяет не только устранить модальность, но и сократить число нажатий. Кроме того, в локусе внимания пользователя в этом случае находится задача, которую он хочет выполнить, а не настройки для ее выполнения. На рис. 5.5 показано типичное диалоговое окно с настройками и кнопкой запуска, которое используется в Microsoft Word. Из рисунка видно, что с диалоговыми окнами этого типа связана и другая проблема: должны ли кнопки переключателя всегда находиться в указанном положении при открытии окна? Должны ли они быть в положении, в котором пользователь оставил их при последнем использовании? Или же они должны быть в некотором положении, установленном пользователем по умолчанию?



Рис. 5.5. Диалоговое окно, снабженное кнопками для соответствующего типа поиска и одной кнопкой запуска

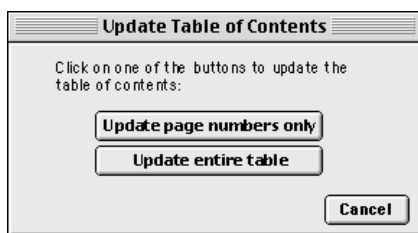


Рис. 5.6. Более эффективное диалоговое окно с несколькими кнопками запуска

Все три варианта являются ошибочными. Если пользователь всегда обновляет оглавление полностью, то при первом варианте ему придется делать каждый раз по два щелчка мышью (или один щелчок и одно нажатие на кнопку <Return>). Если переключатель остается в том положении, в котором окно использовалось последний раз, пользователь не сможет пользоваться этим окном привычным образом, поскольку каждый раз ему придется останавливаться и проверять, в каком положении переключатель находится. Если же переключатель может устанавливаться пользователем в положение по умолчанию (см. раздел 3.2.2), то тем самым включается режим.

Диалоговое окно, изображенное на рис. 5.6, позволяет решить сразу все эти проблемы. Кроме того, по закону Фитса кнопки большого размера имеют преимущество по сравнению с набором переключателей.

Если такое окно сделать прозрачным, то можно также отказаться от использования кнопки <Cancel> (см. раздел 5.2.3). В этом случае используемые две кнопки не должны быть прозрачными, чтобы пользователь мог видеть, что они являются активными.

Диалоговые окна для поиска с разделителями обычно снабжены устройством для сохранения текущего шаблона сразу после обнаружения последнего экземпляра. Такое устройство можно назвать «искать еще» или «найти далее». В некоторых вариантах оно запускается с помощью той же кнопки, которая использовалась для начального поиска. Если поиск является пошаговым, то команда для повторного поиска того же объекта является необходимой, поскольку в этом варианте нет никакой команды для запуска начального поиска. Применять для повторного поиска клавишу включения квазирежима <Search> не желательно, т. к. пользователь может нажать эту клавишу, а затем передумать и отпустить ее. В этом случае будет начат поиск, который пользователь не хотел запускать. В результате пользователь может потерять место в тексте, в котором он находился. Хотя использование системы глобального отката может исправить ситуацию, все же лучше вообще не создавать этой проблемы. Специальный метод для выполнения повторного поиска будет рассмотрен в разделе 5.6.

*В больших по размеру текстах поиск может осуществляться по кругу (циклично) не только в локальном документе, но и далее, в автоматически расширяющихся областях вплоть до всего Интернета.*¹ После того как поиск был выполнен по всему локальному документу, он может быть продолжен в отношении последующих документов папки, а затем перейти на начало первого документа папки и продолжиться до текущего, уже просмотренного документа. После циклического поиска внутри папки рассматривается следующая локальная область поиска и т. д. Если во время пошагового поиска, выполняемого таким образом, пользователь поймет, что процесс поиска слишком расширился, он может остановить его, убедившись, что в ближних областях искомого экземпляра нет. Текущую область поиска определить обычно легко, т. к. пользователь может видеть результаты поиска в своих контекстах, а не просто список файлов, как это делается во многих современных поисковых системах.

В общем случае пользователи будут применять более эффективные стратегии поиска, чем просто полагаться на такой метод автоматически расширяющегося поиска. Например, если вы ищете какой-то документ в текущей папке, то, скорее всего, вы станете искать символы документов, чтобы быстро просмотреть начало или заголовок каждого документа. Если нужный документ обнаружен, запускается пошаговый поиск в отношении целевого объекта. Таким образом, обеспечива-

¹ Для того чтобы среднее время ответа не превысило 250 мс, можно использовать такие методы, как предпросмотр и индексирование.

ется порядок, по которому поиск производится в первую очередь в выбранном документе, что позволяет применять более короткий шаблон на меньшей площади поиска. Если же вы не знаете, в каком документе искомый объект находится, или если вы не хотите искать с начала документа, вы можете применить сплошной поиск, который в любом случае обнаружит искомый объект.

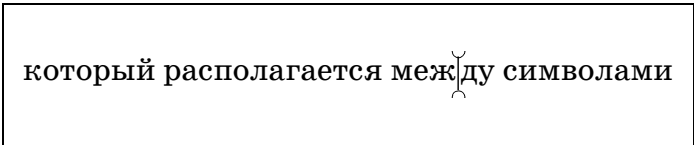
5.5. Форма курсора и методы выделения

Цель как поиска с ограничителями, так и пошагового поиска строки обычно заключается в том, чтобы обнаружить в тексте некоторую строку и выделить ее. Пользователи стараются использовать как можно более короткие последовательности в шаблонах поиска, т. к. длинные последовательности трудно набирать, и в большинстве систем они должны быть набраны с точностью до каждого символа, чтобы соответствовать целевому объекту. Поэтому поиск строки обычно не применяется для выделения даже средних по размеру целевых объектов, длина которых больше, чем 10–15 символов, не говоря уже о действительно больших блоках текста. Поиск строки применяется для того, чтобы помочь пользователю обнаружить место, где расположена искомая выборка, после чего пользователь может применить уже другой метод для обозначения этой выборки (например, использовать ГУВ для перемещения курсора от одного конца выборки до другого). Однако, если края выборки нельзя видеть одновременно, следует использовать другой метод. Он состоит из следующих шагов: (1) обозначение одного края выборки с помощью той техники, которая используется в данной системе; (2) сделать видимым другой край выборки с помощью полос прокрутки; (3) обозначить другой край выборки. В большинстве систем обозначение второго края выборки позволяет выделить всю выборку.

Более эффективный подход заключается в том, чтобы создать такой механизм поиска, который позволили бы размещать курсор на конкретном символе. Два таких размещения курсора можно использовать для обозначения первого и последнего символа выборки. Таким образом, все множество механизмов, обычно используемых для поиска краев выборки (а именно: перемещение курсора, прокрутка, различные механизмы поиска страниц, поиска по шаблону и т. д.) и их обозначения заменяются одним механизмом, который используется два раза, что упрощает процесс изучения, использования и формирования привычек, а также упрощает внедрение.

Рассмотрим теперь графическую форму курсоров. В настоящее время наиболее распространенной формой текстовых курсоров является курсор, который помещается между символами, как это показано на рис. 5.7. Одной из проблем стандартного текстового курсора является то, что пользователи пытаются поместить его точно между символами, целясь на небольшой горизонтальный объект, который по размеру

меньше, чем требуемый, что в соответствии с законом Фитса приводит к временным затратам. Кроме того, во время тестирований, проведенных в компании Information Appliance, мы с удивлением обнаружили, что эта распространенная форма курсора создает интересную когнитивную проблему, состоящую в том, что пользователь должен располагать курсор по-разному, в зависимости от того, какое действие он собирается совершить далее. В частности, чтобы удалить существующий символ с помощью клавиши <Backspace>, требуется разместить курсор справа от символа (в английском или любом другом языке, который читается слева направо). Чтобы вставить какой-то символ рядом с существующим символом, необходимо поместить курсор слева от существующего символа (в этом случае существующий символ сдвинется вправо). Наше удивление было связано с тем, что использование стандартного курсора хорошо знакомо каждому пользователю, и поэтому никто не мог подумать, что курсор можно было бы рассматривать как потенциальный источник проблем.¹ Любой, кто пользовался курсором, знает, что разобраться с тем, как им пользоваться, несложно. Тем не менее, у начинающих пользователей компьютеров может возникнуть некоторая путаница и ошибки при первых попытках его использования. Путаница усугубляется еще и тем, что пользователи не видят локусы действия применяемых команд.



который располагается между символами

Рис. 5.7. Стандартный курсор, помещаемый между символами

Введение режима, при котором удаление происходит в направлении, обратном обычному, не решает проблемы. Такой метод, называемый правосторонним удалением (forward erase), позволяет всегда размещать курсор слева от символа, к которому будет применено действие. Однако в этом случае вы будете иногда неожиданно для себя удалять в неправильном направлении, т. к. текущее направление удаления не находится в локусе вашего внимания. Если требуется наличие двух направлений удаления, желательно использовать для этого две разные клавиши или же ввести режим для противоположного направления удаления.

Использование курсора, который визуальнo показывает как (1) область вставки, так и (2) символ или символы, которые могут быть удалены клавишей <Backspace>, может быть ценным улучшением интерфейсов, ориентированных на работу с текстами. Вторая форма под-

¹ Этот пример еще раз показывает, насколько важно проводить тестирование интерфейсов с помощью аудитории, на которую продукт ориентирован.

светки может быть аналогична той, которая используется для подсветки выборок. Один из способов достижения этого показан на рис. 5.8.

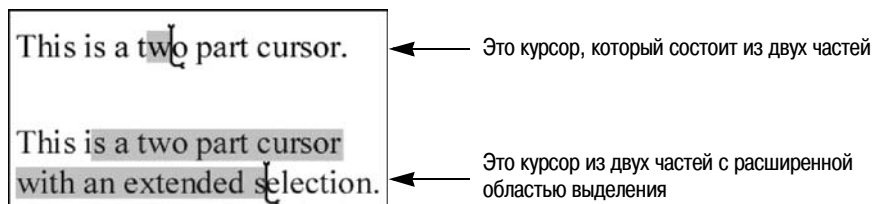


Рис. 5.8. Курсор из двух частей, выделяющий как отдельный символ, так и расширенную выборку. Часть курсора, которая обозначает область вставки, мигает для того, чтобы курсор можно было легко обнаружить на экране

Перемещение курсора к объекту с помощью ГУВ является многократным процессом, в котором пользователь сначала визуализирует целевой объект и затем перемещает курсор к этому объекту. Если пользователь не помнит, где находится курсор, ему также придется визуализировать его местонахождение. После этого он опять ищет целевой объект и т. д. до тех пор, пока объект и курсор не окажутся рядом в одном поле зрения шириной приблизительно 5°. Другими словами, для перемещения курсора с помощью ГУВ пользователь должен выполнить несколько процессов, требующих времени. Тем не менее, ГУВ необходимо для указания на графические объекты или при работе в полностью графической среде.

Неповоротливость ГУВ при работе с текстами усугубляется в случаях, когда возникает необходимость указать на объект, находящийся за пределами страницы. В этих случаях использование только ГУВ вынуждает пользователя использовать полосы прокрутки, переключатели страниц и другие механизмы, чтобы перейти к тому содержанию, на которое нужно указать. Каждый из этих методов требует изучения, и многие из них работают довольно медленно.

Функция LEAP дает пользователю когнитивное преимущество, которое избавляет его от необходимости выбирать между разными методами в зависимости от расстояния до объекта. (Alzofon и др., 1987 г.). Наличие клавиши <LEAP> и ГУВ означает, что имеется два основных способа перемещения курсора. Однако пользователь выбирает тот или иной метод в зависимости от содержания объекта, а не от расстояния между курсором и этим объектом. Обычно содержание уже находится в локусе вашего внимания, поэтому когнитивный выбор между клавишей <LEAP> и ГУВ сделать сравнительно просто.

Использование функции LEAP может быть особенно полезным в системах, управляемых голосом, а также для пользователей с двигательными нарушениями или повторными стрессовыми поражениями, для которых важно минимизировать работу с клавиатурой. Специальная кнопка (например, установленная на микрофоне) может служить для

установки квазирежима LEAP. Объектом для функции LEAP всегда является некоторый символ в тексте, и этот символ (независимо от того, смотрит ли пользователь на него или думает о нем) обычно находится в локусе внимания пользователя во время того, как он применяет функцию LEAP. В отличие от использования ГУВ, в этом случае пользователь не должен визуально искать целевой объект, чтобы переместить к нему курсор. Это свойство функции LEAP настолько выражено, что ею могут пользоваться даже слепые пользователи, что было установлено в ходе тестов, проведенных в больнице Ведомства по делам ветеранов в Пало Альто (результаты не опубликованы).

Еще более важным является то, что эта функция может использоваться в системе, построенной на ее основе, так широко, что вскоре оно становится для пользователя автоматичным. В структуре этой функции нет ничего, что могло бы препятствовать формированию ценных автоматических навыков (привычек) или что могло бы привести к проблемам после того, как такие привычки сформировались. Функция LEAP должна работать быстро, всегда обнаруживая следующий экземпляр текущего шаблона поиска за время, не превышающее время реакции пользователя, чтобы не давать ему возможности и времени выполнить какое-то действие. Необходимая скорость может быть достигнута с помощью методов, рассмотренных в разделе 5.4.

Для сравнения ГУВ и функции LEAP по временным затратам можно использовать метод GOMS-анализа скорости печати. При условии, что руки пользователя изначально находятся на клавиатуре, использование ГУВ для указания на какую-то букву в тексте требует выполнения следующих операций: *Н Р К*. С учетом правила размещения операторов *М* получаем *Н М Р К*, или $0.4 + 1.35 + 1.1 + 0.2 = 3.05$ с. Время, необходимое на использование функции LEAP, зависит от количества символов, которые следует ввести, чтобы перейти к требуемому целевому объекту. Тестирование показало, что среднее число символов, используемых при применении функции LEAP, составило около 3.5 для тех пользователей, которые были протестированы в первую неделю использования компьютера Canon Cat. Для перемещения курсора к некоторому целевому объекту требуется выполнить следующие операции: нажать на клавишу <LEAP>, после чего набрать 3.5 символа. Таким образом, применение функции LEAP в среднем состоит из 4.5 символов. Добавление, в соответствии с правилами, оператора *М* дает в результате $1.35 + (4.5 * 0.2) = 2.25$ с.

В другом эксперименте по хронометрированию, в котором участвовали опытные пользователи, сравнивалась мышь с функцией LEAP. Задача состояла в том, чтобы переместить курсор между двумя случайно выбранными символами, отображенными на экране размером 25 строк по 80 символов. Отсчет времени не начинался до тех пор, пока пользователь не начинал двигать мышью, или до тех пор, пока он не нажимал на клавишу <LEAP>. Среднее время составило приблизительно

3.5 с для мыши и 1.5 с для клавиши <LEAP>. Увеличение значения по сравнению с расчетным, вероятно, связано с небольшими размерами целевых объектов (в качестве которых использовались отдельные символы), что было обусловлено законом Фитса. Неожиданно малые значения для клавиши <LEAP>, вероятно, могут объясняться небольшими размерами текста, в котором возможный шаблон поиска в среднем мог состоять приблизительно из 2 символов. Участникам эксперимента было дано неограниченное время на планирование своих действий перед использованием мыши или клавиши <LEAP>. Таким образом, *во многих случаях пользователь может быстрее завершить передвижение курсора с помощью клавиши <LEAP>, чем переместить свои руки с клавиатуры к мыши.*

5.6. Позиция курсора и клавиша <LEAP>

Объектом функции клавиши <LEAP> является отдельный символ. С какой стороны символа должен появляться курсор: справа или слева? Размещение курсора слева от символа правильно только в том случае, если вы собираетесь сделать в этом месте вставку. Тогда как размещение курсора справа от символа правильно при условии, что вы собираетесь удалить этот символ. По-видимому, компьютер должен знать о вашем намерении перед тем, как правильно разместить курсор.

Для случая со вставкой рассмотрим старый курсор в виде прямоугольника вокруг буквы или в виде подчеркивания. При использовании клавиши <LEAP> для перехода к какой-то букве курсор должен располагаться *на* самой букве. (Курсор не мешает чтению буквы; рис. 5.9.) Тогда пользователь может точно сделать в этом месте вставку или удаление. Прямоугольный курсор позволяет более точно, чем стандартный, межсимвольный курсор, показать место, где произойдет вставка или удаление.

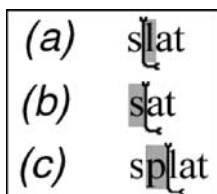


Рис. 5.9. (а) С помощью ГУВ или клавиши <LEAP> курсор был размещен рядом с буквой *l* в слове *splat*. (б) Если нажать клавишу <Backspace>, получится *sat*. (в) Но если вы введете букву *p*, вы получите слово *splat*. Обратите внимание на то, что в первом случае обе части курсора (для вставки и удаления) были размещены на букве *l*, показывая, что вы можете в этом месте сделать либо вставку, либо удаление. После удаления (б) или вставки (в) элементы курсора разделились, показывая теперь, где возможно выполнить вставку или удаление

Такой вариант порождает другую проблему. Предположим, что имеется слово *tongs*. Курсор установлен на букве *o*. Вы вводите букву *h*. Получается *thongs*. Куда должен переместиться курсор? Здесь опять же нужно знать намерение пользователя. Если курсор находится на букве *h*, то неясно: эта буква является местом вставки или удаления? Удалением должно быть действие, противоположное вставке, – так, чтобы нажатие на клавишу <Backspace> привело к удалению буквы *h* и возвращению к слову *tongs*. Но если вы введете другую букву, скажем *r*, такое нажатие должно сместить букву *o* к букве *h*, образуя в результате слово *throngs*. При использовании надсимвольного курсора, наверное, тоже должны быть предусмотрены средства предварительного определения направления его действия.

Проблема может быть решена следующим образом. Когда вы передвигаете курсор, он должен помещаться на отдельный символ. При вводе или удалении курсор должен разделяться на две части, связанные с двумя прилегающими символами, как это показано на рис. 5.9. Первый символ обозначен удаляющей частью курсора, а второй обозначен снизу указателем вставки. Фокус отчасти заключается в том, что используется курсор из двух частей. Он сделан таким образом, что когда его части объединяются (как они должны это делать после нажатия клавиши <LEAP> или перемещения курсора каким-то другим способом), пользователю графически ясно, что они располагаются на одном символе независимо друг от друга. Все это может показаться сложным, когда описывается словами, но на самом деле для начинающего пользователя научиться пользоваться курсором LEAP проще, чем обычным курсором, изображенным на рис. 5.7. Все, что нужно знать начинающему пользователю – это то, как выглядят маркеры для выделения и вставки, и что клавишей <Delete> удаляется выделенное, а вводимые символы появляются рядом с маркером вставки.

Символы располагаются скорее последовательно, чем смежно, т. к. хотя два последовательных символа обычно и могут быть смежными, они не являются таковыми, если первый из них находится в конце строки. Для языков, которые читаются слева направо, курсор удаления обычно располагается справа от вставляемого символа, а для языков, которые читаются справа налево, – порядок обратный. Для вертикально-ориентированных языков курсор вставки размещался бы под курсором удаления, и для двумерной печати (boustrophedonic scripts) в каждой строке они бы менялись местами.

Восприятие целевых объектов становится проще при использовании ГУВ с курсором из двух частей, поскольку объекты, или символы, больше по размеру, чем пробелы между ними, и поэтому активные области легче обнаружить. (Активная область для курсора PARC распространяется приблизительно до середины каждого символа по обе стороны от пробела между ними. Однако визуально они никак не разделяются, отчего целевой объект кажется меньше по размеру.

На практике многие пользователи стараются поместить курсор между символами, тем самым вызывая ограничение закона Фитса, связанное с небольшими размерами целевых объектов.)

Как мы уже говорили, начинающие пользователи иногда испытывают трудности в понимании того, как работает стандартный межсимвольный курсор. Этот переходный процесс связан с интерфейсным элементом, настолько знакомым каждому пользователю, что редко (если вообще когда-то) у кого-нибудь возникает мысль подвергнуть его сомнению. С курсором из двух частей, который при переходе на какой-то символ складывается, упомянутая неясность в том, на какой символ курсор указывает, исчезает и таким образом снимает всю проблему.

В некоторых случаях может быть полезным ограничить действие функции LEAP. Ограничение поиска может быть выполнено с помощью: (1) выделения области поиска; (2) выделения команды или текущей выборки (как это описано в разделе 5.2.1), ограничивающей последующий переход (с помощью клавиши <LEAP>) к этой области, которая становится старой выборкой; (3) использования клавиши <LEAP>. Однако ограничение функции LEAP не должно выполняться путем введения режима. Отсутствие возможности определить, включен режим или нет, делает использование функции LEAP чрезвычайно неудобным – пользователь больше не сможет легко находить необходимые ему объекты. Компания Canon использовала в компьютере Canon Cat механизм, названный «Local Leap» (локальный прыжок), который вызывал именно эти проблемы, о которых читатель этой книги теперь уже знает и может их предупредить.

Внедрение функции LEAP также требует наличия другой функции – функции LEAP AGAIN (Прыгнуть еще раз), с помощью которой выполняется переход к следующему экземпляру того же шаблона в том же направлении, в котором функция LEAP была применена последний раз. В компьютерах SwyftWare и Canon Cat поиск обычно осуществлялся в виде перехода (LEAP) к шаблону и затем использования несколько раз клавиши <LEAP AGAIN> до тех пор, пока не обнаруживался искомый экземпляр.

Функция LEAP позволяет унифицировать выполнение поиска и перемещения курсора внутри текста с учетом того, что текст может включать таблицы, графические элементы и любые другие алфавитно-цифровые символы. После некоторого периода использования функции LEAP пользователь перестает замечать процесс перехода (LEAPing) так же, как оператор, владеющий десятипальцевым методом набора, не замечает механического движения пальцев и концентрируется только на создании содержания. Если вы хотите разместить какой-то целевой объект на дисплее, вы «перескакиваете» (LEAP) к нему, и вам не нужно для этого останавливаться, чтобы найти этот объект внутри содержания системы. Вам не нужно проходить по иерархическим структурам или открывать папки. Вы просто «перепрыгиваете» прямо к тому,

что вам нужно. Однако когда люди, привыкшие к стандартным файловым структурам, впервые сталкиваются с системой, снабженной функцией LEAP, они часто продолжают думать с точки зрения того, где нужный объект располагается иерархически, и поэтому ищут его, пытаясь установить его общее местонахождение, и затем перемещаются к искомому экземпляру по определенному пути. Если среда, внутри которой вы осуществляете поиск, хорошо индексирована, для пользователя становится не важным, находится ли искомый элемент в памяти, в локальном или сетевом хранилище или же в локальной или глобальной сети.

Говоря о методах поиска, следует упомянуть наблюдения, которые были сделаны Ландауэром (Landauer) и его коллегами. Ими было показано, что наиболее распространенные формы расширенного текстового поиска, в котором поиск определяется булевыми комбинациями строк или регулярными выражениями, менее эффективны в сравнении с полнотекстовым механизмом поиска. Последний оказывается и более быстрым, и более простым, и с помощью него пользователи находят больше подходящих элементов, по которым они производят поиск (Landauer, 1995). Как показывает GOMS-анализ и другие измерения эффективности, LEAP имеет преимущества над методами поиска, использованными Ландауэром, и поэтому обладает еще большей ценностью.

5.7. Ликвидация приложений

С каждой новой версией ваша любимая программа выполняет данную задачу в два раза дольше.

Линкольн Спектор

Современная структура программного обеспечения, состоящая из операционной системы и выполняемых под ней приложений, изначально является модальной. Это означает, что для того, чтобы интерфейс был немодальным, требуется подход, при котором не используются приложения в их современном виде.

Поскольку жесты (например, те, с помощью которых вызываются команды) из одного приложения не могут быть доступными в другом приложении, вы должны знать о том, какое приложение в данный момент является активным. Но вы не можете знать это наверняка, когда в локусе вашего внимания находится задача, которую вы пытаетесь выполнить. Поэтому иногда вы будете применять жесты, которые либо не будут давать результата, либо будут давать неправильный результат. Отдельной, создаваемой прикладными программами, стоит проблема, что средства одного приложения являются недоступными, когда вы находитесь в другом приложении. Например, ситуация может быть такой: вы хотите выполнить некоторую задачу, которую могли бы выполнить в приложении А. Но вы находитесь в приложении В, в котором аналогичной команды нет. Специалист по вычислительной

технике Дан Свайнхарт (Dan Swinehart) назвал такую ситуацию «дилеммой вытеснения» (dilemma of preemption) (Tesler, 1981, с. 90).

Известны три подхода к решению дилеммы вытеснения. Наиболее распространенный метод заключается в том, чтобы каждое приложение снабдить всеми средствами, которые пользователю могут понадобиться. Этот метод обсуждался в разделе 5.1, где было указано, что в каждом персональном компьютере имеется множество различных текстовых редакторов. Большинство текстовых процессоров содержит в себе множество разных текстовых редакторов. Например, типичный текстовый процессор имеет более простой текстовый редактор для ввода шаблона в диалоговое окно поиска, чем тот, который используется для работы с основным текстом. При таком подходе приложения расширяются до колоссальных размеров, поскольку каждое из них должно решать огромное множество задач, которые имеют второстепенное значение с точки зрения его основной задачи. Например, мой текстовый процессор имеет встроенную программу для рисования, с помощью которой я могу создавать простые рисунки, без необходимости выходить из редактора. В то же время, моя программа для рисования графики имеет встроенный текстовый редактор, с помощью которого я могу включать в рисунки форматированные блоки текста без необходимости выходить из программы рисования. Средства рисования в текстовом редакторе и средства редактирования текстов в программе для рисования хуже, чем программы, которые разработаны специально для соответствующих задач. В идеале все команды и возможности как программы для рисования, так и текстового редактора должны быть доступными для пользователя в любой момент.

Аналогичным образом в каждой программе имеются средства для сохранения и открытия именованных файлов. Однако эти средства работают по-разному и имеют разные возможности в разных программах. Это приводит к путанице, трудностям в использовании и создает огромный объем избыточного программного обеспечения, которое требует оплаты, усилий на изучение, ведения документации, а также адекватного размера оперативной памяти и места на диске. То же самое касается и множества других возможностей (например, печати), которые дублируются (или очень похожи) в разных программах.

Эти проблемы в некоторой степени были учтены производителями и разработчиками. Ряд компаний разработали программное обеспечение, которое позволяет включать в единый **составной документ** (compound document) части, созданные в разных приложениях. Когда вы щелкаете мышью по любой точке в таком документе, автоматически активизируется приложение, в котором эта часть составного документа была создана. Этот метод позволяет избежать открытия явным образом соответствующих приложений при работе с составным документом. Конечно, для создания такого документа пользователю все равно приходится вручную запускать приложения, создавать части состав-

ного документа и затем собирать их в единое целое (обычно с помощью операций вырезания и вставки или перетаскивания).

Хотя этот метод и создает некоторое, самое небольшое удобство, дилемма вытеснения с помощью него не решается, что можно видеть на разных примерах (OpenDoc компании Apple, NextWave компании HP, OLE компании Microsoft и их производные). Когда вы работаете в одной из частей составного документа, у вас в распоряжении нет программных средств, которые используются для создания других частей. Более того, такой документ как бы не имеет границ, и его поведение неожиданным образом меняется в зависимости от места. Таблица и электронная таблица могут выглядеть одинаково, но одна действует по правилам текстового процессора, а другая – по правилам программы электронных таблиц. Это является злодейской модальностью, поскольку единственным признаком вашего местоположения является изменяющееся при щелчке меню, которое обычно расположено далеко от локуса внимания. Как мы уже знаем, это является неэффективным средством извещения пользователя о текущем состоянии системы. Хотя необходимо сказать, что абсолютного средства, конечно, здесь быть не может.

Оригинальным методом устранения режимов, которые присущи приложениям, было использование оконной парадигмы. Алан Кэй (Alan Kay) из компании Xerox PARC предложил перекрывающиеся окна (overlapping windows) для того, чтобы частично устранить модальность приложений. Он также хотел убрать разделение между операционной системой и приложениями, но успеха добился, главным образом, в том, чтобы сделать функционирование операционной системы видимым в форме так называемого рабочего стола. На то время это было действительно продвижением вперед, но, как выразился Ларри Теслер (Larry Tesler) из компании PARC, «окна, в некотором смысле, – это режимы в овечьей шкуре» (Tesler, 1981, с. 94). Это означает, что окна, по сути, не устраняют модальность приложений, но позволяют сделать видимыми и доступными одновременно множество приложений. Идея, предложенная Кэем, и другие идеи, возникшие на основе использования окон, стали важным шагом вперед, результатом которого люди пользуются более десяти лет. Однако проблема режимов и дилемма вытеснения, порожденная существованием приложений, так и не были решены. С того времени овечья шкура поизносилась, и из нее все чаще стал выглядывать на нас волчий зуб. В разделе 5.8 будет рассмотрен один из способов того, как можно окончательно разделиться с этим волком.

Калькулятор или компьютер?

Не секрет, что многие из нас держат рядом со своим компьютером калькулятор. Почему же нам требуется это примитивное устройство, когда в нашем распоряжении имеется целый компьютер? При-

чина в том, что для выполнения простых арифметических действий на компьютере нам приходится предпринять настолько изощренные манипуляции, что они могли бы быть достойны представления в цирке. Представим, что вы сидите за компьютером и набираете какой-то текст в текстовом процессоре и вам необходимо узнать, сколько стоит одна упаковка Phumuxx, при том, что 375 упаковок стоят \$248.93. На моем компьютере мне приходится открывать окно калькулятора. Для этого я переносу руку с клавиатуры на мышь, с помощью которой выполняю команду выбрать и перетащить (click-and-drag), чтобы открыть окно калькулятора. Затем я переносу руки обратно на клавиатуру и ввожу нужные цифры или же долго сначала вырезаю их из текста, а потом вставляю в поле калькулятора. Потом мне нужно нажать еще пару клавиш и, наконец, скопировать полученный результат из окна калькулятора в документ. Иногда дело осложняется тем, что окно калькулятора открывается прямо поверх того места, где находятся необходимые мне числа. В этом случае я должен также использовать мышь, чтобы переместить окно калькулятора в другое место и продолжить операцию. Проще достать карманный калькулятор и с помощью него сделать все намного быстрее.

Более удачным решением было бы использование специальной кнопки Вычислить (Calculate) или сквозной команды меню, с помощью которой можно было бы вычислить выделенное арифметическое выражение (такое, например, как $248.93/375$) в любой среде, будь это текстовый процессор, программа коммуникации или рисования, презентационное приложение или просто рабочий стол. Другими словами, это пример универсальной функции, применимой в любом месте.

С помощью эксперимента, в котором участвовал опытный оператор компьютера (и владелец калькулятора), я смог определить, что общее время, которое понадобилось оператору, чтобы во время использования текстового процессора достать калькулятор, включить его, выполнить простую операцию сложения и переместить руки обратно на клавиатуру компьютера для продолжения работы, составило около 7 с. Затем я измерил время, которое понадобилось ему на использование встроенного калькулятора. При этом оператору было необходимо переместить курсор к меню в верхней части экрана, найти программу калькулятора, открыть ее, получить сумму и затем щелкнуть по окну текстового процессора, чтобы вернуться к своей работе. Это заняло около 16 с.

