Валерий Дмитриевич Магазанник

Человеко-компьютерное взаимодействие

«Текст предоставлен правообладателем» http://www.litres.ru
«Человеко-компьютерное взаимодействие: Учебн. пособие»: Логос; Москва; 2007
ISBN 978-5-98704-241-0

Аннотация

Освещены теоретические подходы и практические аспекты человеко-компьютерного взаимодействия. Рассмотрены требования к компоновке компьютеризированных рабочих мест и планировке офисных помещений, а также к мониторам и средствам ввода информации. Описаны современные средства моделирования рабочих мест. Видное место отведено подходам, методам и инструментарию формирования и оценки пользовательского интерфейса. Раскрыта роль итерационного прототипирования интерфейса, рассмотрены виды прототипов и, что особенно важно, существующие программные пакеты для прототипирования и их сравнительные возможности. Даны методы мультимедиа при создании пользовательского интерфейса. Особое внимание уделено построению навигационных структур, в том числе с помощью средств мультимедиа, что наиболее актуально в web-сайтах. Представлены примерная учебная программа дисциплины и два приложения, содержащие основные понятия человеко-компьютерного взаимодействия, а также стандарты ISO, регламентирующие требования к пользовательскому интерфейсу.

Для студентов высших учебных заведений, обучающихся по направлению «Прикладная информатика», а также по другим направлениям и специальностям в области информатики и информационных технологий. Может использоваться при подготовке кадров по широкому кругу направлений и специальностей, связанных с разработкой и использованием средств вычислительной техники и программного обеспечения. Представляет интерес для практических специалистов в области технических средств и программного обеспечения компьютерных систем.

Валерий Дмитриевич Магазанник

Человеко-компьютерное взаимодействие

ПРЕДИСЛОВИЕ

Предлагаемое учебное пособие посвящено сравнительно молодой и очень актуальной области – человеко-компьютерному взаимодействию. Скорость распространения компьютеров существенно превышает рост компьютерной грамотности населения. Это приводит к неэффективному использованию программного обеспечения и, как следствие, к растущему разрыву между возможностями новых программных продуктов и уровнем знаний и навыков потенциальных пользователей. Происходит это в значительной степени по причине несоответствия предлагаемых в пользовательском интерфейсе средств деятельности человека с его представлениями, знаниями, привычками. Экономический аспект этой проблемы весьма впечатляет, составляя, по некоторым данным, сотни миллионов долларов, т.к. значительная часть оплаченных возможностей программных продуктов реально не используется. Отрицательно влияет это и на индустрию программного обеспечения, уменьшая и делая все менее ясными сегменты рынка, на которые рассчитаны новые разработки.

Указанные причины обусловили интенсивный рост числа и диапазона работ, посвященных HCI (принятая аббревиатура человеко-компью-терного взаимодействия – Human-Computer Interaction). За последние 15-20 лет эти исследования выделились в самостоятельную междисциплинарную область. К ближайшим смежным дисциплинам можно отнести, конечно, прикладную информатику, психологию, физиологию труда, эргономику и ряд других. Специалисты в области HCI активно востребованы во всем мире, и в настоящее время без них немыслимы сколько-нибудь серьезные программные разработки.

Учебное пособие охватывает основной диапазон вопросов человеко-компьютерного взаимодействия. Автор собрал и обобщил большое количество публикаций, материалов конференций, периодики, сведений, размещенных на соответствующих сайтах и форумах, касающихся чело-веко-компьютерного взаимодействия, использованы также учебные материалы по HCI ряда университетов США.

Много внимания уделено профилям пользователей и вообще подходам и методам описания характеристик потенциальных пользователей. Изучение этой темы для студентов, получающих образование в той широкой области, которую часто называют компьютерными науками, будет особенно полезно, ибо учит с самого начала разработки любого программного продукта подробно представлять себе будущего пользователя, тщательно его описывать и постоянно держать его образ в голове.

Подробно описываются показатели, методы и процедуры оценивания интерфейсов на каждой стадии разработки системы. Центральный аспект оценки интерфейса – периодическое юзабилити-тестирование изложено достаточно полно. Построению прототипов интерфейса посвящен большой раздел, и это оправданно. Существующие программные пакеты построения прототипов вносят много специфики и делают про-тотипирование захватывающим творческим процессом, образующим вместе с юзабилити-тестированием стержень разработки пользовательского интерфейса.

К несомненным удачам автора следует отнести хорошее структурирование очень сложной темы о возможностях и методах мультимедиа при создании пользовательского интерфейса. Сначала отдельно и подробно рассматриваются визуальная и акустическая среды с классификацией форм представления информации. Затем показаны средства и требования к интеграции этих сред.

В целом учебное пособие будет очень полезным для студентов, получающих образование в области вычислительной техники, программирования и информационных технологий.

А. М. ЕМЕЛЬЯНОВ

доктор физико-математических наук, профессор, полный профессор Государственного университета штата Джорджия (США), Департамент компьютерных и информационных наук

ВВЕДЕНИЕ

Человеко-компьютерное взаимодействие – сравнительно молодая область междисциплинарных исследований.

Наиболее бурное ее развитие пришлось на последние 10-15 лет и связано с резким увеличением числа пользователей компьютерами и с возрастанием роли последних в жизни и деятельности людей. Круг специалистов, занимающихся разными аспектами взаимодействия человек-компьютер, достаточно велик. Прежде всего это дизайнеры (особенно web-дизайнеры), педагоги и наиболее продвинутые программисты. Практика, однако, показывает, что человеко-компьютерное взаимодействие de facto стало самостоятельной профессией, ибо это область столь специфичная и обширная, что ни одна из традиционно существующих дисциплин не охватывает ее всю целиком.

Область человеко-компьютерного взаимодействия, развиваясь, вовлекает в себя все новые сферы знания. Грубо ее можно разделить на две неравные части: первая, сравнительно небольшая, относится к планировке офисных помещений и к особенностям компоновки рабочего места человека при работе с компьютером; вторая, основная часть, – к формированию пользовательского интерфейса (ПИ).

В России весьма активно развивается индустрия заказной разработки программного обеспечения, где все большее внимание уделяется интерфейсу. Это внимание обусловлено ужесточающейся конкуренцией на рынке программного обеспечения (ПО) и как следствие акцентированием внимания на нуждах потенциальных покупателей. Кроме того, развиваются Интернет и разработка сайтов. Возросший спрос рождает предложение: в результате появляется большое количество web-дизайне-ров, совершенно не знающих принципы конструирования интерфейсов. Таким образом, возникли две категории людей (программисты и web-ди-зайнеры), профессиональный успех которых напрямую связан с качеством создаваемого ими пользовательского интерфейса.

Если три года назад среди объявлений о поиске персонала встретить вакансию разработчика интерфейсов было практически невозможно, то сегодня подобные объявления появляются не реже чем два-три раза в месяц. О росте интереса также свидетельствует и содержание сайта http:// HYPERLINK «http://forum.usability.ru/»forum.usability.ru/. Если раньше там общался очень узкий круг профессионалов, то теперь все чаще появляются новички.

Сегодня Интернет стал для многих людей мощной побудительной причиной покупать компьютеры. И все громче раздаются критические голоса об интерфейсе, трудно понимаемом непрофессионалами, не сильно продвинутыми пользователями, т.е. большинством людей. Можно с уверенностью предсказать, что дальше количество пользователей будет увеличиваться только за счет любителей.

К сожалению, как всякое модное слово, термин «пользовательский интерфейс» незамедлительно начали использовать в качестве рекламного слогана, в результате чего его смысл стал размываться. В данное понятие входит не только и не столько картинка на экране (трехмерная, анимиро-ванная, какая-то другая), сколько способы взаимодействия пользователя с системой. Дизайн интерфейса имеет подчиненное значение, главная же цель ПИ – облегчить работу пользователя.

Хороший интерфейс похож на удобную обувь: никто его не замечает, а если обратить на него внимание, в ответ получишь равнодушное «Ну и что такого?». Зато плохой интерфейс у всех на виду и на устах. В самом деле, хороший интерфейс пользователями замечается подсознательно, и, когда он нравится, симпатии переносятся и на функциональную часть программы.

Основная функция хорошего интерфейса – сокращение информационной нагрузки на пользователя за счет упорядочения данных и знаний. Один из основателей направления «информационная архитектура» Ричард Сол Вурмен еще 10 лет назад писал: «На берега цивилизованного мира обрушивается информационное цунами. Это гигантская волна разрастающихся данных наплывающих как пена прибоя – произвольная, неуправляемая и ни с чем не согласованная. Ни одну их часть нельзя связать с другой и ни к одной нельзя применить

уже готовые методы построения структур. А теперь хорошая новость… В океане появился волнолом, возникший в последние моменты плавания по XX веку. Этот волнолом встал на пути информационного цунами и придал волне более упорядоченное движение, при котором легче стало искать ответы на вопросы и вырабатывать идеи. Эта преграда состоит из нового поколения дизайнеров…, чьим страстным желанием стало сделать запутанное ясным». Волна «информационного цунами», о которой писал Р.С. Вурмен, продолжает нарастать ежедневно, и сегодня каждый, кто использует информационные технологии (а это фактически весь цивилизованный мир), должен стать преградой на пути этой волны.

Ряд интерфейсных проблем связан с конкурентной борьбой на рынке программ. Пожалуй, главная из них – какие формы должно принимать авторское право на интерфейсные решения. С одной стороны, ясно, что придумать и реализовать хороший интерфейс – очень сложная задача, и авторы такого интерфейса должны получить не только моральное вознаграждение. С другой стороны, если защитить такое решение патентом с последующими лицензионными выплатами, это может спровоцировать авторов новых продуктов искать свои, нехоженые и зачастую худшие пути в интерфейсе. В качестве яркого примера можно попробовать представить себе последствия патентования использования клавиши F1 для вызова справки.

Лицензионная защита интерфейсных решений – прямой путь к тому, что одни и те же интерфейсные функции будут реализовываться в разных продуктах по-разному, а это не в интересах пользователя. Как бы мы ни относились к фактической монополии фирмы Microsoft на рынке операционных сред, следует отметить, что положительной чертой этой монополии явилась фактическая стандартизация интерфейса под Windows.

Имеется множество технических решений, учитывающих человеческий фактор в целом и различия между людьми в частности. Например, бордюры в местах перехода во многих странах делают более низкими, чтобы облегчить людям (а особенно инвалидам, пожилым и детям) переход. Мощение около бордюров делают несколько иное, скажем гребенчатое, более грубое, чтобы люди с ослабленным зрением почувствовали близость бордюра и не споткнулись. Множество современных зданий имеет двери, автоматически открывающиеся при подходе к ним, и лифты, снабженные как визуальным, так и звуковым сигналом при достижении нужного этажа. Благодаря таким инженерным решениям и заданиям здания и лифты становятся доступны для более широкого круга людей, особенно же для людей с физическими недостатками; важно, однако, что и остальным людям это облегчает жизнь. Круг потенциальных пользователей расширяется, что и является конечной задачей человекоориенти-рованного проектирования. Никто не может быть исключен из рассмотрения как потенциальный пользователь. Аналогично разрабатываемое программное обеспечение, как и любой товар, рассчитанный на широкое потребление, должно удовлетворять запросам как можно большего круга потребителей.

Стоимость разработки ПИ колеблется обычно от 5 до 50% стоимости всего программного продукта. И это вполне нормально: технологии у всех схожи, а бизнес-эффект достигается во многом за счет качества интерфейса. Опыт показывает, что объем и глубина работ, а значит, и их стоимость могут варьировать в отдельных проектах очень сильно. Выгоды от разработки хорошего ПИ – это гарантия успешности продукта, снижение затрат на разработку (как это ни парадоксально), удешевление поддержки продукта, увеличение конкурентных преимуществ, снижение вероятности критических ситуаций; самое заметное и очевидное – увеличение экономического эффекта от использования продукта.

Иногда спрашивают: в чем отличие разработчика интерфейсов от обычного программиста? Дело в том, что разработчик интерфейсов – это программист, который привык иметь дело не только с программами и машинами, но и с людьми. Разработчик интерфейсов должен уметь использовать результаты социологических исследований, проводить интервью, полевые исследования (т.е. наблюдение за работой пользователей в естественной обстановке), хорошо знать работы в области человеко-ком-пьютерного взаимодействия, знать и уметь еще тысячу вещей, которые не имеют прямого отношения к чисто техническим дисциплинам, а находятся на стыке дизайна, психологии, социологии и информационных технологий. Довольно трудно ждать от обычного программиста наличия всех этих знаний и навыков, потому что его профессиональная деятельность имеет весьма отдаленное отношение к потребностям,

особенностям и слабостям людей.

Культура профессионального проектирования интерфейсов в России только начинает развиваться. Этой специальности почти не обучают в российских вузах, и настоящих специалистов в данной сфере всего несколько десятков на всю страну. Потому приходится заниматься переподготовкой имеющихся специалистов. Но осознание того, что разработчик пользовательских интерфейсов – это область, требующая знаний, навыков и образования, несколько выходящих за рамки знаний, навыков и образования программиста или графического дизайнера, поможет более эффективно выбирать сотрудников для переподготовки. Можно зафиксировать растущий интерес в России к профессии «разработчик пользовательских интерфейсов».

Настоящее учебное пособие во многом является переработкой учебного пособия, вышедшего в 2005 г. в Твери (издательство «Триада»). Опыт его использования показал, во-первых, потребность освещения в нем таких актуальных направлений, как роль мультимедиа в разработке пользовательского интерфейса, во-вторых, необходимость обновления ряда разделов в связи с появлением новых технических устройств и программного обеспечения да и в связи с интенсивным развитием самого направления – вышло много новых книг, статей, материалов конференций по человеко-компьютерному взаимодействию. В настоящем издании учтены оба эти фактора.

Учебное пособие содержит краткие описания основных задач и инструментов человеко-компьютерного взаимодействия. Структурно выделены основные проблемы, хотя полнота их раскрытия не всегда соответствует их реальному значению в разработке ПИ. Это объясняется, ограниченным объемом пособия, а также, неустоявшимся предметом изложения, спорностью и подчас отсутствием доказательности многих излагаемых принципов и положений. В процессе написания пособия автор использовал множество книг, статей и данных разных конференций, форумов и сайтов по рассматриваемым темам, ну и, конечно, некоторый личный опыт.

Список контрольных вопросов и использованной литературы приводится в конце каждого раздела. Завершается учебное пособие приложениями, которые содержат примерную учебную программу по человеко-компьютерному взаимодействию, список терминов и понятий, а также перечень стандартов ISO в области человеко-компьютерного взаимодействия.

ТЕМА 1. РАБОЧЕЕ МЕСТО ЧЕЛОВЕКА ПРИ РАБОТЕ НА КОМПЬЮТЕРЕ

Изучаемые вопросы:

• Влияние разных рабочих поз на утомление и возникновение скелетно-мышечных расстройств.

• Общая компоновка рабочего места и планировка рабочего помещения.

• Расположение монитора и клавиатуры на рабочем месте.

• Требования к монитору и клавиатуре.

• Преимущества и недостатки различных устройств ввода информации.

• Конструктивные особенности рабочего кресла, обеспечивающие активный комфорт для человека.

• Рекомендуемые характеристики рабочего кресла, обеспечивающие активный и пассивный комфорт для человека.

• Рекомендуемые параметры рабочей поверхности.

• Требования к рабочему помещению.

1.1. Общая компоновка рабочего места

Полные требования к компоновке рабочего места, планировке рабочих помещений можно найти в литературе по индустриальной эргономике, в специальных разделах, посвященных антропометрическим размерам и прикладной биомеханике. Здесь мы остановимся только на особенностях компоновки рабочих мест при работе на компьютере. Обычно в книгах по

человеко-компьютерному взаимодействию такой раздел отсутствует, все посвящено пользовательскому интерфейсу. На наш взгляд, это неправильно, ведь человеко-компьютерное взаимодействие, в отличие от разработки интерфейса, включает эту тематику. Кроме того, резкий рост числа пользователей в последние годы показывает все возрастающую значимость этой проблемы, что проявляется в увеличении числа молодых людей со скелетно-мышечными проблемами, в особенности с проблемами позвоночника.

Сидячая работа (особенно продолжительная) вредна человеку в принципе: вы сутулитесь или подаетесь вперед, ваш позвоночник деформируется, травмируя диски. Вы поднимаете плечи и сгибаете руки, держа их в напряжении, и, естественно, они начинают болеть. Пережимая сосуды, вы перегружаете сердце, а о хронических растяжениях сухожилий кистей рук и постоянно ухудшающемся зрении можно и не говорить. Поза, следовательно, производительность труда и здоровье зависят в значительной мере от размеров и дизайна рабочего места.

Человек за свою жизнь проводит сидя в среднем около 80 000 ч! На работе, во время учебы, еды, в автомобиле, в самолете, у экрана телевизора, в театре и даже в свободное время мы сидим, из нашей жизни исчезает движение. Природа же создала Человека динамичного, находящегося в постоянном движении, для которого состояние покоя в целом нехарактерно. Сидячая поза увеличивает нагрузку на мышцы спины, и только когда человек откидывается на спинку, нагрузка несколько уменьшается (рис. 1.1). Видно, что 100% нагрузки имеет место в положении стоя, плечи отведены назад, руки по швам, т. е. ровная прямая стойка (рис. 1.1, а). Типичная поза при работе сидя – на краешке стула, но спина ровная (рис.1.1, б), и нагрузка сразу возрастает до 130%.

Еще более типичная поза при работе на персональном компьютере (ПК): на краю сиденья, сильно наклонившись вперед (рис. 1.1, в). И только при откидывании на спинку кресла нагрузка уменьшается и становится равной 75% (рис.1.1, г).

Рис. 1.1. Нагрузка на мышцы спины: *а* – 100%; *б* – 130%; *в* – 200%; *г* – 75%

При длительном сидении возникают боли в позвоночнике, ногах, головные боли. Вот статистика жалоб, причиной которых становится долгое неправильное сидение: головная боль – 14%; боль в шее и лопатках – 24; боль в позвоночнике – 57; проблемы в области копчика – 16;

боль в бедрах – 19; боль в коленях и стопах – 29%.

Общий вид правильно организованного рабочего места при работе на компьютере показан на рис.1.2.

Расположение монитора

Монитор, как правило, располагается чрезмерно близко. Рекомендуется ставить монитор на расстоянии вытянутой руки, т. е. располагать его на расстоянии 40-100 см от глаз. Существует ряд научных теорий, по-разному определяющих значимые факторы и оптимальное расстояние от глаз до монитора. Монитор должен располагаться на 10-20 см ниже горизонтальной линии взора (чтобы смотреть на него немного свысока) и быть наклоненным чуть вверх, как книга при чтении.