Тому же оператору было предложено выполнить арифметическую операцию в середине документа при использовании компьютера Canon Cat со встроенной клавишей <Вычислить> и возможностью выполнять арифметические операции внутри документа. Время составило 6 с (цифры вводились с помощью верхнего ряда клавиш). Та-

ким образом, в этом случае не было выгоды во времени от использования внешнего калькулятора. Кроме того, в компьютере Canon Cat после вычисления результат оставался в документе на случай, если пользователю требовалось вставить туда результат. В то же время, этот результат оставался выделенным, поэтому пользователь при необходимости мог легко его удалить нажатием на клавишу <Delete>.

В системе должна быть доступной также и другая возможность: в любом месте, где пользователь может ввести цифры, он должен иметь возможность ввести арифметическое выражение и вычислить его. Такие команды, как

- проверить орфографию в текущей выборке;
- использовать текущую выборку как арифметическое выражение и вычислить его;
- передать текущую выборку по электронной почте;
- передать текущую выборку по факсу;
- перейти по данному URL;
- выполнить текущую выборку как программу, написанную на языке Java или любом другом языке,

должны быть доступными для пользователя в любой момент, что является абсолютно выполнимым с точки зрения разработки.

5.8. Команды и трансформаторы

Хороший дизайн является более важным, чем вы думаете.

Рекс Хейфмэн

Когда сложность некоторого продукта или части программного обеспечения, или компьютерной системы превышает наши в ней потребности и создает трудности, это вполне оправданно вызывает в нас раздражение. Даже если нам требуется поработать с каким-то простым текстом, мы вынуждены разбираться в сотнях или, как в случае с Microsoft Office, даже тысячах команд и методов, которые нам не нужны. С другой стороны, если бы мы могли пользоваться этими командами по мере возникновения в них необходимости, это сняло бы ощущение избыточности и огромного когнитивного груза, даже если бы вся система стала от этого столь же сложной, каким являлось до этого само приложение.

Мечта о том, чтобы продукты были изначально действительно (а не внешне) просты и в то же время гибки, может быть достигнута на основе подхода, в котором система не должна быть сложнее, чем ваши потребности в данный момент, и при котором возможности системы можно наращивать постепенно. Чтобы понять, как это сделать, вспомним, что в разделе 5.1 было сказано о том, что почти все действия, вы-

полняемые с помощью компьютера, включают содержание, которое вы вводите или получаете, и набор операций, которые вы применяете к этому содержанию. Также вспомним, что интерфейс для любой такой операции состоит из двух частей: выборе содержания и вызова операции. Например, в игре требуется убить монстра, или, выражаясь более прозаическим языком, выполнить операцию по изменению изображения монстра на изображение взрыва. Содержание выбирается с помощью курсора (курсор подводится к монстру), а операция вызывается с помощью клавиши на ГУВ. Между прочим, это довольно хороший интерфейс – быстрый и результативный.

Вызов команды для применения к выборке может дать три возможности:

1. Операция может быть применена к выборке.
2. Операцию бессмысленно применять к данной выборке.
3. Операция может быть применена к данной выборке при условии, что эта выборка будет модифицирована.

В первом случае операция выполняется и содержание видоизменяется. Во втором случае содержание остается без изменения. В третьем случае для выполнения операции должен быть вызван другой процесс, с помощью которого выборка предварительно модифицируется.

Предположим, что вы выбираете часть фотографии, на которой изображен вид городской улицы, и пытаетесь применить к ней команду проверки орфографии. Вызов этой команды предполагает наличие некоторой последовательности символов (текста), но здесь обнаруживается фотография (растровое изображение). Одной из возможностей компьютера является возможность преобразовывать содержание из одного типа данных в другой.¹ В этом случае данные являются растровым изображением, а команда применима только к текстам, поэтому компьютер должен определить, имеется ли какой-нибудь трансформатор для преобразования растровых изображений в текст. Вообще такие трансформаторы существуют. Это так называемые программы для оптического распознавания символов (optical character recognition programs). Обычно программы для оптического распознавания знаков (OCR-программы) используются для конвертирования данных, полученных с помощью сканера, в текст, который можно редактировать. (Некоторые из цитат, которые приводятся в этой книге, были получены с помощью сканирования соответствующих статей и последующего использования OCR-программы для конвертирования полученного рас-

¹ Этот класс функций иногда называется *фильтрами*. Однако термин *фильтр* подразумевает скорее отбор, чем преобразование. Поэтому я предпочитаю термин *трансформатор*, который редко используется в науке о компьютерах (и часто используется в описаниях схем), тогда как термин *фильтр* используется гораздо чаще (как, например, в выражении *программа цифровой фильтрации* (digital filter program)).

трово́го изображе́ния в текст.) Допустим, что в компьютере имеется такая программа. Тогда эта программа автоматически запускается и анализирует растровое изображение. Если программа обнаруживает какие-то распознаваемые символы (например, знак остановки и частично читаемый знак с надписью «Первая северная улица»), то начинается проверка орфографии этих текстов, в результате которой выдается сообщение о том, что обнаружено неизвестное слово «улица», и предлагаются возможные варианты исправления: «улица», «утица», «ушица», «лица».

Программное обеспечение вместо того, чтобы состоять из операционной системы и набора приложений, рассматривается человекоориентированным интерфейсом как набор команд, некоторые из которых являются трансформаторами, автоматически вызываемыми в тех случаях, когда тип данных, предусмотренных командой, не соответствует типу данных выбранного объекта. При этом может быть вызвано более одного трансформатора. Предположим, что с помощью одного трансформатора производится конвертация из A в B , а с помощью другого – из B в C . Если команда предусматривает обработку данных типа C , а данные выборки представляют собой тип A , то перед выполнением команды должны быть запущены два трансформатора. Если необходимых трансформаторов не имеется, никаких изменений с выборкой не производится. Пользователю при необходимости выдается специальное сообщение, а выборка остается без изменений.

При таком подходе вместо прикладных программ разработчики будут поставлять наборы команд, представляющие собой совокупность взаимосвязанных операций. Например, вместо программы для обработки фотоизображений разработчик будет предлагать ряд отдельных команд, которые в совокупности будут давать те же возможности, что и программа. Пользователь сможет устанавливать столько команд, сколько ему необходимо, вместо того, чтобы устанавливать целое приложение, из которого ему может понадобиться только некоторая часть. С помощью Интернета передовой разработчик сможет продавать свое программное обеспечение отдельными командами и даже предусмотреть скидки на покупку определенных наборов или определенного числа команд.

Когда пользователи сетуют на невероятную сложность приложений и просят, чтобы программы были проще, «без этих ненужных колокольчиков и свистков», разработчики отвечают, что облегченные («lite») версии программного обеспечения зачастую не имеют успеха на рынке. Неудачи с облегченными версиями программных пакетов могут быть объяснены следующим образом. Поскольку пользователь никогда не знает, какие именно возможности полного пакета могут ему понадобиться, он покупает весь пакет, т. к. это единственный способ получения этих возможностей. Если пользователь приобретет ограниченную версию по меньшей цене, единственный способ, каким он смо-

жет обновить эту версию, — это купить полный пакет, даже если пользователь хочет получить только какую-то одну небольшую функцию из этой программы. Поэтому выходит, что лучше сразу покупать полную версию и примириться с ее сложностью. В результате не удивительно, что мы оказываемся как бы в ловушке. Более гуманным подходом было бы предоставление возможности покупать команды по отдельности.

В различных обзорах сообщается о большом количестве никогда не используемых элементов того или иного программного обеспечения. В последнее десятилетие это число возросло приблизительно с 15% до почти 50%. Это создает большой беспорядок. Если же вместо приложений использовать наборы команд, каждая из которых может быть установлена независимо от других, пользователи сведут показатели этой статистики почти до 0. Другими преимуществами такого подхода для разработчиков является возможность поступенчато улучшать свои продукты, а также более простым образом и чаще поставлять (т. е. продавать!) новые элементы приложений, поскольку в этом случае производителям не придется каждый раз выпускать весь пакет заново. Интернет является самым подходящим средством для осуществления таких многократных поштучных продаж.

Разработчики (необязательно те же, кто выпускает наборы прикладных команд) также могут производить трансформаторы. И они тоже могут продаваться поштучно. Если большинство пользователей регулярно использует большинство команд текстовой обработки, выпущенных разработчиком А, и разработчик В придумает какую-то полезную команду, которую разработчик А не предоставляет, то разработчик В сможет продавать эту команду покупателям продукта, производимого разработчиком А. Однако разработчик В может использовать другую структуру данных. В этом случае он может предложить также и трансформатор для перевода из одной структуры данных в другую, и наоборот. Если продуктом, производимым разработчиком А, пользуется большое число потребителей, разработчик В может создать версию данной команды, работающую непосредственно со структурой данных, предусмотренной разработчиком А. Кроме того, третий разработчик (С) может специализироваться на разработке трансформаторов. Для пользователей может стать обычной практикой покупать трансформаторы именно у таких разработчиков, а для разработчиков команд — давать лицензии на соответствующие трансформаторы. Эта коммерческая структура частично уже существует сегодня, и различные компании (например, DataViz) специализируются на разработке трансформаторов.

Элементы такой компьютерной среды «команды плюс трансформаторы» также существуют, и их нетрудно будет собрать в единую рабочую систему. Для пользователей такие системы будут проще в использовании и более гибкими в сравнении с сегодняшними разработками, огра-

ниченными применением приложений. Такое решение может уменьшить степень избыточности и сложности, снять проблему совместимости между приложениями и необходимость многократно решать одну и ту же проблему. Со временем операционная система может полностью исчезнуть из поля зрения пользователя. Если этого подхода будут придерживаться должным образом, то даже общепринятая версия операционной системы – рабочий стол – не сохранится в ее сегодняшнем виде.

Конечно, не все программы должны быть построены по описанному принципу. Игры, например, должны просто запускаться и работать самостоятельно. Запуск должен осуществляться обычным путем – с помощью выбора (мышью или другим способом) имени игры (возможно, из текста, включающего список имен игр) и затем использования команды Выполнить (Execute). В разделе 6.2 также будет описан альтернативный метод.

Учитивое программирование: приложения как гости

Представим, что вы были приглашены погостить в доме ваших хороших друзей Гримблсов. И даже их собака кажется вам очень милой. Единственная проблема – это портрет их любимой тетушки Астабьюлы, который висит над вашей кроватью. Этот портрет вызывает у вас и вашей жены нервную дрожь, и вы не можете даже спать в комнате, в которой на вас смотрит с портрета тетушка Астабьюла. Вы не решаетесь сказать обо всем этом Гримблсам. Как бы вы поступили в такой ситуации, будучи гостем в доме своих друзей?

1. Сняли бы картину и сожгли ее.
2. Спрятали бы ее в винном погребе, чтобы Гримблсы после месяца поисков обнаружили ее там.
3. Поставили бы ее в шкаф, чтобы Гримблсы нашли ее немного раньше.
4. Поставили бы картину в шкаф и потом, перед отъездом, вернули бы ее на свое место.

Любой учтивый гость знает, что вариант 4 является самым правильным. Принцип заключается в следующем: если ты делаешь по тем или иным причинам изменения в чьей-либо среде, позаботься о том, чтобы поставить все вещи на свои места до того, как вернется хозяин.

Современные компьютеры имеют множество параметров изменения рабочей среды, в том числе громкость динамиков, разрешение экрана и глубину изображения, вид меню, шрифт по умолчанию. В компьютере Macintosh имеются сотни настроек, а в IBM-совместимых компьютерах, работающих с операционной системой Windows и пакетом Microsoft Office, это число превышает тысячу. В чем же

здесь параллель с визитом к Гримблсам? Если вы используете чью-либо машину и изменяете в ней какие-либо настройки, то перед уходом вы, как человек учтивый, должны вернуть все изменения обратно.

Многие программы для нормальной работы требуют определенного разрешения экрана, определенного числа бит на пиксел или какой-то другой настройки параметров. Если же параметры настроены в системе неправильно, поведение программы может варьироваться в диапазоне от благовоспитанного до грубоватого или даже варварского. Ниже приводятся реакции различных протестированных мной программ в случаях, когда параметры монитора не соответствовали требуемым:

1. Полный отказ системы, после которого приходилось вручную производить перезагрузку компьютера.
2. Полный отказ системы с выдачей непонятного числового сообщения, после чего приходится нажимать на кнопку Перезапуск, чтобы произвести перезагрузку компьютера.
3. Выдается сообщение об ошибке, в котором говорится о необходимости настроить параметры экрана. Когда вы нажимаете на кнопку ОК, это приводит к полному отказу системы.
4. Выдается сообщение об ошибке, в котором говорится о необходимости настроить параметры экрана. После нажатия на кнопку ОК вы можете открыть панель управления и выполнить необходимые настройки.
5. Выдается запрос о том, нужно ли выполнить изменение параметров экрана. Если вы нажимаете на кнопку ОК, настройки автоматически изменяются. Если вы нажимаете на кнопку Отмена, запуск программы прекращается, а параметры экрана остаются без изменения.
6. Параметры экрана изменяются без всякого предупреждения, и программа запускается.
7. В специальном диалоговом окне выдается сообщение о том, что можно изменить настройки экрана. Если вы нажимаете на кнопку ОК, настройки изменяются, и программа запускается. После завершения работы программы настройки автоматически возвращаются в начальное положение.

Читатель из будущего может подумать, что полный отказ системы — это уж слишком. Ни один из вышеприведенных вариантов нельзя назвать достойным вежливого гостя, хотя вариант 7 приближается к тому.

Наилучшим вариантом является сочетание, включающее запрос, упомянутый в пункте 5, и способ выхода из программы, описанный в пункте 7. В маловероятном случае того, что изменение настроек

экрана может помешать прохождению другого, параллельного или фоновому процессу, такой запрос позволит отказаться от запуска программы.

8. Параметры экрана изменяются без всякого предупреждения, и программа запускается. По завершении работы программы настройки устанавливаются в начальное положение.

Другими словами, при запуске программы она будет просто выполняться предусмотренным образом, не влияя на работу других программ. Это может служить определением благовоспитанной компьютерной системы. Но что произойдет, если две программы, требующие разных экранных настроек, одновременно будут выдавать какие-то изображения? Вероятно, это будет ситуация, когда лошадь будет отображаться с одной цветовой палитрой, а рыба с другим разрешением.

Вариант из пункта 7, который распространен и может показаться приемлемым, является неудачным, поскольку в нем выдается «диалоговое» окно, которое, по сути, не дает никакого диалога. Это просто злоупотребление, потому что на сообщенную информацию пользователь не может дать качественный ответ, что только приводит к пустой трате времени. (Метод прозрачных сообщений, описанный в разделе 5.2.3, позволяет решить эту проблему.)

Прикладные программы, по сути, являются бременем для пользователя, но раз уж нам приходится работать с ними, давайте сделаем так, чтобы они работали как следует (см. также Raskin, 1993). Обобщая сказанное, повторим, что программа, или в будущем команда, должна автоматически выполнять перенастройку параметров, если это необходимо для ее нормальной работы. После завершения работы программы (или прерывания пользователем) все параметры должны быть восстановлены в исходное положение. Если восстановление невозможно или может привести к нежелательным побочным последствиям, должно выдаваться специальное предупреждение, в котором сообщалось бы об этих последствиях и предлагалось бы пользователю решить: продолжить ли действие или нет.

В настоящее время мы не можем применять к одному экрану разные режимы разрешения одновременно. Но это уже проблема другого рода. Тем не менее, на мой взгляд, пользователи были бы довольны, если бы работа каждого интерфейса руководствовалась пунктом 8.

6

Навигация и другие аспекты человекоориентированных интерфейсов

*Средний человек испытывает большие страдания
от муки новой идеи.*

Адмирал Уильям С. Симс

Одним из самых хвalebных терминов, используемых в отношении интерфейсов, является слово «интуитивный». При ближайшем рассмотрении это понятие исчезает так же бесследно, как шарик в наперстах, и заменяется более простым термином «знакомый».

Существующие сегодня системы навигации, которые, нужно сразу сказать, не являются удачными в любом случае, абсолютно не подходят для целей обработки терабайт информации, что мы вынуждены просматривать. Однако люди и животные имеют тысячелетнюю практику перемещения внутри сложных сред и за это время сумели приобрести некоторые полезные навыки. Эти способности, сложившиеся в течение многих эпох, могут быть привлечены для наших целей — для создания того, что можно назвать масштабируемым интерфейсом.

6.1. Интуитивные и естественные интерфейсы

Нападающий на всеобщее мнение поступает во всех отношениях опрометчиво и должен быть достаточно удачлив и необычайно силен на случай, если будет услышан.

Джон Стюарт Милл «Покорение женщин»

Многие требования, предъявляемые к интерфейсам, предполагают, что конечный продукт должен быть интуитивным, или естественным. Однако такой способности, как человеческая интуиция, не существует в том смысле, в котором это слово обычно используется, т. е. как некое знание, имеющееся изначально, до знакомства с понятием, и дающее возможность пропустить процесс познания и не использовать рациональное мышление. Когда какой-то специалист с помощью того, что мы обычно называем интуицией, делает некоторый вывод со скоростью и точностью, которые превышают способности обычных людей, мы знаем, что его вывод основан на его знаниях и опыте. Зачастую специалист просто использует методы и подходы, которые неспециалисту неизвестны. Специалист по какой-то задаче часто использует такую информацию, которую другие не осознают и не понимают. Таким образом, знания и опыт, в отличие от интуиции, являются реальными.

Когда пользователи говорят, что какой-то интерфейс является интуитивным, они имеют в виду, что он работает так же, как и какой-то другой метод или программа, с которыми они знакомы. Иногда это слово означает «привычный», как в предложении: «Использование этих средств редактирования со временем становится все более интуитивным». Или же оно может означать «уже знакомый», как, например, было сказано об одном из новых аэронавигационных устройств: «Как и любое другое, это устройство можно изучить, но для того чтобы использовать его интуитивно, потребуется большой опыт» (Collins, 1994 г.).

Другим словом, которое я стараюсь не использовать в отношении интерфейсов, является слово «естественный». Так же как и «интуитивный», его смысл, как правило, не определен. На обычном языке элемент интерфейса является «естественным», если он работает так, что пользователю совсем не требуется объяснять, как им пользоваться. Обычно это означает, что есть какое-то другое знакомое человеку действие, которое выполняется аналогичным образом. Однако здесь трудно определить смысл слова «аналогичный». Подобия или аналогии могут быть различными. Когда курсор движется влево при перемещении мыши влево и когда он движется вправо при перемещении мыши вправо – это, несомненно, пример естественности. В данном случае термин «естественный» равнозначен выражению «который очень легко изучить». Хотя естественность, вероятно, невозможно определить количественно, совсем нетрудно измерить время, требуемое на изучение.

Часто говорится, что использование мыши и есть пример интуитивности и естественности. Сейчас, конечно, трудно повторить такой экс-

перимент, поскольку мышь является самым распространенным ГУВ, но когда это устройство было менее известным, я просил людей, не сталкивавшихся с ним, попробовать применить мышь в среде Macintosh. В ходе эксперимента использовалась программа под названием «Люк» (The Manhole) – увлекательная и хорошо разработанная лабиринтная игра для детей, в которой не требовалось ничего, кроме нажатия на разные области экрана. Убрав клавиатуру, я показывал на мышь и говорил: «Это мышь. С ее помощью вы можете играть в эту игру. Попробуйте сделать несколько попыток». Если мне задавали какие-то вопросы, я старался на них не отвечать определенно и говорил что-нибудь вроде: «Ну, попробуйте». Реакция одной умной учительницы из Финляндии, которая до этого никогда не видела подобного устройства, хотя в других отношениях была компьютерно грамотной, была довольно типичной: она взяла мышь в руку и сняла со стола.

Сегодня это кажется нелепым, но этот же момент был показан в одном фильме из научно-фантастического сериала «Star Trek». Инженер космического корабля попал из будущего в наши дни и оказался рядом с компьютером Macintosh. Он берет мышь в руки, подносит ее ко рту, как будто это микрофон, и говорит в нее с сильным шотландским акцентом: «Компьютер, ...» Эта ошибка вызывает смех зрителей. Я порадовался за создателей фильма, которые показали, что мышь не является чем-то очевидным для каждого, кто пытается ею воспользоваться. В случае с моей финской знакомой ее следующим действием было перевернуть мышь и попытаться покрутить шарик. Ничего не произошло. Тогда она потрясла ее, после чего стала держать ее одной рукой, а другой стала нажимать на кнопку, что опять же не дало никакого результата. В конце концов, она приспособилась играть, удерживая мышь в своей правой руке и пальцами покручивая шарик снизу, а левой рукой нажимая на клавишу.

Эти эксперименты показывают, что скорость изучения и легкость использования какого-то интерфейса не связаны с воображаемыми свойствами интуитивности и естественности. Научиться пользоваться мышью очень легко. Все, что мне нужно было сделать в экспериментах с участием людей, не знакомых с этим устройством, – это положить мышь на стол, подвигать ее и щелкнуть по чему-нибудь с ее помощью. Через 5–10 с каждый мог понять, как пользоваться мышью. Это действительно быстро и легко, но это не значит, что использование мыши является интуитивным или естественным.

Убеждение, что интерфейсы могут быть интуитивными и естественными, часто оказывается препятствием для их улучшения. Меня часто просят в качестве консультанта помочь улучшить интерфейс какого-либо продукта. Обычно интерфейс можно улучшить так, чтобы с точки зрения времени изучения, скорости работы (продуктивности), частоты ошибок и простоты исполнения он был лучше в сравнении как с текущей версией продукта, так и с аналогичными продуктами кон-

курентов. Тем не менее, даже если мои предложения считаются значительными улучшениями, они зачастую отклоняются на том основании, что не являются интуитивными. Это классический случай Уловки-22¹: клиент хочет получить нечто, что должно значительно превосходить то, что предлагается конкурентом. Но если это должно быть превосходящим, то это должно быть другим (обычно чем значительно улучшение, тем сильнее разница). Поэтому результат не может быть интуитивным (т. е. знакомым). Клиент хочет получить интерфейс, который бы не сильно отличался от существующей практики разработки интерфейсов (что почти неизбежно означает Microsoft Windows) и, в то же время, каким-то образом стал бы значительным усовершенствованием. Это может быть достигнуто только в редких случаях, когда исходный интерфейс имеет какой-то существенный недостаток, который можно исправить простыми средствами. (Этот раздел частично основан на материалах книги Раскина, 1994 г.)

6.2. Улучшенная навигация: ZoomWorld

Если бы вы хотели разработать навигационную схему, приводящую к путанице, вы могли бы начать с создания интерфейса в виде лабиринта. При использовании такого интерфейса вы бы оказывались в небольшой комнате с рядом дверей, ведущих к разным путям. На дверях повешены таблички с краткими, кодированными обозначениями или пиктограммами, которые могут изменяться или исчезать² в зависимости от вашего местонахождения. Вы не можете увидеть, что находится за дверью, до тех пор, пока не зайдете за нее, и если вы зашли за какую-то дверь, вы уже не можете видеть комнату, из которой вы только что вышли. В некоторых случаях может совсем не быть пути, по которому можно было бы вернуться непосредственно назад. В некоторых комнатах могут быть карты части или всей системы комнат, но чтобы ими воспользоваться, вам необходимо сопоставить изображение на карте с теми комнатами, которые вы видели. Кроме того, карты не совсем подходят для ситуации, когда имеется трехмерная сеть. В этом описании комнаты соответствуют окнам компьютерного интерфейса и веб-сайтам, а двери – это вкладки, меню или ссылки, предназначенные для перехода в другие окна или на другие сайты.

Как известно из древних легенд и мифов, человек всегда очень плохо ориентировался в лабиринтах. Если бы мы могли легко в них разбираться, они не использовались бы для игр и головоломок. При работе с какой-либо сложной программой часто случается, что для решения возникшей проблемы необходимую команду или флажок можно най-

¹ В книге Джозефа Хеллера «Уловка-22» одной из основных линий сюжета является пункт Устава под номером 22, который невозможно обойти, поскольку он замыкается сам на себе. – *Примеч. науч. ред.*

² Адаптивные меню имеют это неприятное свойство.

ти где-нибудь глубоко в подменю. Если я опять сталкиваюсь с этой же проблемой через несколько недель, мне трудно вспомнить, в каком окне я нашел для нее решение. Мы плохо запоминаем длинные последовательности поворотов, – именно поэтому лабиринты являются хорошими головоломками, и наши современные навигационные схемы, используемые как в компьютерах, так и в Интернете, так часто приводят пользователей в замешательство. Многие жалобы на современные системы связаны с их навигацией. Разработаны частичные решения, – такие как Избранное (Favorites) в браузерах.¹ Однако мы можем хорошо запоминать положение ориентиров и позиционные подсказки, – эти черты развились в нас с эволюцией, и они могут быть использованы в разработке интерфейсов.

Противоположностью лабиринтов является ситуация, в которой вы можете видеть цель и путь к ней, который позволяет сохранить чувство ориентации во время перемещения и при необходимости вернуться назад. Изящным решением является принцип масштабируемого интерфейса ПМИ (zooming interface paradigm, ZIP), который во многих случаях дает возможность решить проблемы навигации, а также проблеме ограниченного экранного пространства, которая возникает при использовании любой существующей системы отображения. Представьте, как легко можно было бы передвигаться по лабиринтам, если бы вы имели возможность увидеть их сверху, рассмотреть их план и переместиться сразу туда, куда нужно. Принцип масштабируемого интерфейса как раз и дает такую возможность при решении многих задач, выполняемых на компьютере. Хотя ПМИ не может подойти для всех случаев, я все же остановлюсь на его положительных аспектах для того, чтобы показать, что существуют альтернативы, которые более эффективны по сравнению с пользовательским графическим интерфейсам, основанным на применении рабочего стола.

Пример масштабируемого интерфейса, который будет описан, называется ZoomWorld (дословно – масштабируемый мир). Его идея заключается в том, что пользователь имеет доступ к безграничной плоскости информации с неограниченной степенью разрешения. Эта плоскость является масштабируемой средой ZoomWorld. Все, к чему вы можете обратиться, находится где-нибудь на плоскости этой среды – будь это ваш компьютер, локальная сеть, к которой ваш компьютер подключен, или сеть сетей (как, например, Интернет).

Чтобы увидеть как можно большую площадь масштабируемой среды, нужно как бы пролететь над ней на большей высоте. Чтобы увидеть отдельный элемент, вы, наоборот, спускаетесь ниже. В масштабируемой

¹ Это решение эффективно до тех пор, пока этих ссылок не станет столько, что вы не сможете запомнить, что означает каждая из них, после чего придется определять «избранные из избранных» или другую схему, чтобы систематизировать все ссылки.

среде также имеется механизм поиска содержания. Вся метафора этой среды заключается в том, что вы можете находиться на определенной «высоте» над ней, поднимаясь «вверх» для увеличения масштаба и спускаясь «вниз» – для уменьшения. Навигация по системе осуществляется как с помощью изменения высоты, так и с помощью поиска.

ZoomWorld в принципе аналогичен тому, как используются стены в так называемых комнатах планирования (project planning rooms). По мере продвижения проекта стены в комнате планирования покрываются заметками, записками, листами, прикрепленными кнопками, стикерами, фотографиями и т. д. – словом, всем тем, что помогает запомнить или сформулировать идеи. Когда вы входите в такую комнату, вы можете встать посередине и оглядеть ее, выбрать какое-то место на стене, подойти и рассмотреть имеющийся там материал детально. Или можете при входе сразу направиться к тому месту, где, как вы уже знаете, находится информация, которую вы хотите получить или изменить.

Мы можем находить нужные нам вещи в такой комнате потому, что имеем склонность запоминать их относительное местоположение и другие ориентиры. Эта способность иногда называется пси-эффектом (psi effect) и уже давно известна специалистам по психологии. Например, вам могут сказать: «Листок по сбыту товара находится в нижней части правой стены, ближе к дальнему углу». В другом случае вы идете прямо к какому-то документу, т. к. вы помните, что он находится сразу слева от оранжевого листка, вывешенного Авивой. Во время работы в такой комнате вы можете иногда немного отходить от стены, чтобы увидеть, где сейчас находитесь. Вы можете отойти так, чтобы не видеть мелкий шрифт, но сохранить возможность читать заголовки и видеть большие таблицы. Отойдя еще дальше, вы сможете разобрать только некоторые, самые большие заголовки, увидеть цвет страниц, общий вид областей на стене и сможете сказать, где имеются иллюстрации, диаграммы или рисунки, даже если изображенное на них нельзя будет разобрать.

В масштабируемой среде можно легко прикрепить подпись к изображению или группе изображений, но она не навязывает никакой структуры иерархического или другого рода, кроме объединения по принципу близости. Я предполагаю, что большинство пользователей будет стремиться организовывать группы изображений в кластеры и некоторые пользователи будут создавать произвольные иерархии. Например, при приближении к большому заголовку «Личные фото» над группами фотографий могут открываться заголовки меньшего размера: «Дети», «Отпуск», «Животные», «Хобби», «Друзья», «Родственники» и т. д. Увеличение масштаба, примененное к тексту под заголовком «Дети», может показать имена детей: Агата, Гидеон, Гермийон. Профессиональный фотограф, скорее всего, будет иметь тщательно организованную и подобранную коллекцию. Обратите внимание, что вам не нужно запоминать имена – вы обнаруживаете их, когда увели-

чиваете масштаб и пытаетесь определить, могут ли искомые вами изображения находиться в данной категории. То же самое можно повторить для коллекций фильмов или звуков, хотя в этом случае вам придется запустить просмотр фильма или воспроизведение звука, чтобы определить то, что вам нужно.

Даже неорганизованный человек может пользоваться такой системой: он может просто выкладывать изображения любым удобным для себя способом и выполнять поиск с помощью изменения масштаба. Приблизительное местоположение нужного изображения вы сможете вспомнить благодаря пространственной памяти, которая поможет ускорить поиск.

Если вам требуется увеличить размер символов в документе, то вы, безусловно, можете сделать это, изменив масштаб, хотя, возможно, не сможете при этом поместить строки целиком по ширине дисплея, что усложняет чтение с экрана. В этом случае можно воспользоваться командой увеличения шрифта, тем самым уменьшая длину строки за счет сокращения числа символов в одной строке.

Присущую нам способность запоминать местоположение и ориентиры можно использовать для ZoomWorld. Перемещение по такой среде не будет производиться с помощью неудобных полос прокрутки (читателю предлагается попробовать самостоятельно посчитать, используя закон Фитса (см. раздел 4.4), какой должен быть размер у кнопок со стрелками). Также для этого не будут использоваться кнопки «Увеличить масштаб» и «Уменьшить масштаб» или специальные меню, которые, как и кнопки со стрелками, будут также работать медленно. Это скорее похоже на эмулирование действий человека, когда он находится в комнате планирования: чтобы увидеть большую область стены, он отступает назад; затем он подходит к нужному месту и, в конце концов, придвигается вплотную, чтобы прочитать мелкий текст, или даже использует увеличительное стекло, чтобы разглядеть деталь фотографии.

Увеличение масштаба (для того чтобы можно было рассмотреть детали), вероятно, лучше всего производить с помощью квазирежима (см. раздел 3.2.3). В этом режиме нажатие кнопки ГУВ вызывает увеличение масштаба в реальном масштабе времени в любом месте, где расположен курсор, так чтобы центр изменения масштаба был связан с позицией курсора. Для этой функции может быть использована вторая кнопка ГУВ. Для совместимости с системами, в которых не используются приложения (см. разделы 5.7 и 5.8), квазирежимы изменения масштаба могут включаться разными способами: например с помощью второго ГУВ, клавиатуры или особым образом обозначенных дополнительных кнопок ГУВ (см. приложение А).

Каким бы способом ни осуществлялось изменение масштаба, точка, которую вы увеличиваете, совпадает с позицией курсора и может регулироваться в процессе изменения масштаба с помощью ГУВ, используемого в роли устройства указания позиции. Другими словами, в процес-

се изменения масштаба система изменяет положение ZoomWorld таким образом, что позиция курсора находится в центре дисплея или рядом с ним. Если масштабирование происходит быстро (по крайней мере, 2 раза в секунду по линейным размерам) и внешне непрерывно, то увеличение и уменьшение масштаба относительно позиции курсора будет достаточно для получения графического навигационного инструмента.

В комнате планирования вы можете помещать над основными областями большие надписи, которые можно читать из любого места в комнате. Функция изменения масштаба работает аналогичным образом: размер заголовка и текста определяет, насколько сильно нужно увеличить, чтобы увидеть детали. Этот метод может заменить (или улучшить) иерархические каталоги. Важным дополнением к этому может быть достаточно быстрый текстовый поиск (например, функция LEAP), который может использоваться в тех случаях, когда требуется быстро найти какой-то текстовый элемент. Можно также использовать небольшое количество различных геометрических ориентиров (пояснения о том, что подразумевается под выражением «небольшое количество», см. в разделе 6.3, посвященном эффективности использования пиктограмм). Например, с помощью большого красного креста можно обозначать область (как в комнате планирования, так и в масштабируемой среде), в которой можно получить экстренную помощь.

Пользователи быстро понимают, какие особенности имеет тот или иной вид работы, даже если они не указываются непосредственным образом. Электронные таблицы, простые таблицы, тексты, растровые изображения, рисунки и другие виды содержимого обладают своими визуальными характеристиками, даже если смотреть на них с большого расстояния. Продукты, создаваемые группами сотрудников, разработчиков и других авторов, часто тоже легко узнаваемы.

Масштабируемое пространство обладает большой гибкостью с точки зрения компоновки содержания. Если документ увеличивается по длине и ширине, масштаб может просто автоматически равномерно измениться так, чтобы весь документ мог поместиться в той же области экрана, что и была раньше. Тот же метод организации пространства информационной плоскости может использоваться, если простая или электронная таблица увеличивается. Текст всегда можно сделать достаточно большим, чтобы его можно было читать, поскольку имеется возможность масштабирования. Обратный процесс происходит, если документ уменьшается в размерах. Свободное пространство не ограничено, если система построена как следует. С помощью особой команды можно в любом месте создать новый документ (команда позволяет копировать чистый документ и является аналогом команды Создать (New), которая используется в обычных пользовательских графических интерфейсах). Документы могут перекрываться или ограничиваться смежными документами. Внутренние ссылки и указатели на веб-сайты (URL) позволяют сразу переместиться к другому документу,

передавая его в то место и придавая ему тот размер, которые были определены при создании ссылки. На кнопках может быть столько информации, сколько угодно, в том числе и полное описание с примерами, и это нисколько не будет усложнять их использование, если смотреть на них при небольшом масштабе. Таким образом, каждый элемент может иметь встроенное описание или руководство по применению.

Изменение масштаба может происходить нелинейно во времени, начинаясь медленно и затем ускоряясь до максимальной скорости, что позволяет управлять степенью изменения масштабирования. Кроме того, при определенных значениях масштаба изменение может замедляться или ненадолго останавливаться для того, чтобы можно было легко увидеть символы в их стандартном размере.

Эффективность работы в масштабируемой среде часто позволяет (и иногда требует) использовать методы, которые отличаются от тех, которые применяются в пользовательских графических интерфейсах, оснащенных рабочим столом. Например, одни и те же данные могут рассматриваться с нескольких точек зрения, поскольку экранное пространство не ограничено. Так же как и в эффективном варианте преобразователя температур, рассмотренном в разделе 4.3.2, машина способна выполнять дополнительную работу или подготовку, которая может никогда не понадобиться или в случае необходимости облегчить работу человека. Сравнение документов может выполняться с помощью разделенного экрана, который хотя и требует специальных элементов управления разделением, но позволяет выполнять изменение масштаба отдельных областей экрана независимо друг от друга. Также эта операция может выполняться с помощью другой возможности, присущей масштабируемым средам: перемещения копий двух разных документов или двух копий одного документа таким образом, чтобы они располагались смежно. Возможность перемещения или изменения масштаба любого объекта может быть применена в том числе и к текстовым объектам, что позволяет увеличить документ и тем самым сделать его видимым на более высоком иерархическом (и графическом) уровне.

Таким образом, сноска становится не просто ссылкой – с помощью нее вы можете увеличить весь материал, на который эта сноска сделана. В этом случае изменение масштаба работает как ссылка, за исключением того, что для возвращения к основному документу вы просто уменьшаете масштаб, и поэтому вам не требуется вести запись того, где вы уже были. Для облегчения поиска какого-то набора документов сами эти документы могут быть расположены таким образом, чтобы их можно было сразу узнать при небольшом масштабе. Страница с очень большими буквами, видимыми при небольшом масштабе, может быть использована в качестве заголовка документа. Функция уменьшения масштаба может служить в качестве кнопки Назад (Back) в браузере.

Коллективная работа пользователей может быть легко организована, если все участники коллективного проекта увеличат масштаб одного и того же документа. При этом должны быть разработаны специальные правила для того, чтобы избежать противоречий. Сеть может быть представлена как пространство, в котором работа каждого пользователя помещается в персональной области. В коллективном пространстве масштабируемой среды вы сможете по своему желанию определять степень видимости вашей работы для других пользователей. Невидимые документы могут использоваться как один из способов регулирования доступа к содержимому.¹

Типичное применение ZoomWorld нашел в проекте, разработанном для компании Apricus, которая искала способ компьютеризации большой (площадью приблизительно один квадратный метр) медицинской карты, аналогичной тем, что используются в отделениях интенсивной терапии (intensive care unit, ICU). Все перепробованные до этого методы работали медленнее в сравнении с использованием физической карты и требовали больших усилий на изучение. Также нельзя было применять множество мониторов для отображения всей карты целиком. ZoomWorld смог не только вместить карту – этот интерфейс позволил автоматизировать всю базу данных отделения интенсивной терапии и стал использоваться в качестве базы данных для всего предприятия, и все это без существенных дополнительных затрат на разработку. Использование нового интерфейса позволило расширить модель бизнеса компании и область ее применения без разработки более сложного интерфейса или внедрения каких-либо дополнительных средств, хотя, конечно, потребовало базы данных большего объема. На рис. 6.1–6.8 показаны образцы из ZoomWorld, которые были разработаны в компании Apricus.² При тестировании выяснилось, что даже больничные сиделки могут научиться пользоваться этой системой всего лишь после одной минуты тренировки.

На рис. 6.1 интерфейс масштабируемой среды показывает отделение интенсивной терапии в окружении других больничных отделений. Вы можете увеличить масштаб изображения, чтобы увидеть разные виды

¹ Интересный вариант масштабируемого пользовательского интерфейса (МПИ) (zooming user interface, ZUI) под названием PAD++ (сейчас он называется *Jazz*) был впервые разработан в университете штата Нью-Мексико. См. <http://www.cs.umd.edu/hcil/pad++/>. Я благодарен доктору Дональду Норману (в то время работавшему в компании Apple) за то, что он указал мне на эту работу.

² Большая благодарность за разрешение описывать их версию метода ZoomWorld, а также использовать некоторые изображения в качестве иллюстраций. Многие подробности разработки были сообщены доктором Дэвидом Мошалом, доктором Имэньюзлом Нойком и группой их сотрудников. Благодарю также Бетти Ньюберн, дипломированную медсестру, за то, что она помогла мне разобраться в больничной среде.

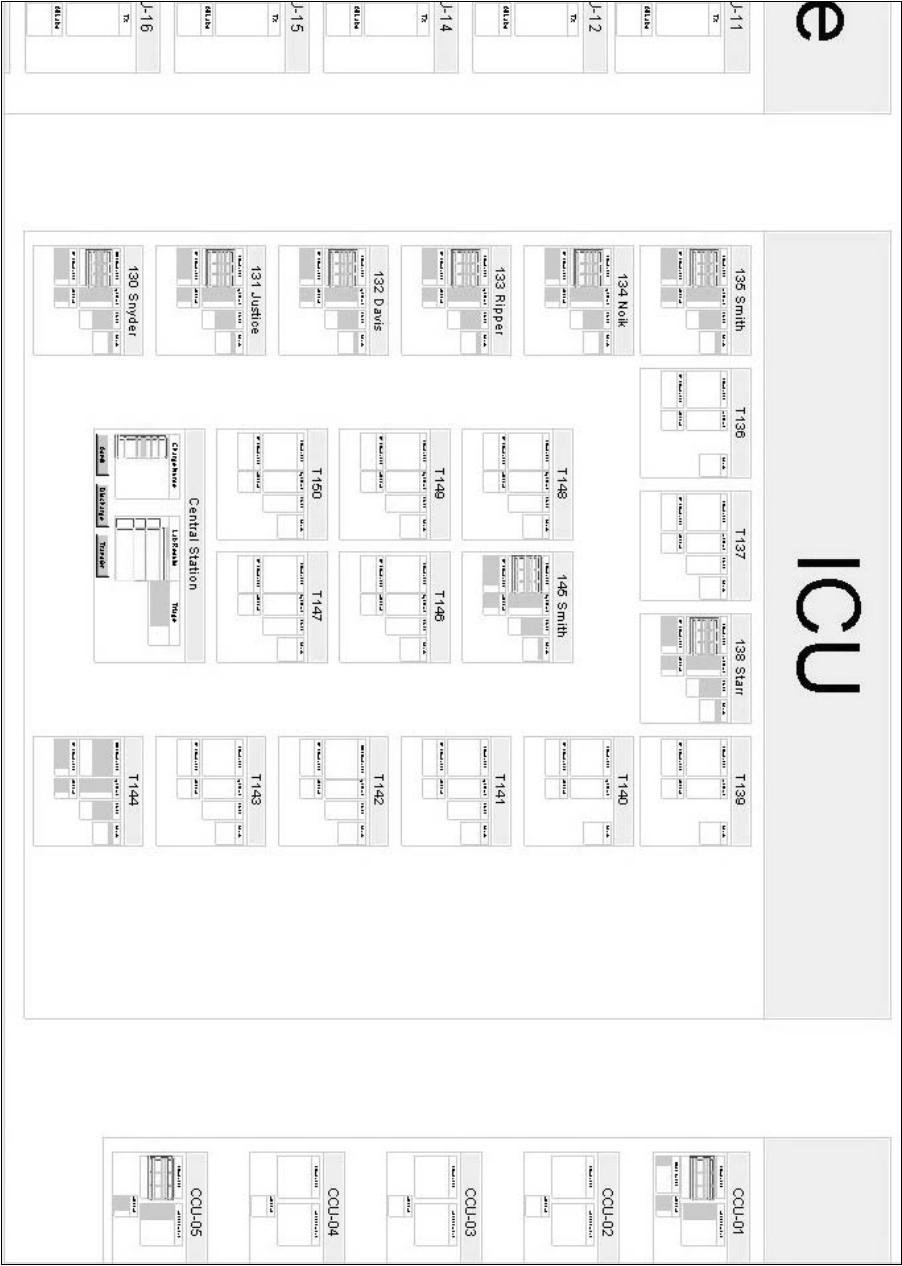


Рис. 6.1. Изображение отделения интенсивной терапии в масштабируемом интерфейсе. Палаты изображены в виде нумерованных прямоугольников с именами находящихся в них пациентов (имена выдуманы)

данных, относящихся к любой палате (можно видеть, что некоторые палаты не заняты). На рис. 6.2 показана крупным планом палата 132 и основные таблицы и диаграммы о состоянии пациента.

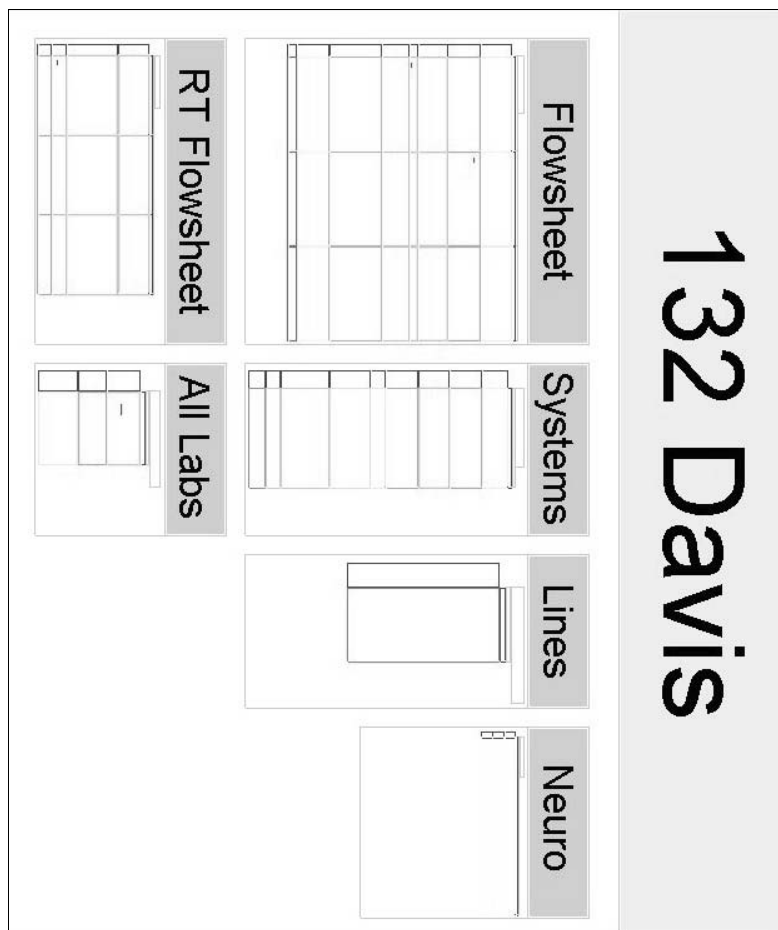


Рис. 6.2. Данные о состоянии пациента можно увидеть, если увеличить масштаб палаты, в которой пациент находится

Процесс увеличения масштаба можно продолжить, чтобы увидеть содержание каждой отдельной таблицы. Когда текст становится достаточно крупным, чтобы его можно было прочесть, появляется возможность его редактировать. На рис. 6.3 показаны автоматически появляющиеся горизонтальная шкала времени и вертикальный набор надписей легенды, которые плавают над фоновым изображением, поэтому если вы перемещаетесь по таблице, шкалы остаются на месте относительно экрана. Кроме того, они автоматически исчезают, если вы уменьшаете масштаб настолько, что когда текст нельзя прочесть, или когда вы увеличите масштаб одной из записей таблицы.

Patient: Davis, Shawn Med Record: 44444444 Physician: Demo, Doctor DOB: 8/7/1942 Admitted: 2/21/1999 Dx: Sepsis Sex: Male												
2/21/1999												
8:00 AM 9:00 AM 10:00 AM 11:00 AM 12:00 PM 1:00 PM 2:00 PM 3:00 PM												
Glasgow Scale												
Eye Opening	3	4	3	3	3							
Verbal Response	3	4	4	4	2							
Motor Response	4	4	5	5	4							
GCS Total	10	12	12	12	9							
Pupils: Size												
R	2mm	2mm	1mm	1mm	1mm							
L	2mm	3mm	2mm	2mm	2mm							
Pupils												
R	Strong	Weak	Norm	Norm	Strong							
L	Norm	Weak	Norm	Weak	Norm							
Pain Assessment												
Score	0	1	unable to eval.	0	unable to eval.							
Location	Neck	Neck	Head	Neck	Head							
Observation	Grimacing	Grimacing	Verbalizes	Intermittent Crying	Calm							
Note 1												
Circulatory/Hemodynamics												
Pulse												
Arterial BP												
S	200	190	189	160	139							
D	120	100	90	80	80							
Mean	146	130	123	113	99							
IABP												
PAP												
S												
D												
Mean												
PCW												
PAP/CVP												
CO												
CI												
SV02												
PVR												
SVR												

Рис. 6.3. Данные в таблице о состоянии пациента теперь видны, и их можно изменять. Заголовки строк и столбцов являются плавающими, поэтому определить смысл значений можно в любом месте

Например, вы можете приблизить одно из значений и получить другие полезные данные (например, нормальный диапазон данного измерения или даже объемный комментарий из медицинской книги). Обра-

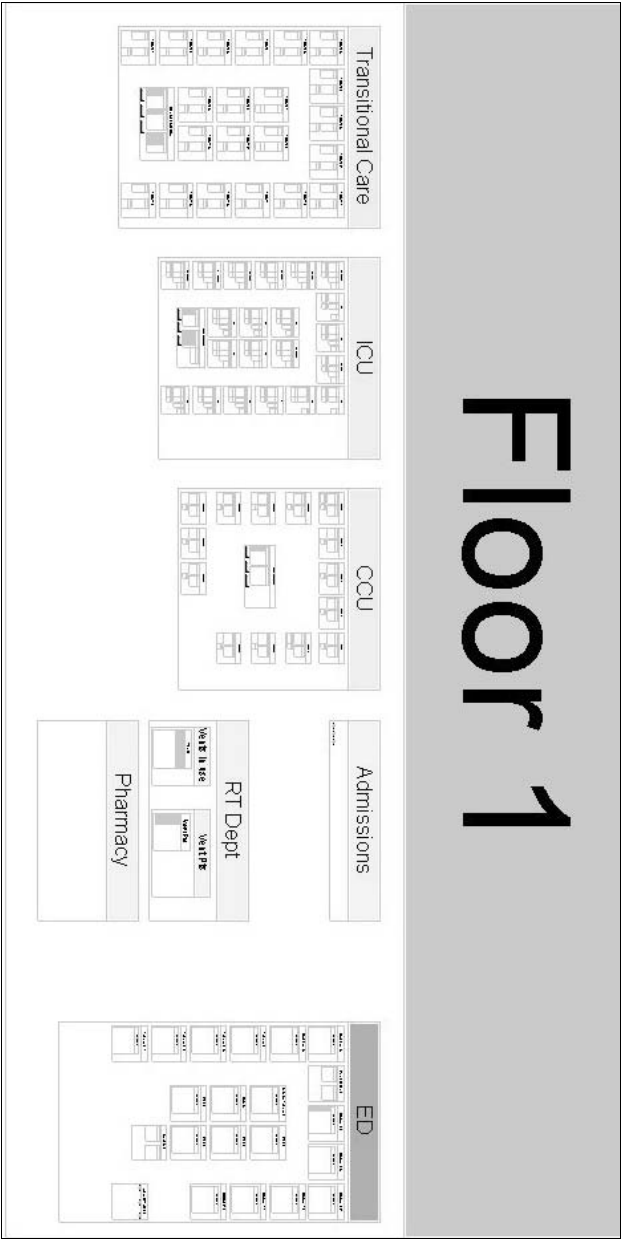


Рис. 6.4. Уменьшенный масштаб изображения на рис. 6.1. Становится видимым весь первый этаж

тите внимание, что хотя эти дополнительные данные и не занимают какого-либо видимого пространства, они, тем не менее, находятся там, где они могут понадобиться.

Изменение масштаба может происходить и в обратном направлении. Если посмотреть выше над тем, что было изображено на рис. 6.1, мы сможем увидеть, что отделение интенсивной терапии находится на первом этаже, на котором расположены и другие отделения. Кроме того, видно, как эти отделения физически расположены друг относительно друга (рис. 6.4).

Если уменьшить масштаб над первым этажом, то можно увидеть всю больницу с одноэтажным сектором приемных отделений и трехэтажным больничным корпусом (рис. 6.5).

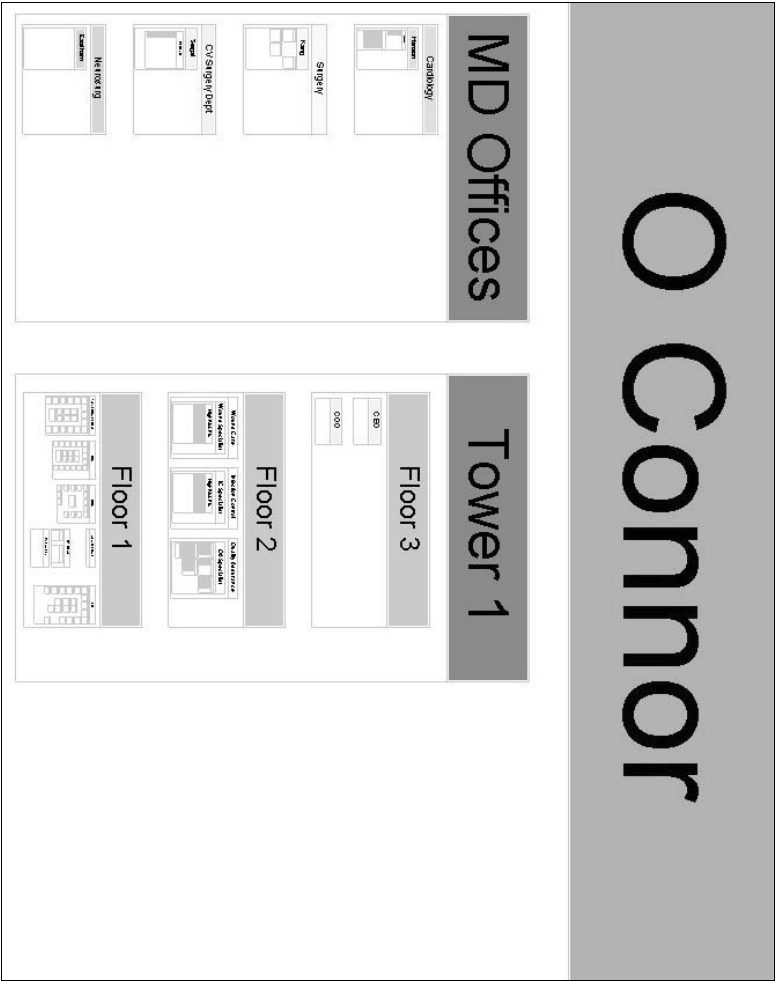


Рис. 6.5. Больница О'Коннор

Если взбираться на еще большую «высоту», то мы сможем увидеть весь комплекс больниц (рис. 6.6), расположенных приблизительно по их географическому местоположению. Перейти к данным о состоянии

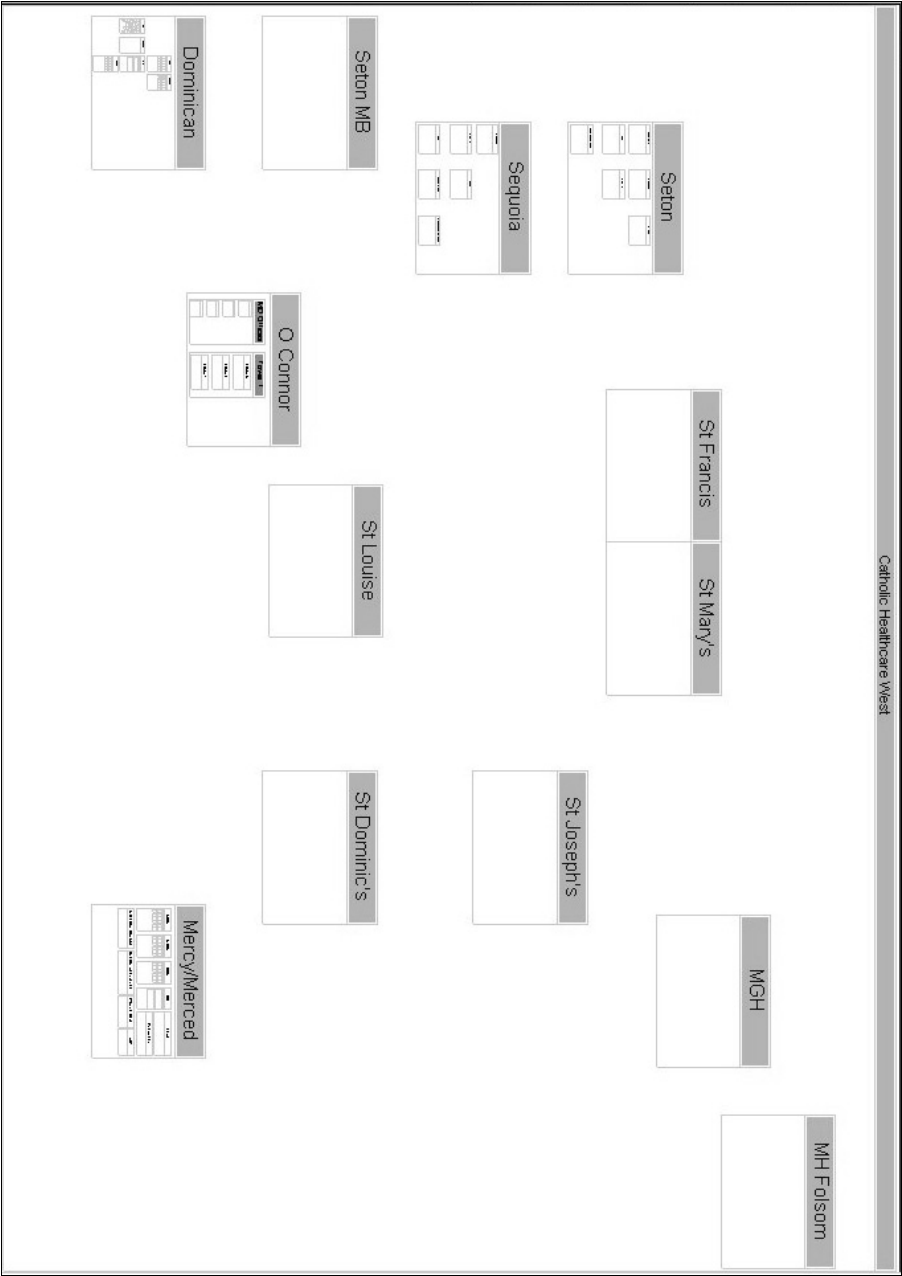


Рис. 6.6. На этом уровне можно видеть весь больничный комплекс

любого пациента в любой больнице можно всего за несколько секунд, а средства для осуществления этого перехода становятся понятными сразу после краткого объяснения. Вы сможете правильно пользоваться этим интерфейсом даже по тому описанию, которое было дано здесь.

Из изображения больничного комплекса в интерфейсе ZoomWorld можно получить не только данные о состоянии пациента. Если вы вошли в систему с соответствующим уровнем доступа, вы сможете увеличить масштаб административного отдела, чтобы проверить бухгалтерские книги, инвентарные записи о лекарствах и поставках, личные дела сотрудников и любые другие стороны предприятия.

Очевидно, что система ZoomWorld может служить в качестве поискового интерфейса для баз данных в предприятиях и объединениях, государственных организациях, школах, каталогах научных данных и т. д. Трехмерные объекты представлены здесь в виде двумерных проекций или других преобразований на дисплее, однако могут быть созданы и трехмерные интерфейсы ZoomWorld.

Хотя перемещения в масштабируемой среде обычно осуществляются с помощью изменения масштаба в каком-то месте, необходима также возможность выполнения небольших изменений для подстройки читабельности. В версии масштабируемой среды, внедренной разработчиком Имэньюзлом Нойком, была предусмотрена возможность задержки начала изменения масштаба. Если сделать движение ГУВ в течение этой короткой (несколько сотен миллисекунд) задержки, вместо изменения масштаба производилось перемещение изображения. Во время тестирования эта возможность хорошо себя показала, хотя, в общем, использование таких задержек может вызывать проблемы (см. раздел 6.4.5): вместо передвижения может произойти изменение масштаба и наоборот. Более быстрый и надежный способ – перемещение изображения путем нажатия обеих кнопок ГУВ. Однако следует отметить, что и тот и другой способы лучше, чем обычный метод перемещения с помощью полос прокрутки, который кроме уже указанных проблем может потребовать от пользователя определения порядка перемещения (сначала по горизонтали, потом по вертикали или наоборот) и затем только способа: либо с помощью кнопок со стрелками, либо с помощью бегунков на полосах прокрутки, либо с помощью клика по полосе прокрутки выше или ниже бегунка. По закону Хика все эти способы приводят к задержкам. Да и вообще процесс прокрутки сам по себе является медленным. Действия пользователя ограничиваются скоростью прокручивания, которая не может быть увеличена, поскольку пользователь не сможет разглядеть, что проскакивает в окне, тогда как изменения масштаба могут происходить быстро, т. к. в этом случае сами изображения не изменяются, и вы не теряете ориентации в своем местоположении. Обратите также внимание на то, что при увеличении масштаба область, в которой вы находитесь, становится ак-

тивной в тот момент, когда вы уже можете с ней работать. Вам не требуется щелкать мышью или каким-то другим образом сигнализировать системе о том, что вы хотите использовать элемент, который уве-



Рис. 6.7. Портал ZoomWorld во Всемирной сети

личили. Как в масштабируемой среде, так и при использовании других методов, описанных в этой книге, понятия «открытия» или «закрытия» документов или приложений уже перестают быть необходимыми.

Если уменьшить масштаб изображения, показанного на рис. 6.6, картина больничного комплекса уменьшится и будет, как остров, окружена пустым пространством. При этом могут стать видимыми и другие «острова», и если увеличить масштаб одного из них, мы увидим, что это портал Всемирной сети (рис. 6.7). Из этого портала мы можем перемещаться в любую область Всемирной сети (рис. 6.8).