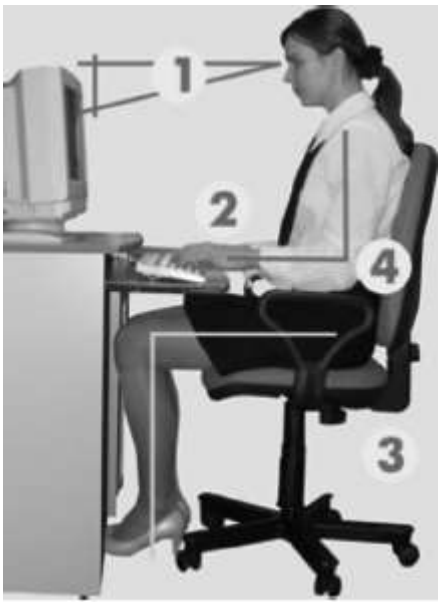

Рис. 1.2. Общий вид рабочего места

Клавиатура

Неправильное положение рук при печати на клавиатуре приводит к хроническим растяжениям кисти. Важно не столько отодвинуть клавиатуру от края стола и опереть кисти о специальную площадку, сколько держать локти параллельно поверхности стола и под прямым углом к плечу. Клавиатура должна располагаться в 10-15 см (в зависимости от длины локтя) от края стола. В этом случае нагрузка приходится не на кисть, в которой вены и сухожилия находятся близко к поверхности кожи, а на более «мясистую» часть локтя.

Клавиатура должна свободно перемещаться по рабочей поверхности, но во время использования она не должна двигаться. Угол наклона клавиатуры от горизонтали может варьировать от 0 до 25°. Важно иметь возможность регулировать этот угол для достижения наиболее удобного положения кистей рук. Поверхность клавиш и клавиатуры должна быть матовой, чтобы исключить блики. Горизонтальный размер поверхности клавиш должен составлять не менее 12 мм, расстояние между центрами клавиш по горизонтали 18-19 мм, а по вертикали – 18-20 мм.

Рабочее кресло

Высота кресла должна быть отрегулирована под горизонтальное расположение бедер, вертикальное расположение голеней, а ступни должны твердо находиться на полу. Диапазон регулировки сиденья по высоте не менее чем 38-52 см. Важно предусмотреть подставку для ног, которая должна иметь небольшой наклон (5-15°), быть нескользкой, достаточно тяжелой, чтобы не двигаться по полу с каждым движением стоп, компактной и желательно регулируемой по высоте. Поверхность сиденья не менее 45 см шириной и 38-43 см глубиной.

Подушка сиденья должна обеспечивать равномерное давление на бедра и ягодицы и иметь максимальную глубину продавливания 1,2-2,5 см. Желательно, чтобы передняя кромка сиденья имела закругленный край и была несколько поднята. Спинка кресла должна обеспечивать возможность наклона назад до 125-130°. Предпочтительна высокая спинка, поддерживающая

всю спину и шею. Рекомендуемая высота спинки 45-51 см. Для предотвращения деформации поясничного отдела позвоночника (наиболее страдающая область при длительном сидении) рекомендуется использовать кресло с профилем спинки, имеющим поясничный изгиб.

Кресло должно быть поворотным и на роликах, чтобы легко перемещаться по полу. Подлокотники поддерживают предплечья и снижают утомляемость плеч, шеи и верхней части торса. Подлокотники также помогают садиться и вставать. Если подлокотники ограничивают комфорт или неудобны, они теряют свою практичность. Подлокотники являются необходимым элементом рабочего кресла, при этом крайне желательно, чтобы они были легкосъемными. Некоторые кресла, предусматривают и регулировку подлокотников по высоте.

Оптимальным с точки зрения максимальной стабильности является кресло с пятью точками опоры. Иллюстрировать приведенные требования можно примером плохого кресла, показанного на рис. 1.3.

На рис. 1.3 видно, что сиденье и спинка непомерно широкие и глубокие, поясничный изгиб спинки отсутствует, подлокотники слишком тонкие, высокие и широко расставленные, нет регулировок, имеются только четыре точки опоры.

Рис. 1.3. Пример плохого с точки зрения биомеханики кресла

Рабочая поверхность

Высота рабочей поверхности над полом, а также ее размеры и форма являются важными факторами, влияющими на работоспособность человека. Несоблюдение требований к рабочей поверхности приводит к повышенной утомляемости, физическому дискомфорту, к целому ряду медицинских проблем и как следствие к снижению производительности труда. Физиологические нарушения происходят в шейно-плечевой области, в области предплечья и кисти, в поясничном отделе позвоночника, в тазобедренной области. Эти проблемы существенно возрастают, если имеются следующие нарушения:

- высота клавиатуры над полом слишком большая или слишком маленькая;
- предплечье и кисть не имеют должной опоры и, следовательно, не отдыхают;
- голова оператора слишком наклонена;
- из-за недостаточного пространства для ног бедра операторов наклонены под столом.

В целом рекомендуемые параметры рабочей поверхности зависят от решаемых задач. Как показывает практика, наиболее распространенная высота рабочей поверхности 76 см над уровнем пола – это слишком высоко для операторов среднего роста. Желательно иметь две поверхности – одну для клавиатуры, другую для монитора, письма и чтения. Если рабочая поверхность одна, то ее высота должна регулироваться в диапазоне 60-80 см. Для большинства случаев оптимальная высота – 70 см над уровнем пола, но регулируемая высота кресла позволяет подогнать высоту под человека любого роста. При двух независимых поверхностях высота поверхности для клавиатуры должна регулироваться в диапазоне 59-71 см, а для монитора – 90-115 см. Под рабочими поверхностями необходимо предусмотреть достаточное

пространство для ног оператора. Минимальная ширина пространства для ног 51 см, предпочтительнее 61 см. Минимальная глубина этого пространства на уровне колен 38 см от края рабочей поверхности, а на уровне стоп – 59 см.

Влияние двухуровневых рабочих поверхностей на положение тела и рук при работе на клавиатуре показано на рис. 1.4.

Одноуровневая рабочая поверхность. Клавиатура на краю стола – верный путь к травме костно-связочного аппарата и мышц кистей и предплечий. Тендовагиниты, «теннисный локоть», другие нарушения – частые следствия такого расположения клавиатуры (рис. 1.4, *а).*

Двухуровневая поверхность. Главное, чтобы локоть и кисть составляли одну линию и были параллельны поверхности стола. Угол в коленном и тазобедренном суставах – 90° (рис. 1.4, б).

Рис. 1.4. Положение клавиатуры и рук оператора

Планировка рабочего помещения

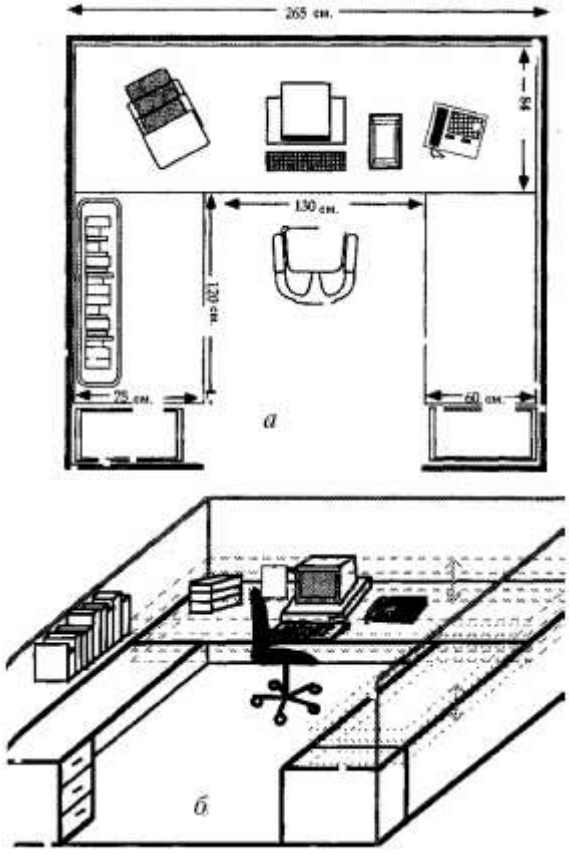

Рис. 1.5. Схема планировки рабочего помещения с рекомендуемыми размерами: *а* – в плане; *б* – в трехмерном пространстве[1]

Активный и пассивный комфорт

Важным фактором при выборе кресла является продолжительность работы. От этого зависят и конструкторские рекомендации. Так, при длительности работы, не превышающей 3 ч, достаточными оказываются требования, обеспечивающие пассивный комфорт, а большая

[1] Из: Handbook of Human Factors and Ergonomics / Ed. by Gavriel Salvendy. 2nd Ed. Purdue University. John Wiley Sons, Inc. – N.Y., 1997. P. 1669.

продолжительность влечет за собой соблюдение требований, обеспечивающих активный комфорт. Так, например, пассивный комфорт для рабочего кресла определяет качества кресла, не связанные с каким-либо механизмом или с регулируемой системой. При соблюдении указанных выше размеров целесообразны такие конструктивные решения, как представленные на рис. 1.6.

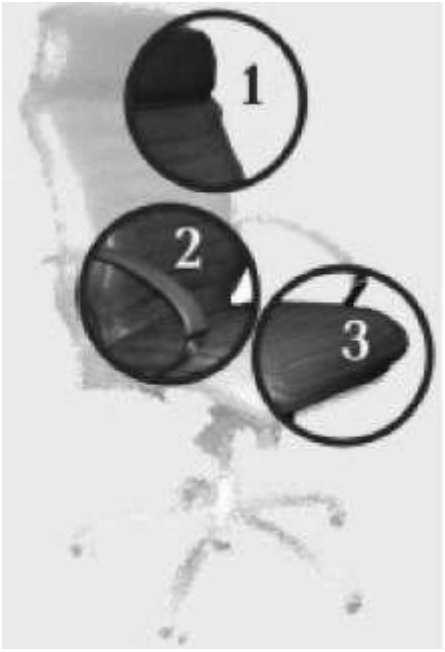

Рис. 1.6. Конструктивные решения пассивного комфорта

1. Утолщения спинки и сиденья способствуют наилучшему распределению давления тела на кресло.

2. Подлокотники, которые могли бы регулироваться по высоте и/или по ширине, позволяют уменьшить усталость плеч и шейных позвонков.

3. Утолщенный изгиб, расположенный на переднем крае сиденья (система «водопад»), помогает устранить нежелательное давление на ноги под коленями.

Требования активного комфорта в дополнение к указанным выше включают следующие конструктивные решения (рис. 1.7).

1. Обычная регулировка. Все рабочие вращающиеся кресла и стулья оснащаются пневматической регулировкой высоты сиденья. Их спинка может регулироваться по высоте и/или наклону, обеспечивая оптимальную поддержку поясницы.

2. Регулировка глубины сиденья. Дополнительно некоторые кресла могут иметь механизм, регулирующий глубину сиденья на длину хода 50 или 70 мм, в зависимости от их модели. Это особенно важно для пользователей крупной комплекции.

3. Механизм постоянного контакта. Спинка кресла находится в постоянном контакте со спиной пользователя, при этом сиденье не меняет своего положения. Спинка кресла может фиксироваться в любом положении или в нескольких, в зависимости от модели.

4. Механизм качания с центральной осью. Наклон кресла вперед-назад осуществляется механизмом, расположенным по центральной оси сиденья. Упор спинки регулируется в зависимости от веса пользователя.

Рис. 1.7. Конструктивные решения активного комфорта

5. Механизм качания со смещенной осью. Наклон кресла вперед-назад осуществляется механизмом, ось которого смещена к переднему краю кресла, что позволяет пользователю все время упираться ногами в пол. Упор кресла регулируется в зависимости от веса пользователя.

6. Синхронный механизм. Самое лучшее решение! Механизм позволяет согласованно изменять положение спинки и сиденья в зависимости от положения пользователя. Упор спинки регулируется в зависимости от веса пользователя. Могут фиксироваться все положения кресла или только некоторые, в зависимости от модели. Спинка кресла оснащена системой антивозврата. При всем этом необходимы пять точек опоры:

• ступни ног устойчиво стоят на полу;

• спина находится в постоянном контакте со спинкой кресла независимо от положения тела (наклон вперед-назад, прямо);

• глубокая посадка – полностью занимать всю плоскость сиденья;

• обе руки лежат на поверхности стола;

• мышцы шеи и затылка имеют опору в виде подголовника или высокой спинки кресла.

Усилия конструкторов по созданию наилучшего с точки зрения антропометрии и биомеханики рабочего места «человек-компьютер» порождают иногда интересные решения. Так, например, компания Personal Computing Environments решила исправить и в 2002 г. запустила в серийное производство персональную компьютерную среду (рис. 1.8). Комплекс включает компьютер, удобное сиденье и подставку для ног, что придает ему некоторое сходство с зубоврачебным креслом. Специальные кронштейны позволяют удобно разместить две жидкокристаллические (ЖК) панели, а кресло легко регулируется под индивидуальные требования оператора.

Рис.1.8. Персональная компьютерная среда

1.2. Требования к клавиатуре

Правильный выбор органов управления ПК не менее важен, чем выбор монитора. Следует иметь в виду, что интенсивный ежедневный труд приводит не только к болезням глаз, но и к

заболеваниям рук. Наиболее часто встречаются такие профессиональные недуги:

• синдром запястного канала – немеют ладони и запястья, в них возникает покалывание, ползут мурашки, немеют большой, указательный и средний пальцы – возникает после интенсивной длительной работы в условиях дефицита рабочего пространства для рук;

• крепитирующий тендовагинит (появляются отеки и болезненность в области сухожилий пальцев, чаще всего большого пальца) вызывается профессиональным перенапряжением сухожилий и окружающих их тканей. Это может происходить, например, из-за частого нажатия на клавишу «Пробел» большим пальцем правой руки или клавишу «Ввод». Надо заметить, что большой палец в качестве пальца-антагониста даже когда не участвует в наборе знаков, то все равно постоянно испытывает нагрузку;

• «теннисный локоть» (воспаляется общее сухожилие мышц-разгибателей, расположенных возле локтя и подвергающихся чрезмерной нагрузке) развивается в результате неправильного положения рук, как правило расположенных выше, чем следует.

В зависимости от вида используемых органов управления пользователи обычно опираются:

• локтем – при широких движениях кистью вместе с предплечьем (сенсорный экран, световое перо);

• предплечьем – при движениях кистью (мышь, джойстик, клавиатура);

• запястьем – при движениях пальцами (клавиатура, трекбол, сенсорная панель).

Инженерная мысль весьма интенсивно работает в направлении создания новых, более приспособленных к человеку клавиатур. Появляются весьма причудливые формы: клавиатуры как бы разламываются надвое или изгибаются в плоскости клавиш, вводятся всевозможные подставки для кистей. Однако следует помнить, что при интенсивной работе *стереотип поведения бывает важнее кажущегося удобства,* ведь если пользователь молодой, то лучше учиться на новой клавиатуре, а если это опытный пользователь (когда-то печатавший на пишущих машинках), то не стоит осваивать новинки – переучиваться будет сложнее! Любая оригинальная клавиатура вряд ли будет одинаково хороша для решения всех задач. Конструкция клавиатуры и место ее размещения в рабочей зоне, определяющиеся антропометрическими характеристиками человека, должны обеспечивать следующие условия:

• возможный наклон поверхности клавиатуры должен быть в пределах от 5 до 15° (в общем случае, а если коллектив в основном женский или слишком «интеллигентский», т.е. размеры кистей у всех небольшие, то угол может быть и меньше);

• высота среднего ряда клавиш не более 30 мм;

• свободное пространство от нижнего ряда кнопок до передней кромки клавиатуры должно иметь ширину 80-100 мм тогда, когда кромка возвышается больше чем на 20 мм (для больших рук). Такая площадка может выполняться либо в виде специального «пристяжного» конструктивного элемента, либо в виде мягкого валика для поддержки кистей рук, наполненного гелеобразной массой;

• часто используемые клавиши должны располагаться в центре, внизу и справа, редко нажимаемые – вверху и слева (если приходится часто работать с «макросами», то отведенные для них клавиши лучше располагать слева);

• группы функциональных клавиш должны выделяться размером, формой и местом расположения;

• функциональные клавиши для печати «слепым» методом должны кодироваться рисками и точечными бугорками, а вот кодирование цветом нецелесообразно, поскольку при работе с клавиатурой в основном задействовано периферическое зрение;

• верхняя поверхность клавиши должна быть вогнута и профилирована по горизонтали, тогда подушечке пальца будет более удобно фиксировать ее;

• размер контактной плоскости клавиш, рассчитанный на антропометрические характеристики отечественного пользователя, по горизонтали должен быть не менее 13 мм, по вертикали – 15 мм (у китайцев, например, могут быть другие величины);

• расстояние между контактными плоскостями клавиш не может быть меньше 3 мм, что определяется точностью позиционирования пальцев и тремором рук;

• равный для всех клавиш рабочий ход – от 1 до 5 мм;

• ко всем клавишам должно прикладываться одинаковое усилие нажатия 0,25-1,5 Н;

• клавиатура должна свободно перемещаться относительно монитора в пределах 0,5-1,0 м.

Как уже отмечалось, для выделения функциональных зон клавиатуры цветовое кодирование применять нецелесообразно, лучше использовать оттенки основного цветового тона. Связано это с тем, что палочки и колбочки по сетчатке глаза распределены неравномерно: на периферии больше палочек, в центре – колбочек. Функционирующие при дневном свете колбочки обеспечивают восприятие цвета (хроматические ощущения), поэтому наиболее чувствительной будет область под углом 10-20° к периферии от центра сетчатки, т.е. центральная. Значит, цветовое кодирование зрительных элементов информационной модели следует производить в центральной области зрения, а клавиатура находится скорее на периферии. Сказанное относится как к выделению функциональных зон, так и к применению цветового разделения символов на клавишах. Например, при русской и английской раскладке алфавитов основное влияние оказывает даже не цвет, а места расположения знаков, цвет воспринимается больше по яркостному контрасту.

Кстати, о контрасте. Следует ограничивать неравномерность распределения яркости в поле зрения пользователя ПК, однако яркостный контраст между рабочими поверхностями не должен превышать 0,67-0,8, а между рабочими поверхностями и поверхностями стен и оборудования – 0,9. Поэтому следует выбирать все оборудование в комплекте, учитывая имеющийся или планируемый интерьер. При этом знаки на клавишах обязательно должны иметь прямой яркостный контраст и не вызывать постоянной переаккомодации глаз, приводящей к быстрой утомляемости. Это требование не всегда соблюдается, и на рынке часто можно встретить клавиатуры со светлыми знаками на темном фоне, темные или прозрачные панели и клавиши (иногда это даже называется фирменным стилем!). Яркостный контраст для позитивного изображения определяется по формуле

$$К = (L\phi - L)/ L\phi ,$$

где L – яркость элемента; $L\phi$ – яркость фона.

Надписи на клавишах должны иметь яркостный контраст в пределах 0,6-0,7. Минимальные угловые размеры знаков должны лежать в диапазоне 12-35° (в линейных величинах – 2-5 мм). А вообще размер и яр-костный контраст знаков станут оптимальными для восприятия при их полном совпадении с аналогичными светотехническими характеристиками текста на экране монитора (опять же для уменьшения переаккомодации глаз оператора). Здесь уместно отметить, что контрастность на ЖК-мониторах и мониторах на электронно-лучевых трубках (ЭЛТ) существенно различается. Например, яркостный контраст у современных ЖК-мониторов достигает значения 0,997-0,998, чего конкурирующим с ними мониторам на ЭЛТ уже не достигнуть никогда (для монохромных мониторов – 0,33-0,67, для цветных – 0,6-0,99).