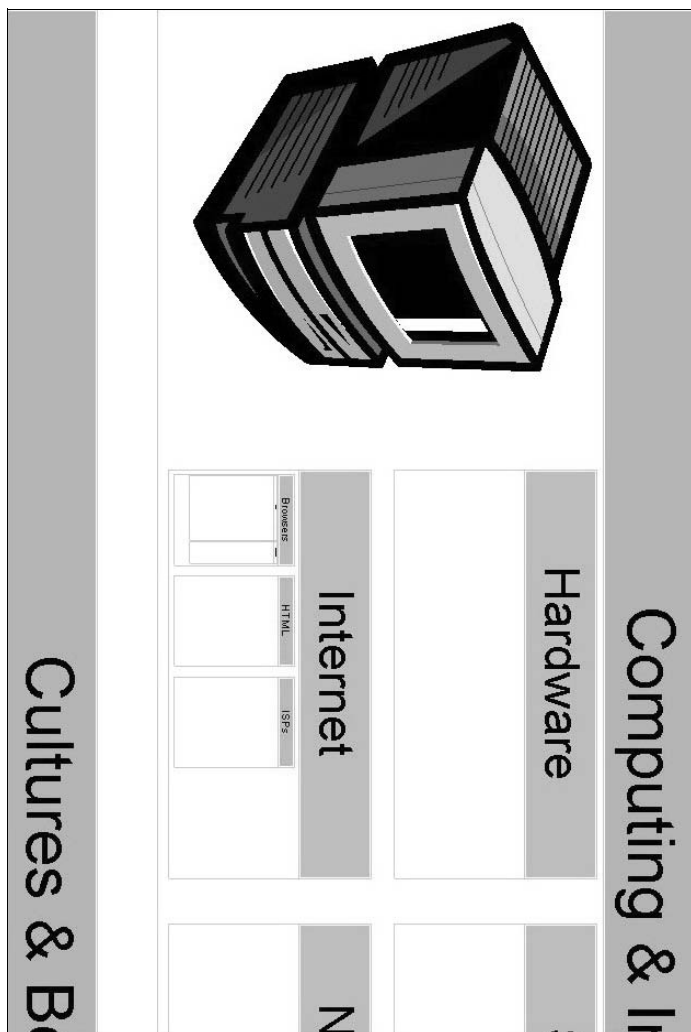


Рис. 6.8. Увеличенный масштаб одной из областей Всемирной сети

Как перемещаться по Всемирной сети с помощью масштабируемого интерфейса ZoomWorld, вполне понятно (рис. 6.9). Другой «остров» — это ваша локальная система вместе с хранящейся в ней информацией, а также всей другой информацией, которая может быть получена без обращения к сетевым ресурсам. При подключении других источников информации (например, DVD) они появляются рядом с «островом» вашей локальной системы.

Очевидный вывод, который можно сделать из вышеприведенного описания, заключается в следующем: *масштабируемый интерфейс может заменить браузер, метафору рабочего стола и традиционную операционную систему*. При использовании такого интерфейса прило-



Рис. 6.9. Увеличенный масштаб списка браузеров. Несомненно, интерфейс типа ZoomWorld сам по себе является превосходным браузером

жения как таковые исчезают. В сочетании с методами, описанными ранее в этой книге, масштабируемая среда ZoomWorld позволяет в целом упростить использование компьютерных систем. Масштабируемая среда, построенная с учетом принципов когнитивной психологии, может полностью соответствовать требованиям, налагаемым когнитивными возможностями человека, что упрощает процесс разработки, изучения и использования в сравнении с современными методами создания программного обеспечения.

Принцип масштабируемого интерфейса позволяет также смягчить другую редко упоминаемую проблему разработки интерфейсов. В наших интерфейсах мы достигли какой-то стильной лаконичности в попытках подобрать одно идеальное слово для обозначения элемента меню и вынуждены использовать непонятные шифровки, чтобы поместить на экране как можно больше надписей. Иногда все же для этой цели лучше использовать несколько слов или даже целое предложение. При общении, конечно, следует использовать как можно меньше слов, но в компьютерных интерфейсах это выходит за рамки возможного, вплоть до непостижимости. При использовании масштабируемого интерфейса экранное пространство становится более доступным, и с ним мы можем позволить большую ясность без экономии пикселей. Конечно, даже без учета принципа масштабируемого интерфейса *мы всегда должны стремиться больше к ясности, чем к краткости в формулировках надписей.*

Существующие инструменты для разработки интерфейсов не позволяют создавать масштабируемые среды. Для этих целей программирование должно проводиться на таких языках, как Perl, C, C++ или Smalltalk. Модель, имитирующая такую среду, может быть создана с помощью Macromedia Flash. Если программисты и дизайнеры, участвующие в проекте, не знакомы с новыми принципами разработки, применяемыми при создании масштабируемой среды или функции LEAP, они имеют склонность в мелких деталях возвращаться к старым методам. Много раз в моей практике случалось так, что когда я возвращался к проекту кого-то из моих клиентов, чтобы посмотреть на его продвижение, обнаруживалось, что кто-то умудрился встроить в интерфейс переключатель или другой какой-нибудь модальный элемент или добавить диалоговые окна. Именно наши старые привычки в разработке являются одной из трудностей с точки зрения сохранения качества интерфейсов такого рода. *Единственным способом удержать интерфейс на верном пути является тщательное изучение и внимание к основам его разработки с учетом когнитивности. Во многих случаях это означает необходимость того, чтобы вся рабочая группа, начиная от менеджеров и заканчивая дизайнерами и тестерами, прошла дополнительные курсы с целью изучения той части когнитивной технологии, которая соответствует специальности работника. Такие курсы должны включать изучение приемов разработки интерфейсов, которые не были рас-*

смотрены в этой книге, но которые известны в среде специалистов по человеческому фактору.

Что касается аппаратного оборудования, то для работы в масштабируемой среде мышь может иметь три кнопки: две кнопки, обозначенные сверху как Выделить и Активизировать, и третью кнопку, обозначенную сбоку как Масштабировать, с помощью поворота которой (по часовой стрелке или против нее, аналогично направлению резьбы в стандартном шурупе) можно было бы увеличить или уменьшить масштаб.

6.3. Пиктограммы

Пиктограммы (icons), эти знакомые всем маленькие картинки, служащие для обозначения кнопок и других объектов, являются неотъемлемым признаком современных интерфейсов. Компания Apple Computer, известный лидер в области разработки интерфейсов, сообщает нам, что «пиктограммы могут существенным образом увеличить ясность и усилить привлекательность приложения. Кроме того, использование пиктограмм позволяет намного упростить процесс перевода программ на другие языки. Всякий раз, когда требуется добавить объяснение или надпись, попытайтесь вместо текста использовать пиктограмму» (Apple Computer 1985, с. I-32). В более поздних версиях этого руководства подход к применению пиктограмм уже не был столь догматичным, однако создаваемый им вред уже нельзя было исправить.

Пиктограммы делают интерфейс более привлекательным в визуальном отношении и, при определенных условиях, могут способствовать большей понятности. Однако со временем стали понятны и недостатки пиктограмм. Например, как в операционной системе Macintosh, так и в Windows сейчас уже используются средства для объяснения значения пиктограмм. Если вы наводите курсор на какую-то пиктограмму, появляется небольшое окно с текстом, в котором дается ее описание. Возникает очевидный вопрос, который я неоднократно слышал от пользователей, впервые встретивших такие текстовые окна: «Почему вместо пиктограмм сразу не использовать текст?» В самом деле почему бы и нет? Ведь, по сути дела, *вместо того чтобы объяснять, пиктограммы зачастую сами требуют для себя объяснений*. Использование пиктограмм вместо слов вполне подходит для того, чтобы скрыть или зашифровать какую-то информацию от посторонних глаз. Проблему пиктограмм можно рассматривать как проблему ограниченной видимости. В интерфейсе показана пиктограмма, но ее смысл невидим, или же ее изображение может дать неверное сообщение для тех, для кого это изображение незнакомо или кем это изображение может быть истолковано по-другому. Например, пиктограмма, изображающая ладонь поднятой руки, в Соединенных Штатах означает «стоп», а в Греции – «вот вам экскременты на ваше лицо» (Horton, 1994, с. 245).

Пиктограммы могут вызвать раздражение не только у специалистов по разработке интерфейсов. По поводу автомобилей часто можно услы-

шать следующий комментарий: «Чтобы понять назначение ручек настройки [автомагнитолы], нужно доставать инструкцию по использованию, так как никаких надписей (вроде надписи «громкость») на кнопках не имеется» (Hotchkiss, 1997, с. 14А).

Иногда использование пиктограмм обосновывается их независимостью от языка, что является ценным свойством для программного обеспечения с точки зрения расширяющегося международного рынка. Однако следует учитывать, что, как было указано, смысл пиктограмм зависит от культуры, в которой они используются. Но даже если пиктограмма и не требует перевода, средства помощи, основанные на использовании языка, и пояснительные окна, объясняющие значение пиктограммы, должны быть переведены, что сводит на нет упомянутое преимущество. Как бы то ни было, используется ли текст или пиктограмма, необходимо, чтобы перевод выполнялся человеком, который является носителем языка, используемого целевой аудиторией, и который знаком с соответствующей культурой, а редакция должна осуществляться людьми, которые выросли и жили в этой культуре. Слова могут быть полностью поняты, по крайней мере, в одном языке. Что же касается пиктограмм, то они могут быть с равной возможностью как поняты, так и нет, причем недостаточное понимание может быть получено сразу в целых языковых семействах.

Приведу пример, когда смысл пиктограммы был понятен только небольшой группе людей. На задней стороне компьютера Apple IIe каждый разъем был обозначен специальной пиктограммой. Одна особенно таинственная пиктограмма представляла собой горизонтальную линию, под которой находилась прерывистая линия той же длины, состоящая из трех отрезков (рис. 6.10). Из приблизительно тысячи человек, которых я опрашивал в основном во время различных конференций, только около 10 человек смогли объяснить значение этой пиктограммы, и еще меньшее число людей смогли рассказать о ее происхождении.

На самом деле эта пиктограмма означает след на экране осциллографа. Осциллограф (рис. 3.5) – это прибор, используемый электротехниками для перевода электрических параметров (например, напряжения) в их видимое отображение. Прерывистая линия, изображенная на рис. 6.10, означает нулевое напряжение, а сплошная линия показывает то, что будет видно на экране осциллографа, если подать на его вход переменное напряжение от источника питания.



Рис. 6.10. Чрезвычайно загадочная пиктограмма

В предисловии к «Книге о пиктограммах» (The Icon Book) ее автор Уильям Хортон (William Horton) говорит: «Около десяти лет я пользуюсь системами с графическими пользовательскими интерфейсами и предпочел бы щелкать мышью по понятным картинкам, чем вводить техническую тарабарщину в виде команд, – даже если бы я мог запомнить, как правильно эти команды пишутся» (Horton, 1994). Из упомянутых Хортоном двух, далеко не самых удачных, вариантов пиктограммы являются, наверное, более предпочтительными, особенно для начинающих или нерегулярных пользователей. Однако Хортон не говорит о другом варианте, а именно: щелкать мышью можно и по кнопке, обозначенной одним или двумя хорошо подобранными словами. Однако необходимо сказать, что Хортон также утверждает, что в некоторых случаях следует использовать как текст, так и пиктограмму, и далее он верно отмечает важность проведения в связи с этим тестирования, независимо от того, используется ли в интерфейсе текст или пиктограмма, или и то и другое вместе. Боб Хорн (Bob Horn) (Jacobson, 1999) создал стиль, в котором текстовые и пиктографические атрибуты образуют комбинированные символы, подтверждающие высказывание, что *текст часто является самой лучшей визуальной подсказкой*. Мы хорошо умеем визуально отличать одно слово от другого, и, в то же время, слова могут передавать сложный смысл. Также имеют значение такие эргономические факторы, как регистр, кегль, цвет и другие атрибуты шрифта.

Мэйхью (Mayhew, 1992) упоминает ряд исследований, посвященных использованию пиктограмм. К сожалению, в большинстве исследований не проводилось сравнений между пиктограммами и надписями. Однако из этих и других исследований мы можем сделать вывод, что пиктограммы являются наиболее эффективными, если количество пиктограмм, которые видно одновременно, не превышает 12. Кроме того, необходимо, чтобы пиктограммы:

- визуально отличались друг от друга;
- хорошо отражали соответствующее понятие;
- имели разумно большой размер (обычно больший, чем могла бы быть текстовая надпись).

Во всех исследованиях, в которых рассматривался данный вопрос, было показано, что смысл пиктограмм труднее понять, чем смысл надписей, особенно при первом восприятии, что противоречит одной из наиболее часто упоминаемых причин использования пиктограмм, а именно их понятность для начинающих пользователей. В графических пользовательских интерфейсах мы часто можем встретить окна с множеством одинаковых пиктограмм, имеющих надписи. Пиктограммы имеют небольшой размер, и их может использоваться множество. Существуют десятки разных видов пиктограмм. В современных компьютерных системах нельзя достичь тех ограниченных условий, при которых пиктограммы могут быть эффективными.

Действительно, пиктограмма небольшого размера занимает меньше экранного пространства, чем надпись. Тем не менее, возникает вопрос: какой ценой? Чем меньше экранная кнопка, тем больше времени требуется на то, чтобы ею воспользоваться, и тем труднее обнаружить ее на экране. Кроме того, пиктограмму небольшого размера трудно сделать различимой. Другой важный момент заключается в том, что пиктограммы требуют больше времени на создание, чем надписи.

Основной проблемой, связанной с пиктограммами, является то, что их зачастую трудно обозначить, – например, в разговоре с другими людьми или в ситуации, когда вы пишете о них или просто хотите дать им словесное определение. Каким образом пиктограммы можно сортировать или классифицировать? Куда их можно поместить в индексированном списке? Например, одна из пиктограмм, которая используется на клавиатуре компьютера Macintosh, выглядит как транспортная развязка типа «клеверный лист». В литературе эту пиктограмму называли и клеверным листом, и пропеллером, и даже кренделем, хотя в руководствах компании Apple эта клавиша называется управляющей клавишей (Command key). Это обстоятельство приводит к естественному вопросу: почему на клавиатуре, на которой имеются клавиши, обозначенные Shift, Option, Return, компания Apple не поместила на эту клавишу слово *Command*? Какое преимущество имеет здесь пиктограмма? Как пользователь может, глядя на нее, определить, что это именно управляющая клавиша? В эту же категорию торжества рыночных свойств над юзабилити может быть отнесена и клавиша с небольшим изображением окна, которая помещена на современную Windows-клавиатуру.

Если быть последовательным, то нажатие на клавишу с подобной веселой картинкой должно приводить к появлению этого символа в активном документе, тогда как клавиши, обозначенные словами, имеют другие функции (клавиши со стрелками в данном случае являются, вероятно, полезным исключением). Мне кажется, что такая «лапша» вряд ли может давать прирост в продажах, однако трудности, порождаемые ею, бесконечны (вплоть даже до того, что для обозначения такого знака в руководствах по продуктам, в которых он используется, в шрифтах должен быть предусмотрен еще один символ).

Попытки описания пиктограмм при помощи слов могут приводить к невообразимым ухищрениям. Возьмем сложный словарь, при помощи которого описываются геральдические символы английской знати. Вот пример геральдического описания щита: «В зеленом поле серебряный на задних лапах лев от золотого поля отгорожен».¹ Неужели та-

¹ Понимать это надо так: щит разделен на две половины – левую (золотую) и правую (зеленую). В центре зеленого поля расположен серебряный геральдический лев, стоящий на задних лапах примерно под 45° к горизонтали; голова льва повернута вправо и вверх.

кой язык понадобится для описания пиктограммы, когда вы позвоните кому-нибудь, чтобы вам помогли заставить компьютер работать, как положено?

Конечно, слова не всегда оказываются подходящим средством. Например, это касается цветовых палитр. В одной из программ на моем компьютере палитра передается с помощью слов, представленных в (главным образом!) алфавитном порядке. Вот часть этого списка:

Cyan (голубой)	Bittersweet (паслёновый)
Blue (синий)	Blue Green (сине-зеленый)
Magenta [именно так – Д. Р.]	Blue Gray (сине-серый)
Apricot (абрикосовый)	Blue Violet (сине-фиолетовый)
Aquamarine (аквамарин)	Brick Red (кирпично-красный)
Burnt Orange (жженный оранжевый)	Gray (серый)
Burnt Sienna (жженная охра)	Green Blue (зелено-синий)
Cadet Blue (синий кадет)	Green Yellow (зелено-желтый)
Carnation Pink (телесный)	Indian Red (индийский красный)
Corn Flower (васильковый)	Lemon Yellow (лимонный)
Forest Green (травянистый)	Maize (бледно-желтый)
Gold (золотой)	Maroon (коричнево-малиновый)
Goldenrod (золотарниковый)	

Какой цвет означает название Bittersweet? А что делать тем, кто не знаком с ботаникой¹ и не может отличить одуванчик от василька? Индекс для библиотеки графических фрагментов часто является более эффективным, если содержит свернутые, пиктографические изображения каждого фрагмента, тогда как список категорий изображений или другие формы высокоуровневой организации должен быть текстовым. Изображение некоторого количества цветков не передает ту же информацию, что и слово «цветы». Это изображение может обозначать «лето», «список торговцев цветами» или «чат-комнату для тех, кто болен сенной лихорадкой».

На рабочем столе моего персонального компьютера типа Macintosh одна из часто используемых пиктограмм обрамлена красной рамкой. В системе имеется опция изменения цвета рамки. Такое цветовое выделение позволяет быстрее находить какую-то из более дюжины пиктограмм, которыми завален мой рабочий стол. Однако я бы хотел, чтобы имелась возможность выделить красным только слово. Хортон (Horton, 1994) отмечает, что «выделение цветом может быть бесполезным, если используется слишком много цветов или если имеется слишком много пиктограмм одного цвета». Хортон также уточняет,

¹ В названиях цветов использованы названия растений – «василек», «паслён», «золотарник». – *Примеч. науч. ред.*

что если пиктограммы расположены близко друг к другу, не следует использовать более семи цветов; если же пиктограммы разбросаны по всему экрану, то число цветов не должно превышать четырех. На мой взгляд, это правильное уточнение, и предложенные им ограничения относятся также и к цветовому выделению слов. Тем не менее, создается ощущение, что очень многие разработчики графических пользовательских интерфейсов совсем не читали Хортонa.

В ранних руководствах, используемых в компании Apple, утверждается, что «для наиболее эффективного использования экранного пространства в приложениях Macintosh графика используется весьма широко даже в тех местах, где в других приложениях применяется только текст. Насколько это возможно, все команды, элементы и параметры приложения, а также все данные пользователя существуют на экране как графические объекты». (Apple Computer, Inc. 1985, с. I-31). Однако с точки зрения пользователя самыми важными качествами являются ясность и простота использования. Тенденция к чрезмерному использованию графики стала препятствием для разработки хороших интерфейсов.

Разработчики интерфейса для одной из телекоммуникационных программ должны были предусмотреть в нем отображение следующих состояний: набор номера, звонок, занятость линии, отсутствие сигнала в линии, чрезмерный шум на линии, попытка соединения, успешное соединение. После нескольких месяцев подбора для этой цели пиктограмм и их тестирования разработчики пришли к выводу, что невозможно создать такой набор пиктограмм, который мог бы однозначно передать пользователю требуемую информацию. Когда ко мне обратился за помощью по поводу этой проблемы, я задал вопрос: «Как можно графически отобразить звук сигнала «занято»?», после чего предложил использовать список соответствующих слов вместо пиктограмм. Выполнив мои рекомендации, они провели тестирование, которое подтвердило эффективность этого решения. То, что эти разработчики не смогли сами прийти к такому решению, вероятно, является результатом того «промывания мозгов», которое делается советами по разработке интерфейсов, наподобие вышеприведенного.

Применение пиктограмм иногда не дает разработчикам увидеть возможность прямого манипулирования. Чтобы выбросить документ, в большинстве интерфейсов вам необходимо переместить пиктограмму этого документа к электронной мусорной корзине или урне. Если же в этот момент вы можете видеть документ, то почему же нельзя переместить сам этот документ? В данном случае пиктограмма документа служит ненужным заместителем и представляет собой лишний уровень абстракции, который пользователю требуется понять и запомнить. Можно поступить даже проще и совсем не использовать пиктограмму мусорной корзины. Документ, подлежащий удалению, просто выделяется, после чего нажимается кнопка Delete. Такой метод позволит

сделать процесс удаления документов монотонным и может привычно использоваться вместе с другими формами удаления.

Пиктограммы вызывают у разработчиков соблазн к поиску образного отображения, модели или аналога некоторой задачи и затем к оперированию с этим образным представлением, а не с самим объектом, который им отображается. В некоторых случаях (см. выше) это приводит к ошибкам проектирования. Иногда пиктограммы вызывают у разработчиков соблазн создать еще больше пиктограмм. Однажды разработчик, который занимался созданием библиографической базы данных, попросил меня высказать свое мнение по поводу пиктограмм, которые были отобраны специалистами его группы. Я ответил, что большинство тех людей, которые пользуются библиографиями, могут читать, и поэтому лучше использовать слова. На рис. 6.11 показаны некоторые из тех нескольких десятков пиктограмм, которые были созданы этими разработчиками. Читателю предлагается самостоятельно расшифровать их значения.¹

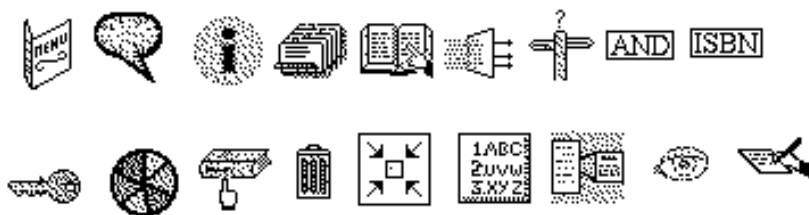


Рис. 6.11. Библиографические пиктограммы (взяты из <http://www.scran.ac.uk/iconstd/>). Вы можете сами убедиться, что многие из них трудно понять, если попытаться определить, какую библиографическую категорию или операцию представляет каждая из них

Самыми понятными из них, наверное, являются те, на которых есть надписи Menu и AND, а также, особенно если вы профессионально занимаетесь книгами, ISBN. Прочитав толкования всех этих символов всего только за два дня до того, как написать этот абзац, я все же не могу вспомнить, что все они означают, и в особенности тот, который выглядит, как пляжный мяч.

Таким образом, как это ни удивительно, пиктограммы нарушают принцип видимости, поскольку именно *смысл* пиктограмм остается

¹ *Ответы:* показать основное меню, изменить язык, показать информацию об этом элементе, просмотр, тезаурус, уменьшить количество полученных результатов, показать информацию о местонахождении текущего объекта (например, URL веб-страницы или область хранения в библиотеке), найти элементы, для которых действительны оба булева выражения, стандартный международный номер книги, ключевое слово, тема, заголовок, убрать текущую информацию, уменьшить обзор текущей информации, отобразить краткий список результатов по текущему поиску, показать все поля, показать результаты текущего поиска, автор.

невидимым. Используйте пиктограммы только в тех случаях, когда исследования показывают, что их применение может быть более эффективным. В других случаях слова лучше.

6.4. Способы и средства помощи в человекоориентированных интерфейсах

Будьте водителем, а не механиком.

Из рекламы автомобилей

Интерфейс с рабочим столом имеет довольно низкую эффективность, так как, находясь на его уровне, вы не можете выполнять свои задачи. В человекоориентированном интерфейсе, в котором нет ни рабочего стола, ни приложений, пользователь может в любой момент работать с необходимым ему содержимым.

Как уже говорилось, мы можем отказаться от использования файлов и файловых имен, оставив только одно пространство – пространство содержимого. В такой среде не требуется открывать или закрывать документы – вы просто уменьшаете масштаб системы до их размера и начинаете работать. Вам не требуется открывать приложения – вы просто копируете чистый документ (или что-то еще). Игру не требуется запускать, вы просто масштабируетесь до нее (причем, если игра многопользовательская, это можно сделать прямо по ее ходу). Для разделения текста на определенные пользователем области используются разделители из набора символов или выбранные пользователем слова и коды, или же разделение достигается с помощью особой формы расположения содержимого.

Как показало исследование, на основе которого были разработаны системы SwyftWare и Canon Cat, начинающие пользователи считают, что системы, в которых нет рабочего стола или файловых имен, очень просты в использовании. Тем не менее, опытным пользователям современных систем эти принципы построения интерфейсов могут показаться настолько необычными, что без некоторых объяснений они могут прийти в замешательство. Крайним примером этой ситуации может быть тот случай, когда компания IBM пыталась рассматривать компьютер Canon Cat на предмет возможного интереса к его интерфейсу. Для выполнения оценки интерфейса компания IBM выбрала двух своих бывших сотрудников (чтобы избежать заключения соглашения о неразглашении полученной информации), которые были довольно опытными пользователями персональных компьютеров. Образ действия (или *modus operandi*), которого придерживались эти эксперты, заключался в том, чтобы определить, смогут ли они быстро изучить интерфейс без помощи руководств, онлайн-овой системы подсказок или пособий.

В конце концов, у них ничего не получилось. Они пытались запустить текстовый процессор с помощью целого набора команд, начиная от ко-

сих черт из языка управления задачами с использованием перфокарт IBM (этот безобразный стиль разделителей, как ни удивительно, сохранился до сих пор и встречается в веб-адресах) и заканчивая командами IBM DOS. Естественно, это приводило лишь к набору и удалению текста, хотя они и делали свои попытки, как им казалось, на «уровне системы».

Интерфейс должен быть по возможности самообучающимся. Это не означает, что интерфейс должен быть интуитивным, а означает, что пользователь может легко найти понятные объяснения или инструкции в тех случаях, когда он в них нуждается.

Для упрощения процесса изучения при первой активизации продукта на экране должен отобразиться экран с текстом инструктивного содержания. Учебник и полное руководство по использованию должны быть частью интерфейса, доступной в любой момент. Справочные экраны должны быть просто частью содержания, и для их использования не должны применяться какие-то особые методы или средства. Для поиска необходимой информации должно быть достаточно лишь использовать функцию LEAP по любому ключевому слову или же несколько раз нажать на клавишу <LEAP Again>. Также можно сначала просто увеличить масштаб справочной области, а затем той ее части, которая вам необходима. Справочные системы в пользовательских графических интерфейсах являются дополнением к основной части интерфейса, внедряются отдельно и работают по своим собственным правилам, что создает дополнительные трудности как для пользователей, так и для разработчиков. Однако куда вы можете обратиться, если вам требуется получить справку о том, как пользоваться справочной системой в обычных пользовательских графических интерфейсах, – в справочную систему справочной системы? Аналогично списку команд, которые были вынесены из области, доступной только для программирования, и стали обычным текстом (см. раздел 5.4), в человекоориентированном интерфейсе со справочными компонентами можно работать без помощи каких-то специальных методов.

Допустим, вы пытаетесь что-то ввести в заблокированный текст

С помощью клавиши <LEAP> или непосредственно мыши вы можете переместить курсор в заблокированный текст. Если теперь вы попытаетесь ввести какой-то текст, как должна реагировать система?

Более или менее традиционным методом реагирования в данном случае является подача звукового сигнала и, возможно, мелькание экрана или панели меню для того, чтобы показать, что вы делаете что-то неправильно; после чего иногда может появиться диалоговое окно с сообщением, что совершается недопустимое действие. Однако такая реакция не подходит для человекоориентированного ин-

терфейса, поскольку выдаваемое сообщение необходимо закрывать, и любые введенные символы теряются, что приводит к нарушению нашего аналога Первого закона робототехники Азимова, суть которого сводится к тому, что все содержание пользователя является священным.

Также неприемлемо, чтобы система изменяла выбранную вами точку ввода на какую-то другую внутри документа или за его пределами. Вы уже решили, что хотите видеть, и машина не должна изменять это решение. Ряд вариантов позволяет не нарушить когнитивные принципы разработки человекоориентированных интерфейсов. Например, экран может разделяться, и в одной его части может отображаться заблокированный текст с курсором, а в другой части может появляться вводимый вами текст, скажем, сразу после заблокированного текста. Другим решением может быть выдача поверх текста полупрозрачного слоя с сообщением о том, что вы пытаетесь изменить текст, который является заблокированным; причем вводимый вами текст может появляться в этом же слое в уже выделенном виде так, чтобы вы могли при желании сразу же перенести его в любое другое место. Как только вы перенесете этот текст, полупрозрачный слой автоматически исчезает.

В большинстве систем передача электронной почты осуществляется с помощью особого приложения. В обсуждаемой здесь системе процесс отправки электронного сообщения состоит из выбора текста, выбора адреса и выполнения команды Send (Отправить). Так как выполнение почти всякой операции начинается с определения выборки и заканчивается назначением какой-то команды, стандартная последовательность действий становится привычной, и поэтому операция по отправке электронного сообщения кажется состоящей из одного шага, а не из трех разных шагов.

Каталог адресов электронной почты в данном случае содержит просто список электронных адресов или любой список, включающий имена и соответствующие адреса. Такой каталог является обычным текстом, а не специальным механизмом. Допустим, что вы хотите отправить электронное сообщение своему дяде Герману. Вы вводите текст и выделяете его. Затем с помощью клавиши <LEAP> переходите к пункту *Герман Джэксон* или к другому имени, под которым ваш дядя фигурирует в списке, и выделяете его электронный адрес. И, наконец, вы вводите или выбираете из меню команду Send и выполняете ее, как было описано выше, с помощью клавиши <Command>. Вместо одного имени, конечно, вы можете выбрать целый список имен для осуществления рассылки.

Другой способ заключается в том, чтобы специальная команда отправки электронных сообщений использовала первую строку выделенного текста как адрес, вторую строку – как тему сообщения, а остальную

часть выборки – как тело сообщения. Какая-нибудь другая команда может использоваться для прикрепления выборки к электронному сообщению. Безусловно, разные разработчики будут предлагать несколько разных методов отправки электронных сообщений, из которых разумный пользователь приобретет только один.

Что касается получения электронной почты, здесь можно предложить несколько необычный метод, который, однако, на практике работает лучше, чем это может показаться сначала. В тот момент, когда приходит электронное сообщение, перед текущей позицией курсора сразу же вставляются два символа документа, между которыми размещается поступающее содержание получаемого сообщения. Естественно, во время приема сообщения вы можете не прерывать свою работу. Никакое специальное окно или сообщение «Вам пришла почта» не требуется. Вы можете игнорировать полученное электронное сообщение или прочитать его в процессе получения. В любой момент, как во время получения электронного сообщения, так и после этого, вы можете выделить его и переместить куда-нибудь. Опытный пользователь, скорее всего, предусмотрит для этого специальное место, обозначив его надписью вроде «E-mail-хранилище» или «Здесь e-mail» или как-нибудь иначе, чтобы сюда можно было быстро перейти с помощью клавиши <LEAP> и в свободное время прочитать полученные сообщения (которые могут сохраняться в этой области автоматически, так же как и оказываться в фокусе по их получении). Ценность этого метода состоит в том, что электронная почта вместе с присоединенной к ней информацией поступает автоматически и видимым образом и становится частью вашего текста, – о том же, как переместить текст в любое место, вы уже знаете. Загрузка присоединенных файлов происходит в фоновом режиме, поэтому работа пользователя не прерывается, а сама информация, во избежание вирусного заражения, может быть помещена в неисполняемую карантинную область. Разработчики также могут предусмотреть команду для сбора всей недавно полученной почты и помещения ее в какую-то область в пользовательской среде.

Шифрование осуществляется через выделение какого-то текста и применение к этой выборке команды шифрования. Зашифрованный текст может быть оставлен на том же месте, переслан по электронной почте или использован другим образом. Расшифровка производится аналогичным способом.

Любые возможности, поставляемые современными приложениями, также соответствуют этой ментальной модели. Рассмотрим электронные таблицы. Если выборка синтаксически построена как алгебраическое выражение, то при использовании команды Calc это выражение сохраняется, и на его месте отображается результат вычисления. Другая команда позволяет выполнить обратный процесс и показать сохраненное выражение, которое при необходимости можно отредактировать и вычислить снова. Полностью функциональную электронную

таблицу можно получить в случаях, если: в выражениях допускается использовать переменные, величинам можно давать имена, синтаксис допускает использование относительных ссылок, ряды данных можно подписывать и выражения можно организовывать в таблицы. Однако это несколько больше, чем просто электронные таблицы, поскольку ссылки на использованные в них значения могут быть вставлены в любом месте внутри текста, и любое значение в тексте можно под определенным именем включить в электронную таблицу. Таким образом, даже без помощи специальных механизмов результаты, помещенные в электронную таблицу из какого-то отчета, могут быть также использованы во введении к этому отчету, а значения, указанные в тексте отчета, могут быть использованы в электронной таблице. В результате между текстом, почтой, электронными таблицами или любым другим типом содержания в этом смысле нет резкого разделения.

6.4.1. Вырезать и вставить

В обычных интерфейсах также имеется и другая проблема, связанная с методом «вырезать и вставить» (cut-and-paste). При использовании этой функции многие пользователи сталкивались с ситуацией, когда какая-то часть работы пропадала после случайного повторения операции вырезания до того, как первый вырезанный отрывок был вставлен. Когда текст удален, он не должен исчезать в небытии, и особенно он не должен перемещаться в невидимый буфер вырезания. Одно из решений может заключаться в том, чтобы помещать вырезанный текст в конце какого-то документа, в котором собираются удаленные элементы. Такой документ может быть самым обычным текстовым документом. (Желательно также, чтобы имелась специальная команда для более радикального удаления.) Важным здесь является то, что с удаляемым текстом не происходит ничего мистического, и для обнаружения документа, содержащего удаленный текст, не требуется использовать какие-то специальные команды или методы. Для этой цели такой документ может начинаться с обычного текста, например: «Этот документ содержит удаленный текст». Пользователь может ввести любую фразу в качестве целевого объекта поиска в этом документе, и, конечно, любая фраза из удаленного текста также может быть использована для выполнения поиска, как будто она и не является удаленной.

Любой человекоориентированный метод удаления

- *не отличается в работе от других команд;*
- *ничего не подвергает риску при удалении или перемещении текста;*
- *не использует особый буфер или другое скрытое место «системного уровня», в которое текст перемещается;*
- *удаляет отдельные символы таким же образом, как и отрывки, состоящие из множества символов;*
- *допускает отмену или повторение действия.*

6.4.2. Сообщения пользователю

Всегда поступайте правильно. У некоторых людей это будет вызывать одобрение, у остальных – удивление.

Марк Твен

Когда вы пытаетесь написать текст сообщения об ошибке, пожалуйста, остановитесь и переделайте интерфейс таким образом, чтобы условие, при котором это сообщение об ошибке вызывается, не возникло. Другими словами, сообщение об ошибке сигнализирует именно об ошибке, но о той, которая имеется обычно в структуре системы или интерфейса, а не совершается со стороны пользователя. В некоторых случаях работа по устранению сообщений об ошибках, которую мы выполняли с моими коллегами, приводила нас к осознанию, что основополагающие проектные решения были неверными и поэтому требуется внести в проект соответствующие поправки. В этом, и только лишь в этом отношении, сообщения об ошибках могут быть весьма полезными. Например, при разработке одного пакета для выполнения арифметических вычислений, на первый взгляд, казалось, что не существует способа избежать выдачи сообщения об ошибке в тех случаях, когда пользователь пытается выполнить деление на 0. Однако более удачным методом является выдача значения, названного «*неопределимо*»¹. В арифметическом стандарте №754 Института инженеров по электронике и электротехнике (IEEE (Institute for Electrical and Electronic Engineers)) для этой цели используется выражение NaN, что означает «не является числом» (not a number). Следует отметить, что арифметические операции, применяемые к *неопределимым* значениям, являются вполне определенными (например, *неопределимо* + 3 = *неопределимо*), а полученные в таких вычислениях результаты являются более полезными и информативными, чем просто остановка вычислений. Использование значения *неопределимо* также позволяет решить проблему, возникающую в тех случаях, когда команда Calculate применяется к объекту, который синтаксически не является арифметическим выражением. В этом случае информация о том, что что-то произошло неправильно, становится локусом внимания пользователя, т. е. искомым результатом. Еще более информативным методом была бы замена значения *неопределимо* на *деление на 0* или другие сообщения в зависимости от конкретного случая. Арифметически они все будут вести себя так же, как значение *неопределимо*.

Другим примером, когда желание устранить сообщение об ошибке повлияло на выбор аппаратного обеспечения, является проект по созданию системы Macintosh. Этот же пример иллюстрирует решение типа «ставка на все состояние», которое часто избирается при создании нового продукта. В тот момент мы пытались подобрать для Macintosh запоминающее устройство. Жесткие диски были тогда слишком доро-

¹ undefined

гими. Накопители на гибких дисках 5,25 дюйма были наиболее широко распространенными, однако ряд других технологий начинал уже вытеснять их. Группа разработчиков системы Macintosh решила использовать накопители 3,5 дюйма, что впоследствии оказалось правильным решением, т. к. остальной мир персональных компьютеров поступил так же. Если бы компания IBM, например, приняла другое решение, пользователям, возможно, было бы труднее достать дискеты для «Мака» (Mac – сокр. от Macintosh).

Тем не менее, наш выбор в пользу накопителя компании Sony был предопределен с точки зрения человекоориентированности интерфейса. В большинстве марок дисководов, которые мы рассмотрели, извлечение дискеты производилось с помощью нажатия на кнопку на корпусе дисковода. К сожалению, в этих дисководах ничего не было предусмотрено для предотвращения преждевременного извлечения дискеты до выполнения сохранения текущей работы. Поэтому требовалось использовать специальное сообщение, которое предупреждало бы пользователя о том, что он совершил ошибку: «Вы извлекли дискету, с которой был считан текущий документ. Для сохранения вашей работы на дискете, пожалуйста, вставьте ее снова в дисковод». Естественно, если пользователь уже отошел от компьютера, унеся дискету с собой, то это создаст трудности для следующего пользователя. Позже я узнал, что существует дисковод без кнопки «Извлечь». Процесс извлечения в этом дисковде выполнялся только по команде компьютера. Теперь, если вы хотели извлечь дискету, то подавали об этом сигнал компьютеру, а компьютер проверял, можно ли выполнить эту команду. Если команда на извлечение диска может привести к потере данных или другим проблемам, система исправляла ситуацию, прежде чем выполнить команду. Итак, мы решили использовать дисководы с самоизвлечением, и в то время, когда царствовали накопители на гибких магнитных дисках, такое решение было одним из многих факторов, сделавших компьютер Macintosh более простым в использовании, чем его конкуренты.

Если выдается сообщение, которое не требует ответа от пользователя, оно может быть устранено. Если по каким-то причинам сообщение непременно должно быть выдано, оно может быть отображено как прозрачный слой, как об этом говорилось в разделе 5.2.3. Если сообщения будут выдаваться в виде прозрачных слоев, лежащее под ними изображение на экране останется видимым, и работа с ним может быть продолжена, как если бы сообщения и не было вовсе. Другими словами, для удаления сообщения с экрана не требуется выполнять никаких специальных действий. Вы можете просто продолжить свою работу, глядя на экран сквозь выданное сообщение. Само же сообщение исчезает при первом выполненном действии (перемещение курсора в данном случае не является действием). В отличие от стандартного диалогового окна, для удаления сообщения с экрана пользователю не требуется выполнять дополнительных действий. Сообщение никогда пол-

ностью не заслоняет информацию на экране, что является общей проблемой современных интерфейсов.

Иногда утверждается, что в отдельных случаях пользователь все же должен ответить на выданное сообщение (например, из юридических соображений). Однако, по моему мнению, в суде это может быть подвергнуто сомнению, ибо пользователи закрывают окна с такого рода сообщениями по привычке (автоматично), и поэтому закрытие окна сообщения не может означать, что пользователь прочитал или хотя бы заметил выданное сообщение (см. раздел 2.3.2).

Сообщения: учебный пример

Одним из интересных решений, полученных в результате устранения набора сообщений об ошибках, является набор методов, разработанных доктором Джеймсом Уинтером (James Winter) для сохранения и загрузки информации из внешней памяти в компьютере Canon Cat. Пользователям графических интерфейсов знакомы сообщения, которые в большом количестве выдаются в процессе сохранения или загрузки информации. Например, пользователю может быть дано предупреждение о том, что он пытается закрыть не сохраненный файл. Изначально я предложил, чтобы имелось две команды для сохранения и открытия данных, каждая из которых подается с помощью специальной клавиши. Как я уже говорил, файловая структура в системе не предусмотрена.¹

Доктор Уинтер показал, что требуется только одна команда и что такой интерфейс будет более безопасным. Моей первой мыслью было, что такая система не будет работать – это обычная реакция на все хорошие и радикальные идеи. «В конце концов, – думал я, – что может быть проще, чем предложенные мной две команды?» Однако метод доктора Уинтера работал именно так, как он говорил. Метод был использован в промышленно выпускаемой модели и оказался вполне успешным и популярным. В частности, были устранены многие ошибки, которые совершаются при сохранении и загрузке информации и зачастую приводят к потере данных.

Для реализации идеи доктора Уинтера должны быть определенные предпосылки. В частности, в компьютере Canon Cat уже не использовалось понятие файлов в привычном смысле этого слова. Вся информация сохранялась в пользовательской рабочей среде. Съемные запоминающие устройства (в то время это были устройства на

¹ В идеале для сохранения данных вообще не должны использоваться специальные команды. Весь материал, включая все его промежуточные состояния, должен автоматически сохраняться, если только пользователь не удалил какую-то его часть намеренно. В то время возможности нашего аппаратного обеспечения еще не позволяли достичь этого.

гибких дисках) содержали то же количество информации, что и память, поэтому вся рабочая среда могла быть сохранена как в памяти, так и на гибком диске. В то время возникла ментальная модель, в соответствии с которой можно было попеременно использовать разные рабочие среды. Эта идея многим пользователям казалась очень удобной. После переключения на другую рабочую среду компьютер фиксировал все внесенные в нее изменения, и каждый диск имел свой уникальный серийный номер, полученный из суммирования имеющихся на нем данных.

Идея Уинтера заключалась в следующем. Использовалась только одна команда под названием DISK. При ее выполнении производился анализ ситуации, и автоматически совершалось необходимое действие. Для того чтобы показать, как работает эта система, Уинтер составил простую таблицу, в которой отразил, какие действия будут выполнены системой при различных условиях применения команды DISK. Далее приводится центральная часть этой таблицы.

Состояние диска	Состояние памяти		
	Не измененное	Измененное	Пустая память
Не измененное	нет действия	сохранение	нет действия
Измененное	загрузка	выдача предупреждения	загрузка
Пустой диск	сохранение (дублирование)	сохранение	нет действия

Если вы загрузили в память компьютера рабочую среду с диска и не изменили ее, состояние памяти остается «не измененным». Если этот диск находится в машине, и вы нажали на клавишу <DISK>, больше ничего делать не нужно – содержание памяти автоматически сравнивается с содержанием диска. Если же диск был извлечен и заменен на другой («измененное» состояние диска), система стирала память и производила загрузку с нового диска, т. к. системе известно, что копия текущей рабочей среды имеется на дискете и поэтому не будет потеряна. В третьем случае дискета может быть чистой (пустой, не использованной), что означает, вероятно, желание пользователя сделать копию рабочей среды, что системой и выполняется. Форматировать дискеты пользователю не требуется, поскольку эта операция выполняется автоматически при необходимости. Если система совершает ошибку в каком-либо из этих случаев, данным не причиняется ущерба.

Не трудно разобраться и с остальной частью таблицы. Выражение «выдача предупреждения» означает извещение пользователя о том, что рабочая среда и содержание дискеты были изменены, и любое последующее действие может привести к потере данных, и что для

сохранения изменений система рекомендует пользователю вернуться к начальному диску или же сохранить рабочую среду на чистом диске. После выполнения одного из этих двух действий нажатие на клавишу <DISK> позволит загрузить новую рабочую среду. Также был предусмотрен способ принудительной загрузки. Кроме того, система автоматически выполняла команду <DISK>, если рабочая среда была изменена, но не использовалась в течение нескольких минут, что увеличивало уровень безопасности в системе.

Вначале Уинтер предложил даже более простой метод, основанный на использовании накопителя, способного определять, вставлен ли в него диск, и извлекать его по программной команде. К сожалению, эти устройства были произведены компанией Sony, и мы не смогли убедить компанию Canon, которая конкурировала с Sony, использовать их, что является одним из примеров того, как пользователей вынуждают страдать из-за корпоративной гордости. Если бы был использован накопитель компании Sony, нам тогда вообще не понадобились бы команды управления дисководом. Действия системы определялись бы только с помощью кнопки «Извлечь» на самом дисковом и по состоянию используемого диска в соответствии с вышеприведенной таблицей. В компании Apple, которая не производила дисководов для гибких дисков, мы смогли бы использовать дисковод компании Sony, но в тот момент, когда я составлял технические требования для проекта Macintosh, я еще не знал о ценной идее доктора Уинтера.

При тестировании в учебных классах учителя часто давали хорошие отзывы о команде DISK. Кроме того, что она почти не требовала времени на изучение и что при ее использовании не тратилось время на форматирование дискет, эта команда также позволяла избежать одной из самых частых ситуаций, в которых происходила потеря данных: если один студент уходил, забыв сохранить свои данные, а другой студент вставлял свою дискету и нажимал на клавишу <DISK>, на экран выдавалось предупреждение. Второму студенту в этом случае приходилось либо бежать на поиски первого, либо сохранять данные на пустом диске, поскольку до тех пор, пока это сохранение не выполнено, второй студент не сможет загрузить данные со своей дискеты!

Команда DISK, разработанная Уинтером, довольно проста в использовании. Все инструкции сокращаются до следующей: если вы хотите выполнить какое-либо действие с участием дисковода, нажмите на клавишу <DISK>. Также эта команда позволяла автоматически сохранять информацию. Концепция, формулируемая как «один диск – одна рабочая среда», также позволила существенно упростить пользовательскую ментальную модель. Компьютер Canon Cat стал неким «окном», с помощью которого можно было как бы заглянуть в дискету и увидеть ее содержание.

Было высказано предположение, что команда DISK создает модальность. Такая ситуация может возникнуть, только если воспринимать команду DISK как комбинацию команд Загрузить и Сохранить, представленную одной клавишей. Однако если вы не знаете ничего об этих командах, используемых в обычных системах, вы можете воспринимать клавишу <DISK> как команду типа «делай то, что нужно сделать с помощью дисководов». В этом случае, как показало тестирование, модальных ошибок не происходит.

Если в вашем компьютере все еще используются дисководы для гибких дисков, то вы, наверное, знаете, что перед использованием дискет их необходимо сначала отформатировать либо приобрести предварительно отформатированные дискеты. В компьютере Canon Cat форматирование дискеты производилось перед сохранением на ней данных таким образом, что, казалось, это не занимает времени, и пользователь мог даже не думать об этом процессе. Так как пользователь не получает никакой пользы от возможности контролировать процесс форматирования, он может даже и не знать о его существовании. Это является еще одним примером, иллюстрирующим следующий принцип: *не предоставляйте пользователю тех средств управления, которые должны работать всегда или никогда.*

6.4.3. Упрощение входа в систему

При входе в большинство систем пользователи выполняют больше работы, чем это необходимо. Сначала требуется сообщить, кто вы, для чего вы указываете свой «идентификатор», «онлайновое имя» или «системное имя», и затем вводите свой пароль. Как предполагается, с помощью имени вы сообщаете системе о том, кто вы, а пароль позволяет предотвратить несанкционированный доступ к вашей учетной записи.

На самом деле вы дважды сообщаете системе о том, кто вы. По логике вещей все, что от вас должно требоваться в этом случае, – набрать пароль. И это не снизит безопасность вашей системы. Вероятность угадывания чье-либо имени и пароля зависит от выбора пароля, его длины и т. д. Тогда как определить чье-либо онлайновое имя можно довольно просто. На самом деле оно обычно не скрывается, чтобы через него можно было взаимодействовать с его владельцем. Плохо подобранный пароль (например, использование в качестве пароля клички вашей собаки) является самой частой причиной недостаточной безопасности.

Утверждение, что ввод двух разных цепочек символов увеличивает надежность, ошибочно. Если онлайновое имя состоит из j символов, а пароль – из k символов, то пользователь для входа в систему должен ввести $j + k$ символов, и только k символов из этой суммы являются неизвестными для потенциального нарушителя вашей безопасности.

Если пароль был выбран случайным образом (что является самым лучшим методом) из набора в q символов, вероятность угадать учетную запись пользователя с первой попытки равна $1/q^k$.

Удлинение пароля даже на один символ и отказ от использования имени уменьшают вероятность угадывания пароля на величину q и избавляют пользователя от необходимости вводить лишние $j - 1$ символов, к тому же в двух полях (вместо одного). Убрав поле для имени и увеличив минимальную длину пароля на один символ, мы получаем большую степень безопасности, используем меньше экранного пространства и достигаем большей простоты использования. Мы не теряем ничего. Для некоторых программ имеет смысл использовать менее базовые методы обеспечения безопасности (как, например, использование «отпечатков голоса» (voiceprinting) или пальцев, или других неизменяемых физических характеристик пользователя), хотя в этом случае вы не сможете сказать своему доверенному лицу, как войти под вашей учетной записью.

Здесь возникает следующий вопрос: как обеспечить уникальность символьных паролей? Как избежать случаев, когда два или более пользователей выберут одинаковый пароль? Можно предоставить системе самой назначать пароли. Однако это приведет к тому, что генерируемые системой пароли будут плохо запоминаемыми (как, например, *2534-788834-003PR7* или *ty6*>fj`d%d*).

Существует много способов создания запоминающихся паролей, и вы всегда можете дать пользователю возможность выбора из пяти или шести таких способов. Например, вы можете предложить компьютеру случайно выбрать два прилагательных и одно существительное из большого словаря и предоставить вам такой список:

- эксклюзивный уродливый тюлень;
- вкусный человекоподобный оракул;
- старая свободная папайя;
- цветущий маленький лабиринт;
- скверная репообразная история.

из которого пользователь сможет сам выбрать наиболее ему понравившийся вариант. В английском словаре возможно около двух триллионов подобных комбинаций. Даже пробуя по миллиону комбинаций в день, вы будете подбирать пароль более 25 лет. Такая безопасность достаточна.

Когда предлагается идея улучшить интерфейс веб-сайта или компьютерной системы, упростив процесс входа запросом только пароля, она обычно отклоняется по одному или двум основаниям. Либо программисты говорят, что так делать нельзя, либо они говорят, что в этом случае они теряют возможность контролировать процедуру входа. Тем не менее, кто-то другой, конечно, все-таки сохраняет этот контроль.

6.4.4. Автоповтор и другие приемы работы с клавиатурой

В настоящее время, скорее всего, рядом с вашим компьютером имеется обычная буквенно-цифровая клавиатура. Множество попыток реформирования клавиатуры (например, раскладка Дворака) оказались бессильными для того, чтобы преодолеть инерцию многих миллионов людей, которые научены вслепую набирать на клавиатуре с раскладкой QWERTY. Все, что мы можем сделать в качестве разработчиков интерфейсов, – это ходить где-нибудь с краю и делать небольшие улучшения, не требующие серьезного переучивания. Приведу некоторые из тех улучшений, которые мы можем предпринять.

Для того чтобы начался автоматический повтор, в большинстве клавиатур требуется удерживать клавишу в нажатом положении в течение 500 мс. Это является примером фиксированной задержки. Однако имеет смысл не использовать в интерфейсах фиксированный интервал задержки. Любая фиксированная задержка может оказаться как слишком большой, так и слишком малой, в зависимости от пользователя и обстоятельств. В данном случае задержка в 500 мс может быть слишком короткой в тех ситуациях, если вы задумались над тем, что собираетесь ввести дальше. Когда вы очнетесь, то можете обнаружить, что на странице появились пара строк с одним символом, наподобие sssssssssssssssssssss. (Мой кот является мастером в использовании компьютера таким образом.) Для пользователя, имеющего небольшую скорость набора или страдающего от какого-нибудь неврологического или физиологического расстройства, задержка в 500 мс перед автоповтором может также быть слишком короткой.

С другой стороны, интервал задержки в 500 мс является слишком длинным, хотя бы потому, что задержка есть задержка – пользователю приходится ждать, пока возникнет необходимый эффект. Например, у пользователей интерфейса Macintosh следующая ситуация вызывает особенное раздражение: чтобы изменить имя файла после открытия тома или папки, вам требуется щелкнуть по имени и подождать полсекунды до тех пор, пока не появится специальное обрамление или изменение цвета, указывающее на переход в режим редактирования. Это было предусмотрено для того, чтобы пользователь мог выбрать имя файла одним нажатием на кнопку мыши без риска случайного его изменения. После перехода в предварительный режим редактирования вы должны щелкнуть по имени еще раз, чтобы перевести систему в состояние редактирования. Раздражение пользователей подтверждается как проведенными интервью, так и неоднократными статьями в журналах с описаниями способов обхода этих задержек. Пользователи не любят, когда их вынуждают ждать.

Джон Бумгарнер (John Bumgarner), работавший в компании Information Appliance, предложил хорошее решение проблемы автоповтора. Он заметил, что в большинстве фонетических языков одна буква поч-

ти никогда не встречается три раза подряд. Он также заметил, что автоповтор редко используется, если букву требуется повторить менее пяти раз (в этом случае пользователь просто нажимает на клавишу необходимое число раз). При использовании метода Бумгарнера автоповтор начинается, если клавиша удерживается более 100 мс после третьего подряд нажатия на клавишу. Другими словами, чтобы получить строку, состоящую из знаков равенства, требуется нажать следующую последовательность:

= = =↓

После этого клавишу со знаком равенства следует удерживать в нажатом положении до тех пор, пока не появится необходимое количество символов, и затем отпустить.

Многократное нажатие на одну и ту же клавишу выполнить быстрее, чем набирать разные символы, и GOMS-анализ показывает, что задержка перед началом автоповтора падает с 700 мс при обычных методах до 400 мс в предложенном методе. Метод автоповтора Бумгарнера достаточно прост в использовании и, как показали тестирования, такой автоповтор никогда не запускается случайно (даже если ваша кошка сядет на клавиатуру). Отрицательной стороной, которая свойственна также и стандартному методу автоповтора, является то, что автоповтор работает как невидимая функция, которая нигде в системе не обозначена.

Хорошо разработанные компьютеры и информационные устройства снабжаются **аккордными клавиатурами** (chord keyboards), чтобы в программном обеспечении можно было предусматривать распознавание одновременного нажатия нескольких клавиш. Старые и более примитивные компьютеры имели клавиатуры, в которых только некоторые специальные клавиши (например, <Shift>) можно было одновременно нажимать вместе с другими клавишами. Аккордные клавиатуры позволяют решить ряд сложных интерфейсных проблем. Например, рассмотрим наложение символов. Требуется логически ясный метод создания двух символов в одном месте. Например, для того чтобы сделать знак доллара с помощью наложения буквы s и вертикальной черты (|), должна быть предусмотрена возможность одновременного нажатия клавиш:

s↓ |↓ |↑ s↑

Это не мешало бы совмещенному нажатию на клавиши, которое обычно происходит при большой скорости набора и при котором клавиша, нажатая первой, отпускается только после того, как нажимаются одна или несколько других клавиш. Слово *the* часто набирается не так:

t↓ t↑ h↓ h↑ e↓ e↑

а вот так (приведем один из множества возможных вариантов):

t↓ h↓ e↓ t↑ h↑ e↑

Современные клавиатуры и их программное обеспечение допускают использование таких совмещенных нажатий клавиш, что называется циклическим буфером (rollover). Большинство клавиатур имеют ***n*-клавишный циклический буфер**. Это означает, что система сможет различить *n*-е количество одновременно нажатых клавиш. С учетом человеческой анатомии коэффициент *n* вряд ли должен превышать 10, хотя с технической точки зрения вообще нет надобности его ограничивать, если компьютер оснащен аккордной клавиатурой.

С учетом общепринятого метода создания наложений с помощью нажатия на одну клавишу во время того, как удерживается другая, знаки ударения и диакритические символы могут также рассматриваться как налагаемые символы и вводиться аналогичным образом. Например, в компьютере Macintosh букву *й*, как в слове Dupй, можно набрать с помощью сложной последовательности клавиш:

Option↓ *e*↓↑↑ *e*↓ *e*↑

Обратите внимание на то, что здесь используется модальный метод типа «глагол–существительное», что является нарушением собственных принципов компании Apple. Кроме того, этот метод работает непоследовательно. Если вы используете нижеприведенную последовательность, то будет получена кавычка, за которой стоит буква *t*, а не буква *t* со знаком ударения, как можно было бы ожидать¹:

Option↓ *t*↓↑↑ *t*↓ *t*↑

Если же наложение выполнять с помощью квазирежима, ввод знаков акцента и других диакритических символов упрощается и делается более последовательным:

e↓ '↓ '↑ *e*↑

Вы нажимаете на букву *e* и, удерживая ее, нажимаете на символ ударения. То же самое сочетание можно получить и в обратном порядке:

'↓ *e*↓ *e*↑ '↑

Логически нет никакой разницы, в каком порядке вы выполняете эту операцию.

Кроме того, наложение символов полезно для написания математических и других специальных символов, а также в языках программирования (например, APL). Может возникнуть вопрос: почему вместо наложения символов нельзя просто включить нужные символы в шрифтовые наборы, ведь наши дисплеи являются полностью растровыми? Действительно, так можно поступить, однако не все захотят тратить время на разработку новых символов и их установку в каждый шриф-

¹ Если вы используете *Option*↓ *e*↓↑↑ *t*↓ *t*↑ для того, чтобы напечатать *t* с ударением, вы опять будете неправы.

товой набор, в котором мы хотим их использовать. С другой стороны, вряд ли в современном компьютере нельзя получить те же возможности, которые были легко доступны с помощью обычных механических печатных машинок.

Наложение не должно ограничиваться только двумя символами. Любые символы могут налагаться друг на друга, как, например, в следующей последовательности:

Shift↓ *s*↓ *Shift*↑ | ↓ / ↓ / ↑ | ↑ *s*↑

Такая последовательность даст в результате знак доллара, перечеркнутый кривой чертой. Функция наложения символов должна ограничиваться скорее только лишь эстетическими соображениями и доступностью для чтения, чем аппаратными или программными соображениями.

Если используются *n*-клавишный циклический буфер и описанные выше методы наложения, то для обратной связи во время набора интерфейс может временно отображать пару налагаемых друг на друга символов в виде смежных символов. Смысл этого заключается в том, что интерфейс не может отличить одновременное нажатие клавиш при быстром наборе от одновременного нажатия с целью наложения символов друг на друга до тех пор, пока клавиши не отпущены, после чего слияние накладываемых символов происходит автоматически. Также хочу добавить, что требуется радикальная реформа клавиатуры, связанная с удалением клавиши <CapsLock>. Эта клавиша порождает режим.

6.5. Письмо от одного пользователя

Когда я работал над проектом для большой компании, один опытный пользователь программного обеспечения, производимого этой компанией, написал письмо, которое иллюстрирует некоторые идеи этой книги. Приводимые ниже высказывания в кавычках взяты из этого письма.

- «Этот программный пакет показался мне более развитым продуктом». В разговоре с программистами выяснилось, что приоритет был отдан больше плану выполнения работ, чем качеству. Поэтому покупателям была предложена скорее мечта руководителей изначального проекта. Скорее всего, в условиях жесткого плана работы был создан «минимальный полезный краткий вариант». Многие важные детали были пропущены, т. к. инструменты для разработки интерфейса были выбраны заранее и поэтому не дали возможности реализовать задуманные формы взаимодействия с пользователем.
- «Пользователь должен нажимать клавишу <Enter> или щелкать кнопкой мыши намного чаще, чем это требуется для ввода полезной информации». При вводе данных в поле *нет* необходимости в

том, чтобы пользователь нажимал клавишу <Enter> или <Return> или вообще что-либо делал еще. *Когда пользователь переходит к следующему полю или окну или использует меню или кнопку, система должна автоматически принять введенные данные.*

Использование клавиши <Tab> вместо клавиш со стрелками для перемещения по полям также создавало проблемы. Два поля на экране допускали свободное форматирование текста. В этих полях пользователь мог применять клавишу <Tab> для создания отступов или списков, и поэтому клавиша <Tab> не давала возможности перейти на следующее поле. Тяжело было смотреть на пользователя, который много раз и безуспешно нажимал на клавишу <Tab>, чтобы попытаться перейти на следующее поле.¹

Эти примеры иллюстрируют две распространенные проблемы в интерфейсах. Первая связана с использованием клавиши <Return> для разделения полей. Эта привычка уходит далеко в то время, когда несколько десятков лет назад в прикладных системах, работающих в режиме разделения времени, а также в приложениях для микрокомпьютеров использовались ограничения, принятые для телетайпных машин. Вторая проблема связана с функциональной перегрузкой клавиш <Return> и <Tab>, в результате которой в полях, допускающих свободный ввод текста, они означают одно, а в более коротких полях – другое.

- «Когда выбирается опция поиска, курсор должен появляться в соответствующем текстовом окне так, чтобы пользователь мог начать вводить информацию без необходимости щелкать мышью внутри поля или нажимать на кнопку Tab». Это частный случай следующего общего принципа: *если пользователь в следующий момент может выполнить только одно действие, пусть это действие выполнит компьютер.*
- «Ненужные диалоговые окна, наверное, являются главной причиной бесполезной траты времени и вызывают раздражение у пользователя». Речь идет о тех диалоговых окнах, которые предназначены для сообщения пользователю о том, что произошло, и для своего закрытия требующие нажатия кнопки мыши или клавиши <Enter>. Другого выбора не остается – продолжить можно, только лишь кликнув по окну. Это другой частный случай приведенного выше принципа (если пользователь далее может выполнить только одно-единственное действие, пусть его выполнит компьютер). Как пишет автор в другом месте своего письма: «Важно, чтобы всякий раз, когда пользователь должен взаимодействовать с каким-то диалоговым окном, это взаимодействие давало полезный результат». Это можно обобщить до следующего утверждения: *каждый раз, ког-*

¹ Возможно, клавиша <Следующее поле> (Next Field) была бы полезным дополнением к современным клавиатурам.

да пользователь должен вступить во взаимодействие с компьютером, это взаимодействие должно предполагать получение полезного результата. Перемещение к следующему шагу в работе само по себе не является полезным результатом.