Подсветка клавиш целесообразна тогда, когда оператор должен немедленно реагировать на поступающий сигнал, а не при смене режима набора, что бывает сейчас чаще всего. Но это замечание скорее относится к пользовательскому интерфейсу, а не к клавиатурам.

Расстояние между соседними кнопками рассчитывается исходя из антропометрических и физиологических свойств человека-оператора, в первую очередь должен учитываться тремор рук. Суставной тремор, проявляющийся при работе на клавиатуре, можно представить в виде суммы двух синусоидальных колебаний с частотой 10 Гц при амплитуде 10-20° и 1-2 Гц при амплитуде 1°. Легко подсчитать, что при оптимальном расстоянии до сенсорного экрана, равном 300-500 мм, расстояние между сенсорными кнопками не должно быть менее 7-11,5 мм. Говорить о таких светотехнических характеристиках, как яркость и контраст, вообще не приходится: можно установить любые удобные для пользователя значения. Примеры наиболее продвинутых клавиатур приведены на рис. 1.9.

Рис. 1.9. Примеры наиболее совершенных клавиатур. На нижнем рисунке показан мягкий валик для кистей рук, причем высота этого валика регулируется (см. стрелки на нижнем рисунке)

В последнее время появились образцы беспроводной лазерной виртуальной клавиатуры. Такая клавиатура представляет собой небольшой черный параллелепипед размером 9 х 3,5 х 2,5 см, и внешне ничем не напоминает клавиатуру (рис. 1.10, а). При включении питания над поверхностью, где находится этот параллелепипед, появляется рисунок клавиатуры красного цвета размером со стандартную «ноутбучную» (рис. 1.10, б). Эта клавиатура – аппарат, излучающий два лазерных луча, один из которых проецирует изображение клавиатуры на поверхность, а другой используется для считывания касаний клавиш. Нажатия клавиш сопровождаются короткими звуковыми сигналами, что в некоторой степени заменяет тактильную реакцию с обычной клавиатуры. Чувствительная плоскость находится немного выше плоскости стола, и при некоторой тренировке можно работать, не касаясь поверхности стола вообще. Клавиатура дополняется зарядным устройством для встроенного аккумулятора, mini-CD с драйверами и чехлом.

а *б*

Рис. 1.10.

Виртуальная лазерная клавиатура: *а* – переносной блок 9 х 3,5 х 2,5 см, проецирующий изображение клавиатуры; *б* – изображение клавиатуры на поверхности

В такой клавиатуре легко реализуется смена изображения букв различных алфавитов и даже иероглифов. Достаточно иметь микросхему памяти для хранения пиктограмм и можно в режиме он-лайн загружать любые наборы букв или символов. Подключение клавиатуры к основному устройству, будь то системный блок или ноутбук, карманный компьютер или смартфон, при наличии интерфейса BlueTooth затруднений не вызывает. Клавиатура поддерживает два формата подключения. Первый из них – HID (human interface device), который вообще не требует установки драйверов под Windows 2000 или Windows XP. Второй – SPP, с использованием эмуляции последовательного порта и необходимостью установки соответствующего драйвера. Первый вариант удобнее, так как требует меньше манипуляций при работе, зато второй позволяет осуществлять все настройки через компьютер при помощи специальных утилит. Среди таких настроек и яркость отображения клавиш, и степень чувствительности, и скорость повтора при удерживании клавиш, и отражение уровня зарядки аккумулятора и др. Кроме этого, настраиваются функции энергосбережения, когда после некоторого времени бездействия отключается лазер развертки или устройство полностью.

Несомненные достоинства такого рода клавиатуры – это компактные размеры, малая масса, отсутствие проводов, возможность совмещения с различными устройствами. Но имеются и недостатки. Так, необычные тактильные ощущения приводят к случайным нажатиям, ведь сенсорная плоскость «висит» над столом и приноровиться нажимать клавиши без ошибок получается не сразу. При этом большинство опечаток связано не с неправильным попаданием в кнопки, а с перемещением рук при печати. Можно уменьшить количество опечаток настройкой чувствительности сенсора, но чтобы совсем их избежать, требуются время и практика. Кроме того, расположение некоторых клавиш не совсем стандартное. Это связано с тем, что в русском алфавите букв больше, чем в латинице, а ограниченный размер клавиатурного поля заставляет некоторые находящиеся по краям буквы помещать в непривычные места.

Виртуальные клавиши

В последнее время появилось много новых устройств с такими виртуальными органами управления, как сенсорные экраны. Чтобы сохранялся стереотип деятельности оператора, размеры и расположение управляющих элементов на этих экранах при дублировании стандартных клавиатур не должны отличаться от них. Пользователь может формировать параметры сенсорных экранов, самостоятельно подстраивая их под свои антропометрические

характеристики. При эксплуатации ПК группой пользователей следует исходить из того, что минимальный размер кнопок должен быть рассчитан на человека с антропометрическими параметрами руки по 95 перцентилям. Расположение и вид виртуальных органов управления подчиняются общим принципам формирования информационной модели.

1.3. Требования к мыши

Мышь является важнейшим средством ввода-вывода данных, в последние годы, в связи с распространением Интернета и игр, даже более часто употребляемым, чем клавиатура. Поэтому число разработок разных вариантов мыши, трекбола и других средств координатного ввода растет очень быстро. Основными показателями при этом являются как характеристики деятельности, т. е. скорости и точности наведения на точку, так и степень нагрузки на связочный и мышечный аппарат кисти. И конечно, соответствие сложившимся стереотипам, т. е. отзывы пользователей.

Учитывая, что практически все предлагаемые решения в той или иной мере соответствуют антропометрическим требованиям, решающую роль при выборе варианта начинают играть субъективные предпочтения, привычки и т. п. Вместе с тем тестирование таких вариантов проводится по многим параметрам, и появляются все более и более совершенные разработки. Так, например, лучшие результаты тестирования были у вертикальной мыши (рис. 1.11).

Рис. 1.11. Вертикальная мышь

Рис. 1.12. Вращающийся шарик вмонтирован в клавиатуру

Рис. 1.13. Мышь с шариком для вращения пальцем

Рис. 1.14. Трекбол с большим шаром, облегчающим движение кисти

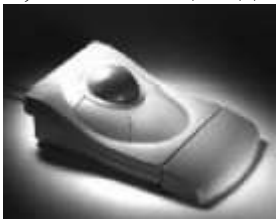

Рис. 1.15. Трекбол с мягким валиком для кисти

Рис. 1.16. Двухклавишная ножная мышь

Рис. 1.17. Трекбол, встроенный в мягкий валик для кисти

Рис. 1.18. Мышь для детей

Рис. 1.19. Оптические мыши, наиболее привычные для большинства. Слева – проводной вариант, справа – беспроводной

Рис. 1.20. Вертикальная мышь

Принцип работы с вертикальной мышью: человек кладет руку на мышь не сверху, а сбоку. В таком состоянии запястье не изгибается в неестественном для себя виде, и, соответственно, человек не испытывает дискомфорта при длительной работе. Мышь имеет 3 кнопки и вертикальное колесо прокрутки.

Отправной точкой для всех модификаций мышей, трекболов, джойстиков и пр. являются поперечные размеры ладони, зрительно-моторная координация движений кистью и пальцами и биомеханика движений кисти и пальцев. Материал, из которого сделана мышь и клавиатура, тоже важен. При изготовлении периферийных устройств (мышь, клавиатура, монитор) традиционно используется пластик (как легкий и недорогой материал), позволяющий обеспечить большинство потребительских свойств. Известно, однако, что существует категория людей, у которых пластик вызывает аллергию – именно для них компанией Holzkontor (США) созданы клавиатура, монитор и мышь в корпусах из дерева и камня, натуральных минералов и акриловых полимеров. Мониторы, клавиатуры и мыши, выполненные из этих материалов, предлагаются на выбор в различном цветовом исполнении. Круг потенциальных пользователей ограничивает, конечно, более высокая цена на эти устройства.

Мы осветили физические устройства ввода информации. Однако на Web-сайтах и во многих программных приложениях взаимодействие с системой осуществляется на других уровнях. Их соответствие друг другу отражено в табл. 1.1.

Таблица 1.1

Уровни взаимодействия и их соответствие друг другу

Программные приложения	Web-сайты	Физические устройства
Самостоятельные элементы экрана (кнопки, списки, комбобоксы, ползунки и пр.)	Элементы формы, тэги и связи	Кнопки, цифронаборники, осветительные приборы, дисплеи
Разработка экрана	Разработка страницы	Физическая структура
Разработка навигации	Разработка структуры сайта	Основные режимы работы
Другие приложения и системы	Сеть, браузер, внешние связи	Реальный мир

Преимущества и недостатки различных устройств ввода информации представлены в табл. 1.2.

Таблица 1.2

Преимущества и недостатки различных устройств ввода информации

Устройство	Хорошо для ...	Плохо для ...
Клавиатура	• алфавитно-цифрового ввода данных; • управления дискретными перемещениями курсора;	• гибкости; • скорости; • свободы пространственных перемещений

Устройство	Хорошо для ...	Плохо для ...
	• точного позиционирования пиксела	
Функциональные клавиши	• гибкости – одни и те же клавиши могут использоваться для разных функций; • того же, что и для клавиатуры	• легкости понимания
Мышь	• выбора, протаскивания и манипуляции объектами; • функциональной универсальности, гибкости; • удобства	• свободного рисования; • общественных терминалов (устройство отделено от основного блока)
Трекбол	• малой области указания и выбора; • портативного оборудования; • скорости и минимизации усилий	• действий протаскивания; • свободного рисования; • большой области указания и выбора
Сенсорные экраны	• легкости понимания; • простых задач выбора; • отсутствия устройств, отделенных от основного блока	• длительного использования (усталость); • мелких элементов экрана; • заслонения экрана рукой
Графические планшеты	• свободного рисования; • точной оцифровки графики	• небольшого пространства и горизонтальной плоскости; • любых задач, кроме рисования/выбора
Джойстик	• задач слежения; • задач позиционирования и указания; • длительного использования	• задач, требующих высокого разрешения и точности
Световое перо	• указания объекта и простого ввода; • минимизации пространства; • высокой скорости позиционирования	• продолжительного использования (усталость предплечья и кисти); • заслонения экрана рукой

1.4. Требования к монитору и к освещению рабочего места

Все поверхности в зрительном поле оператора должны освещаться примерно одинаково. Имеется в виду однородность временная и пространственная. Колебания освещенности в зрительном поле серьезно ухудшают работоспособность. Такие колебания возникают, если оператор по роду работы должен попеременно смотреть то на яркую, то на темную поверхность либо если источник света имеет колебания. Дело в том, что восприятие изменений освещенности требует определенной задержки (т.е. имеет место рефрактерный период, когда восприятие блокировано), и при определенных частотах колебаний аккомодация не успевает адаптироваться и восприятие нарушается. Важнейшим аспектом восприятия информации на

мониторе является общая и местная освещенность.

Различают четыре типа освещенности: прямую, непрямую, смешанную (прямую и непрямую), переливчатые шары. Прямая освещенность – основной источник освещения в рабочем помещении. При прямом освещении световой поток имеет вид конуса, направленного на объект. Этот вид освещения образует блики. Непрямое освещение – это отраженный свет. Отражается он от потолка и стен, при этом бликов, как правило, не образуется, однако требуется дополнительное освещение. При смешанной (прямой и непрямой) освещенности часть света (около 40%) исходит во всех направлениях, а остальной свет направлен либо прямо на объекты, либо отражается от стен и потолка. Переливчатые шары испускают свет равномерно во всех направлениях, но из-за своей яркости они часто дают блики.

Современные источники света – это в основном лампы накаливания и электролюминесцентные трубчатые лампы. Свет от ламп накаливания состоит из красных и желтых лучей, т.е. области длинных волн спектра. Поэтому он изменяет естественные цвета объектов и, следовательно, не может использоваться, если восприятие цветов важно для работы. Такие лампы имеют недостаток – значительное тепловыделение. Кроме того, лампы накаливания потребляют больше энергии, чем люминесцентные лампы и, следовательно, более дороги в эксплуатации, но в то же время большинству нравится мягкий свет ламп накаливания как более комфортабельный.

Свет люминесцентных ламп образуется при прохождении тока через газ. Этот свет имеет меньшую яркость, чем у ламп накаливания, и поэтому в значительно меньшей степени является источником бликов. Их спектр близок к дневному свету. Можно, впрочем, выбрать и иной спектр освещения люминесцентных ламп, если это желательно при оформлении помещения. Вместе с тем стандартный свет этих ламп воспринимается как холодный, бледный и в целом некомфортный. Этим лампам свойственно также крайне неприятное мерцание, если они старые или не вполне исправные.

На рабочем месте важно как вертикальное, так и горизонтальное освещение. В помещении, оборудованном потолочными светильниками, отношение вертикального освещения к горизонтальному составляет 0,3-0,5. Таким образом, если освещение такой комнаты равно, скажем, 500 лк, то горизонтальное – 500 лк, а вертикальное – между 150 и 250 лк. При работе на компьютере наиболее предпочтительная величина горизонтальной освещенности 300-700 лк. Если оператору необходимо читать информацию не только с экрана, но и с листов бумаги, то величина общей освещенности рабочей зоны должна быть 500-700 лк, а если только с экрана – 300-500 лк. При работе в одном помещении нескольких операторов и разных видах работы у каждого из них горизонтальная освещенность должна быть 300-500 лк, но необходимо предусмотреть местные источники света, расположенные так, чтобы не допускать бликов на экране.

Снижение производительности отмечено как при недостаточной, так и при избыточной освещенности. Если при недостатке освещенности наблюдается замедление ручного ввода данных, то при избыточной освещенности – рост числа ошибок.

Очень важны при установке осветительных приборов характеристики яркости. *Яркость* – это количество света, отраженного от поверхности (стена или потолок), излученного поверхностью или проходящего сквозь поверхность (как, скажем, солнечный свет проходит сквозь занавеску). Яркость измеряется в канделах на квадратный метр (кд/м2). Весьма высокие значения яркости имеют окна, поэтому следует избегать такого расположения рабочих мест, когда окна находятся на периферии поля зрения. В центральном поле зрения соотношения яркостей разных поверхностей (или объектов) должно быть около 3: 1, в то время как для периферического поля зрения это отношение не должно превосходить 10: 1. Яркость зависит как от величины освещенности, так и от коэффициента отражения конкретной поверхности. Соотношение между яркостью и освещенностью называется коэффициентом отражения и выражается формулой

$$K = \frac{0{,}32\ [\text{кд/м}^2]}{lx}\ \%.$$

Для восприятия информации на экране очень важна различимость символов и других

элементов изображения. Она зависит от их яркости, контраста и начертания (дизайна в общем случае). Хорошая различимость и читаемость символов имеет место, если расстояние между символами составляет 20-50%, а межстрочный интервал – от 100 до 150% их высоты.

Шрифты с размером кегля меньше 12 обычно слишком малы для чтения на экране. При расстоянии от глаз пользователя до экрана 50 см минимальная высота шрифта должна быть 2,6 мм (соответствует кеглю 10), при расстоянии 60 мм такая высота должна быть 3,1 мм (соответствует кеглю 11), а при расстоянии 70 мм высота шрифта не должна быть менее 3,7 мм (соответствует кеглю 12). Различный дизайн символов виден в многочисленных фонтах, среди которых можно выделить как наиболее читабельные «Times New Roman», «Bookman Old Style», «Courier New» и др.

В последнее время появились мультимедийные комплексы, где функции монитора выполняют специальные очки. Такого рода очки разработала израильская компания «Lumus». К легким (менее 50 г) очкам сбоку прикреплен экран LCD, достаточно крупный, полноцветный, с высокой разрешающей способностью VGA с 640 x 480 пикселей. Очки снабжены линзами, которые не вызовут косоглазия при просмотре видео. Изображение проецируется из прикрепленного сбоку источника и по принципу зеркальных фотокамер передается на оптические линзы толщиной всего 2 мм, где микропризма и специальное напыление увеличивают изображение до приемлемого размера для человеческого глаза. Человеку кажется, что он смотрит на 60-дюймовый экран телевизора на расстоянии 3 м. Преимущество таких очков состоит в том, что передаваемое изображение не блокирует для зрителя происходящее вокруг. Экран остается прозрачным, таким образом, пользователь может видеть как бы сквозь него, при необходимости переключая внимание на окружающую обстановку.

1.5. Средства моделирования компоновки рабочего места

В последние годы разработаны программные средства моделирования, визуализации и анализа данных о движениях людей, позах, усилиях и т.п., т.е. о любых аспектах моторной активности. Таких средств довольно много, мы здесь упомянем следующие: Observer (фирма «Noldus Information Technology», http://www.noldus.com) и CSM-анализ (фирма SGI, www.sgi.com/industries/csm/).

Observer – это система для сбора, анализа, систематизации, которая обладает широкими возможностями обработки данных о позах, движениях, положениях, даже выражениях лица и социальной активности и существенно облегчает анализ поведенческих проявлений людей и животных. При этом данные могут вводиться и с портативного, и с обычного компьютера, и с видеопленки. Обработка данных включает сравнение с наиболее близкими образцами, оценку средних времен и усилий разных действий, статистический анализ, оценку надежности разных действий, анализ последовательностей действий с запаздыванием, экспорт-импорт данных в/из электронных таблиц и, следовательно, использование всех возможностей последних, графическое и табличное представление данных и др.

CSM (Computational Structural Mechanics)-анализ – это система трехмерного моделирования пространственного положения и взаимных перемещений любых физических тел. Система предоставляет широкие возможности для любых биомеханических построений, включая даже столь тонкие и сложные манипуляции, как работа кисти и пальцев. Человеческое тело представлено в виде манекена, параметры которого могут задаваться. Манекен может ходить, бегать, сидеть, сгибаться и разгибаться, совершать все физиологически возможные движения конечностями, производить мышечную работу с оценкой утомляемости и пр. Важнейшей характеристикой системы является возможность ее интегрирования с известными системами для проектирования (Autocad, MicroStation и др.). CSM широко используется в автомобилестроении.

Компоновка рабочих мест и средств деятельности при работе за видеотерминалом (с компьютером). Здесь существует множество разработок. Приведем в качестве примера разработки компаний 3M (Англия, www. mmm.com.ergonomics) и GWS Systems (Финляндия, филиалы в Германии, Швеции, Великобритании, Франции, США, а сейчас и в России).

Компания 3M предложила компьютерный стол и кресло к нему, особенность которых

состоит в возможности их моделирования и подгонки под конкретного пользователя. Легко изменяются высота расположения монитора, клавиатуры и кресла, положение и наклон клавиатуры, взаимные положения клавиатуры и коврика для мыши и другие параметры. Пользователь может изменять сидячее положение тела на, скажем, стоячее, при этом легко за 1-2 мин меняя параметры рабочего места.

Эта же компания предложила очень существенное, на наш взгляд, дополнение формы компьютерной клавиатуры и коврика мыши – мягкий широкий валик по всей длине переднего обреза клавиатуры и такой же валик перед ковриком для мыши. Это значительно уменьшает напряжение и утомление в лучезапястном суставе, позволяя кисти свободно лежать при работе на клавиатуре и находиться в физиологически благоприятном положении. Предусмотрена также подгонка этого валика по высоте (см. рис. 1.9). Компанией 3M предложены также специальная панель для удержания перед глазами листа или тетради. Панель крепится к боковой стороне монитора, вращается в горизонтальной и вертикальной плоскостях. Это удобное приспособление хорошо известно российским пользователям (его ранние варианты).