Далее автор письма продолжает сетовать, что одно из диалоговых окон просто «сообщает пользователю, что элемент уже введен в список» в том случае, когда название или номер существующего предмета вводится в окно для новых элементов. Чтобы продолжить, вам требуется убрать это диалоговое окно. Вместо этого автор предложил, чтобы появились три кнопки: оставить элемент, удалить элемент из списка или перейти к его редактированию. Хотя вариант автора является лучшим, чем изначальный, мы все же можем предложить еще более лучшую схему. Описанная проблема отчасти связана с идеей, что ввод нового элемента отличается от редактирования или удаления элементов. Предложим более простой метод: пользователь вызывает форму и вводит дескриптор элемента. Если он является новым, элемент вводится, и пользователь продолжает работу, как предполагалось. Если элемент уже имеется в списке, данные о нем сразу же вызываются, чтобы пользователь мог увидеть, что элемент уже существует. После этого пользователь может редактировать их. Естественно, удаление – это один из способов редактирования.

- Автор отмечает, что экран очень быстро заполняется одинаковыми пиктограммами, которые можно различить только по именам, указанным внизу каждой из них. Он предложил, чтобы пиктограммы различались между собой в большей степени, поскольку «среда, по сути, является *визуальной*». Автор письма прав в том, что если экран заполнен множеством одинаковых пиктограмм, то от пиктограмм мало толку – они только занимают место. Он предложил, чтобы использовались четыре пиктограммы. Однако следует заметить, что если будут использоваться только четыре пиктограммы, на экране все равно будет слишком много одинаковых пиктограмм. Решение заключается в том, чтобы понять, что пиктограммы на самом деле можно не использовать. При создании графических интерфейсов мы должны помнить, что *текст тоже может быть визуальной подсказкой*. Текст – это очень сильная подсказка с довольно подробным содержанием, которое мы все можем легко понять (см. раздел 6.3).
- «Если вы открыли окно с формой для заказа на покупку и хотите внести в него какой-то элемент, перед вами открывается диалоговое окно со следующим содержанием: «Данное приложение не может работать одновременно с окном «Создать/обновить бланк заказа». Естественной реакцией пользователя может быть вопрос: «Почему не может?» Здесь разработчики просто не имели полного представления о том, как в действительности может проходить работа пользователя.

Общий принцип состоит в том, что почти любой чрезмерно структурированный подход к процессу взаимодействия с пользователем рискует стать препятствием при выполнении им той или иной задачи, которая требует совсем другого подхода. В данном случае интерфейс превращается из помощника для пользователя в диктатора. *Компьютер должен быть слугой для пользователя. Он не должен быть равным человеку или быть его начальником.*

- «В электронной промышленности существует тенденция к согласованию, независимо от того, насколько это может быть продуктивным... Согласование и использование стандартов очень важно, т. к. позволяет пользователю быстрее работать. Но если согласование и стандартизация создают бесполезные вещи, то такой проект можно считать неудачным». Именно это несколько лет назад и было сказано в известной статье Грудина (Grudin) «Дело против согласования пользовательских интерфейсов» («The Case Against User Interface Consistency», Grudin, 1989). Очевидно, что анализ, проведенный Грудиним, не был воспринят электронной индустрией. *Следует отказаться от стандарта, если он явным образом снижает продуктивность или является неудобным для пользователя.*
- «При разработке этого программного обеспечения использовался стандартный метод построения меню, принятый в операционной системе Windows». Автор приводит пример: «Все меню изначально содержат в себе какую-то долю бесполезности. В одном из меню содержится команда Выход, которая вставлена в меню Файл». Другими словами, автор говорит, что команда Выход была *единственной* командой в том меню. «Команду Выход необходимо поместить в строку главного меню, а меню Файл следует убрать». Как автор пишет в своем письме, «список не может состоять из одного элемента». *Бессмысленна ситуация, когда необходимо открывать меню, в котором нельзя сделать выбор.*
- Автор письма делает много конкретных предложений по улучшению некоторых деталей, тогда как в этих случаях требуется более серьезная переделка. Например, при формировании заказа на покупку определенного товара пользователь сначала получает окно под названием «Ввод заказа на покупку (Добавить)». В этом окне пользователь должен указать количество. Значение количества, принятое по умолчанию, равно 0, и, как указывает автор письма, «значение по умолчанию должно быть равно 1, поскольку вряд ли кто может сделать заказ на нулевое количество». Автор, конечно, прав, но, по моему мнению, все это окно является ошибочным. Пользователю должен быть представлен список элементов, который он может прокручивать и выполнять в нем поиск. В этом списке пользователь может просто изменять значения количества, и тогда нулевое значение по умолчанию становится необходимым. В некоторых приложениях в списке должны сохраняться значения,

выбранные пользователем (например, в последний раз), в качестве начальных значений для последующих случаев использования. В зависимости от необходимости может быть предусмотрена кнопка для обнуления всех значений. Такая кнопка должна быть ясно и отчетливо выделена.

В этой системе также имеется окно под названием «Ввод заказа на покупку (Убрать)». Это окно является ненужным: установка нулевого значения напротив элемента списка автоматически удаляет этот элемент из заказа на покупку.

Кроме того, эти окна содержат и другую бесполезную условность. В них имеются кнопки с указанными внизу обозначениями *Сохранить* и *Выход*. Большинство пользователей не могут точно сказать, что эти кнопки делают. Кнопка *Сохранить* выполняет сохранение и выход (в этом случае она так и должна быть названа: *Сохранить и выйти*) или пользователь должен нажимать их по очереди для того, чтобы выйти, предварительно сохранив введенные данные? Если выйти без сохранения, будет ли выдано предупреждающее сообщение наподобие «Хотите ли вы сохранить изменения перед выходом?» или данные будут потеряны? В любом случае наличие двух кнопок является бесполезным. Если вы перемещаете курсор за пределы этого окна и приступаете к какому-то другому делу, система не должна мешать вам и автоматически сохранить содержание предыдущего диалогового окна.

- Если покупатель тратит свое время на тщательный анализ вашего продукта и делает конструктивные предложения для его улучшения, обязательно отнеситесь к этому с вниманием! Это нельзя рассматривать как попытку сделать вам выговор или нанести оскорбление. Такой человек не является вашим врагом. Этим он демонстрирует свою лояльность и интерес к вашему продукту.

7

Проблемы за пределами пользовательского интерфейса

Утверждение, что следует развивать привычку думать о том, что мы делаем, часто повторяется в учебниках и в речах известных людей и является абсолютно ошибочной избитой фразой. Верно совершенно обратное. Развитие цивилизации связано с увеличением числа важных операций, которые мы можем выполнять не задумываясь.

Альфред Норф Уйтхед

В этой главе будет представлено попури из случаев применения нестандартного мышления (или не-мышления), которые могут быть полезными для разработчиков в разных областях технологии. В разделе 7.1 будет затронута проблема, связанная с тем, что среды программирования имеют, наверное, самые худшие интерфейсы в компьютерной индустрии. Мы рассмотрим два аспекта, в которых могут быть сделаны улучшения. Один из них заключается в том, что сложность системного окружения и сред программирования достигла таких высоких уровней, что это отбивает у начинающего программиста стремление к экспериментированию и изучению среды в процессе работы. Второй аспект связан с тем, что хотя всем известно, что документирование является полезным, оно по большей части не выполняется. Небольшое изменение в языках программирования могло бы позволить упростить весь процесс.

В разделе 7.2 рассматривается проблема избытия кабелей (или шнуров), которые растут из наших компьютеров подобно змеям из головы медузы Горгоны. Зачастую оказывается трудно подобрать кабель требуемого типа, с необходимым переходником, вилок или разъемом.

Эту проблему можно было бы решить, если бы кабели не имели разные концевые соединения типа «папа» и «мама», а каждый кабель, предназначенный для определенной функции, соединялся бы с любым другим кабелем или разъемом в компьютере, что вполне осуществимо.

В разделе 7.3 рассматриваются вопросы этики. При создании интерфейсов разработчик находится в близком и продолжительном контакте с разумом и телом пользователя. Это влечет за собой определенную ответственность. В учебных программах и тренингах для специалистов в области разработки интерфейсов уже много было сказано о том, что должно быть предусмотрено для защиты интересов пользователей. Но мало кто говорит о том, какие охранные меры и какая социальная защита требуются для того, чтобы дать возможность компетентному разработчику делать свою работу хорошо.

7.1. Более человекоориентированные среды программирования

7.1.1. Системное окружение и среда разработки

Исследования в области когнетики для сред программирования нашли еще меньшее применение, чем для пользовательских интерфейсов. Нечего и говорить, что современные системы становятся все более сложными и что инструменты программирования должны соответствовать этой сложности. Простые вещи стали излишне сложными, и мы не смогли пока создать достаточно хорошие инструментальные программные средства для облегчения этих сложностей работы в современных компьютерных средах.

Я начну с простого примера. В давно исчезнувшем компьютере Apple II для того, чтобы написать программу сложения двух чисел, требуется включить компьютер (время загрузки не заметно!) и нажать <Control>+, после чего вы переходите в BASIC. Если вы теперь наберете PRINT 3+4 и нажмете <Return>, то сразу же и без трудностей получите 7. С момента запуска BASIC до получения результата прошло 5 с. Как хорошо известно, в компьютерной промышленности простота использования достигается с помощью довольно больших ресурсов памяти и скорости. Поэтому мы понимаем, что компьютер Apple II способен выполнять вычисления с такой быстротой и простотой именно потому, что он обладает мощнейшими ресурсами: имея 2-мегагерцовый 8-битный процессор, 48 Кбайт памяти (и это все, что можно вместить!) и 400 килобайтовый диск, машина работает как зверь. В 1999 году на выполнение этой операции у компьютера с 400-мегагерцовым 32-битным процессором, 192 мегабайтовый RAM-памяти и несколькими гигабайтами памяти на жестком диске уходит более 3 минут. Судя по разрядности системной шины и тактовой частоте процессора, новая машина работает приблизительно в 1500 раз быстрее, чем старая. Если же оценивать по времени, которое требуется для написания

программы, новая машина оказывается медленнее приблизительно в 36 раз.

Я попросил двух профессиональных программистов написать программу на Visual Basic (VB), которая бы выполняла сложение $3 + 4$ и выдавала результат на экран. Первый программист начал жаловаться, что у машины всего только 8 Мбайт памяти и что у нее устаревший 75-мегагерцовый 32-битовый процессор. Не считая времени загрузки (2 мин.), среда программирования была открыта через 54 с. После этого требовалось открыть модуль вставки (Insert Module), потом открыть окно опций (Option Box) и установить соответствующие настройки, создать кнопку и рабочую форму, после чего программист должен был набрать среднюю строку из следующего текста:

```
Private sub Command1_Click ()  
    MsgBox 3 + 4  
End Sub
```

Для того чтобы воспользоваться этой программой, ее нужно было сначала запустить, а потом нажать кнопку для ее включения. В течение этого процесса, который занял 3 мин. 40 с (опять же, не считая времени начальной загрузки), было сделано всего только две или три ошибки.

Другой программист, работавший на 64-битовом процессоре с тактовой частотой 75 Мгерц и 40 Мбайт памяти, запустил VB и выполнил ту же задачу за 28 с (что приблизительно в 5 раз медленнее, чем на компьютере Apple II). Программа, созданная несколько иным способом, была следующей:

```
Private sub Form_Load ()  
    MsgBox Str (3 + 4)  
End Sub
```

Я спросил у этого программиста, почему он не написал вторую строку так же, как ее написал первый программист:

```
MsgBox 3 + 4
```

Он ответил, что не был уверен, что это будет работать. Другими словами, он не знал точно, как VB будет работать в этом случае. Здесь нет ничего странного: как и другие современные компьютерные языки, VB имеет довольно сложное и непоследовательное построение. Оправданием его громоздкости может быть то, что он позволяет делать большие проекты проще, однако это не может быть оправданием для того, чтобы делать простые вещи сложными. Большие вещи состоят из множества малых, поэтому чем проще сделаны составные задачи, тем проще становится вся задача в целом. Именно плохая организация системы и языка является причиной того, почему один опытный программист допустил ошибки, а другой – не был уверен в правильности синтаксиса простой программы. Те же самые результаты я получил и с

тремя другими программистами, работающими со Smalltalk; это показывает, что данные проблемы относятся не только к VB. Очевидно, что каждый из этих языков обладает множеством преимуществ, но если бы они и особенно их среды были хорошо разработаны с точки зрения человеческих факторов, эти преимущества достигались бы с меньшими неудобствами и меньшим числом ошибок со стороны человека.

Таким образом, было утрачено нечто удивительно непосредственное, в частности, немедленная обратная связь, в которой человек нуждается для того, чтобы иметь возможность быстро создавать эффективные программы. Я не настолько наивен, чтобы думать, что мы можем вернуться к прежней простоте и достичь того уровня сложности, который требуется от современных программ. Но я уверен, что мы можем сделать в этом много улучшений.

7.1.2. Важность ведения документации при создании программ

Во многих источниках сообщается, что для программистов важно подробно документировать машинный код, который они пишут. Для этого обычно приводятся две причины: во-первых, чтобы помочь понять программу при ее чтении (Knuth, 1992, с. 99), и, во-вторых, чтобы упростить адаптацию программы к новым условиям (Weinberg, 1971, с. 164). Обычно в программе рядом со строками кода можно встретить комментарии (чаще однострочные). Многие программы бывают почти полностью лишены комментариев.

Как отметил Кнут (Knuth), написание комментария *до* или во время создания кода может помочь процессу написания, улучшить структуру алгоритмов, снизить число ошибок и повторных написаний программы, необходимых для завершения проекта, а также дает другие преимущества, которые обычно упоминаются по поводу комментариев. По всей видимости, правильность выводов Кнута имеет основания с точки зрения когнитивной психологии.

Когда мы как опытные программисты разрабатываем алгоритмы и пишем код, этот процесс отчасти происходит на бессознательном уровне. Как уже было отмечено, эта ментальная область может быть подвержена противоречиям. Предполагаю, что причина некоторых программных ошибок заключается в том, что когнитивное бессознательное переживает противоречие между тем, что мы хотим сделать, и тем, что должен сделать компьютер в соответствии с нашим кодом.

Однако для того, чтобы ясно выразить свои намерения на естественном языке, нам следует сделать процесс мышления сознательным, и именно в когнитивном сознательном эти противоречия быстро становятся очевидными. Даже если эта гипотеза неправильна, написание комментариев вынуждает нас тщательнее обдумывать решаемую задачу для разных условий и с разных точек зрения.

К сожалению, среды программирования были разработаны таким образом, что вносить комментарии в создаваемые программы не просто. Например, комментарии во многих языках программирования ограничиваются одной строкой, а если многострочные комментарии допустимы, то в них часто не используется перенос текста или другие возможности, которые должны быть в простейших текстовых редакторах (исключения составляют UCSD Pascal и Oberon). Чтобы в языке Visual Basic, появившемся в 90-х годах, написать комментарий размером с абзац, вам приходится вручную делать перенос текста. По легендам, программисты не особенно грамотно умеют писать, поэтому проверка орфографии должна быть включена в каждую среду программирования, однако в программных средах эта служба почти никогда не встречается. В существующей версии Mathematica, которая является в общем превосходной программой для работы с математическими выражениями, комментарии были убраны из программ и помещены в специальное окно, что является совершенно неправильным шагом. Только некоторые системы (как, например, созданная Кнудом программа WEB (1992)) были разработаны таким образом, чтобы способствовать ведению документации. Другой, менее сложный, но достаточно эффективный метод, был использован автором этой книги (Lammers, 1986, с. 226).

Для сохранения работоспособности программы любым вносимым в нее изменениям должны предшествовать изменения в сопровождающих ее комментариях. Аналогично тому, как нет необходимости разделять разные формы использования компьютера в виде приложений, программирование не должно как-то особенно отличаться от других видов работы, которые пользователь выполняет с помощью компьютера. Опыт показывает, что большинство пользователей компьютеров с неохотой относятся даже к попыткам программирования. Наверное, некоторые преимущества программирования были бы более понятны, если бы процесс программирования был настолько интегрирован со средой, что отдельные программные структуры могли бы использоваться без необходимости учить весь язык или среду программирования. Было бы также полезно иметь возможность комбинировать функции из разных языков, однако это не должно быть смесью, подобной PL/I, но, скорее, должен использоваться метод слияния, предложенный в этой книге для приложений. Вероятно, весьма ценным был бы вариант, в котором комбинировались бы структуры LISP для обработки списков, структуры APL (или разновидности этого языка J) для обработки массивов, мощные методы обработки строк из языка SNOBOL, механизмы наследования и объекты из Smalltalk и т. д.

Разработчики языков программирования и эксперты по интерфейсам слишком редко работают вместе, и хотя когнетика позволяет усовершенствовать компьютерные интерфейсы, следует в максимальной степени использовать также сочетание современных методов разработки языков программирования с нашими знаниями о человеческих факто-

рах. Этот вопрос, до сих пор сохраняющий свою актуальность, был рассмотрен в одной из самых ранних книг Вейнберга в области разработки интерфейсов человек-машина «Психология компьютерного программирования» (Weinberg, «Psychology of Computer Programming», опубликована в 1971 г.), которая намного опередила свое время и до сих пор остается для нас откровением.

7.2. Режимы и кабели

Программное обеспечение работает на основе аппаратного обеспечения. Если вы не можете определить, как соединить между собой блоки аппаратного обеспечения, программное обеспечение превращается в бесполезные биты информации. Кабель – несколько проводов или волоконно-оптических линий, соединенных вместе и имеющих на каждом конце по разъему, – является одним из простейших элементов аппаратного обеспечения. Кабели должны присоединяться и отсоединяться от компьютера независимо от того, включен он или нет (кабели, которые нельзя заменять без выключения питания («not hot-swappable»), являются модалными!). У пользователя не должна возникать необходимость изменять конфигурацию устройств, как это требуется при использовании SCSI-соединений. В стандартах USB и FireWire эти проблемы учитываются. Однако есть другие проблемы интерфейсов, которые не учитываются даже в новых стандартах. Например, ситуация, когда кабель имеет неподходящий «пол» разъема, вызывает раздражение. Поскольку кабели могут иметь разные разъемы («папа» и «мама»), и так как некоторые части оборудования могут иметь соединители или для одного типа разъема («папа»), или для другого («мама»), это приводит к тому, что каждый тип кабеля может иметь огромное число модификаций. Многие владельцы компьютеров покупают адаптеры для инверсии «пола», поскольку они дешевле и меньше по размеру, чем кабели. Допустим, что у вас есть только кабель с двумя типами разъемов («папа-мама»), а вам требуется соединить два устройства, в которых есть разъемы только одного типа («мама»). В этом случае вы можете либо приобрести кабель с двумя разъемами соответствующего типа («папа-папа»), либо купить адаптер типа «папа-мама» и присоединить его к одному из концов («мама») кабеля, который у вас уже есть. В результате вы получите кабель («папа-папа»), который будет подходить для соединения двух устройств с симметричными разъемами («мама-мама»).

Эту проблему можно избежать, но методы, которые обычно предлагаются, не работают. Одно из решений, о котором я слышал, заключается в том, чтобы на оборудовании устанавливать разъемы одного типа (скажем, «мама»), а на кабелях все разъемы сделать другого типа (соответственно, «папа»). Но даже в этом случае будут нужны адаптеры типа «мама-мама» для соединения двух коротких кабелей в один длинный, или же производители должны будут подумать о выпуске

кабельных шнуров типа «папа-мама», чтобы их можно было использовать для удлинения. Следуя этой логике, можно прийти к ситуации, в которой потребуется использовать все возможные комбинации между соединениями «папа» и «мама» на кабелях и адаптерах, даже если по стандарту все соединения на оборудовании будут иметь разъемы типа «мама».

Обычная соединительная пара состоит из штекерного разъема («папа») и гнездового разъема («мама»). В результате появляется набор из следующих восьми типов деталей, которые могут использоваться как соединители для оборудования и кабельных шнуров:

- соединитель «папа» для оборудования;
- соединитель «мама» для оборудования;
- соединитель «папа» для кабельных шнуров;
- соединитель «мама» для кабельных шнуров;
- адаптеры «папа-мама»;
- адаптеры «мама-папа»;
- адаптеры «папа-папа»;
- адаптеры «мама-мама».

При использовании гермафродитных соединителей любые два кабеля можно соединить вместе и любой кабель может быть присоединен к любому разъему в оборудовании. Таким образом, весь набор типов соединителей сводится к следующим двум типам:

- соединитель для оборудования;
- соединитель для кабельных шнуров.

С точки зрения электротехники для каждого вида сигнала требуется использовать свой тип кабеля, но в каждом классе кабелей ни с точки зрения электротехники, ни с точки зрения производства ничто не мешает использовать гибридные (так называемые гермафродитные) соединители, не разделяющиеся на тип «папа» или «мама». Любые два соединителя-гермафродита данного класса могут быть соединены вместе. Более того, любой вид соединения для проведения электронного сигнала или питания может быть выполнен с помощью гермафродитных соединителей. Они могут использоваться в качестве многостежерных соединителей, разъемов питания, а также для коаксиальных кабелей.¹

¹ Некоторые кабели, используемые для соединения оборудования, расположенного на больших расстояниях друг от друга (например, фотографические стробоскопические источники света), снабжены соединителями-гермафродитами. Другой пример – это вышедшие из употребления коаксиальные радиочастотные соединители питания для регистра общего назначения.

Если имеются два гермафродитных соединителя данного класса, вы можете их использовать как два разных кабеля или соединить их в один длинный кабель. В некоторых случаях соединитель-гермафродит мог бы быть не сложнее и не дороже соединителя с определенным «полом». Однако это не всегда будет так. Во многих случаях гермафродитный соединитель будет несколько сложнее и дороже в производстве, но это будет оправдываться следующими факторами:

- большее удобство для пользователя;
- более простые инструкции;
- меньший объем производственной наладки;
- меньшее количество деталей, которые нужно складировать продавцам и распространителям.

На рис. 7.1 показана схема четырехпроводного линейного гермафродитного соединителя. При линейном расположении n проводов требуется минимум $2n - 1$ контактов. По схеме проводов на рис. 7.2 можно понять, каким образом работают такие соединители. На рис. 7.3 показаны выходы из гермафродитного коаксиального соединителя. Эту идею можно использовать и для многожильных редакторакоаксиальных проводов. Для того чтобы получить более удобные интерфейсы человек-машина, мы готовы платить за более сложные компьютеры. Этот же принцип должен быть применен к сумасшедшему клубку проводов и кабелей, которым вечно опутан компьютер и пользователь. Может показаться, что нужна какая-то линейная схема с меньшим, чем $2n - 1$, числом контактов, но здесь следует учесть возможность использования кабелей для удлинения.

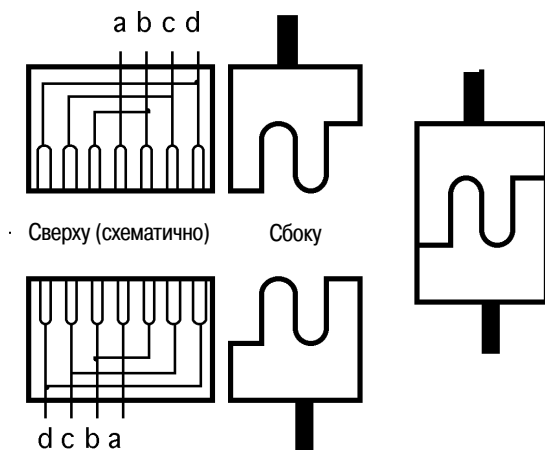


Рис. 7.1. Гермафродитный четырехпроводный соединитель.
Четыре провода обозначены как a , b , c и d

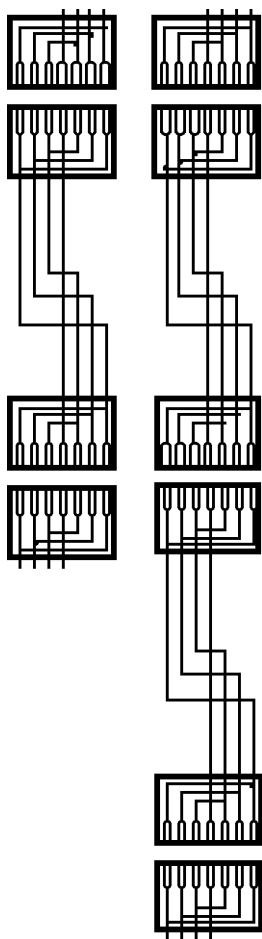


Рис. 7.2. Гермафродитные кабели могут использоваться для соединения компонентов оборудования (см. нижнюю и верхнюю части каждой колонки), а также для удлинения кабелей (см. правую колонку)

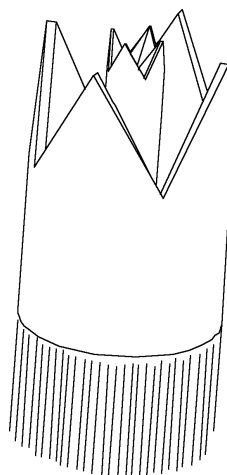


Рис. 7.3. Выходы в гермафродитном коаксиальном соединителе. Внешняя изолирующая оболочка с четырьмя зубцами, аналогичными изображенным, но с увеличенными концами и соединительным расширением снизу (не показаны), служит для сцепления

7.3. Этика и управление разработкой интерфейсов

Разумный человек приспосабливает себя к миру. Неразумный человек стремится приспособить мир к себе. Следовательно, весь прогресс зависит от людей неразумных.

Бернард Шоу

Трудно создать хороший интерфейс, если руководство не понимает, что разработка интерфейса является достаточно важным этапом. В краткосрочной перспективе тщательный подход к разработке интерфейса может увеличить расходы и время на создание продукта. Мой опыт показывает, что краткосрочный подход является неверным даже в краткосрочном периоде, поскольку улучшение пользовательского интерфейса часто упрощает разработку. Тщательное проектирование и детальное определение технических и других требований не замед-

ляют, а, наоборот, ускоряют процесс разработки. Создание качественного интерфейса полезно и с точки зрения долгосрочной перспективы, поскольку в результате приводит к

- большей продуктивности работы пользователя;
- большему удобству для пользователя;
- большей ценности в глазах покупателя;
- уменьшению расходов на поддержку покупателей;
- ускорению и упрощению процесса внедрения;
- преимуществу перед конкурентами на рынке;
- лояльности к данной марке;
- упрощению инструкций и онлайн-помощи;
- более безопасным продуктам.

Разработчики интерфейсов редко когда имеют возможность контролировать, в какой момент в процессе разработки проекта начнется создание интерфейса и какое значение будет придаваться его проблемам. В тех случаях, когда созданию интерфейса отдается главенствующая роль, как это было в проекте Macintosh, это дает поразительные результаты.

Если не учитывать, что данная область является довольно новой, и поэтому мало кто из специалистов в этой области пока поднялся до управляющих должностей, другой проблемой является то, что разработчики интерфейсов имеют небольшое влияние. Однако идет некоторая работа по решению этой проблемы с помощью предложения образовательных стандартов и тестов. Тем не менее, обладание такого сертификата у специалиста еще не является гарантией его компетентности. Здесь речь идет о другой стороне этой проблемы. Даже если разработчик является достаточно компетентным, от него (или нее) часто требуют создавать плохие интерфейсы. В этом отношении можно только позавидовать врачам, потому что для них предусмотрены юридические защитные меры, которые позволяют им выполнять свою работу правильно. Например, врач может предъявить судебный иск за незаконное увольнение при отказе выполнять действия, угрожающие состоянию здоровья пациентов. Строительные инженеры могут обращаться в суд в случае увольнения за отказ нарушить каноны, принятые в их профессии.

Специалисты по разработке интерфейсов работают в области, в которой неправильные решения могут вызвать физические поражения и способствовать психологическим расстройствам. Например, если интерфейс создает необходимость слишком часто нажимать на клавиши или кнопку мыши, это может привести к возникновению или обострению хронического стрессового нарушения (*repetitive stress injuries*). Плохой интерфейс может вызывать психологические расстройства. Таким образом, требуется создание основы для установления юридических норм защиты добросовестных специалистов. Другой необходи-

мостью является установление определенных профессиональных стандартов (речь идет не о стандартах разработки интерфейсов). Меры, упомянутые в этой книге, а также те, которые будут разработаны в будущем, могут помочь установить количественные, объективные нормы. Например, инженер-строитель должна показать, что она спроектировала мост, который отвечает установленным стандартам, в соответствии с которыми этот мост должен выдерживать нагрузку, скажем, в два раза превышающую минимально возможный уровень. Выбросы из автомобиля должны содержать не более 0,2% СО для того, чтобы этот автомобиль мог быть сертифицирован. Аналогичным образом, мы могли бы установить, что интерфейс текстового процессора не может быть принят, если, скажем, его общая информационно-теоретическая эффективность меньше 0,7 или если общая символьная производительность является меньше 0,8, а отдельные элементы имеют эффективность меньше 0,5.

Критерии могут быть также подобраны таким образом, что для некоторого числа наиболее часто используемых задач средневзвешенное время выполнения, а значит, и количество нажатий на клавиши, движений с помощью ГУВ и нажатий на его кнопку в новом текстовом процессоре не должно превышать значений, которые достигнуты в любом предыдущем или современном коммерческом продукте, предназначенном для аналогичной цели. Продукты, которые удовлетворяют данным критериям, могут получать какую-то форму сертификации. Эти критерии будут автоматически изменяться по мере развития интерфейсной технологии. В настоящее время новые продукты часто оказываются сложнее в использовании, чем старые, но это нельзя понять до тех пор, пока вы не попытаетесь это проверить на собственном опыте. Поскольку эти критерии касаются эффективности, т. е. в конечном счете определяют итоговый результат работы пользователя, то руководители проекта должны уделять им особое внимание. От публикации объективных нормативов качества интерфейсов выиграют не только разработчики и руководители проекта, но также и покупатели.

Стив Уайлдстром (Steve Wildstrom), который публикует свои статьи в еженедельнике *Business Week*, указывает, что «производители компьютеров и, в особенности, разработчики программного обеспечения, часто думают, что требования Единого коммерческого кодекса (Uniform Commercial Code) касаются не их, а кого-то другого» (частный разговор, октябрь 1998 г.). Многие современные лицензии на программное обеспечение, навязываемые покупателям, не гарантируют им даже того, что это программное обеспечение будет выполнять задачу, о которой сообщалось в рекламе. Во многих таких документах прямым образом отрицается понятие *merchantability* (годности для продажи), которое означает буквально следующее: продажа данного продукта автоматически предполагает, что этот продукт способен выполнять задачу, для которой он, по заявлению продавца, предназначен. В некоторых штатах США были приняты законы, запрещающие

отказ от подтверждения «годности к продаже» для продуктов компьютерного производства. Все верховные власти должны поступить так же.

Система оценки качества интерфейсов, осуществляемая независимой организацией, может быть полезной для покупателей тех продуктов, в которых интерфейсный компонент выполняет значительную роль. Сама разработка пользовательского интерфейса не должна как-то регулироваться или ограничиваться. Следует избегать применения принципов, основанных на использовании конкретных интерфейсных механизмов, чтобы не подавлять стремление к нововведениям. Однако введение относительных количественных нормативов продуктивности для продуктов одного типа побудит разработчиков двигаться в правильном направлении.

Нужно найти тонкий баланс между созданием настолько нового продукта, что опытные пользователи, привыкшие к обычным интерфейсам, почувствуют неудобство в его использовании, и созданием продукта с интерфейсом, который настолько не отличается от стандартного графического пользовательского интерфейса, что его никак нельзя считать результатом нашего желания в максимальной степени помочь пользователю. С одной стороны, мы должны избежать новизны как таковой, хотя с другой стороны, мы не должны терять ценную возможность выиграть на рынке из-за того, что декалькируем аналогичные существующие продукты.

В бизнесе разработки интерфейсов давно существует миф о том, что «расширение функциональности и сохранение простоты использования не могут быть совмещены в одном интерфейсе» (Microsoft, 1995, с. 8). Действительно, добавление множества специальных, созданных именно для данного случая сервисов, уменьшает простоту использования. Но как раз это является плохой разработкой. Часто, но не всегда, возможно увеличить функциональность, не увеличивая степень сложности интерфейса. Добавление нового сервиса, как правило, может быть сделано таким образом, что это не прибавит сложности в интерфейсе (здесь следует отметить разницу между сложностью интерфейса и сложностью задачи). Если добавляемая функция позволяет объединить в единое целое разрозненные элементы интерфейса, то такой интерфейс может стать проще.

«Одним из способов сохранить простоту заключается в сокращении объема предъявляемой информации до того минимума, который необходим для адекватного взаимодействия» (Microsoft, 1995, с. 8). Это действительно так, за исключением того, что слово «адекватного» следует заменить словом «нормального». Однако в этой компании ошибаются, когда утверждают: «Например, не используйте словесных описаний командных имен или сообщений» (Microsoft, 1995, с. 8). Здесь возникает вопрос: что же можно считать минимумом для нормального взаимодействия? В большинстве современных интерфейсов акцент

делается на краткость в ущерб ясности. Почему мы должны заниматься расшифровкой непонятного названия «Список» в выпадающем меню в текстовом процессоре, когда можно было бы использовать более понятное «Создать указатель или оглавление»? (Необходимо учесть, что выпадающее меню не занимает места в документе, поскольку оно исчезает сразу же, как только вы уведите от него курсор или выбираете какую-то из опций.) Не следует путать простой внешний вид экрана с простотой использования интерфейса.

Навигация против свободного пространства

Кажется, что мы просто боимся отображать информацию в наших интерфейсах. Известно, что чем меньше группа вещей, тем легче найти в ней нужный предмет. Однако из этого не следует, как это может некоторым показаться, что чем меньше элементов на экране, тем лучше. Если сотни элементов разбросаны в десятках экранов, то вы больше времени потеряете на перемещения между ними, чем на сам поиск конкретного элемента, даже если этот элемент находится среди моря других аналогичных. Поиск чего-то в длинных, однообразных списках не всегда может быть трудным.

Если бы люди не могли быстро находить короткие элементы в длинных списках, журнал «Уолл стрит джорнал» уже давно бы разорился. Предпочли бы вы, чтобы котировки акций печатались по 15 строк на странице, каждая из которых оформлена наподобие экрана из современного интерфейса, в придачу даже со схемой поиска нужной страницы вроде:

Ценные бумаги	Страница
AA-AD	1
AD-AS	2
AT-AZ	3
BA-BK	4
и так далее	

Такая схема показалась бы несерьезной, неэкономной и странной. Однако иногда мы используем больше экранных пикселей на то, чтобы сделать аккуратные рамки с тенями, чем на отображение полезной информации. Если человек имеет мотив (обусловленный личным интересом или зарплатой) для того, чтобы искать какие-то нудные данные, длинные списки не могут представлять для него никакой проблемы. Визуальный дизайнер Эдвард Тафт (Edward Tufte, 1983, с. 105) разработал принципы отображения информации, среди которых первыми тремя являются следующие:

- данные следует показывать прежде всего остального;
- следует максимально увеличивать долю чернил, используемых для отображения данных;

- следует максимально уменьшать долю чернил, которые не используются для отображения данных.

Для того чтобы приложить эти принципы к устройствам с дисплеями, требуется всего лишь заменить в них слово *чернила* на слово *пикселы*. Серьезный, профессиональный пользователь желает, чтобы экраны были до отказа заполнены полезным содержанием. Экраны должны быть хорошо обозначены, снабжены простыми механизмами для осуществления поиска и получения информации, отражающей суть данного экрана. (В конце концов, раз уж мы сели за компьютер, мы должны извлечь из этого максимальную пользу.)

Сегодня существует множество исследований о дизайне экранных изображений. Многие ранние, но до сих пор не утратившие свою ценность исследования, рассмотрены в обзоре Туллиса (Tullis, 1984). Некоторые из результатов до сих пор могут быть использованы (например, время поиска в списке элементов составляет приблизительно 30 мс на каждый элемент (с. 126)). Основные результаты, полученные Туллисом, относились к 24×80 алфавитно-цифровым дисплеям и позволяли дать количественные оценки.¹ Если эти результаты применить к современным растровым дисплеям и получить критерий оценки времени поиска целевого объекта, то это позволило бы не только оптимизировать отдельные, изолированные беззаконные экранные изображения (в соответствии с ограничениями Туллиса), но также достичь более глобальной оптимизации, связанной с оценкой навигационной структуры. Как отмечает Туллис (с. 132), почти всегда необходимо искать компромисс между сложностью экрана и сложностью навигации. Этот компромисс зависит от скорости и простоты работы навигации и от структуры данных. Когда для выполнения внутриэкранного поиска вместо визуального сканирования используется поисковое устройство (как, например, функция LEAP), то для оценки эффективности должны быть разработаны другие виды критериев. Здесь имеется широкая область для дальнейших исследований.

В любом случае, если довести до логического конца популярную философию экранного дизайна, которая сводится к принципу, что «чем больше пустого пространства, тем легче читать», то мы увидим, что тогда на каждом экране должен помещаться только лишь один элемент данных. В этом случае пользователь уж точно сможет визуально распознать этот элемент с наименьшим возможным усилием.

Имея опыт улучшения множества продуктов с помощью сокращения количества экранов и увеличения доли информации на остав-

¹ Критерии, предложенные Туллисом, не учитывают качества содержания, но только его внешний вид. Можно построить такие образцы изображений, которые по этим критериям будут иметь высокие оценки, но будут сложны в использовании.

шихся экранах, – или, другими словами, с помощью улучшения логической структуры дизайна, приводящего к такому сокращению, – я пришел к выводу, что почти во всех коммерческих программах мы допускаем ту ошибку, что помещаем на экраны слишком мало информации.

Лучший способ заставить интерфейс вашего продукта отличаться – это сделать так, чтобы он работал. В книге Нормана «Невидимый компьютер» (Norman, «The Invisible Computer», 1998) можно найти хорошо написанное и убедительное обоснование важности вопросов разработки интерфейсов, которое в большой степени обращено к тем, кто осуществляет руководство проектами.

Заключение

Он почувствовал жалость к псу, которому по окончании долго действовавшего «Распоряжения о мордах» сняли намордник, и он теперь плохо понимает, что же ему делать.

*К. Г. Грей (из справочника Джейн
«Все самолеты мира», 1919)*

Если вы стремитесь сделать интерфейсы как можно более простыми в использовании с учетом человеческих возможностей и ограничений, следует выполнить следующие два действия. Первое – понять, что мы можем делать и чего не можем, изучить карту человеческого мышления на основе когнитивной психологии и проследить ее влияние с точки зрения такой прикладной дисциплины, как когнетика. В данной книге было рассмотрено одно из основных направлений на этой карте – исследования показывают, что наши способности разделяются между когнитивным сознательным и когнитивным бессознательным; что мы обладаем только одним локусом внимания; и что формирование привычек играет центральную роль в том, как мы реагируем на те или иные виды функционирования интерфейсов. Мы также узнали, что индивидуальные отличия при формировании привычек невелики по сравнению с теми отличиями, которые существуют между индивидуумами в других отношениях.

С помощью науки когнетики мы изучили, что режимы (или модальности), которые, как это давно известно, являются нежелательными, становятся причиной некоторых наиболее неприятных проблем в современных компьютерных интерфейсах. Для решения этих проблем требуется, чтобы интерфейс был немодальным и в максимально возможной степени обладал характеристикой монотонности. Следует учитывать имеющееся единообразие между пользователями. Повысить производительность можно за счет уменьшения времени выполнения задачи, что приводит к использованию классического метода исследования – количественного GOMS-анализа в приложении к интерфейсам. GOMS-анализ позволяет определить, какие детали интерфейса замедляют или ускоряют его работу, и приводит нас к вопросу о том, насколько же быстрым и эффективным может быть интерфейс, а также к необходимости установления количественных критериев эффективности.

Пройдя по дороге понимания человеческих способностей, мы затем рассмотрели современное компьютерное оборудование и способы его использования. В результате был определен набор элементарных действий и методов, которые могут быть применены к широкому и разнообразному кругу приложений. Эту форму единообразия также следует использовать.

Наши исследования вели нас в незнакомом направлении, которое оказалось лучшим путем к созданию удобного в использовании интерфейса. Ускорение работы достигается с помощью быстрых методов поиска в сочетании с устранением таких ненужных механизмов, как файловые имена и URL, иерархичные файловые структуры и приложения. Масштабируемая ZIP-среда позволяет нам «лететь» над содержанием, больше видеть и быстро перемещаться в необходимое место внутри него. Мы также затронули некоторые побочные направления, например способы упрощения использования соединительных кабелей.

Эта книга содержит много пробелов. Например, наверняка я мог пропустить какую-то из существующих работ, которую мне следовало бы прочитать, и, возможно, при описании не моих собственных идей я мог неправильно назвать имена их авторов. В некоторых местах этой книги рассматриваются области, в которых при наличии соответствующих ресурсов я бы хотел провести эксперименты, чтобы протестировать мои выводы и предположения. Читателям предлагается рассматривать эти области как поле для дальнейших исследований.

Спасибо за то, что прочитали эту книгу и вместе со мной пытались найти решения для создания более гуманных интерфейсов. В тех случаях, когда наше путешествие заходило в неисследованные земли, я мог в том или другом месте сделать поворот в неверную сторону. Тем не менее, я убежден, что мой компас работает верно, и что главное направление, которое я избрал, является правильным.

На основе наших знаний о познавательной человеческой способности мы пришли к необходимости фундаментального изменения интерфейсов «человек-машина». Все другие методы здесь будут бесполезны.



Однокнопочная мышь: история и будущее

Разнообразие любят люди тоже.¹

Чосер «Рассказ Сквайра»

За создание однокнопочной мыши и некоторых основных методов ее использования я получал как критику, так и одобрения. Вопросы читателей черновика этой книги показали, что, на их взгляд, система, разработанная в Macintosh, работала аналогично системе с использованием мыши, которая была ранее создана в исследовательском центре PARC компании Xerox. В данном приложении будут описаны системы с использованием мыши, которые я видел в исследовательском центре Palo Alto Research Center (PARC) компании Xerox. Еще ранее мышь была использована исследовательской группой под руководством Дугласа Инглбарта (Douglas Englebart) из Станфордского исследовательского института для создания системы, которая во многих отношениях на несколько десятилетий опередила свое время. Эта система содержала в себе ценные идеи, которые до сих пор не нашли широкого применения. Однако программное обеспечение, созданное Инглбартом, часто было модальным и иногда являлось неэффективным при подсчете нажатий на клавиши.

Немногие пользователи современных персональных компьютеров помнят, через что надо было пройти, чтобы сделать операцию «выделе-

¹ *Men loven of propre kynde newefangelnesse* – строфа 610 из второй части стиха «Рассказ Сквайра», поэма «Кентерберийские рассказы» («Canterbury Tales») Джеффри Чосера (1340–1400), перевод О. Румера (БВЛ, серия 1 т. 30, изд-во «Художественная Литература» М., 1973). – *Примеч. науч. ред.*

ния» в системе, созданной в исследовательском центре PARC, (например, в их самом популярном текстовом редакторе BRAVO). Далее нажатия на каждую из трех кнопок мыши системы PARC будут обозначены буквами *L*, *M* и *R* (левая, средняя и правая). В текстовом редакторе BRAVO квазирежимы для кнопок мыши не использовались.

Для выделения символа: указать на нужный символ, нажать *L*.

Для выделения слова: указать на нужное слово, нажать *M*.

Для выделения произвольной строки символов: указать на первый символ строки, нажать *L*; указать на последний символ строки, нажать *R*.

Для выделения строки слов: указать на первое слово, нажать *M*; указать на последнее слово, нажать *R*.

Обычной ошибкой было нажатие на кнопку *L*, а потом на кнопку *M*, что приводило только лишь к повторному началу процесса выделения в том месте, которое, как вам казалось, будет концом требуемой вам выборки. Это очень раздражало, особенно если выборка была большого размера. Тем не менее, обратите внимание, что вам никогда не требовалось «отменять» выборку – в любой момент вы могли свободно начинать новую. Это было действительным преимуществом в сравнении с другими современными системами, в которых перед тем, как сделать другую выборку, требовалось сначала нажать на кнопку ESC или совершить другое действие для отмены текущей выборки.

В исследовательском центре PARC ранее был разработан и другой редактор, о котором я в то время еще не знал. Для выделения текста в нем использовался метод click-and-drag (щелкнуть и перетащить). Используя систему обозначений, описанную в разделе 3.1, для выделения какого-то текста, требовалось выполнить следующие действия: указать на верхнюю точку текста, $L\downarrow$, указать на нижнюю точку текста, $L\uparrow$. Однако эта идея не была распространена на другие типы выделения.

В компании Apple я показал, каким образом можно более широко использовать одну кнопку мыши для выделения. Для выделения любой сплошной области на экране, независимо от того, является ли она текстом или нет, нужно указать на один угол области, $L\downarrow$, и указать на другой край, $L\uparrow$. Этот метод стал называться «щелкнуть и перетащить».

Другие методы в текстовом редакторе BRAVO были более сложными, чем те, которые использовались в компьютере Macintosh.

Для удаления текста: выделить текст и нажать *d*. Проблема здесь заключается в том, что вам требуется знать о том, что выделен какой-то текст, перед тем как нажимать клавишу с буквой *d*. В компьютере Macintosh я решил использовать клавишу Delete для удаления текста как во время набора, так и при редактировании.

Для вставки текста: указать на точку вставки и нажать *L*; нажать клавишу с буквой *i*; ввести новый текст и нажать клавишу ESC. Нажатие клавиши ESC снимает режим вставки. В компьютере Macintosh не использовался режим вставки, и метод, который был применен в нем, стал общераспространенным: для вставки текста указать и щелкнуть мышью в точке, где необходимо сделать вставку, т. е. *L*; ввести новый текст. В этом случае вместо использования явного разделителя для начала ввода текста, как в редакторе BRAVO, вы сразу же начинаете вводить нужный текст вставки. Вам не требуется в конце ввода вставки нажимать клавишу ESC или другую кнопку – начало выполнения новой задачи автоматически заканчивает предыдущую. Замена текста в текстовом редакторе BRAVO происходила следующим образом: выделить текст, нажать клавишу с буквой *r*, ввести новый текст, нажать клавишу ESC. Для замены текста в компьютере Macintosh: выделить текст, нажать на клавишу Delete и ввести новый текст. Этот метод отличается от того, который теперь используется в системах Macintosh и Windows для замены текста и который является проще: выделить текст, ввести новый текст.

Однако, как пользователям известно, этот последний метод часто приводит к случайной потере текста. В предыдущем способе требуется на одно нажатие клавиши больше, но он никогда не вызывает потери данных, и поэтому текстовый редактор является более безопасным и пользователь ощущает его именно таким. Следовательно, одно дополнительное нажатие на клавишу здесь оказывается весьма полезным.

В отличие от текстового редактора BRAVO и некоторых других редакторов, разработанных в исследовательском центре PARC, в компьютере Macintosh и, в значительной степени, в более поздних системах ввод простой буквы никогда не действует как команда. Сейчас уже очевидно, что так и должно быть, но в то время так не казалось. Мой проект интерфейса, основанного на использовании однокнопочной мыши, уточнялся и расширялся во время обсуждений с моими коллегами, чаще с Брайеном Хоурдом (Brian Howard) и Биллом Аткинсом (Bill Atkinson), и, конечно, многие изменения были сделаны на основе наблюдений, сделанных во время пользовательских тестирований и дальнейшего процесса разработки. Для некоторых пользователей оказалось сложным одновременно удерживать кнопку и двигать графическое устройство ввода, но это отчасти зависит от его конструкции, а также от того, используется ли мышь или другое устройство. В проекте Macintosh эта проблема была облегчена тем, что на мышь была установлена одна большая кнопка, требующая небольшого усилия для нажатия и хорошо воспринимаемая на ощупь. (Некоторые из последних моделей сенсорных панелей, особенно переносные, имеют плохие кнопки, и поэтому с их помощью трудно делать перетаскивание, что часто приводит к ошибкам.) Кроме того, общее улучшение и сокращение ошибок, достигаемое путем устранения режимов, часто перевешивает

ошибки, возникающие во время перетаскивания, даже если используются не совсем оптимальные графические устройства ввода.

В то время, однако, я не понимал еще, что мышь может работать нормально даже со множеством кнопок *при условии, что они как-то обозначены*. Если бы мышь Macintosh была снабжена множеством кнопок, и если бы кнопки имели постоянные обозначения и использовались только лишь для обозначенной функции, такая мышь могла бы стать более совершенным вариантом. Улучшенная мышь могла бы иметь сверху две кнопки, обозначенные как Select (Выделить) и Activate (Активизировать), и сбоку кнопку, управляемую большим пальцем руки. Эта последняя кнопка могла бы быть обозначена как Grab (Схватить). На некоторых современных мышах есть сверху колесо, которое в основном используется для прокрутки. Было бы лучше, если бы в этом месте находился небольшой трекбол. Таким образом, с помощью мыши контролировалась бы позиция курсора, а трекбол использовался бы, например, для перемещения объектов или для выбора элементов из плавающих меню.

В

Теория работы интерфейса для SwyftCard

Некоторые из принципов, рассмотренных в этой книге, впервые были опубликованы в 1984 году в руководстве для SwiftCard. Система SwiftCard, предназначенная для довольно успешного в то время Apple II, была (по сегодняшним стандартам) простой. Приложение к ее руководству имело необычное содержание. Вместе с обычным, традиционным изложением принципа работы электронной начинки этого устройства предлагалось теоретическое описание работы программного обеспечения, а также информация о теории работы пользовательского интерфейса. Это, пожалуй, был первый случай, когда информация такого рода сопровождала коммерческий продукт. В некотором смысле то приложение можно считать началом этой книги. Нижеследующий материал взят из второго издания руководства (Alzofon и Raskin, 1985).

Подходы, на основе которых была создана SwiftCard, разрабатывались с целью решения целого ряда проблем, которыми страдают почти все современные системы. Большинство из этих проблем сами по себе являются довольно небольшими, но в совокупности они приводят к тому, что процесс изучения и использования существующего программного обеспечения чрезмерно замедляется, что вызывает раздражение и усталость от использования компьютеров.

Например, мы всегда удивлялись, почему пользователю приходится форматировать диски. Разве компьютер не способен самостоятельно определить, форматирован диск или нет, и при необходимости выполнить форматирование? Кроме того, нам кажется, что клавиши для управления курсором работают слишком медленно, а если учесть то количество дополнительных команд, которое они требуют (для перемещения к следующему/предыдущему слову, предложению, абзацу,

странице; для перемещения к началу или концу строки, документа или файла), то, на наш взгляд, они являются также и слишком сложными. ГУВ существенного улучшения не приносит, поскольку большая часть этих устройств заставляет пользователя убирать руки с клавиатуры, а также требует много экранного пространства для меню, полос прокрутки и других механизмов, связанных с использованием ГУВ. Необходимость работать с меню вместо того, чтобы сразу делать то, что нам необходимо, вызывает раздражение, а огромное число команд, которое используется в большинстве систем, ставит нас в затруднительное положение. Мы ненавидим дисковые системы, которые могут допустить потерю информации из-за обычных человеческих ошибок. Нас удивляет, что многие текстовые процессоры не успевают воспринимать информацию, вводимую с клавиатуры человеком.

SwiftCard демонстрирует собой, что хороший проект позволяет решить все эти, а вместе с ними и многие другие вопросы и проблемы, которые беспокоили нас в течение многих лет. Эта система способна работать на недорогом компьютере, имеющем только один диск, и требует минимальных ресурсов памяти. Наш продукт выполняет то, что требуется для большинства пользователей, однако без помощи операционной системы, без ценника с большими цифрами, без капризов в работе.

Основные принципы разработки включают много новшеств и учитывают опыт, использованный в других продуктах.

1. Концепция курсора с функцией LEAP, которая позволяет перемещаться к целевому объекту со скоростью, в среднем в три раза превышающей скорость работы самого лучшего общераспространенного устройства типа мышь.
2. Сам курсор состоит из двух частей, которые точно показывают места, где будет появляться вводимый текст и где будет происходить удаление. Этот же курсор сворачивается при перемещении так, чтобы было удобно удалять.
3. Ограниченный набор основных операций позволяет легко выполнять широкий диапазон задач.
4. Отказ от использования операционной системы позволяет выполнять все операции непосредственно из редактора без необходимости переходить в различные режимы.
5. Устранение режимов в целом способствует формированию привычек, поскольку пользователю не приходится задумываться, в каком состоянии находится система, чтобы определить доступные в этом состоянии команды. Это свойство называется безмодальностью.
6. Отказ от использования множества способов выполнения той или иной задачи – опять же для того, чтобы пользователю не приходилось задумываться о том, какой способ выполнения задачи выбрать. Этот принцип мы называем монотонностью. Так же как и безмо-

дальность, монотонность способствует формированию привычек в использовании.

7. Акцент на формирование привычек сам по себе является основополагающим принципом, однако разработчики зачастую его неоправданно не учитывают. Мы считаем важным, чтобы после короткого периода обучения пользователь мог использовать систему, не задумываясь о ее работе.
8. Команда DISK позволяет снять сложности, которые обычно возникают при использовании дисковой операционной системы (DOS). Используется только одна простая команда. Она же позволяет защитить данные от наиболее частых ошибок, приводящих к потере информации. Возможность для создания такой команды дает метод, при котором один диск соответствует одному Тексту.
9. Создание зависимости скорости работы от частоты использования (чаще используемые задачи выполняются быстрее, реже используемые – медленнее).
10. «То, что вы видите, то вы и получите» (What you see is what you get). Другими словами, изображение на экране будет выглядеть так же и на бумаге. (Из-за ограничений в оборудовании Apple этот принцип, тем не менее, был нарушен для операции подчеркивания.)
11. Построение команд по методу «существительное-глагол». Сначала пользователь определяет, с чем он собирается работать (что дает время на проверку и при необходимости исправление), и затем применяет некоторое действие к выбранному объекту. В некоторых системах используется обратная последовательность («глагол-существительное») или даже смешиваются оба метода, что еще хуже.
12. Другой общий принцип заключается в том, что в системе трудно что-либо испортить или стереть. Это возможно, но очень трудно. Ошибки не происходят случайно или по невнимательности пользователя.
13. Включение средств программирования и коммуникации в среду общего назначения, внутри которой выходные данные помещаются в область редактирования или поиска.
14. Необходимость учета в плане работы времени на проведение длительных (в течение нескольких месяцев) тестирований и наладки. Покупатели системы не должны использоваться в качестве субъектов тестирования.

Вышеизложенное является только упрощенным наброском (описание системы насчитывает около 50 страниц), но мы надеемся, что это даст вам некоторое представление о том, что привело нас к идее сделать систему SwiftCard такой, какая она есть.

Библиография

Где-то я мог неправильно интерпретировать или понять смысл цитируемых мной работ. Заранее прошу меня извинить за любые ошибки такого рода.

Accot, Johnny, and Shumin Zhai. «Beyond Fitts' Law: Models for Trajectory-Based HCI Tasks» (www.dgp.toronto.edu/~accot/Common/Articles/CHI97/chi.html, 1997).

Alzofon, David, David Caulkins, Jef Raskin, and James Winter. *Canon Cat How-To Guide* (Tokyo: Canon, 1987).

Alzofon, David, and Jef Raskin. *SwyftCard*, 2d ed. (Menlo Park, CA: Information Appliance, 1985).

Anderson, J. R. *Rules of the Mind* (Hillsdale, NJ: Lawrence Erlbaum Associates, 1993).

Apple Computer. *Inside Macintosh*, Vol. 1 (Cupertino, CA: Apple Computer, 1985).

Apple Computer. *Human Interface Guidelines: The Apple Desktop Interface* (Reading, MA: Addison-Wesley, 1987).

Ashlar. *Vellum 3D Manual* (Sunnyvale, CA: Ashlar, 1995).

Asimov, Isaac. *I Robot* (New York: Bantam Books, 1977).

Baars, Bernard J. *A Cognitive Theory of Consciousness* (Cambridge, U.K: Cambridge University Press, 1988).

Business Week. «Special Report on Information Appliances» (22 Nov. 1993), p. 110.

Buxton, William. «Chunking and Phrasing and the Design of Human-Computer Dialogs,» *Information Processing '86: Proceedings of the IFIP 10th World Computer Congress* (Amsterdam: North-Holland, 1986).

Card, Stuart K., Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction* (Hillsdale, NJ: Lawrence Erlbaum Associates, 1983).

Cohen, Jonathan D., and Jonathan W. Schooler, eds. *Scientific Approaches to Consciousness* (Hillsdale, NJ: Lawrence Erlbaum Associates, 1997).

Collins, Richard. *Flying* 121:10, p. 67 (October 1994).

- Cooper, Alan. *About Face* (Foster City, CA: IDG Books Worldwide, 1995).
- Dennett, Daniel C. *Consciousness Explained* (Boston: Little, Brown, 1991).
- Dijksterhuis, E. J. *The Mechanization of the World Picture* (London: Oxford University Press, 1961).
- Drori, Offer. «The User Interface in Text Retrieval Systems,» *SigCHI Bulletin* 30:3 (1998).
- Eriksson, H., and P. Magnus. *UML (Unified Modeling Language) Toolkit* (New York: John Wiley & Sons, 1998).
- Garrison, Peter. *Flying* 121:12, p. 112 (December 1994).
- Garrison, Peter. «Drifting Off Centerline,» *Flying* 122:1, p. 43 (January 1995).
- Gray, Wayne D., Bonnie E. John, and Michael E. Atwood. «Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance,» *Human-Computer Interaction*, 8:3, pp. 237–309 (1993).
- Grudin, J. «The Case Against User Interface Consistency,» *Communications of the ACM*, pp. 1164–1173 (1989).
- Hewlett-Packard. *User Interface Design Rules for the New Wave Office System* (Cupertino, CA: Hewlett-Packard Personal Software Division, 1987).
- Hotchkiss, B. «The Car Column,» *Pacifica Tribune*, 12 Nov. 1997, p. 14A.
- Horton, William. *The Icon Book* (New York: John Wiley, 1994).
- IBM. *System Application Architecture, Common User Access, Panel Design and User Interaction* (Boca Raton, FL: IBM, 1988).
- Jacobson, Robert, ed. *Information Design* (Cambridge, MA: MIT Press, 1999).
- John, Bonnie E. «Why GOMS?» *Interactions*: pp. 80–89 (October 1995).
- Johnson, J., and G. Englebeck. «Modes Survey Results,» *SigCHI Bulletin* 20:4, pp. 38–50 (1989).
- Kaplan, Justin, ed. *Bartlett's Familiar Quotations*, 16th ed. (Boston: Little, Brown, 1992).
- Knuth, Donald E. *Literate Programming* (Stanford, CA: Center for the Study of Language and Information, 1992).
- Lammers, Susan. *Programmers at Work* (Redmond, WA: Microsoft Press, 1986).
- Landauer, Thomas K. *The Trouble with Computers* (Cambridge, MA: MIT Press, 1995).
- Laurel, Brenda, ed. *The Art of Human-Computer Interface Design* (Reading, MA: Addison-Wesley, 1990).
- Lewis, C., and D. A. Norman. «Designing for Error,» in D. Norman and S. Draper, eds., *User Centered System Design* (Hillsdale, NJ: Lawrence Erlbaum Associates, 1986).

- Linzmayr, Owen. *Apple Confidential* (San Francisco: No Starch Press, 1999).
- Loftus, Elizabeth F. *Eyewitness Testimony* (Cambridge, MA: Harvard University Press, 1979).
- Loftus, Elizabeth F. *Memory* (Reading, MA: Addison-Wesley, 1980).
- Mackenzie, I. S. «Movement Time Prediction in Human-Computer Interfaces,» in R. M. Baecker, W. A. S. Buxton, J. Grudin, and S. Greenberg, eds., *Readings in Human-Computer Interaction*, 2d ed., pp. 483–493 (Los Altos, CA: Kaufmann, 1995).
- Malone, Michael S. *Infinite Loop* (Chicago: Doubleday, 1999).
- Mayhew, Deborah. *Principles and Guidelines in Software User Interface Design* (Englewood Cliffs, NJ.: Prentice-Hall, 1992).
- Microsoft. *The Windows Interface Guidelines for Software Design* (Redmond, WA: Microsoft Press, 1995).
- Miller, George A. «The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information,» *Psychological Review* 63, pp. 81–97 (1956).
- Moore, J. S., and R. S. Boyer. «A Fast String Searching Algorithm,» *Communications of the Association for Computing Machinery* 20:10, pp. 762–772 (1977).
- Norman, Donald A. «Categorization of Action Slips,» *Psychology Review* 88:1, pp. 1–15 (1981).
- Norman, Donald A. «Design Rules Based on Analyses of Human Error,» *Communications of the ACM* 26:4, p. 255 (1983).
- Norman, Donald A. *The Psychology of Everyday Things* (New York: Basic Books, 1988).
- Norman, Donald A. *The Invisible Computer* (Cambridge, MA: MIT Press, 1998).
- Penrose, Roger. *The Emperor's New Mind* (London: Oxford University Press, 1989).
- Raskin, Jef. «Looking for a Humane Interface: Will Computers Ever Become Easy to Use?» *Communications of the ACM* 40:2, p. 98 (Feb. 1997).
- Raskin, Jef. «The Quick-Draw Graphics System.» Ph.D. diss. (State College, PA: Pennsylvania State University, 1967).
- Raskin, Jef. «FLOW: A Teaching Language for Computer Programming,» *Computers and the Humanities* 8:4 pp. (July 1974).
- Raskin, Jef. «Computers by the Millions,» *SIGPC Newsletter* 5:2 (1982).
- Raskin, Jef. «Systemic Implications of an Improved Two-Part Cursor,» *Proceedings of CHI 89: Human Factors in Computing Systems*, Austin: 30 April 1989, pp. 167–170 (New York: ACM Press, 1989).
- Raskin, Jef. «Down with GUIs,» *Wired* pp. (December 1993).