Другое предложение этой же компании – оптический фильтр, помещаемый перед экраном монитора, значительно сужающий угол наблюдения за экраном, т.е. не позволяющий видеть содержание экрана человеку, даже стоящему чуть сбоку от пользователя. Также предложены фильтр, предотвращающий появление бликов при боковом освещении экрана, специальная фактура поверхности коврика для мыши, не допускающая скольжения шарика мыши, и др.

Компания GWS Systems Oy (www.gwssystems.ru) разрабатывает модульные рабочие места, позволяющие оптимизировать их методом физического макетирования при конструировании самых разных рабочих мест и рабочих поз. Система позволяет также учитывать нагрузку на разные группы мышц при разных положениях тела человека, оценивать динамику утомления и делать сравнительный анализ разных вариантов рабочих мест. Система легко может быть дополнена модулями, соответствующими конкретным рабочим местам. Быстрота, легкость и эффективность создания всех возможных компоновок рабочих мест, а также сравнительная дешевизна системы обусловили ее широкое распространение в промышленном и офисном дизайне, при разработке практически любых промышленных объектов. Система GWS пользуется большой популярностью у дизайнеров.

Институт профессионального здоровья Финляндии предлагает самую дешевую систему конструирования рабочих мест. Это обильно иллюстрированный и сравнительно краткий справочный буклет; специальные бланковые формы, в которых учтены практически все существенные параметры; номограммы и таблицы для определения величин и времени допустимых мышечных нагрузок, зон досягаемости, расстояний от глаз и углов обзора для разных видов деятельности, формы сидений и спинок, расположений рабочих поверхностей и пр. Для удобства в табл. 1.3 обобщены требования, а также рекомендации для обустройства рабочего места.

Таблица 1.3

Требования и рекомендации к компоновке рабочего места при работе на компьютере

Компоненты рабочего места	Рекомендации
Рабочее кресло	• Должно позволять откидываться назад для расслабления мышц спины и снятия нагрузки с межпозвоночных дисков • Ширина сиденья не менее 45 см, глубина – от 38 до 43 см • Высота поверхности сиденья: легко регулируемая между 38 и 52 см • Высота спинки: 45–51 см над поверхностью сиденья • Регулируемое отклонение назад – до 130°

Компоненты рабочего места	Рекомендации
	• Профиль спинки – слегка вогнутый в грудной области и выгнутый (небольшой валик) в поясничной • Регулировка глубины сиденья – от 50 до 70 мм • Наличие постоянного контакта спинки кресла со спиной пользователя и возможность фиксации спинки кресла • Упор спинки должен регулироваться в зависимости от веса человека • Наличие системы антивозврата спинки кресла • Синхронный механизм качания, позволяющий согласованно изменять положение спинки и сиденья в зависимости от положения пользователя
Расположение монитора	• Высота центра экрана над полом – от 90 до 115 см • Отклонение экрана назад от горизонтальной плоскости – от 88 до 1050 • Расстояние экрана от края стола – от 50 до 75 см • Расстояние от глаз оператора до экрана – от 45 до 50 см (не более 70 см)
Клавиатура	• Высота над полом – от 59 до 71 см (до 80 см) • Расстояние от края стола до клавиатуры – от 10 до 26 см • Изменение наклона поверхности клавиатуры должно быть в пределах от 5° до 15° • Высота среднего ряда клавиш – не более 30 мм • Свободное пространство от нижнего ряда кнопок до передней кромки клавиатуры должно иметь ширину 80–100 мм, тогда как кромка возвышается больше чем на 20 мм (для больших рук). Такая площадка может выполняться либо в виде специального «пристяжного» конструктивного элемента, либо в виде мягкого валика для поддержки кистей рук • Часто используемые клавиши должны располагаться в центре, внизу и справа, редко нажимаемые – вверху и слева (если приходится часто работать с макросами, то отведенные для них клавиши лучше располагать слева) • Группы функциональных клавиш должны выделяться размером, формой и местом расположения

Компоненты рабочего места	Рекомендации
	• Функциональные клавиши для печати «слепым» методом должны кодироваться рисками и точечными бугорками • Кодирование цветом нецелесообразно, поскольку при работе с клавиатурой в основном задействовано периферическое зрение • Верхняя поверхность клавиши должна быть вогнута и профилирована по горизонтали, тогда подушечке пальца будет более удобно ее фиксировать • Размер контактной плоскости клавиш, рассчитанный на антропометрические характеристики отечественного пользователя, по горизонтали должен быть не менее 13 мм, по вертикали – 15 мм (у китайцев, например, могут быть другие величины) • Расстояние между контактными плоскостями клавиш не может быть меньше 3 мм, что определяется точностью позиционирования пальцев и тремором рук • Равный для всех клавиш рабочий ход – от 1 до 5 мм • Ко всем клавишам должно прикладываться одинаковое усилие нажатия – от 0,25 до 1,5 Н • Возможность для перемещения относительно монитора в пределах 0,5–1,0 м
Пространство для ног	• Глубина на уровне колена не менее 38 см от края стола • Глубина на уровне стоп не менее 59 см • Ширина пространства не менее 51 см
Освещение	• Окна и другие источники яркого света не должны располагаться ни сзади, ни спереди операторов • Уровень общей освещенности должен быть 300–700 лк • Уровень освещенности должен соответствовать решаемым задачам • Яркость в центральном поле зрения 1 : 3, а в периферическом – 1 : 10 • Яркость символов на экране по отношению к фону 7 : 1

Контрольные вопросы

1. Каковы основные причины возникновения утомления, обусловленные конструкцией рабочих мест и позой работающего человека?

2. Расскажите об основных видах скелетно-мышечных расстройств при работе на компьютере, их причинах и способах предотвращения.

3. Что вы знаете о параметрах расположения монитора и клавиатуры на рабочем месте?

4. Определите параметры компоновки рабочего места.

5. В чем заключаются требования к характеристикам монитора (яркость, контрастность, размеры символов и пр.)?

6. Что такое освещенность рабочего помещения и рабочего места? Типы освещенности, разновидности источников света, соотношение между яркостью, освещенностью и коэффициентом отражения.

7. Каковы требования к монитору и клавиатуре? Разновидности клавиатур.

8. Требования к устройствам координатного ввода (мыши, трекболу, джойстику).

9. Преимущества и недостатки различных устройств ввода информации.

10. Особенности конструкции рабочего кресла, обеспечивающие активный комфорт для оператора.

11. Что вы знаете о рекомендуемых характеристиках рабочего кресла, обеспечивающих активный и пассивный комфорт для человека?

12. Требования к размерам рабочей поверхности.

13. Требования к общей планировке рабочего помещения.

Литература

Основная

1. Эргономика. Принципы и рекомендации: Метод. рук-во. – М.: ВНИИТЭ, 1981.

2. Человеческий фактор / Под ред. *Г. Салвенди:* В 6 т.: Пер. с англ. – М.: Мир, 1991. Т. 2, 5.

3. Handbook of Human Factors and Ergonomics. / Ed. by Gavriel Salvendy). 2nd Ed. Purdue University. John Wiley Sons, Inc. – N.Y., 1997. Ch. 50.

4. ISO 9241-1: Ergonomic requirements for office work with visual display terminals (VDTs) – General Introduction.

5. ISO 9241-3: Visual display requirements.

6. ISO DIS 9241-4. Keyboard requirements.

7. ISO DIS 9241-5. Workstation layout and postural requirements.

Дополнительная

1.www.ergonomic.ru/

2.www.fentek-ind.com/ergo.htm

3.penza.fio.ru/personal/36/3/8/ergos.htm

4.www.mebelstyle.ru/furniture/ergonomic/

5. zoom.cnews.ru/ru/publication/index.php?art_id80=1104 page80=1

6. www.cnews.ru/news/top/index.shtml?2006/12/22/229529

7. www.gwssystems.ru/

8. www.mmm.com/ergonomics

9. www.zoom.cnews.ru/ru/publication/index.php?art_id80=1104

10. www.noldus.com

11. www.sgi.com/industries/csm/

12. *Bangor A.* LCD vs. CRT: Effects on Performance of Office Work. Proc. of the Human Factors and Ergonomics Society 48th Annual Meeting-2004, p. 745

13. *Wurman Richard Saul.* Information Architects.

Watson-Guptill Publ., 1997

ТЕМА 2. ЭВОЛЮЦИЯ ВЗГЛЯДОВ И ИНСТРУМЕНТАРИЯ ЧЕЛОВЕКО-КОМПЬЮТЕРНОГО ВЗАИМОДЕЙСТВИЯ

Изучаемые вопросы:

• Функциональные возможности четырех классов интерфейсов (языка команд, заполнения меню и специальных форм-таблиц, прямой манипуляции, антропоморфных).

• Принцип WYSIWYG и функции, которые он обеспечивает.

• WIMP-интерфейс, его составные части и возможности.

• Возможности трехмерного интерфейса.

• Понятие «виртуальная реальность».

• Характеристики ПИ, обеспечивающие виртуальную реальность.

• Способы создания объемных изображений, их сравнительная оценка, варианты технической реализации.

2.1. Общие положения

Со времени появления и широкого распространения компьютеров сильно изменились взгляды на формирование пользовательского интерфейса и его возможности – от командной строки первых интерфейсов до современных мультимедиа-систем с почти безграничными возможностями звука и визуализации. Взгляды на стиль взаимодействия человека с компьютером прошли некоторую эволюцию. В соответствии с ней различают четыре класса интерфейсов:

1) язык команд;

2) заполнение меню и специальных форм-таблиц;

3) прямая манипуляция;

4) антропоморфный.

Интерфейс языка команд – наиболее старый и наиболее трудный в использовании. От пользователей требовалось помнить и печатать определенную последовательность команд, подчас имеющих зашифрованный вид и сложный синтаксис. Это вызывало напряжения памяти, тем более что системы этого типа были очень чувствительны к малейшим ошибкам во вводе. Подробнее об этом типе интерфейса будет сказано в разд. 2.2.

В интерфейсе заполнения меню многие такие проблемы отсутствуют, для выбора предъявляются только пункты меню, и пользователь легко может их выбрать «кликаньем» мышью либо одним-двумя нажатиями на клавишу. Разработка эффективных интерфейсов такого типа требовала рассмотрения множества факторов: структуры меню, последовательности и формулировки пунктов меню, сокращений для часто используемых вариантов наборов меню, компоновки меню и его графического оформления, темпа предъявления и требуемой скорости ответа, механизмов выбора и т.п. Этот тип интерфейса рассмотрен в разд. 2.3.

Интерфейс прямой манипуляции основан на положении, что пользователи должны иметь возможность управлять интерфейсом как объектами в пространстве. Этот тип интерфейса описывается следующими тремя характеристиками:

1) имеет место визуальное и непрерывное представление объектов и результатов действий с ними;

2) управление объектами осуществляется прежде всего с помощью физических действий или нажатия определенных клавиш, а не путем ввода последовательности команд сложного синтаксиса;

3) действия быстры и обратимы; их влияние на объект можно наблюдать непосредственно в ходе действий.

Примеры интерфейсов прямой манипуляции достаточно многочисленны: текстовые процессоры, настольные издательские системы, система проектирования Autocad, имитаторы полета, видеоигры и пр. Такие интерфейсы используют принцип WYSIWYG (What You See Is What You Get), т.е. принцип прямого соответствия объектов и результатов действий с ними на экране. Эти интерфейсы имеют свои преимущества и недостатки.

Преимущества достаточно очевидны и состоят в том, что пользователь испытывает гораздо меньше беспокойства при работе и поощряется при обучении. Обучение упрощается, так как основано на аналогиях, а необходимый для запоминания синтаксис весьма незначителен, что ведет к лучшему сохранению операциональных знаний. Другое достоинство состоит в том, что минимизируется потребность в сообщениях об ошибках, так как результаты действий наблюдаются визуально.

Недостатки интерфейсов прямой манипуляции связаны с общими проблемами использования метафор и значков (иконок). Проблемы эти состоят в том, что пользователь не всегда понимает смысла метафор и значков, т.е. у него возникают другие ассоциации, чем

предполагал разработчик, а также в неэффективном использовании части экранного пространства. Другой недостаток проистекает из основного достоинства этих интерфейсов – наглядного, а не цифрового контроля своих действий пользователем. В ситуациях, когда нужно точное соответствие, скажем, углов, положений в пространстве, длин, площадей и т.п., ввод соответствующих данных с клавиатуры оказывается более эффективным, чем наглядные действия. Этот тип интерфейса соответствует трехмерному (пространственному) интерфейсу вместе с WIMP-интерфейсом (разд. 2.4).

Антропоморфный интерфейс – это и есть цель человеко-ориентиро-ванного проектирования интерфейса. Предполагается, что интерфейсы такого рода должны включать диалог с пользователем на естественном языке, содержащем знаковые элементы, визуальную информацию (в том числе выражение лица), и учитывать такие вещи, как, скажем, направление взора пользователя (по локализации его зрительных фиксаций).

2.2. Командная строка

Командная строка – это средство прямого указания компьютеру со стороны пользователя. Обычно используются функциональные ключи, отдельные символы, аббревиатуры либо команды, состоящие целиком из слов. Это прямой доступ в систему, часто сочетаемый с набором дополнительного инструментария для повышения гибкости: множество опций или параметров, которые могут применяться для многих объектов одновременно, избавляя от повторений команд. Схему командной строки можно представить так:

команда – опция… аргумент1 аргумент2…

Командная строка – весьма мощное средство воздействия, так как занимает значительно меньше памяти машины и не требует промежуточных программ-посредников. Однако эта быстрота и мощь сопряжены с трудностями в освоении и запоминании языка взаимодействия (вспомните MS DOS, особенно замечательный интерфейс Norton Commander). По мере набора команды могла выдаваться зависящая от контекста подсказка. Для отображения подсказки использовалась панель, расположенная непосредственно под полем ввода. В зависимости от контекста подсказка могла содержать:

• полный список команд с краткими описаниями (команд немного, так что такой список гарантированно уместится на панели);

• перечень команд, начинающихся с уже введенного префикса. Например введено «по», предлагаются варианты: показать, помощь, порубать_в_капусту (большинство команд однозначно определяется первыми одним-двумя символами);

• перечень допустимых опций;

• описание опции;

• описание аргументов команды;

• объяснение, почему текст в командной строке не является корректным.

Проблему пытались решить, вводя смысловую мнемонику, аббревиатуры (вспомните Norton Commander). Это, конечно, упростило диалог, но даже незначительное расширение возможностей (и системы, и пользователя) приводило к неоднозначности смысла выполнения тех или иных команд, возникали противоречия в понимании этих мнемонических аббревиатур для разных программ и в разном окружении, что порождало ошибки, устранение которых существенно усложняло процесс обучения.

2.3. Меню, заполнение форм-таблиц, диалог по вопроснику

В меню наборы возможных команд отображаются на экране, и пользователь выбирает их мышью либо функциональными клавишами, либо алфавитно-цифровыми. Это не требует запоминания всех команд и их опций и облегчает процесс взаимодействия. Часто меню организованы иерархически, что упрощает вызов нужной команды. Такие меню базируются на названиях действий на естественном языке и предлагают опции как нумерованный выбор (они могут быть и графическими).

Последнее время меню чаще всего являются частью WIMP-интерфей-са. По-разному

может быть организовано и предъявление меню. Иногда даже главное меню спрятано и всплывает при определенных действиях пользователя, т.е. в зависимости от контекста. Вообще, всплывающие меню используются при необходимости контекстно-зависимых действий. Как вариант всплывающих меню используются выпадающие меню: когда курсор оказывается в зоне какой-то опции главного меню, то самостоятельно либо при нажатии на кнопку мыши выпадает подменю. Меню может быть активно или нет и отражать это в определенной подсветке опций (но сами опции видны, и это важно). Меню неэффективно, если в нем слишком много вариантов выбора, поэтому используется иерархическое группирование, позволяющее иметь наверху всего 6-8 опций, но каждая из них раскрывается в систему подменю. Полоса с меню обычно располагается наверху экрана (наверху каждого окна).

Имеется и другой подход к отображению меню – расположение всего главного меню по кругу. Когда курсор в центре, расстояния до каждой опции равны, но как только он сдвинулся по направлению к какой-то опции, появляется (выпадает) соответствующее подменю. Такая организация позволяет иметь несколько большую область для каждой опции (и это очень хорошо), но занимает много места на экране. Видимо, из-за этого недостатка такие формы меню не нашли распространения. Есть много вариантов разных видов организации меню на экране. Например, выяснено, что полоса меню внизу экрана требует больше времени для идентификации и «кликанья» мышью, чем такая же полоса вверху экрана.

Основная проблема при организации меню – это иерархическое группирование разных опций, т.е. построение дерева опций. При этом учитываются такие аспекты, как частота использования и важность, противоположные по смыслу действия (скажем, как сохранить и уничтожить – save и delete). Это группирование – отражение нашего представления о тех оперативных единицах мышления пользователя, которыми он реально оперирует. Их выявление и объективизация – основное содержание работы по дизайну интерфейса.

Использование естественного языка в меню – это, конечно, очень удобно и для пользователей, и для разработчиков. Но возникает проблема – многозначность слов естественного языка, их нечеткая определенность. Язык – это, по сути, иерархическая система обобщений. Как и во всякой иерархической системе, обработка поступающей на вход информации состоит в ее классификации. Но отличие этой системы от других подобных систем – попадание практически каждого слова, поступившего на «вход», в некоторое множество классов, т.е. имеет место полисемия или многозначность практически каждого слова. Это скорее не исключение, а правило для языка.

Точное опознание имеет место только на основе контекста. А это самое трудное и для систем искусственного перевода, и вообще для формальных систем понимания языка. Поэтому в полной мере использовать естественный язык пока невозможно. Но отдельные слова, даже фразы (заранее определенные) широко используются. Так, например, огромную популярность приобрел универсальный язык запросов SQL, основанный на словах и фразах естественного языка. Вместе с тем для определенного типа задач более подходящим оказался графический язык запросов QBD (Query By Diagram), хотя его применение более ограниченно.

Вопрос-ответ (диалог по вопроснику) – пользователь отвечает поочередно на вопросы, где ответом, как правило, являются да-нет, множественный выбор или какие-то коды. Таким интерфейсом легко пользоваться, но он ограничен заданным набором вопросов и ответов, т.е. совсем не обладает гибкостью. Такой интерфейс может быть рассчитан на любого человека, в том числе совершенно не имеющего опыта работы на компьютере. Пример – справочные системы на вокзалах и в аэропортах, другие системы массового потребления. Для подобных целей такой интерфейс незаменим. Огромный круг людей, способных использовать интерфейс этого типа, – залог его большой коммерческой перспективности. Распространение систем с таким интерфейсом будет весьма активным (это всевозможная транспортная информация, торговля, справочные системы, связь, бытовые приборы, платежные системы, охрана и т.д.).

Заполнение специальных форм-таблиц часто сочетается с диалогом по вопроснику. Пользователь продвигается по бланку, заполняя соответствующие поля. Вид и содержание бланка, как правило, хорошо знакомы пользователю. Это несколько более развитый вариант по сравнению с диалогом по вопроснику, более гибкий, так как пользователь может вернуться и исправить ошибку либо передумать и, скажем, указать другое значение. Такая организация

интерфейса хороша для новичков, она удобна для массового использования и широко применяется.

Пример.
Туристическое бюро «Светлый путь»
Пункт отправления – *Урюпинск*
Пункт назначения – *Сиэтл*
Через – *Москва, Берлин, Лондон*
Первый класс /Второй класс/Бизнес-класс/
Для некурящих/Для курящих
Место №
Возврат в начало

2.4. WIMP-интерфейс

WIMP-интерфейс – это наиболее часто используемый интерфейс в настоящее время. Все продукты компании «Microsoft» основаны на нем (все версии Windows, в том числе и для операционной системы UNIX, и многие системы для Макинтошей). Аббревиатура WIMP – это Windows (окна), Icons (значки, иконки), Menus (меню), Pointers (указатели мыши). Иногда – Windows, Icons, Mice (мышь), Pull-down menus (выпадающие меню). Опишем по очереди каждый элемент – Windows, Icons, Menus, Pointers. Отметим, что кроме этих элементов широко используются кнопки, инструментальные панели, палитры, диалоговые окна и др. Их принято называть *widgets,* что лучше всего перевести, как «прибамбасы». Наборы таких «прибамбасов» образуют инструментарий взаимодействия человека с компьютером. Современные оконные системы используют одинаковые наборы «прибамбасов» для реализации основного принципа – видеть и чувствовать.

Windows. Окна могут по-разному располагаться, быть наложенными друг на друга или располагаться каскадом, менять размеры, развертываться во весь экран, свертываться в иконку. Важная часть окна – наличие скролл-бара для перетаскивания содержимого в видимую часть (что делает окно аналогом реального окна).

Icons. Маленькая картинка служит «представителем» закрытого окна, позволяя развернуть его, когда потребуется. Такая картинка – напоминание о существовании окна, экономит место на экране. Иконки также служат для отображения других элементов системы, являясь их «сверткой», например мусорная корзина, разные диски, папки и пр. Варианты изображения иконки могут быть самыми разнообразными и служат мнемоническими правилами, позволяющими по виду иконки понять, что за ней скрывается. Разные варианты иконок показаны на рис. 2.1.

Рис. 2.1. Разные варианты иконок
Menus. Описание меню дано в разд. 2.3.
Рис. 2.2. Разные варианты форм курсоров

Pointers. Указатели – основа функционирования WIMP-интерфейса. Большое значение имеет форма курсора. Скажем, стрелка, либо вертикальная полоска, либо курсор меняют форму при изменении функции (стрелка заменятся крестом при рисовании линий). Форма указателя может также информировать о состоянии системы: иметь вид песочных часов, если выполняется программа и надо подождать. Форма указателя подобна иконке, но в дополнение к ней указатель имеет так называемую hot-spot (активную область); если указатель в форме стрелки (любого вида), то центр этой области – на острие стрелки, если в виде указывающего пальца – на кончике этого пальца, но есть формы, когда активная область не столь очевидна (и это плохо). При выборе формы указателя следует всегда заботиться об очевидности активной области. Варианты указателей показаны на рис. 2.2.

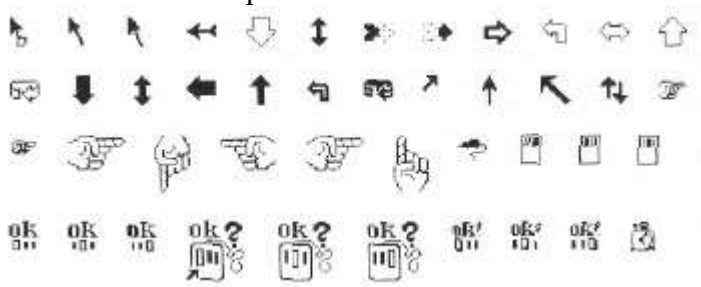

Рис. 2.2. Разные варианты форм курсоров

Используются и другие элементы, например диалоговые окна (они могут быть также всплывающими или нет, иметь текст, иконку или и то и другое), наборы инструментальных кнопок (toolbars), при нажатии на которые происходят какие-то часто встречаемые действия.

2.5. Трехмерный (пространственный) интерфейс и виртуальная реальность

Такой тип интерфейса все более востребован последнее время. Самая убедительная иллюстрация этого – виртуальная реальность, но этим отнюдь не исчерпывается круг возможных приложений 3D (3-Dimensions, т.е. трехмерных образов). В целом же пространственный интерфейс сочетается с другими возможностями, особенно в WIMP-интерфейсах.

Простейший случай – использование в WIMP-интерфейсе элементов объемности, к примеру затенения (для кнопок, полос прокрутки и др.). Разумно используемый эффект трехмерности позволяет подсветить активные (или какие-то нужные) области. К сожалению, во многих интерфейсах злоупотребляют этими эффектами, скульптурно выделяя каждый фрагмент текста, меню, границы, в результате чего теряется смысл это мощного средства. В пространственных интерфейсах объекты обычно плоские, но путем изменения их размера создают ощущение удаленности-приближенности. Для создания ощущения трехмерности пользуются углом освещения, размерами (о которых мы сказали) и наложениями одних изображений на другие.

Трехмерные интерфейсы в большинстве случаев относятся к интерфейсам прямой манипуляции. Смысл в том, что пользователь не отдает команды системе, а манипулирует объектами, что более естественно. Первым популярным применением этого метода была корзина для удаления файлов на Macintosh (начиная с Windows 95 такая корзина стала стандартом и в мире Windows, хотя существовала и раньше). Ограничимся констатацией того простого факта, что, если перетащить в нее пиктограмму файла, этот файл будет удален.

Важно понимать еще две вещи. Во-первых, для достижения достаточной эффективности не следует стараться наиболее реалистично отражать действие, значительно важнее более реалистично отразить объект, над которым это действие совершается. Например, компьютерную панель управления работой осветительных приборов не обязательно снабжать точными изображениями выключателей. Главное – реалистично отразить на ней план помещения и расположение источников света, равно как и показать прямую (читай – непосредственную) связь между этой информацией и собственно выключателями. Во-вторых,

бывают ситуации, когда эффективность непосредственного манипулирования уравновешивается неэффективностью физических действий пользователя.

Термин «виртуальная реальность», или «виртуальное окружение», означает, что создается такое компьютерное изображение, которое, находясь в согласии с прошлым опытом человека, интуитивно воспринимается им как естественное. Виртуальная реальность отличается от множества других компьютерных образов выполнением следующих условий:

• графическое изображение должно позволять выполнять пространственные преобразования, зависящие, скажем, от пространственного положения пользователя;

• все три оси, описывающие положение объектов в пространстве, должны быть активны, т.е. задействованы;

• изменения объектов на экране должны соответствовать их реальным изменениям в трехмерном пространстве;

• должно быть реализовано большинство интуитивно возможных взаимодействий с предъявляемым объектом в реальном мире;

• должны быть возможны преобразования объектов по всем реальным или требуемым степеням свободы;

• реакция объектов на воздействие должна происходить в квазиреальное время.

Виртуальная реальность есть результат отделения (можно даже сказать, отрыва) логического представления процессов от их физической сути. Для человеко-компьютерного взаимодействия под логическим представлением понимается представление пользователя о системе. Наиболее ярким примером этого являются файловые системы – мы давно уже не задаемся вопросом, какие конкретно блоки на дисках занимают те или иные данные, более того, содержимое файлов совершенно не зависит от того, как именно и на каких носителях они лежат. Современным примером в этой области являются виртуальные массивы EVA, существенно изменившие подход к управлению ресурсами в сетях хранения данных. Но сегодня развиваются еще и виртуальные центры обработки данных, значит, кроме виртуальных систем хранения данных внедряются и виртуальные машины.

Виртуализация требует наличия средств преобразования логического представления в физическое и обратно. В качестве ресурсов, подвергающихся виртуализации, могут выступать как программные, так и аппаратные средства. Главное – возможность разделения объекта виртуализации на отдельные блоки, каждый из которых можно сравнительно легко преобразовать в требуемую форму. Скажем, если мы хотим использовать многопоточную обработку, например, на многопроцессорной машине, приложение должно быть разделено на потоки (thread) – относительно независимые нити выполнения, которые можно исполнять параллельно как на нескольких процессорах или даже машинах, так и на одном процессоре (но с точки зрения приложения тоже параллельно). В этом случае преобразователь может быть встроен в операционную среду, чтобы обеспечивать взаимодействие и изоляцию приложений и их частей.

На уровне приложений, например, виртуализация существует давно: еще в 1960-1970 гг. ею пользовались для создания псевдомногозадачных сред. Примерно тогда же появились виртуальные машины в мэйнфреймах IBM. Это позволило перейти от пакетной обработки данных к интерактивным приложениям и обеспечить параллельную работу нескольких пользователей или приложений. Сегодня большинство операционных систем поддерживает параллельную обработку приложений и может обслуживать одновременно несколько пользователей, а средства виртуализации продолжают развиваться, например, в сторону обеспечения одновременной работы нескольких операционных систем на одной машине. Современный уровень развития средств виртуализации позволяет временно не задействованным ресурсам приносить реальную пользу. Виртуализация дает конкурентные преимущества, именно поэтому она столь популярна.

Для человеко-компьютерного взаимодействия основной смысл виртуальной реальности состоит в создании автономных программных приложений, чьи абстрактные репрезентации ориентированы в большей степени на представления пользователей, чем на реальные объекты. В этой связи уместно помнить, что в виртуальной реальности могут происходить и невыполнимые преобразования. Создать же абсолютную имитацию естественного мира с его

бесконечным разнообразием невозможно, потому что это означало бы создание совершенной психической модели мира.

Степень погружения пользователя в виртуальную реальность зависит от правдоподобности этой реальности. Погружение имеет технические измерения: содержательно-изобразительное, индивидуально-психологическое. Цель погружения достигнута, когда пользователь верит, что находится не в том мире, где располагается его физическое тело. Различают внешние и внутренние факторы погружения; внешние факторы определяются используемой технологией, а внутренние отражают субъективный опыт взаимодействия с подобными объектами.

Виртуальное окружение должно соответствовать субъективным ожиданиям человека, т.е. быть почти абсолютной метафорой (см. разд. 4.4) и таким образом расширять его возможности овладения предметом и принятия решений. Разработка ПИ и инструментария должна при этом отвечать следующим принципам:

• трехмерности объектов на экране;

• прямой манипуляции и интуитивному взаимодействию с объектами вместо ввода команд;

• интерактивности, а не формализованной последовательности действий;

• мультисенсорной адресации, а не только зрительно воспринимаемой.

Эти требования вытекают из основной цели – человеко-компьютер-ное взаимодействие должно быть аналогичным взаимодействию с реальным миром, где важны и формы, и фактура, и звуки, и осязательные ощущения, и запахи, и окружение. Все сенсорные каналы человека в идеале должны участвовать во взаимодействии.

В последнее время широкий круг разработчиков стремятся создать трехмерный, т.е. объемный, интерфейс, который смог бы стать по-настоящему массовым продуктом. Несмотря на то что уже известно множество способов создания объемных изображений, в этой области постоянно патентуются новые изобретения. При этом одни исследователи концентрируют усилия на создании изображений не на экране, а в воздухе, другие – на создании трехмерных изображений, пусть даже и на экране.

В 2001 г. в Южной Корее были представлены первые разработки так называемого Walk-thru Fog Screen («прогулка сквозь туманный экран») – плоского экрана, состоящего из частиц, напоминающих дым или туман, на который проецируется изображение. Экран состоит из ламинарных потоков воздуха, которые нагнетаются сверху и перемещаются упорядоченными слоями, параллельными направлению течения. Между слоев воздуха закачивается туман, на который и проекцируется изображение. В результате получается плавающая в воздухе картина, сквозь которую можно проходить или даже изменять ее, используя специальную аппаратуру. Создатели предлагают использовать свое изобретение в качестве декораций для театров, улучшения дизайна помещений, визуальных презентаций, рекламы.

В том же году корпорация «IO2 Technology» презентовала устройство, проецирующее в воздухе изображение, которое можно крутить руками. Это устройство не создает туман или другое вещество, а изменяет свойства воздуха так, что при освещении специальными приборами на нем проецируется изображение. Изображение, как и в случае с ламинарным экраном, не имеет физической глубины, является плоским, двумерным. Изобретение выпущено в продажу в виде проектора с диагональю 30 дюймов и разрешением до 1024 x 768 dpi (и стоимостью около 25 тыс. долл.).

Другим направлением стало создание мониторов, изображение на которых выглядит объемным. Трехмерное восприятие здесь достигается за счет того, что изображение, поступающее в левый глаз отличается от изображения, поступающего в правый. Этот эффект используется в объемном кино: камера снимает с помощью двух объективов, расставленных как человеческие глаза, а при демонстрации фильма зрителю нужно надеть специальные очки с соответствующими светофильтрами, разными для каждого глаза.

Сотрудники Института имени Генриха Герца в Берлине представили на выставке CeBIT-2005 новую технологию создания объемных изображений «Mixed-Reality», позволяющую просматривать 3D-изобр-жения без специальных стереоочков. Первую такую

разработку продемонстрировала компания «Sharp» летом 2004 г. Пример такого дисплея показан на рис. 2.3. В 2004 г. германская компания «SeeReal Technologies» выпустила на рынок трехмерный двадцатидюймовый LCD-монитор с самым высоким в мире разрешением: 1600 х 1200 dpi. Монитор воспроизводит 16,7 млн цветов, имеет яркость 250 кд/м2 и контрастность 400: 1. В основе работы этого дисплея лежит уникальный принцип отслеживания положения глаз пользователя в пространстве: система подстраивает изображение таким образом, чтобы каждый глаз видел свою картинку. При этом никаких очков для использования монитора не нужно. Этот дисплей пригодится не только фанатам игр, но также медикам, физикам и конструкторам, ведь он позволяет моментально получать гораздо больше информации о предмете, будь то новая деталь самолета или белок.

Рис. 2.3. Пример вертикально расположенного 3D-дисплея

Похожую систему представила в 2005 г. компания «Grundig» для телетрансляций. Классическая телевизионная панель была переработана: поверх экрана появился фильтр, позволяющий транслировать разные картинки для левого и правого глаза.

В 2005 г. калифорнийской компанией «EP Industries» запатентовано изобретение, позволяющее снимать объемное кино без использования двух параллельных потоков информации для каждого глаза. Система «Circlescan 4D» работает на основе эффекта Пульфриха: левый и правый глаз видят изображение с задержкой во времени благодаря тому, что на один глаз надевается темный светофильтр. Из-за светофильтра человеческий мозг обрабатывает информацию на долю секунды медленнее, поэтому, если камера смещается во время съемки, в разные глаза попадает разное изображение. Система «Circlescan 4D» работает на основе вращающихся зеркал, которые направляют изображение в объектив кинокамеры, поэтому движение камеры при съемке не обязательно и не влияет на результат. Снятые таким образом фильмы могут быть показаны на самом обычном экране, от зрителя требуется лишь использование затемняющего фильтра для любого глаза.

В конце 2004 г. группа сотрудников Кембриджского университета, образовавшая компанию «Light Blue Optics», разработала первый голо-графический проектор для мобильных устройств, основой которого является чип, способный генерировать до 200 голограмм в секунду.

Перспективной разработкой 3D-дисплеев можно признать разработку японской компании «Toshiba», предложившей технологию, позволяющую создавать объемные изображения с многометровой визуально воспринимаемой глубиной на плоском, горизонтально лежащем дисплее. Созданы два жидкокристаллических экрана размером 15,4 и 24 дюйма, которые располагаются горизонтально и проецируют изображение над собой. Физическое разрешение матриц дисплеев составляет 1920 х 1200 точек, однако изображения, на которые следует смотреть под углом (а не отвесно вниз), имеют разрешение 480 х 300 точек, так как при взгляде с разных положений наблюдателю открываются различные пиксели, отвечающие за тот или иной угол просмотра. Сверху матрицу прикрывает специальное стекло с решеткой из множества маленьких линз, которые обеспечивают распространение различных изображений

одного и того же предмета в правильных направлениях. Кардинальное отличие от предшественников состоит именно в том, что иллюзия объемности изображения действительно создается в разных плоскостях пространства. Правда, пока изображение нельзя обойти вокруг, так как при смещении более чем на 30° картинка растает.

До сих пор стереоскопические изображения можно было просматривать только лишь на вертикально расположенных экранах. Изменение ориентации дисплея, по мнению авторов разработки, позволит улучшить восприятие объемных образов, увеличить угол обзора и создавать целые композиции, состоящие как из виртуальных, так и из реальных объектов, т.е. обеспечить вполне реалистическое ощущение глубины. Специалисты «Toshiba» считают, что зрители испытывают неудобство при просмотре 3D-изображений, если они воспринимаются расположенными всего в нескольких сантиметрах от поверхности дисплея, поскольку бессознательно зритель предполагает, что пространство за дисплеем имеет бесконечную глубину.

В противоположность этому стереоскопические изображения той же высоты будут выглядеть натуральнее на расположенном горизонтально дисплее, поскольку зрители будут считать, что перед дисплеем нет ничего. Благодаря проецированию изображений объектов под разными углами никаких дополнительных приспособлений вроде специальных очков не требуется. Обычно для создания 3D-изображений в каждый глаз наблюдателя проецируется отдельное изображение. При этом, однако, стереоскопический эффект исчезает, как только наблюдатель меняет собственное местоположение. В представленном же «Toshiba» дисплее наблюдатель может перемещаться без потери ощущения объемности. Пример композиции из трех виртуальных и одного реального объекта на горизонтальном 3D-дисплее компании «Toshiba» показан на рис. 2.4.

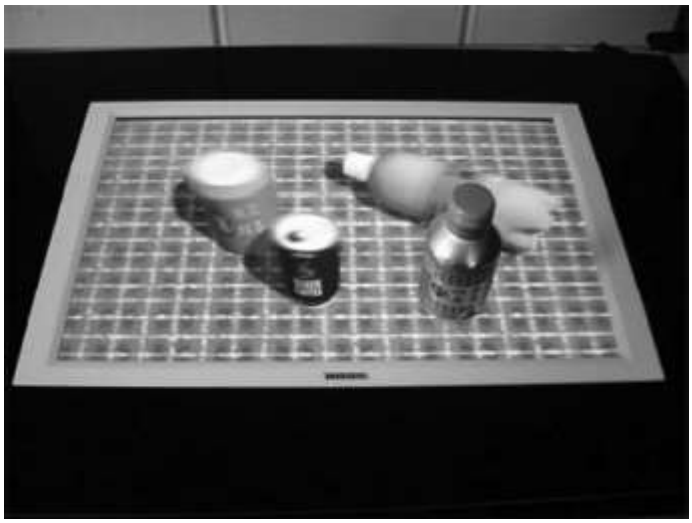

Рис. 2.4. Пример горизонтально расположенного 3D-дисплея компании «Toshiba». Композиция из трех виртуальных и одного реального объекта

Представители компании говорят о возможности создания в ближайшие годы плавающих над столами меню в ресторанах, о новых игровых системах и системах обучения, электронных книгах с трехмерными иллюстрациями и других подобных вещах. Впрочем, уже в настоящее время трехмерные изображения активно внедряются в практику, в частности в системы автоматизированного проектирования (САПР). Мировые лидеры в производстве САПР давно убеждают пользователей переходить на 3D-моделирование. Компания «Autodesk», ставшая ведущей в области двумерного черчения, интенсивно переходит с 2D на 3Б. Следует, впрочем, уточнить, что компания предлагает не вместо 2D, а и 2D вместе. Конечно, совсем без 2D жить невозможно – есть огромный круг задач, которые проще и эффективнее решить в 2D. И широко известный А^юСАБ, содержащийся во всех продуктах компании, поставляется теперь со всеми 3D-решениями по умолчанию.