- Raskin, Jef. «Intuitive Equals Familiar,» *Communications of the ACM* 37:9, pp. (September 1994).
- Raskin, Jef, and James Winter. U.S. Patent No. 5,019,806, Method and Apparatus for Control of an Electronic Display, 1991.
- Reason, James. *Human Error* (Cambridge, U.K.: Cambridge University Press, 1990).
- Shneiderman, Ben. *Designing the User Interface* (Reading, MA: Addison-Wesley, 1987, 1998).
- Sellen, A., G. Kurtenbach, and W. Buxton. «The Prevention of Mode Errors Through Sensory Feedback,» *Human Computer Interaction* 7:2, pp. 141–164 (1992).
- Shannon, Claude E., and Warren Weaver. *The Mathematical Theory of Communication* (Urbana: University of Illinois Press, 1949, reprinted 1963).
- Smith, S. F., and D. J. Duell. *Clinical Nursing Skills*, 3rd ed. (East Norwalk, CT: Appleton & Lange, 1992).
- Stallman, Richard. *GNU Emacs Manual*, 9th ed. (Cambridge, MA: Free Software Foundation, 1993).
- Tesler, Larry. «The Smalltalk Environment,» *Byte* (August 1981).
- Tesler, Larry, and Timothy Mott. *Report on the Xerox Palo Alto Research Center Gypsy Typescript System* (Palo Alto, CA: Xerox, 20 April 1975).
- Thomas, Lewis. *The Lives of a Cell* (New York: Viking Press, 1974).
- Tognazzini, Bruce. *Tog on Interface* (Reading, MA: Addison-Wesley, 1992).
- Tolkien, J.R.R. (Douglas Anderson, ed.) *The Annotated Hobbit* (Boston: Houghton Mifflin, 1988).
- Tufte, Edward. *The Visual Display of Quantitative Information* (Cheshire, CT: Graphics Press, 1983).
- Tullis, Thomas S. «Predicting the Usability of Alphanumeric Displays,» Ph.D diss., Rice University, 1984.
- de Unamuno y Jugo, Miguel. *The Tragic Sense of Life*, Chapter 9 (1913).
- Weinberg, Gerald M. *The Psychology of Computer Programming* (New York: Van Nostrand Reinhold, 1971).

Алфавитный указатель

А

America Online, пакет электронной почты, 62
AND в шаблоне поиска, 208
APL, 223, 233
Apple Computer, 202, 207, 218
Apple iBook, 51
Apple II, 230
 и система SwiftCard, 250
Apple IIe, 203
Apple Newton, 26
Apricus, 190
AutoCAD, 60

В

BASIC, 22, 155, 232
Bob (Microsoft), 81
BRAVO, 247, 248
Business Week, 239

С

С и C++, 201
Canon, компания, 218
Canon Cat, компьютер, 49, 51, 71, 78, 126, 150, 156, 158, 165, 168, 172, 209, 216
 клавиша Use Front на, 79
 клавиша Calculate, встроенная в, 172
 тестирование, 209
 форматирование диска на, 217, 218
 функция LEAP, 169

Д

DataViz, 176
Drafting Assistant, 60

Е

Edsel, 147
 и поглощенность, 45
EMACS, 154
E-mail, 212
Eudora, 126

Ф

FireWire, стандарт, 232

Г

Global Village, компания, 158
GOMS-анализ, 244
 и автоповтор, 222
GOMS-модель, 97

И

iBook, 51
IDE (Borland), 158
IEEE, 214
Information Appliance, 159, 163, 221

Ј

J, разновидность языка APL, 233
JavaScript, 155
 команда Alert в, 111
Jazz, пользовательский интерфейс, 190

Л

LEAP, функция, 158, 164, 188, 201, 210, 242
 выделение документа с помощью, 151
 позиция курсора и, 166
 преимущество, 169
LISP, 233

M

Macintosh, компьютер, 206, 238, 246
 восстановление исходного
 состояния ОЗУ, 56
 однокнопочная мышь и, 248
Macromedia Flash, 201
Mathematica, программа, 233
Microsoft Office, 173
Microsoft Word, 69, 71, 126
 диалоговое окно с настройками
 и кнопкой запуска в, 160
 окно поиска, 157
Mobile Office, журнал, 20

N

NextWave (HP), 171
n-клавишный циклический буфер, 223,
 224

O

Oberon, 233
OLE (Microsoft), 171
OpenDoc (Apple), 171

P

PAD++, 190
Palm Pilot, 26, 128
PARC, 171, 246
Perl, 201
PL/I, 233

S

SCSI-соединения, 234
Smalltalk, 201, 232, 233
SNOBOL, 233
Sony, 215, 218
 радиоприемник 2010, 76
SwiftCard
 теория работы интерфейса для, 250
SwiftWare, 150, 158, 168, 209

U

UCSD Pascal, 233
UML Toolkit, 23
URL, 173, 245
USB, стандарт, 234

V

Vellum, 60, 61, 71, 80
 адаптивные палитры, 80
Visual Basic, 22, 155, 231, 233
Visual C++, 22

W

WEB, 233
Windows 2000 (Microsoft)
 адаптивные меню в, 94

X

Xerox PARC, 171

Z

ZoomWorld, 184
 изображения в, 196–199

A

авиакатастрофы
 вызванные поглощенностью, 44, 46
 предотвращение через устранение
 режима, 64
автоматизм, 41, 43
автоматы по продаже билетов, 87, 88
автомобиль на паровой тяге, 26
автопилот, 89, 90
автоповтор, 57, 221
автоформатирование в Microsoft Word,
 69
адаптация
 адаптивная палитра, 80
 адаптивные меню, 80, 94
адаптеры для перехода пола, 234, 235
Азимов, Первый закон робототехники,
 24
активизация, 133
 содержания, 130
алгоритм Бойера-Муура, 158
анализ задач, сравнительный, 99
анализ с использованием метода
 критического пути GOMS, 98, 99
Андерсон, Дж. Р., 30
аргумент, 104
архитектура, 9, 31
Аткинсон, Билл, 249

Б

Баарз, Бернард, 30, 43, 49
безопасность и вход в систему, 219
бессознательное и сознательное, 29
биты, измерение информации в, 111
броузеры, избранное в, 185
булев метод, 156, 169, 208
Бумгарнер, Джон, 221
быстрый поиск по строке, 158

В

ввод
 в заблокированный текст, 210
 видеоданных, 127
 знаков ударения, 223
 команд, 143
 скорость и, 100
 создание выборки и, 140
веб-сайты, 9
веб-страницы
 передача, 26
 программы обработки, 131
вертикально-ориентированные языки
 курсор удаления для, 167
Вейнберг, Гералд М., 234
видеомагнитофон, легкоустанавливаемые электронные часы в, 21
видимость, 94
 команды и, 139
 недостаток, 134
 пиктограммы и, 202
 состоятельность и, 85
 усиление с помощью надписей, 73
видимый элемент интерфейса, 85
внимание, 35
волоконная оптика, 234
временные интервалы в интерфейсе, 99
временные режимы, 71
время
 бесценность, 24
 запуска (загрузки) компьютера, 25
время обучения
 и естественный язык GOMS, 98
Всемирная сеть, 30
 портал ZoomWorld, 198
 перемещение в, 199
 увеличение масштаба одной из областей в, 199
вставка текста, методы в текстовом процессоре BRAVO, 248

встроенная программа для рисования, 170
второй пилот, 74
вход в систему, упрощение, 219
выборка, 133
выдвигающаяся палитра Ашлара, 80
выделение, 133
 отделение перетаскивания от, 141
 прозрачность выборки, 144
 содержания, 130
выключатели, 59
 трудности подбора подписи к, 58
выполнение задач, одновременное, 40
вырезать и вставить, 213
вычисления, 68, 96, 104, 116, 122, 214
выявление, 155

Г

генерация содержания, 130
гермафродитные проводные кабели, 237
 коаксиальный соединитель, выходы, 236, 237
 четырехпроводный соединитель, 236, 237
гибкие диски, 218
голосовые автоответчики, 20
графические
 данные, 147
 пользовательские интерфейсы, 20, 71, 107, 150–152
 запуск поиска в, 158
 пиктограммы и, 204
 редакторы и модель «существительное-глагол», 83
 фрагменты, 206
графическое устройство ввода данных, 54, 127, 129, 251
 двойной щелчок и, 100
 изменение масштаба и, 197
 квазирежимы при изменении масштаба и, 187
 кнопки, 54, 174
 перемещение курсора, 164
 состоящий из двух частей курсор, 164
Грудин, Дж., 227
ГУВ, 54

Д

двойной квазирежим, 136
двойной щелчок, 54, 100
деление на 0, 214

дело против согласования пользователь-
ских интерфейсов, Грудин (Grudin),
227

Денетт, Даниел, 31

действие-объект, 82

Дженнингз, Карла, 57

Джон, Бонни, 99

джойстик, 54, 64, 127, 130

диакритические символы, 223

диалоговые окна, 225, 226, 228

снабженное кнопками для поиска
и запуска, 160

для поиска с разделителями, 161

диапазон действий и режимы, 67

дилемма вытеснения, 170

дисковая операционная система (DOS),
252

дисковые накопители, 215

дисплеи

дизайн изображений, 243

эргономика, 28

дистанционно-управляемые летатель-
ные аппараты, 64

переключатели в, 66

пульт дистанционного управления
коммерческого аппарата, 64

длительность ответа, поступающего
от компьютера и действия пользова-
теля, 101

документ составной, 170

документация, важность ведения
при создании программ, 232

Дьюэл, Д. Дж., 41

Е

единицы взаимодействия, 159

единичность личности, 48

Единый коммерческий кодекс, 239

естественный интерфейс, 182, 183

естественный язык, 148

GOMS, 98

обычаи, 140

Ж

желаемая числовая клавиатура, 116

жесты, 169

модальные, 63

монотонность и, 90

определение, 57

сравнительный анализ задач и, 99

З

задержки, 50, 56, 221

и другие приемы работы с
клавиатурой, 221

закон Фитса, 97, 120, 124

ограничение, связанное с небольшо-
ми размерами целевых объектов,
168

прокрутка и, 142

стандартный текстовый курсор и,
162

закон Хика, 97, 123, 197

запрос, 131

затухание восприятия, 37, 50

звуковые файлы, 147

значение и информация, 115

значение неопределимо, 214

значения по умолчанию, 227

И

игры, 177

иерархическая файловая структура
устранение, 149

извлечение дискеты, 215, 217, 218

Инглбарт, Дуглас, 247

индикатор хода выполнения задачи, 102

Институт инженеров по электронике и
электротехнике, 214

инструменты по разработке
интерфейсов, 22

инструменты, с помощью которых
трудно прийти к новым идеям, 22

Интернет, 30, 125, 147, 158, 161, 175,
176, 185

интерференция, 40

интерфейс

для поиска, 152

жест и, 57

задержки и, 57

интуитивный, 182, 183, 210

как продукт, 24

количественный анализ, 97

монотонность и, 90

немодальный, 169

определение, 20

человекоориентированного, 25

поглощенность и, 45

приспособление под характеристики
личности пользователя, 81

проблемы с, 226

- интерфейс
 - проектирование перед разработкой, 23
 - самообучающийся, 210
 - сущность, 19
 - унификация, 125
 - «человек-машина», 20
 - эргономика сознания и, 28
- интерфейс Macintosh
 - задержки в, 221
 - принципы создания, 22
- интерфейс Windows
 - принципы создания, 22
- интерфейс для Хола
 - другие решения, 117
 - производительность, 114
- информационная производительность, 110
- информационно-теоретическая
 - производительность, 110, 116
 - диалоговое окно, 111
- К**
- кабели, 245
 - и режимы, 234
- кабина пилотов, 44, 73, 89
- калькулятор против компьютера, 171
- Кард, Стюарт К., 98, 102, 124
- карманные компьютеры, 26
- каталог элементарных действий, 129
- катодная трубка в телевизионном кинескопе, 26
- квазимодалный, 78
- квазирежим
 - для поиска, 158
 - наложение символов, 223
 - при масштабировании, 187
 - режимы и, 77
- квантификация, 97
 - законы Фитса и Хика и, 119
 - измерение эффективности интерфейса и, 109
 - модель скорости печати GOMS и, 98
 - количественные ориентиры, 119
 - количественный анализ интерфейса, 97
- кегель и пиктограммы, 204
- клавиатура, 127–140, 151, 159, 164, 172, 205, 221–223, 251
 - задержка перед началом повтора и другие приемы работы с, 221
 - интерфейсы и, 57
 - комбинации клавиш, 91
 - полная, 136
 - разработка, 29
 - эргономика и, 28
- клавиши, 54, 60, 63, 135, 154
 - <Caps Lock>, 77
 - LEAP Again, 210
 - пробел, 50, 55
- Кларк, Ларри, 60
- ключевое слово, 210
- книга о пиктограммах (Уильям Хортон), 204
- кнопки, 86
 - Заккрыть (Close), 132
 - запуска, 158
 - диалоговое окно с несколькими, 160
 - с изменяемыми названиями, 74
 - с надписями, 73
 - чем меньше, тем лучше?, 74
- Кнуд, Дональд Э., 232
- коаксиальные кабели, 235
- когнетика, 28, 30, 233, 244
 - локус внимания и, 27
 - разработка интерфейсов и, 201
 - эргономика и, 27
- когнитивная теория сознания (Б. Баарз), 30
- когнитивное бессознательное, 29–32, 244
 - свойства, 35
 - формирования привычек, 41
- когнитивное сознательное, 32, 33, 244
 - и когнитивное бессознательное, 35
 - ограничения, 43
 - свойства, 35
- когнитивные различия, 131
- команда
 - Backspace, 63
 - DISK, 217, 219, 252
 - Заблокировать, 144
 - Заблокировать с паролем, 144
 - Копировать, 152
 - Отменить, 134, 136
 - Переместить, 152
 - Повторить, 134
 - Разблокировать, 144
 - Разблокировать по паролю, 144
- командная строка, 72
- команды, 136
 - видимость и, 139

освобождение от приложений, 126
 трансформаторы и, 171–174
 улучшенный метод вызова, 138
 комментарии, 232, 233
 компьютер
 время запуска, 25
 как слуга, 227
 калькулятор против, 171
 необычное поведение и реакция
 пользователя, 47
 получение полезных результатов и,
 226
 проблемы, 17
 трата времени и, 25
 человекоориентированный
 интерфейс для, 128
 компьютерные игры, 86
 компьютерные интерфейсы
 аппаратное обеспечение и, 127
 когнетика и усовершенствование,
 233
 компьютерные системы на основе
 рабочего стола, 51
 копирование содержания, 130
 космология, 34
 кратковременная память, 37, 47, 59, 85
 краткость и ясность, 241
 критерии эффективности, 97
 курсоры, 54, 132, 133, 249–251
 PARC, 167
 закон Фитса и, 120
 закон Хика и, 123
 используемые в Vellum, 61
 перетаскивания, 142
 размещение и компьютер Canon Cat,
 51
 состоящие из двух частей, 164, 167
 стандартные межсимвольные, 162
 удаления для двумерной печати, 167
 форма курсора и методы выделения,
 162
 функция LEAP и, 166, 251
 части вставки и удаления, 167
 Кэй, Алан, 171

Л

лабиринты, противоположность, 185
 Ландауэр, Томас К., 169
 личные настройки, 69
 локальный прыжок, 168

локус внимания, 35, 63, 135, 140, 244
 выбор метода и, 91
 истоки, 47
 кнопки запуска и, 160
 модальность интерфейса и, 63
 модель «существительное-глагол»
 и, 82
 одновременное выполнение задач
 и, 40
 побочный эффект и, 141
 привычные элементы интерфейса
 и, 79
 режим и, 61
 сингулярность, 43
 совпадение с фокусом, 135
 формирование привычек и, 39
 Люк (The Manhole), программа, 183
 Льюис, Томас, 37–38

М

магнитно-резонансное представление,
 метод, 33
 макрос, 68
 Мартин, Пам, 101
 масштабируемая среда,
 эффективность работы в, 189
 масштабируемый интерфейс, 201, 245
 медицинские карты,
 увеличение масштаба, 193
 межсимвольный курсор, 166, 168
 ментальные операторы,
 расстановка (таблица), 103
 меню, 19, 20, 53, 58, 59, 69, 72, 80, 123,
 137, 139, 225, 227
 адаптивные, 80
 в стиле Apple Macintosh и Microsoft
 Windows и закон Фитса, 121, 122
 системы, основанные на, 90
 структуры, 123
 метод
 GOMS-анализа скорости печати
 для сравнения ГУВ и функции
 LEAP, 165
 организации пространства
 информационной плоскости, 188
 «существительное-глагол»
 для построения команд, 252
 щелкнуть и перетащить, 247
 микрофоны, 127
 миф о дихотомии «новичок-эксперт», 92

многовариантный переключатель, 68
многосимвольные команды, 72
многострочные комментарии, 233
многоштекерные соединители, 235
модальное диалоговое окно поиска, 153
модальные ошибки, 61, 62
 уменьшение, 62
 частота, 61
модель
 «глагол-существительное» против
 модели «существительное-
 глагол», 81
 скорости печати GOMS, 98
 «существительное-глагол» против
 модели «глагол-существительное»,
 81
модификация содержания, 130
монотонность, 89, 95, 244, 252
 монотонный интерфейс, 90, 91
Моран, Томас П., 97, 102, 124
Мэйхью, Дебора, 28, 97, 204
мышь, 54, 182
 двойной щелчок, 100
 при работе в масштабируемой среде,
 202
 функция перетаскивания, 142

Н

навигация, 181
 ZoomWorld, 184
 против свободного пространства, 241
надписи
 на кнопках мыши, 249
 против пиктограмм, 204
 усиление видимости с помощью, 73
 ясность и краткость, 201
нажатие, 54, 55
наложение символов, 223, 224
 модальное, 223
невидимые документы и регулирование
 доступа к содержанию, 190
невидимый элемент интерфейса, 85
немодальность, 72, 95, 251
 немодальный интерфейс, 63, 90, 92,
 169, 244
непосредственное восприятие, 37
Норман, Дональд, 12, 30, 62, 97, 243
Нойк, Имэнюэл, 197
Ньюэлл, Аллэн, 102, 124

О

«О лице» (Cooper, About Face), 47
облегченные версии программных
 пакетов, неудачи с, 175
обратимость и модель
 «существительное-глагол», 82
обратная связь от покупателей, 224
обратная совместимость, 89
объект поиска, 120, 121, 153
объекты
 локус внимания и, 135
 одинаковость, 129
 подсветка, 132
 применение действий к, 81
 экранные состояния, 143
объясненное сознание (Денетта), 31
обычай естественной речи,
 подстраивание машины под, 140
ограничения Туллиса, 242
однопользовательская система,
 качество и простота взаимодействия,
 17
однокнопочная мышь, 54, 249
 история и будущее, 246
оконная парадигма, 171
операционная система, 16
 Macintosh, 9
 Windows, 9
 устранение, 251
определение задачи, 23
ориентация на пользователя
 и на человека, 21
ориентация разработок на нужды
 покупателей, 21
ориентиров, 185–188
орфографическая проверка, 128
освобождение команд от, 126
основы, важность, 15
осознание, 35
осциллограф, 203
 Tektronix, 75
 Флюка, 74
отделение интенсивной терапии,
 изображение в масштабируемом
 интерфейсе, 191
 медицинская карта, 190
 уменьшение масштаба, 194
отображение информации,
 принципы Эдварда Туфта, 241

отпечатки

голоса, 220

пальцев, 220

ошибки

двусмысленные обозначения и, 55

порождаемые режимами, 60

при двойном щелчке, 100

с выключателями, 59

связанные с работой интерфейса,
предотвращение, 47

П

память, 32, 34, 37

параллельная обработка, 48

параметры изменения компьютерной
рабочей среды, 177

пароли, 219

педаль тормоза и газа меняются
местами, 39

Пенроуз, Роджер, 31, 43

перевод из одной температурной шкалы
в другую, 104, 114

переключатели, 59, 66, 67, 135

диалоговое окно с, 105

обозначенные прилагательными, 59

перекрывающиеся окна, 171

перемещение содержания, 130

перенастройка параметров программы,
179

перетаскивание объекта (drag-and-drop),
54, 89, 101

переход LEAPing, 168

перцептивная память, 37

пиктограммы, 53, 86, 150, 202, 226

библиографические, 208

особенно таинственные, 203

планшет для письма, 127

планшетный карандаш, 54

побочные эффекты, устранение, 141

поглощенность, 45

подсветка, 132

подтверждение годности продукта к
продаже, отказ от, 240

подтверждения и фиксированные
ответы, 43

позитронно-эмиссионная томография,
33

позиционные подсказки, 185

познаковое взаимодействие, 159

поиск

интерфейс для, 152

по строке, 153

быстрый, 158

механизмы поиска и, 154

пошаговый, 154

с разделителями, 153–155

полнотекстовая система поиска, 147

полный отказ системы, 178

пользователь

письмо от, 224

установка ритма взаимодействия, 26

пользовательские интерфейсы,

принципиальные недостатки в
(*см. также* интерфейс), 9

пользовательские настройки, 68, 69

время, затрачиваемое на изучение
и использование, 70

помехи и поглощенность, 46

портативные устройства, 9

последовательности символов, необхо-
димность единообразия при вводе, 157

пошаговый поиск, 154–162

правила

редактирования, универсальные, 128

для целей, объектов, методов

и выделения, 98

правостороннее удаление, 163

прерванная работа, возобновление, 50

прилагательные, 59

приложения, 67

применение действий к объектам, 81

примеры расчетов по модели GOMS, 104

временные интервалы, 99

интерфейс для Хола

вариант 1. Диалоговое окно, 105

вариант 2. ГИП, 107

принцип

масштабируемого интерфейса, 185

замена броузера, 200

управления, 219

проверка положения переключателей
в дистанционно управляемых
летательных аппаратах, 65

программирование, учтливое, 177

программисты и ведение документации,
231–232

программное обеспечение, 9

лицензии для, 239

неиспользуемые элементы, 176

продажа отдельными командами,
175

программы
 для оптического распознавания
 знаков, 174
 цифровой фильтрации, 174
продуктивность, 23
прозрачное диалоговое окно, 145
производительность
 измерение, 109
 информация и, 110
 термодинамика и, 110
прокрутка, 141, 162, 187, 197, 249, 251
 замена, 142
пропускная способность, 26
простота, 94, 125, 244
 задач, 16, 20
 и модель «существительное-глагол»,
 82
профессиональные стандарты, 239
процесс Маркова, 113
прямоугольный курсор, 166
психология компьютерного программи-
 рования (Вейнберга), 234
пси-эффект, 186

Р

работа, священность, 24
равновероятные сообщения, 115
радиоприемник, 756 76
разветвленный интерфейс, 118
разделители, 104, 148, 151
 в шаблоне поиска, 153, 156
разработка, влияние на, 27
разработка интерфейсов, 9, 16, 19, 20,
 184
 без режимов, 71
 когнитивная технология и, 201
 количественные законы и, 119
 однопользовательских, 17
 оценка процента появления ошибок
 при использовании данной модели
 интерфейса, 106
 применение Первого закона
 робототехники Азимова, 24
 принцип масштабирования и, 201
 режимы и, 53
 сенсорное восприятие и, 37
с точки зрения общего цикла
 разработки, 23

разработка интерфейсов
 сравнительная оценка между интер-
 фейсами по уровню эффективности
 использования, 100
 стандарты, 238
 теория информации и, 109, 114
 упрощение, 126
 формирование привычек и, 37, 39
 человеческие качества и, 92
 этика и управление, 230, 237
разработчики интерфейсов, 21, 98
 видимость и, 86
 дихотомия «новичок-эксперт» и, 92
 общие законы психологии и, 22
 процесс разработки проекта и, 238
разрывы страниц, 148
распределение динамической памяти,
 145
расстояние и закон Фитса, 120
растровые изображения, 174
расчеты по модели GOMS, 102
расшифровка, 212
режимы, 53, 57, 244
 кабели и, 234
 квазирежимы и, 77
 определение, 63
 предотвращение (авиа)катастроф
 через устранение, 64
 пружинный, 77
 с запертой пружиной, 77
 регулирования климата, 74
 сокращение ошибок с помощью
 устранения, 248
 удерживаемый пользователем, 77
 устранение, 251
решение дилеммы вытеснения, 170
рисунки, проблемы использования, 189
ритм взаимодействия, установка, 26
робототехника
 Первый закон Азимова, 24
ручки, 86
 управления в дистанционно
 управляемых летательных
 аппаратах, 66

С

самолеты и кнопки, которые меняются
 за одну ночь, 73
самообучающийся интерфейс, 210
САПР, 60

- Свайнхарт, Дан, 170
световое перо, 54
свободное пространство
 против навигации, 241
сенсорные
 графические планшеты, 130
 панели (тачпады), 248
 экраны, 127
сертификация интерфейсных технологий, 239
символы разделения, 149
синтаксис для команд, 140
система BART и отсутствие
 состоятельности, 86
система оценки качества интерфейсов, 240
системное окружение и среда
 разработки, 230
системные сбои, 24
системы автоматического проектирования, 60
сканирование, 174
скорость
 информационная производительность и, 111
 модель «существительное-глагол» и, 82
 пошагового поиска, 158
 работы функции LEAP, 165
Смит, С. Ф., 41
сноски, увеличение масштаба, 189
совместная среда ПМИ (ZIP), 190
содержание
 квазирежимы и, 81
 определение, 53
 структура как часть содержания, 149
соединители питания, 235
сознание, 30
 эргономика, 28
 модели человеческого разума и, 30
сознательное и бессознательное, 34
сообщения
 об ошибках, 47, 214
 прозрачные, 145
 оценка объема информации в, 113
 пользователю, 214
состоятельность и видимость, 85
сохранение
 образов непосредственного восприятия, 37
 работы, 24
 специальные символы, 157
 справочные системы, 210
 спящий режим, 26
 сравнительная оценка между интерфейсами по уровню эффективности их использования, 100
стандарты
 FireWire, 234
 USB, 232
столы, стулья и эргономика, 28
строка, 104
субъективные оценки и интерфейсы, оптимизирующие продуктивность, 71
сходство, 128
- Т**
- тачпад, 54
текст, 153
 вставка/удаление в текстовом редакторе BRAVO, 247
 вырезать и вставить, 213
 как визуальная подсказка, 204, 226
 команды изменения, 144
 перетаскивание (drag-and-drop), 142
 шифрование, 212
текстовые
 процессоры, 51, 131, 170, 251
 редакторы, 170
 файлы
 содержание как лучшее имя, 147
телевизоры, 26
теория информации, 115
 и разработка интерфейсов, 114
теория работы интерфейса для SwyftCard, 250
терминология и условные обозначения, 53
термодинамика, определение производительности в, 110
термометр, 107
Теслер, Ларри, 171
тестирование, 204, 252
 функции LEAP, 165
трансформаторы и команды, 173–176
трансформация содержания, 130
трекбол, 54, 249
Туфт, Эдвард, 241

У

Уайлдстром, Стив, 239
увеличение масштаба, 187
 списка, 200
удаление
 текста в текстовом процессоре
 BRAVO, 247
 содержания, 130
Уинтер, Джеймс, 216
указание, 132
 содержания, 130
 текста с помощью ГУВ, 164
указатель в Vellum, 61
уменьшение
 избыточности и сложности, 179
 количества ошибок в модели
 взаимодействия, 82
унификация, 125
 имена файлов и файловые
 структуры, 145
 команды и трансформаторы, 173–176
 ликвидация приложений, 169
 позиция курсора и клавиша LEAP,
 166
 поиск строк и механизмы поиска, 152
 форма курсора и методы выделения,
 162
 элементарные действия и, 127
 каталог, 129
управление разработкой интерфейсов,
 237, 243
управленческие функции, использова-
 ние квазирежимов для, 81
управляющая клавиша, 56, 205
условные обозначения
 двусмысленность, 55
 терминология и, 53
устройства
 ввода, компьютерные, 127
 отличающиеся от клавиатуры, 113
 распознавания речи, 127
учебные примеры
 приведение проблемной модели «гла-
 гол-существительное» к модели
 «существительное-глагол», 83
 сообщения, 216
учитивое программирование
 приложения как гости, 177

Ф

Фауэлз, Боб, 69
файловые имена, 126
 описание, 146
 устранение, 147
 файловые структуры и, 145
фиксированные
 задержки, 221
 ответы и подтверждения, 42
фильтры, 174
флажки, 59
фокус, 135
 против локуса, 36
фонарик, 58, 68
форматирование дисков, 250
формирование
 модуля (chunking), 57
 привычек, 37, 40, 41, 244, 251
 закон Хика и, 123
 кнопки и, 75
функциональность и простота, 240
функциональные клавиши, 74

Х

Хорн, Боб, 204
Хортон, Уильям, 204, 206
Хоурд, Брайен, 249
хроническое стрессовое нарушение, 238
хеширование, 145

Ц

цвет и пиктограммы, 205
цветной дисплей, 127
цветовые палитры, 205
центральная ямка, 36, 61, 164
циклический буфер, 223, 224
циклический поиск, 159, 161

Ч

часовой механизм, 31
частота появления ошибок
 в разветвленном интерфейсе, 119
часы с обратным отсчетом, 68
«человек-машина»
 взаимодействие и количественные
 методы, 9, 98
 интерфейс, 27
 закон Фитса и, 120
 локус внимания и, 27

«человек-машина»

интерфейс

модальный, 63

преодоление проблем в, 10

революция в разработке, 245

формирование привычек, 39

человекоориентированные среды

программирования, 230

важность ведения документации

при создании программ, 232

системное окружение и среда

разработки, 230

человекоориентированный

интерфейс, 19

ввод с клавиатуры в, 140

единственный диапазон в, 67

идеальный, 39

определение, 25

способы и средства помощи, 209

трансформаторы и, 175

метод удаления, 213

Ш

шаблоны, 153

поиска, 162

шифрование, 212

Шнейдерман, Бен, 97

Шэннон, Клод Э., 113, 124

Щ

Щелкнуть (click), 54

Э

эвристические подходы, 22

экранная блокировка и разблокировка,
144

экранные состояния объектов, 143

электронная почта, 211

America Online, 62

электронные таблицы, 131, 171, 188, 212

элементарные действия, 129

команды, 136

подсветка, указание, выделение, 132

унификация и, 127

экранные состояния объектов, 143

эмоциональные потребности, 81

эмпирическое сознательное

и бессознательное, 29

энергонезависимая память, 52

эргономика и когнетика

что мы можем и не можем делать, 27

этика и управление разработкой

интерфейсов, 243

эффективность

символьная, 116

интерфейса, измерение, 109

Я

языки, читаемые слева направо

и справа налево, расположение

курсора удаления в, 167

ясность

краткость и, 241

функции, 94

чрезмерное использование графики

и, 207