Отдельно стоит упомянуть о разработках устройств, «обогащающих реальность» (Augmented Reality Systems). Одним из наиболее интересных примеров использования этой

технологии является разработанное в Колумбийском университете устройство «The Mobile Augmented Reality System» (MARS). Пользователь, надевший специальный шлем, получает возможность читать тексты, смотреть изображения и прослушивать музыку об объектах окружающей его реальности, включая людей. Глядя в небо, можно будет прочесть номер рейса пролетающего самолета, а подходя к ресторану, узнать, что сегодня в меню и по какой цене. Водитель, чинящий машину, будет видеть и по ходу ремонта узнавать функцию и схему частей двигателя; полисмен, глядя на прохожих, сможет их идентифицировать и выслеживать интересующих его людей; архитекторы смогут на местности видеть новые здания – в разрезе и в разных ракурсах. Система MARS может обмениваться информацией с системой глобального позиционирования GPS. Общая схема использования системы MARS представлена на рис. 2.5.

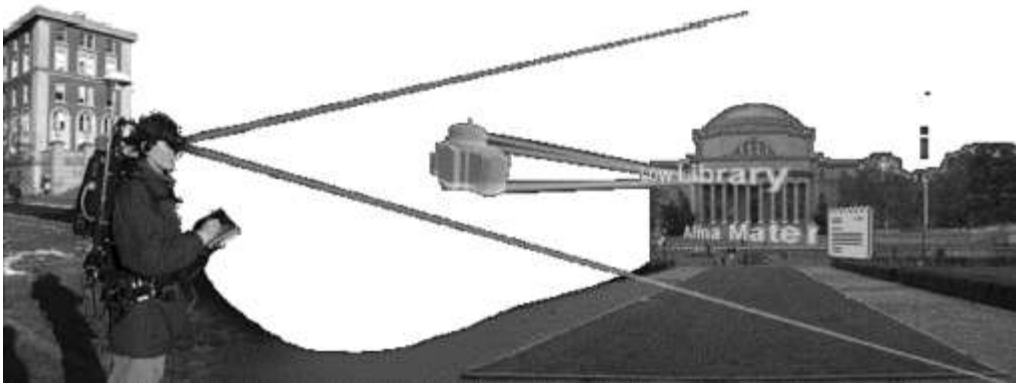

Рис. 2.5. Общая схема использования системы MARS (пример)[2]

Очень интересна разработка Джереми Ньютона (США), который создал интерактивный компьютерный экран, позволяющий демонстрировать разным пользователям различные видеоизображения одновременно. Рассмотрев ряд методик пространственной сепарации изображений, он предложил систему, состоящую из линзово-растрового (лентикуляр-ного) стереоскопического экрана, камеры, отслеживающей положение пользователей относительно него, и управляющего компьютера. Массив из вертикально ориентированных линз позволяет выделять изображения, предназначенные для просмотра двумя пользователями, расположенными соответственно слева и справа друг относительно друга. Новое изобретение позволит покончить с разрушающей семьи борьбой за заветное место у экрана ПК. Схема такого дисплея показана на рис. 2.6.

Однако разработка японского Национального института передовых наук и технологий («Advanced Industrial Science and Technology – AIST») и Университета Кейо совместно с компанией «Burton Inc.» существенно отличается от описанных технологий: найден способ генерировать действительно трехмерное изображение, а не иллюзию, основанную на том, что в разные глаза подается разная информация. Развитие новой технологии способно окончательно стереть грань между реальным и виртуальным мирами и обладает огромным потенциалом, в частности вследствие простоты технологии. Устройство способно создавать в воздухе физически трехмерное изображение. Это принципиальный прорыв в области разработок трехмерных интерфейсов. Разработки японских ученых существенно изменяют представления о пределах возможного, совмещая реальность с киберпространством.

[2] Источник: http://www1.cs.columbia.edu/graphics/projects/mars/mars.html

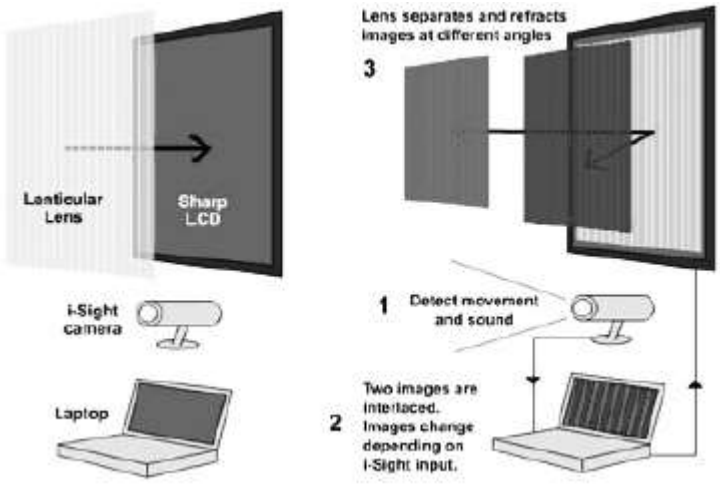

Рис. 2.6. Схема дисплея Джереми Ньютона[3]

Это, по сути дела, не голограмма; никаких специальных оптических эффектов не используется. В данном случае невидимые для человеческого глаза импульсы мощного инфракрасного лазера длительностью около 1 нс (10-9 с) и частотой повторения около 100 Гц, конвергирующие с другим инфракрасным лазерным лучом, формируют прямо в воздухе небольшие сгустки плазмы – отдельные пиксели, словно парящие в воздухе. В устройстве используется явление плазменной эмиссии в окрестностях фокуса лазерного излучения – визуально оно воспринимается как светящаяся точка или шарик. Лазерные лучи отражаются от нескольких рефлекторов, и в общей сложности за одну секунду в воздухе удается сгенерировать до сотни парящих пикселей, которые могут располагаться в нескольких метрах от источника излучения. Как и в случае с «Walk-thru Fog Screen», здесь используется специальный газ особого состава, незаметный для глаз.

Изображение проецируется инфракрасным импульсным лазером, который при помощи системы качающихся зеркал фокусируется на заданных точках пространства и генерирует маленькие шарики светящейся плазмы.

Прибор способен зажигать в любом месте пространства до 100 таких точек в секунду, если там содержится нужный состав. Каждая точка создается одним лазерным импульсом, глаз наблюдает точки одновременно, а не последовательно, за счет эффекта остаточного изображения. Построение массива трехмерных точек осуществляется с помощью механической развертки лазерного луча, а также изменения точки фокусировки. Технология управления лазерным фокусом позволяет обеспечить уже сейчас глубину трехмерного объема в несколько метров, а в перспективе – существенно увеличить ее. И сколь бы малым ни казалось значение 100 пикселей в секунду, изображения, полученные с помощью этого устройства, – прекрасное зрелище. Общая схема этой технологии показана на рис. 2.7.

[3] Источник: www.cnews.ru/news/top/mdex.shtm?2005/06/14/180328

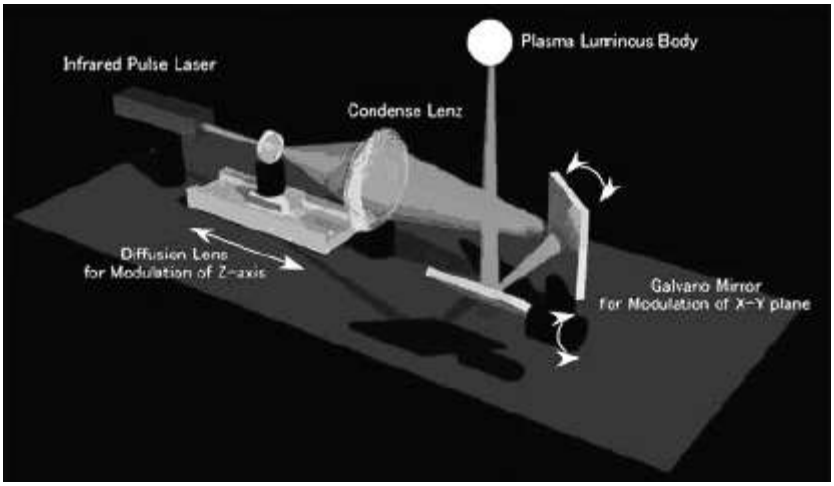

Рис. 2.7. Технология построения растровых изображений в пространстве[4]

При помощи таких светящихся точек можно создавать непосредственно в воздухе сложные трехмерные изображения, которые будут подлинно трехмерными, но при этом нематериальными. Пока прибор способен создавать лишь слегка окрашенные однотонные голубоватые точки, однако в будущем несомненно будут созданы устройства, создающие значительно более качественные изображения. На рис. 2.8 показаны изображения, сделанные по технологии AIST.

Сейчас активно внедряются международные стандарты новых форматов трехмерных изображений в Интернете, которые разрабатываются группой компаний во главе с корпорацией «Intel» и Национальным институтом стандартов и технологий США (National Institute of Standards and Technology – NIST). Эволюция человеко-компьютерного взаимодействия представлена в табл. 2.1.

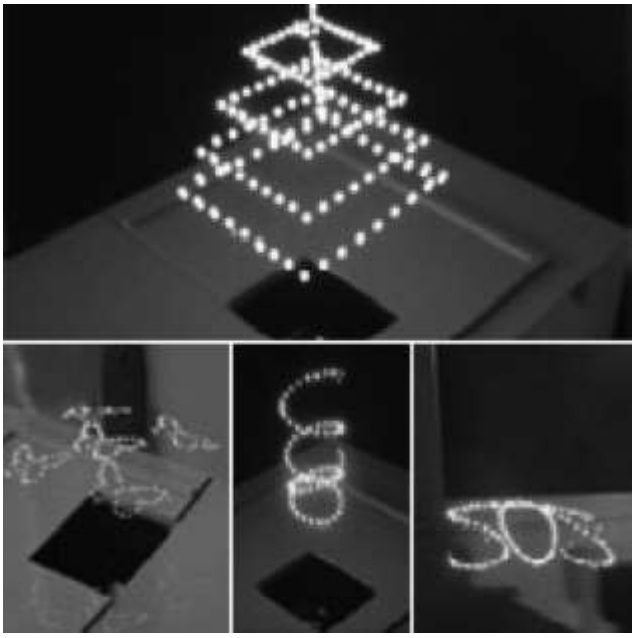

Рис. 2.8. Примеры изображений по технологии AIST[5]

Таблица 2.1

Развитие основных средств человеко-компьютерного взаимодействия

[4] Источник: www.aist.go.jp/aist_e/latest_research/2006/20060210/20060210.html

[5] Источник: www.aist.go.jp/aist_e/latest_research/2006/20060210/20060210.html

Формат данных	Двоичный код	Синтаксис команд	Графические объекты	Представление объектов
Мерность	Одномер-ные	Одномерные	Двухмерные	Трехмерные
Способ взаи-модействия	Основан на кодах	Язык команд	Прямая ма-нипуляция	Сочетанный (речь, жесты, ...)
Средства вза-имодействия	Перфолента	Алфавитно-цифровая клавиатура	Графический интерфейс	Комбиниро-ванное взаимо-действие (язык, прямые мани-пуляции, речь и т.д.)

Контрольные вопросы

1. Классы интерфейсов и их функциональные возможности.
2. Принцип WYSIWYG и функции, которые он обеспечивает.
3. Преимущества интерфейсов прямой манипуляции.
4. WIMP-интерфейс, его составные части, функции каждой из частей.
5. Преимущества трехмерного интерфейса.
6. Основные признаки виртуальной реальности.
7. Характеристики ПИ, обеспечивающие виртуальную реальность.
8. Принципы создания объемных изображений, их сравнительная оценка, варианты технической реализации.
9. Как изменялись формат данных, мерность представления пространства, способ и средства взаимодействия в процессе эволюции человеко-компьютер-ного взаимодействия?

Литература

Основная

1. Человеческий фактор / Под ред. *Г. Салвенди:* В 6 т.: Пер. с англ. – М.: Мир, 1991. Т. 1, 4, 5.
2. *DixA., Finlay J.* (Eds). Human-Computer Interaction. – Printed in Great Britain, Glasgow. Publisher – Prentice Hall: 3rd ed. 2004. Ch.20
3. Handbook of Human Factors and Ergonomics / Ed. by Gavriel Salvendy. 2nd Ed. Purdue University. John Wiley Sons, Inc. – N.Y., 1997.
4. ISO/DIS 9241-13: User guidance.
5. ISO/DIS 9241-14: Menu dialogues.
6. ISO/DIS 9241-15: Command language dialogues.
7. ISO/DIS 9241-16: Direct manipulation dialogues.
8. ISO/DIS 9241-17: Form-filling dialogues.
9. ISO/IEC 10741-1 Dialogue interaction – Cursor control for text editing.
10. ISO/DIS 13407 Human-centered design processes for interactive systems.

Дополнительная

1. www.mebelstyle.ru/news/2003/10/09/01.html#top
2. www.cnews.ru/newtop/index.shtml?2005/04/14/177238
3. www.cnews.ru/newtop/index.shtml?2005/03/14/175822
4. www.cnews.ru/newtop/index.shtml?2004/08/10/162202
5. www.cnews.ru/newtop/index.shtml?2004/12/30/171759

6. www.cnews.ru/news/top/index.shtml?2005/06/14/180328
7. www.cnews.ru/cgi-bin/redirect.cgi?http://jeremynewton.com
8. www.aist.go.jp/aist_e/latest_research/2006/20060210/20060210.html
9. www.cnews.ru/news/line/index.shtml?2004/11/11/167957
10. www.polit.ru/event/2006/02/20/3d_monitors.html
11. www. io2technology.com/salesinquiry
12. www.seereal.com/
13. www.grundig.com/
14. www.optics.com/
15. www1.cs.columbia.edu/graphics/projects/mars/mars.html
16. www.epindustries.com/4Dvideo/index.html

ТЕМА 3. МОДЕЛИ ПОЛЬЗОВАТЕЛЯ В РАЗРАБОТКЕ ИНТЕРФЕЙСА

Изучаемые вопросы:

• Понятие модели пользователей.

• Типы моделей пользователей и их характеристики.

• Разновидности моделей пользователей первого типа, т.е. определяемых социальным и организационным окружением.

• Разновидности моделей пользователей второго типа, т.е. базирующихся на когнитивных процессах.

• Пользовательские профили, их содержание и цель составления.

• Правила описания портрета потенциального пользователя (персоны).

• Метод GOMS и его описание.

3.1. Общие положения

Термин «человеко-компьютерное взаимодействие» говорит о наличии человека как одной из сторон взаимодействия. Однако это не какой-то мифический обычный человек или среднестатистический пользователь. Каждая система, каждый ПИ предназначен для использования изредка одной, а чаще – несколькими категориями пользователей, которые обладают определенными характеристиками. Процесс определения этих характеристик, создание своеобразных портретов пользователей, выделение их целевых групп являются обязательным этапом проектирования любого интерфейса. Не существует интерфейсов, которые были бы одинаково удобны в использовании для всех людей. Желая создать удобную систему, необходимо задаться вопросом: для кого именно она будет удобна и в каких условиях.

В разработке ПИ нет каких-то конкретных и стандартных правил вроде «сделать адекватный, удобный и простой в обучении интерфейс». Правил нет, потому что для разных людей в силу объективных причин удобными окажутся разные интерфейсы. Необходимо знать, кто будет пользоваться системой и в каких условиях это будет происходить. Причем знать это нужно уже на начальных этапах разработки интерфейса, а именно – на этапе начала работы над проектом либо, в крайнем случае, на этапе постановки задачи. Полагаться в этом вопросе только на здравый смысл иногда можно, но в редких, очень простых случаях.

Как правило, разработчик не может при помощи только воображения определить значимые характеристики будущих пользователей. Или, возможно, ему известны вовсе не те характеристики пользователей, которые могут помочь в разработке интерфейса. К примеру, для разработчика сайта, на котором осуществляется заказ пиццы на дом в режиме он-лайн, не имеет особого значения интенсивность прерываний пользователя, тогда как для разработчика офисного приложения подобная информация крайне важна.

Как узнать характеристики будущего пользователя (построить его модель)? Самым общим методом является опрос людей, являющихся (как вы надеетесь) вашими потенциальными пользователями. Если разрабатываемая программа предназначена для широкого круга

пользователей (скажем, редактор web-страниц), можно выбрать наугад пять коллег, друзей, родственников, рассказать им в общих словах о назначении вашей программы. Затем предложите ситуацию: «Ты работаешь над web-страничкой. У тебя есть файл с картинкой под именем Picture1. JPG. Ты хочешь вставить эту картинку на свою страницу. Как ты себе представляешь свои действия?» Далее полезно задавать вопросы по ходу процесса: «Куда делась картинка? Если теперь ты удалишь свой файл Picture1.JPG, сможет ли страница по-прежнему показывать твою картинку?» Можно спросить, где хранятся ярлычки картинок. Конечно, некоторые пользователи не имеют об этом представления и никогда об этом не задумывались, другим все равно, но когда будет обработано много различных мнений, более отчетливо станет решение, которое вас устроит. Просто просите их думать вслух, задавайте вопросы, которые предполагают альтернативные варианты ответов, и попытайтесь построить модель пользователя. Самый часто встречающийся вариант ответа и есть, как правило, нужная характеристика пользователя, важнейшая часть его модели. Так, постепенно варьируя задания и вопросы, можно построить более или менее адекватную модель пользователя.

Следующий этап – проверка сделанных предположений, т.е. предварительной модели. Проверка предположений предполагает обычно создание прототипа ПИ (хотя бы бумажного на этом этапе – см. разд. 4.2.1.) и его апробацию, т.е. первые варианты его юзабилити-тестирования. Попросите группу приглашенных вами пользователей комментировать свои действия во время решения поставленных задач. Цель тестирования заключается в том, чтобы понять, чего они ожидают от программы. Предположим, вы дали задание: вставить картинку. Если вы увидите, что человек пытается мышью затащить картинку в документ, вы поймете, что вам следует поддержать технологию *drag-and-drop.* Если он остановит курсор на кнопке *Вставка* панели инструментов, вы поймете, что было бы полезно разместить в этом меню опцию *Картинка.* Когда же он на панели инструментов будет пытаться заменить слова *Times New Roman* на *Вставить картинку,* вы сообразите, что вам попался редкий персонаж, который ищет командную строку, будучи совсем не знаком с графическими интерфейсами.

Какое количество пользователей следует привлекать к подобным тестам? Неправильно думать: чем больше, тем лучше. Обычно достаточно пяти-семи человек. Подготовка и проведение тестирования на каждом этапе проектирования ПИ (а иногда и несколько раз внутри одного этапа, ведь тестирование – итерационная процедура) – довольно трудоемкое дело, и если привлекать большее количество людей, то будет наблюдаться скорее всего повторяемость результатов. Важно помнить, что профили пользователей, как правило, не должны быть очень сложными. Догадки обычных пользователей о том, как работает та или иная программа, будут скорее простыми, очевидными и, весьма вероятно, будут отличаться от хитроумных задумок программистов.

В этом разделе рассматриваются два типа моделей пользователя. Первый охватывает характеристики человека, определяемые его социальным и организационным окружением. Эти характеристики представлены с помощью:

• социотехнических моделей, представляющих совместно социальные и технические требования;

• методологии разработки программных продуктов, рассматривающей человеческие и организационные аспекты в едином контексте и в широком смысле;

• совместной разработки, предполагающей непосредственное включение пользователя в процесс работы над проектом.

Второй базируется на когнитивных моделях, т.е. акцент делается на процессах восприятия, памяти, мышления, психологических особенностях. Эти модели представлены:

• иерархической моделью, представляющей задачи пользователя и структуру целей;

• лингвистическими моделями, представляющими единый язык человека и программы;

• физическими моделями, представляющими моторные навыки человека.

Оба типа моделей ставят в центр внимания пользователя: первый тип как бы смотрит на мир вокруг человека, второй – сосредоточен на его индивидуальных особенностях. Сначала же рассмотрим пользовательские профили как наиболее простой тип фиксации характеристик будущего пользователя.

3.2. Пользовательские профили

Наиболее простой и распространенной формой представления характеристик будущего пользователя являются пользовательские профили. Для сбора информации о пользователях используются различные методы: качественные (например, проведение интервью с пользователями конкурирующих продуктов) и количественные (если есть возможность, можно провести формализованное анкетирование пользователей). Кроме непосредственного сбора информации разработчики могут описывать пользователям часть программного продукта на основе своего опыта.

Польза от профилей заключается в том, что впоследствии на их основе отбираются объекты тестирования. Это должно, во-первых, облегчить процесс отбора участников тестирования и, во-вторых, обеспечить валидность результатов тестирования (а тестирование принесет мало пользы, если будет проходить без участия реальных будущих пользователей). В результате работ по определению пользовательских профилей разработчики получают описание главных категорий пользователей, причем часто одна из них может определяться как основная. Точное их количество, разумеется, зависит от системы. Для системы, рассчитанной на массовую аудиторию, количество категорий пользователей будет больше, нежели для системы, предназначенной для использования исключительно специалистами. Например, программа для обработки любительских фотографий должна быть сделана так, чтобы ею могли пользоваться как можно большее количество людей: от программиста с двадцатилетним стажем, у которого постоянно не хватает времени, до вашей бабушки, которая решила привести в порядок семейный фотоальбом. В этом случае категорий пользователей будет, как нетрудно догадаться, несколько больше, нежели при проектировании интерфейса для управления атомной электростанцией.

Каждый из профилей содержит подробное описание характеристик пользователя, существенных для работы с проектируемой системой. Сюда должны входить цели пользователя, его социальные характеристики (пол, возраст, образование, профессия и т.п.), характерные для него модели поведения, условия, в которых он будет использовать систему, навыки пользователя, характеристики его компьютера. Другими словами, все то, что окажет впоследствии значимое влияние на предпочтения пользователя в интерфейсе программы. Ведь создать набор характеристик не проблема. Но нужны именно такие характеристики, которые в дальнейшем станут действительно эффективным средством отбора адекватной целевой аудитории для юзабилити-тестирования.

Пример профиля

1. *Социальные характеристики:*

• пол;

• возраст;

• образование;

• уровень занимаемой должности;

• использует ли компьютер только он и (или) другие (члены семьи, коллеги).

2. *Навыки и умения:*

• общий стаж работы с компьютером;

• стаж использования Интернета;

• уровень теоретических знаний об устройстве Интернета;

• уровень практических знаний о внутреннем устройстве Интернета (что конкретно умеет делать).

3. *Рабочая среда:*

• тип подключения к Интернету;

• размер монитора;

• экранное разрешение;

• быстродействие компьютера;

• используемая операционная система;

• язык операционной системы;

• наиболее часто используемые программные приложения;
• количество времени, проводимого ежедневно за компьютером на работе;
• количество времени, проводимого ежедневно за компьютером дома.

4. *Мотивационно-целевая среда:*
• цели пользователя вообще;
• мотивация к обучению работе с программой (сайтом).

К определению целей и мотивации пользователей следует подходить особенно осторожно, так как тут вполне можно столкнуться с тем, что наши стереотипы и представления о целях вовсе не совпадают с реальным положением вещей. Важно не путать реальные цели и мотивации пользователей с декларируемыми целями. Основные вопросы и рекомендации, которые следует всегда держать в поле зрения при создании профиля потенциальных пользователей, следующие:
• кто они;
• возможно, они не похожи на вас;
• поговорите с ними;
• наблюдайте за ними;
• используйте ваше воображение.

Кто они

Они молоды или стары, опытные пользователи или новички? Ответ на этот вопрос не очевиден, и его надо задавать снова и снова по мере расширения знаний о системе и ее окружении. Этот вопрос тем более труден, чем более универсальным является ПО, которое разрабатывается. Если это текстовый процессор, то ясно, что будет много разных пользователей с различными целями и характеристиками. Подобная проблема очень остра во многих web-сайтах, которые посещают очень разные люди. При этом появляется соблазн представить некоего универсального пользователя с универсальными навыками и универсальными целями. Однако более правильно в подобных ситуациях думать о нескольких определенных группах пользователей.

Возможно, они не похожи на вас

При разработке любой системы достаточно просто представлять дело так, как будто вы и будете ее основным пользователем. Часто разработчик говорит: «Но ведь очевидно, как надо делать». Это может быть очевидно для него. Например, предпочтения мужчин и женщин могут существенно различаться, а большинство сотрудников, разрабатывающих программное обеспечение, мужчины. Необходимо иметь в виду, что, несмотря на значительный разброс индивидуальных особенностей, женщины в целом лучше понимают других людей и менее эгоцентричны.

Поговорите с ними

Трудно определить, о чем думает другой человек, поэтому лучше спросить его. Это может происходить в разных формах (структурированные опросы о работе и жизни, открытые обсуждения, вовлечение потенциальных пользователей непосредственно в процесс разработки). Последнее называется совместной разработкой. Привлекая пользователей к разработке, можно получить более полные знание контекста работы и их потребностей. Однако существует и недостаток: вы не можете быть уверенными в том, что используемая вами группа полностью и адекватно отражает портреты всех потенциальных пользователей.

Люди могут сказать вам о том, как что-либо происходит в реальности, а не то, как должно происходить, по утверждению компании. Нужно завоевать их доверие, так как часто фактические действия входят в противоречие с корпоративной политикой. Такие противоречия типичны для методов, касающихся организационных аспектов.

Наблюдайте за ними

Несмотря на большую важность того, что люди вам говорят, этого для вас недостаточно. Когда обладатель черного пояса по дзюдо излагает, как он произвел бросок противника, его

рассказ, как правило, не соответствует тому, что он в действительности делал. Это тем более очевидно в интеллектуальных видах активности, которые всегда были особенно трудны для интроспекции.

Профессионал имеет прочные практические навыки в своей области и может в ней *делать* что-то. Академик в той же самой области может не уметь делать практические вещи, но он *знает* в данной области много. Знания и навыки – не одно и то же. Иногда люди владеют и тем и другим, но не всегда. Лучшие спортивные тренеры могут не быть лучшими атлетами, лучшие живописцы могут не быть лучшими искусствоведами.

Поэтому важно именно наблюдать то, что люди делают, слышать то, что они говорят. Это предполагает наблюдение и регистрацию того, как они работают и вообще проводят день, используя видеокамеру или магнитофон. Такие наблюдения проводятся и над пользователями. Можно, например, попросить их вести записи или через каждые 15 мин подавать звуковой сигнал и просить их записывать, что они делают (структурированная форма отчета побуждает к более точному ответу).

Другой способ узнать, что люди делают, состоит в том, чтобы смотреть, чем они пользуются и что создают. Посмотрите на типичный стол в офисе. Есть газеты, письма, файлы, возможно, степлер, компьютер, разные заметки… Рассмотрение каждого из них по отдельности не поможет понять, почему они все на столе, при решении конкретной задачи. Только пользователь может объяснить их роль. Иначе говоря, наблюдение говорит вам, что именно пользователь делает, а он говорит вам, почему он это делает.

Используйте ваше воображение

Даже если бы вы хотели привлечь множество потенциальных пользователей к вашей разработке, это не всегда будет возможно. Это может быть слишком дорого, или они, возможно, не смогут уделить вам много времени (например, консультант больницы), может быть и так, что их слишком много (например, если вы создаете web-сайт). Однако, не имея возможности привлечь всех реальных пользователей, вы можете, по крайней мере, пробовать вообразить их опыт. Но это довольно опасно. Было бы легко думать: «Если бы я был складским менеджером, я сделал бы то и это». Следует понять не то, что вы бы сделали на месте пользователей, а то, что они сделают. Это требует использования некоторого метода. Вообразите себя складским менеджером. Какой смысл имеет для него опция меню «undo»?

Как показывает опыт, часто бывает полезно «оживить» потенциального пользователя, придумав ему имя, отчество, элементы биографии, определенные черты характера и даже внешний облик. Для этого создаются так называемые персоны (от англ. «persona» – действующее лицо художественного произведения), хотя лучше перевести это на русский как «персонажи». В данном случае персонаж – это конкретное описание воображаемого пользователя, которого мы придумываем сами. Такое описание создается на основе одного из профилей (другими словами, наш персонаж является представителем одной из определенных ранее категорий пользователей). Это помогает более рельефно изобразить себе типичного представителя какой-либо из пользовательских категорий. При помощи такого персонажа гораздо проще понять пользователя, увидеть за набором данных, собранных в профиле, живого человека. Все это не дает разработчику забыть, для кого разрабатывается продукт. Когда дизайнер постоянно смотрит в глаза пользователю (пусть придуманному), деятельность его становится более осмысленной.

Персонажу дают имя, возраст, описывают его цели в зависимости от того, что за систему мы проектируем, кратко описывают либо его рабочий день (если, например, проектируется офисное приложение), либо другой контекст использования системы. При этом если в профиле сказано: «пользователи данного типа работают в условиях частых прерываний основной деятельности», то в профиле будет написано: «на протяжении первой половины дня Владимиру Ильичу приходится часто отвлекаться от редактирования отчета для того, чтобы отвечать на телефонные звонки клиентов». Иногда даже для документа, который описывает персонаж, рисуется небольшой портрет или вставляется фотография. В дальнейшем созданные персонажи могут очень пригодиться для создания пользовательских сценариев.

Ниже приведен пример персонажа Ольги – складского менеджера.

> Ольге 37 лет. Она была менеджером склада в течение пяти лет и работала в компании «Воздушные замки» в течение 12 лет. Она не поступила в университет, но училась по вечерам в бизнес-школе. У нее двое детей в возрасте 15 и 7 лет, она не любит работать поздно. Она закончила часть вводного компьютерного курса несколько лет назад, но была вынуждена прервать, когда была назначена на должность, и больше не могла позволить себе тратить столько времени. У нее отличное зрение, но движения правой руки немного ограниченны после несчастного случая на производстве три года назад. Она работает с энтузиазмом, проявляет готовность как делегировать ответственность, так и брать ее на себя, если от управления компании поступает соответствующее распоряжение. Однако она ощущает беспокойство от плана введения новой компьютерной системы.

Коллектив разработчиков должен иметь несколько таких персонажей, отражающих различные типы пользователей и их роли. Портрет персонажа основан на результатах изучения реальных пользователей, наблюдения за ними, их профилей и т.д. Когда рассматривается какое-то решение ПИ, разработчики могут спросить: «Как Ольга будет реагировать на это?» Только чувствуя, что Ольга является реальным человеком, разработчики могут вообразить, как она будет себя вести.

3.3. Модели, определяемые социальным и организационным окружением пользователя

3.3.1. Социотехнические модели

В целом этот тип моделей сосредоточен на описании внешних факторов, в частности организационных и социальных, и направлен на совмещение социальных и технических аспектов. Существует целый ряд социотехнических моделей, которые используются при разработке программных продуктов, мы рассмотрим основные из них:

• **USTM** (User Skill and Task Match – соответствие навыков пользо-вателяи требований задачи) и ее форму для малых предприятий **CUS-TOM;**

• **OSTA** (Open System Task Analysis – анализ задач открытой системы);

• **ETHICS** (Effective Technical and Hyman Implementation of Computer Systems – эффективная реализация технической и человеческой составляющих в компьютерных системах).

USTM/CUSTOM. USTM (User Skill and Task Match – соответствие навыков пользователя и требований задачи) – это схематическое представление структуры задачи со словесным описанием. Этим достигается объединение структурности и человеческого фактора. Модификация USTM для малых предприятий именуется CUSTOM, где упор делается на учете требований совладельцев; при этом предполагается, что совладельцы не являются конечными пользователями системой. Совладельцы определяются как лица, на которых сказываются (которые зависят от) успех либо неудачи системы. Выделяют четыре категории совладельцев:

1) первичные – используют систему;

2) вторичные – непосредственно не используют систему, но получают ее «выход» или могут определять исходные данные для системы (например, те, кто получает отчеты, сгенерированные системой);

3) третичные – не попадают в категории 1 и 2, но на них влияет успех или неудача системы (например, директор, который может получать пользу (прибыль) либо убыток в зависимости от работы системы);

4) обеспечивающие – участвуют в разработке, развитии и сохранении системы.

Пример – система заказов авиабилетов. Первичные совладельцы – это офисы туристических агентств, центральный офис бронирования авиарейсов. Вторичные – это пользователи в этих агентствах (т.е. агенты), менеджмент авиакомпании. Третичные – это конкуренты, гражданские власти, совладельцы авиакомпании. Обеспечивающие – это команда

разработчиков, управление департамента информационных технологий.

Модель CUSTOM применяется на начальных этапах разработки: в начале работы над проектом и, возможно, на этапе постановки задачи, когда только определены возможности продукта. Это методология, основанная на заполнении формуляров (готовых форм), которая предполагает определенный набор вопросов на каждой стадии разработки. Краткий пример типичного перечня вопросов в модели CUSTOM таков:

• Чего хотят достичь совладельцы и как измеряется успех?

• Что является источником удовлетворения от работы для совладельцев и что – источником неудовлетворения и стресса?

• Какими знаниями и навыками обладают совладельцы?

• Каково отношение совладельцев к работе и к компьютерным технологиям?

• Имеются ли некоторые групповые предпочтения среди совладельцев, которые будут влиять на приемлемость программного продукта?

• Каковы характеристики задачи совладельцев в смысле ее частоты, фрагментации (или декомпозиции) и выбора действий?

• Встают ли перед совладельцами вопросы, касающиеся конкретной ответственности, безопасности или конфиденциальности (исходя из работы системы)?

• Каковы физические условия работы совладельцев?

Иными словами, определяются и описываются совладельцы (по именам), их роль и функции в работе, их первичные цели, реальное влияние на дела, знания, навыки, готовность к новациям, обычные ежедневные задачи и т.д. Аналогично определяются и описываются рабочие группы, при этом уделяется особое внимание связкам «задача-средство». CUSTOM создает полезную канву для понимания потребностей совладельцев путем использования простых бланков и относительно стандартных вопросов (все это может делаться вручную, так как эта работа не очень трудоемкая).

OSTA (Open System Task Analysis – анализ задач открытой системы) описывает прежде всего организационное окружение технической системы. В OSTA пользовательские аспекты системы, такие, как потребительские свойства и доступность, объединены с техническими аспектами, например с системным функционированием. В OSTA различают восемь стадий:

1) в терминах целей пользователя описывается основная задача, которую технология должна реализовать;

2) определяются способы ввода задач в систему. Эти способы могут иметь разные характеристики, что может явиться некоторым ограничением для разработки;

3) описывается внешнее окружение, в котором могут быть представлены физические, экономические и даже политические аспекты;

4) описываются процессы трансформации внутри системы в терминах выполняемых действий или объектов;

5) проводится социологическое описание пользователей, учитывающее существование рабочих групп и отношения внутри и вне организации;

6) описывается техническая система в терминах ее конфигурации и объединения с другими системами;

7) определяются показатели функционирования, охватывающие и технические, и социальные характеристики системы;

8) точно определяется новая техническая система.

Выходы OSTA представляются в виде описаний, понятных разработчикам (например, схемы, графики и текстовые описания).

ETHICS (Effective Technical and Hyman Implementation of Computer Systems – эффективная реализация технической и человеческой составляющих в компьютерных системах). ETHICS, так же как и OSTA, имеет дело с техническими и человеческими требованиями, но отличается от OSTA тем, что использует две принципиально разные команды разработчиков. Одна направлена на техническое решение вопроса, не вдаваясь в человеческие проблемы, другая заботится в основном об адекватности системы и человеческих проблем, не особо вдаваясь в их программную реализацию. Иначе говоря, модель ETHICS основана на двух параллельных и до какого-то времени независимых частях разработки – человеческом и

техническом аспектах. В модели ETHICS соответствующие команды разработчиков работают отдельно и только потом пытаются объединить свои решения. Предполагается, что тем самым уменьшается влияние разных специалистов друг на друга. Суть метода – независимая работа двух команд: человеческих и технических предпочтений. Затем результаты предлагаемых каждой командой проектов пытаются совместить, создавая продукт, удовлетворяющий требованиям обеих команд. В методе ETHICS различают несколько основных стадий:

1) определяется проблема и описываются система, цели и задачи, а также критерии удовлетворительного функционирования. Определяются ограничения системы – как технические, так и эргономические;

2) формируются две команды разработчиков, одна по проверке человеческих аспектов, другая – технических. Цели и задачи, описанные на первой стадии, ранжируются по приоритетности и проверяются на совместимость до того, как принимаются технические и социальные решения;

3) рассматриваются две группы решений – с упором на технические и человеческие аспекты. Эти решения оцениваются по заранее установленному (на первой стадии) критерию, составляется список возможных вариантов, желательно короткий;

4) проверяются на совместимость решения, выделенные на третьей стадии;

5) ранжируются в соответствии с ранее выбранным критерием совместные пары человеко-технических решений;

6) разрабатываются детали проекта.

3.3.2. Методология разработки программных продуктов, рассматривающая в едином контексте человеческие и организационные аспекты

Методология разработки программных продуктов (Soft Systems Methodology – SSM) – второе направление учета характеристик человека при разработке программных продуктов в целом и ПИ в частности. Мы рассматривали социотехнические модели как средство, позволяющее определить потребности, как человеческие, так и технические. Методология разработки программных продуктов рассматривает человеческий и технический аспекты в рамках единой системы, где и люди, и технология разработки есть лишь ее компоненты. Существо SSM – в акценте на взаимном и совместном понимании проблем всеми разработчиками, как ориентирующимися на технологию, так и теми, кому важнее человеческая адекватность. Эта методология включает несколько стадий. При этом различают стадии, относящиеся к реальному миру, и стадии, относящиеся к системе.

Стадия 1 – осознание проблемы и начало анализа, что сопровождается детальным рассмотрением проблемы. Картина должна включать всех участников, их задачи, интересы, группы, в которые они входят, организационные структуры и процессы и может быть выражена по-разному, но всегда ясно, что здесь нет верных или неверных ответов; она должна отражать все аспекты функционирования системы и быть понятной для разработчиков.

Стадия 2 – это движение от реального мира к миру системы и попытка сформулировать «базовые определения» системы. Таких определений может быть несколько, к примеру отражающих разное понимание системы разработчиками. Базовые определения описываются в терминах CATWOE (Clients, Actors, Trasformations, Weltanschauung, Owner, Environment):

• Clients – те, кто получает продукцию и/или прибыль от работы системы.

• Actors – те, кто работает внутри системы.

• Trasformations – трансформация изменений, произведенных системой. Важнейшая часть базовых определений, так как это ведет к деятельности, которая требует включения в следующую стадию. Чтобы выявить трансформации, рассматривают вход и выход системы.

• Weltanschauung (с нем. – мировоззрение). Как понимается система с позиций конкретного базового определения.

• Owner – собственники, т.е. те, кому принадлежит система, кому она подотчетна и кто вправе санкционировать изменения в ней.

• Environment – внешнее окружение, т.е. мир, в котором функционирует система и который влияет на нее.

Пример. *Базовые определения для менеджмента пассажирскими авиаперевозками – системы бронирования билетов.*

Рассматривается некая система бронирования авиабилетов для сотрудников туристических агентств, продающих билеты населению. Эта система принадлежит менеджменту пассажирских авиаперевозок, управляется объединением туристических агентств, работает по всему миру в сети туристических агентств, в рамках международного законодательства в части авиаперевозок:

- Clients – пользователи системой.
- Actors – объединенный центр управления туристических агентств.
- Trasformations – предпочтения, запросы и требования клиентов к путешествию, выражаемые в покупке билетов на авиарейсы.
- Weltanschauung – прибыль может быть повышена более эффективной организацией продаж.
- Owner – менеджмент пассажирских авиаперевозок.
- Environment – международные законодательные акты, касающиеся международных авиаперевозок, и национальные законодательства. Общенациональные требования безопасности.

После того как базовые определения разработаны, формируется концептуальная модель. Эта модель определяет, что именно система должна делать, чтобы соответствовать базовым определениям. В концептуальной модели фиксируются трансформации и рабочие процессы в системе и (что очень важно) все они иерархически структурируются с обязательными пояснениями – что достигнуто и как достигнуто. Это итеративный процесс, и полная ясность достигается, как правило, путем нескольких итераций.

Далее мы возвращаемся к реальному миру с нашим набором базовых определений и сравниваем реальную систему с концептуальной моделью, выявляем расхождения, и таким образом высвечиваем необходимые изменения или потенциальные проблемы.

SSM – гибкий подход, который предполагает детальное рассмотрение всего контекста проектирования. Для эффективного использования SSM требуются навыки. В целом нет жестких методик и единственно правильных подходов к проведению процедур SSM. Это хороший подход, если он помогает разработчику шире понять систему.

3.3.3. Модели совместной разработки

Идея совместной разработки (Participatory design) заключается в том, что пользователи непосредственно включены в процесс разработки как полноправные члены команды проектировщиков. Это приводит к изменению всей работы и организационных процессов, причем любые изменения в систему вносятся только в тех случаях, когда это приемлемо для пользователя. По сути, по ходу разработки все требования к системе с каждой итерацией как бы фильтруются через сито удобства и возможностей пользователя. Этот подход разработан в Скандинавии, где он закреплен законодательно и широко используется.

Процесс совместной разработки содержит ряд приемов, помогающих обмену информацией между разработчиками и пользователями. Это такие методы, как совместный мозговой штурм для решения отдельных задач системы, пошаговая раскадровка работы пользователя в течение всего рабочего дня (или некоторого законченного рабочего цикла). Это также параллельное изложение рабочего процесса пользователем и разработчиком как средство дополнительного получения знаний: разработчик спрашивает пользователя о процессе реальной работы с программой, а пользователь – разработчика об особенностях технологии и возможностях программы. Широко практикуется при этом также использование бумажных прототипов, на которых как бы проигрываются разные варианты работы с системой. Это простой и дешевый способ ранней оценки программных систем вообще и ПИ в частности. Эти методы вовсе не обязательно используются только в совместных разработках, они

универсальны.

3.4. Модели, основанные на когнитивном подходе (с учетом процессов восприятия, памяти, мышления, психологических особенностей пользователя)

Это модели, отражающие механизмы понимания, формирования и функционирования структуры знаний, мотиваций, процессов переработки информации пользователем в процессе взаимодействия с интерфейсом. Различают эти модели по степени адекватности представления компетенции (уровня знаний) и деятельности пользователя.

Модели компетенции предназначены для описания правильной последовательности поведения, но обычно без рассмотрения реального поведения пользователей, это так называемые нормативные модели. Противоположны им *модели деятельности,* которые описывают не только правильную последовательность действий пользователя, но также и необходимые для этого знания и то, как эти знания реально используются при выполнении конкретных задач. Модели компетенции, следовательно, представляют ожидаемое поведение пользователя, но они малополезны для представления реального поведения пользователя. Модели же деятельности сосредоточены на реальном поведении, но в ограниченном круге приложений. Другим отличием моделей компетенции от моделей деятельности является то, что первые основаны на сборе информации и составлении планов действий, а вторые – на выполнении планов и действий. Когнитивные модели представлены обычно тремя разновидностями:

1) иерархической структурой задач и целей пользователя;

2) лингвистико-грамматическими моделями;

3) физическими моделями (в основном исполнительных действий на уровне «человек – пульт управления»).

Первая разновидность имеет дело с четкой структурой (как правило, иерархической), отражающей взаимосвязь целей и задач. Вторая разновидность предполагает представление целей, задач и вообще всех действий на языке пользователей, т.е. на обычном языке, но этот язык должен быть предельно понятен для пользователей. Третья разновидность делает упор на исполнительском, моторном уровне, жертвуя уровнем понимания (т.е. делай так, потом так).

Когнитивные модели, иными словами, оперируют четырьмя основными категориями субъективной структуры знаний пользователя: пониманием, знанием, намерениями, способом преобразования информации.

3.4.1. Иерархическая структура задач и целей пользователя

Многие модели пользователей основаны на представлении о психических процессах как об иерархической структуре, на верхнем уровне которой находятся основные задачи, которые надо решить, на нижележащем уровне – их разбиение на подзадачи, на следующем нижележащем уровне – разбиение уже этих подзадач на подзадачи более мелкие и т.д. до следующего уровня, который представлен обычно простыми исполнительскими действиями. При этом каждой подзадаче соответствуют процедуры ее решения и другие факторы, учет которых необходим для решения. Такого рода разбиения имеют место и в ряде других моделей, но здесь эта черта является главной. Основные аспекты иерархической декомпозиции задач следующие:

• степень детализации:

• где мы начнем;

• где мы закончим;

• обычные приемы при изучении поведенческих аспектов (а не решения проблем):

– задача выделенного блока (модуля) поведения;

• конфликт:

– более одного пути достижения цели;

• ошибки.

Выделяют следующие технологии декомпозиции:

• метод GOMS;

• теорию когнитивной сложности (CCT);

• иерархический анализ задач.

Наиболее распространенным вариантом реализации этих моделей являются модели, объединенные аббревиатурой GOMS (Goals, Operators, Methods, Selections – цели, операторы, методы, правила их выбора), и модели когнитивной сложности, обозначаемые обычно как CCT (Cognitive Complexity Theory).

Составляющие модели GOMS следующие.

Цели – это то, что пользователь хочет достигнуть. Причем они должны быть такими, которыми действительно оперирует пользователь, т.е. адекватно отражать субъективное разбиение им цели на подцели разного уровня.

Операторы – это самый нижний уровень анализа. Он предполагает действия, которые надо произвести для выполнения задачи. Например, нажать клавишу такую-то или совершать какие-то внутренние действия нижнего уровня, скажем прочесть диалоговое окно. Как и в случае с целями, степень детализации зависит от квалификации потенциального пользователя, иначе дробление операций может быть совсем примитивным – для малоквалифицированных пользователей и крупным – для более продвинутых.

Методы, которыми может быть получен тот или иной результат.

Скажем, вызов файла может производиться из подменю «Документы» меню «Старт», если этот файл недавно использовался, или «кли-каньем» на его иконке, если такая иконка выведена у нас на экран, или вызовом определенного диска, поиском в нем соответствующей папки, пока не найден этот файл, или с помощью поиска меню «Найти» и т.д. В методе GOMS каждый из путей представлен в виде иерархической структуры и в связке с соответствующим методом его выполнения.

Выборы – здесь речь идет о выборе того или иного метода, т.е. пути достижения результата. Идеология GOMS предполагает, что выбор метода – не случайный процесс: необходимо предсказать, какой именно метод будет использован и в каких условиях. Конечно, это зависит от нашего представления о потенциальном пользователе, от состояния системы, от конкретной цели.

Из всего этого видно, что анализ в терминах GOMS практически всегда лежит на пути декомпозиции некоей целостной задачи. Типичный GOMS-анализ состоит, следовательно, из какой-либо одной задачи сравнительно высокого уровня, и далее идет ее древовидная декомпозиция до уровня простейших действий. GOMS-анализ может содержать и характеристики деятельности. Глубина цепочки подцелей может использоваться для оценки требований к кратковременной памяти. Кроме того, в GOMS-анализе часто используют значения времени для каждого действия, чтобы прогнозировать временные затраты (см. разд. 4.2). Такие значения предварительно определяют в экспериментальных исследованиях. Методология модели GOMS является, по сути, базовой для большинства когнитивных моделей, используемых в челове-ко-компьютерном взаимодействии. Эта методология хорошо подходит для описания исполнительских действий, а при известных физических устройствах, на которых производятся эти исполнительские действия, она успешно используется и для прогнозирования времени выполнения различных задач. Пример декомпозиции на основе метода GOMS:

ЦЕЛЬ: ЗАКРЫТЬ-ОКНО
• [select GOAL: USE-MENU-METHOD
• MOVE-MOUSE-TO-FILE-MENU
• PULL-DOWN-FILE-MENU
• CLICK-OVER-CLOSE-OPTION GOAL: USE-CTRL-W-METHOD
• PRESS-CONTROL-W-KEYS]

Для конкретного пользователя:

Правило 1: Select USE-MENU-METHOD unless another rule applies Правило 2: If the application is GAME, select CTRL-W-METHOD

Вместе с тем в рамках этой методологии невозможно учесть такие важные вопросы, как соответствие предъявляемой информации уровню знаний пользователя, а значит, и обосновать объем необходимой тренировки, учесть динамику времени выполнения действий при разной тренированности и пр. Отметим, что предшественником этого метода является широко известный метод анализа деятельности Г.М. Зараков-ского (операционно-психофизиологический метод).

3.4.2. Модели, основанные на теории когнитивной сложности

Модель CCT (Cognitive Complexity Theory) является продолжением GOMS-модели, в которую введена параллельная линия описания каждой подзадачи в терминах жесткой привязки ее к конкретным действиям пользователя. Каждый элемент иерархической структуры, сделанной по модели GOMS, является условием реализации определенного действия, т.е. каждый элемент представлен в виде продукционного правила: если *(условие),* то *(действие).* Условием при этом является содержание активной памяти пользователя в данный момент времени, и если условие имеет место, то продукционное правило выполняется.

Описание последовательностей таких продукционных правил, как и всякий ветвящийся процесс, удобнее всего представлять на языке LISP. Это позволяет существенно обогатить картину деятельности и даже количественно сравнивать интерфейсы, считая более простым тот из них, работа с которым представлена меньшим числом продукционных правил. Этот тип моделей имеет еще одно преимущество, заключающееся в возможности представить структуру деятельности пользователя и компьютерной системы в едином контексте, что позволяет видеть те области, где они противоречат друг другу, и добиваться соответствия. Таким образом, основными составляющими CCT-моделей являются:

- параллельное описание;
- представление действий в виде процедур;
- преимущественное внимание к поведению новичков, а не опытных пользователей;
- возможность описания поведения при ошибках.

Меры: глубина структуры целей; число правил; сопоставление с описанием средства технической реализации.

Пример описания в терминах CCT-модели четырех действий по вставке пробела представлен на рис. 3.1.

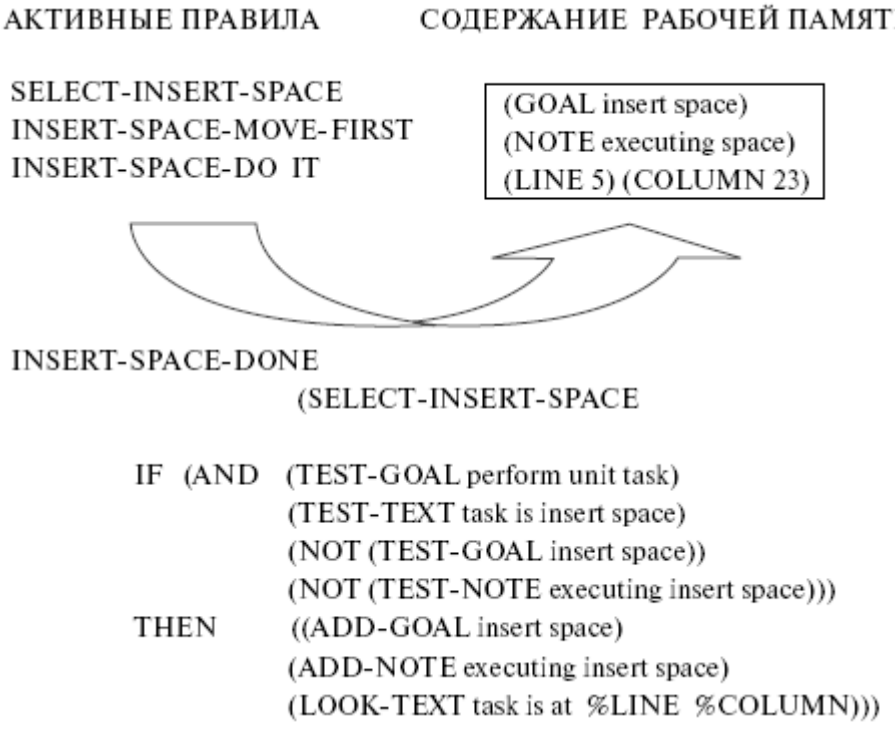

Рис. 3.1. Пример описания в терминах CCT-модели

Конечно, эти модели имеют и свои ограничения. Так, принципиально возможно несколько способов описания одной и той же активности пользователя и изменений интерфейса. Кроме того, структура задач и соответствующих им действий для новичка и опытного пользователя будет разной, но такое различие невозможно провести для описания компьютерной системы. И наконец, объем описаний и продукционных правил для каждого типа интерфейса может оказаться непомерно большим.

3.4.3. Лингвистико-грамматические модели

Эти модели основаны на формализации языка, коим произведено описание человеко-компьютерного взаимодействия. В основном используются формы метаязыка Бекуса-Наура (Backus-Naur Form – BNF), хорошо известные в информатике. Приведем краткое пояснение. Дело в том, что для описания языка-объекта должен применяться некий язык высокого уровня, т.е. метаязык. Но метаязык также должен обладать некоторыми свойствами формального языка, чтобы однозначно определять конструкции языка-объекта. Следовательно, метаязык должен быть сначала описан сам, для чего также нужен язык. Естественно, может сложиться впечатление, что такой процесс никогда не закончится.

Однако доказано, что для описания любого метаязыка можно использовать язык естественный. Таким образом, для построения формального языка необходимо средствами естественного языка описать метаязык, а затем посредством метаязыка описать язык формальный.

Один из широко распространенных метаязыков известен как *нотации Бекуса-Наура.* Для формирования предложений в форме Бекуса-Наура используются универсальные метасимволы: {‹,›,::=, |}. Первые два метасимвола называют угловыми скобками – они служат для обрамления нетерминального символа. Символ «::=» читается «по определению есть»; символ «» – «или». В предложениях, записанных в форме Бекуса-Наура, нетерминальный символ, стоящий в угловых скобках, играет роль определяемой конструкции языка-объекта. Терминальные символы отражают самый нижний уровень действий пользователя, т.е. нажатие на клавишу, «кликанье» мышью или передвижение мыши. Нетерминальные символы отражают более абстрактные действия и определены другими символами – как терминальными, так и нетерминальными.

Описание формального языка строится из последовательности формул, каждая из которых в левой части содержит один метасимвол, обозначающий некоторую конструкцию языка-объекта. Правая часть такой формулы содержит либо перечисление метасимволов и терминальных символов языка-объекта (никаких разделителей при этом не ставится), либо совокупности перечислений, разделенных символом «|». Правая и левая части объединяются в единую формулу знаком «::=».

Язык-объект можно считать полностью определенным в форме Бекуса-Наура, если любой нетерминальный символ можно представить последовательностью терминальных символов. В качестве примера можно рассмотреть определение понятия «идентификатор», которое используется во многих языках программирования. На естественном языке определение звучит следующим образом: «Идентификатор – это любая последовательность букв и цифр, начинающаяся с буквы». В форме Бе-куса-Наура оно будет выглядеть следующим образом:

<идентификатор>::=<буква>|<идентификатор><буква>|<идентификатор><цифра>

<буква>::= a|b|c|d|e... <цифра>::= 0|1|2|3|...|9

Видно, что в определении данного понятия присутствует рекурсив-ность, поскольку понятие «идентификатор» определяется через самое себя. Элементарным оказывается идентификатор из одной буквы.

Достоинство нотаций Бекуса-Наура в том, что они представляются в буквенном виде; неудобны нотации однообразностью способов построения предложений языка-объекта – запись

оказывается громоздкой и плохо воспринимается.

Гораздо более наглядным следует считать другой способ описания формального языка, предложенный Николасом Виртом – создателем языка программирования PASCAL – и получивший название «синтаксические диаграммы». *Синтаксическая диаграмма* – это схема (графическое представление) описания какого-либо нетерминального символа языка-объекта. Схема всегда имеет один вход и один выход. Элементами схемы могут служить терминальные символы языка-объекта, заключенные в окружность (или овал), или нетерминальные символы (понятия) языка-объекта, заключенные в прямоугольник. Элементы соединяются между собой направленными линиями, указывающими порядок следования объектов в определяемом нетерминальном символе. Структура синтаксических диаграмм идентична структурам языков программирования, что позволило широко использовать диаграммы для написания трансляторов различных языков. Первым языком, описанным с помощью синтаксических диаграмм, был язык Паскаль. Нотации Бекуса-Наура и синтаксические диаграммы – это два альтернативных способа описания конструкций метаязыка, с помощью которого строится формальный язык.

После того как построена формальная грамматика и ею порожден язык, последний может быть использован для решения прикладных задач: коммуникации, хранения и обработки информации. Последний класс задач приводит к необходимости формулировки с помощью языков последовательностей обработки информации, т.е. алгоритмов, и их представлению в форме, доступной для понимания и исполнения лицом или техническим устройством, которые обработку производят.

Описание активности пользователя таким языком может отражать только действия пользователя, но не его восприятие интерфейса. Это, конечно, приводит к существенным погрешностям рассматриваемой модели, самой, наверное, общей из всех когнитивных моделей. Частично указанное ограничение смягчено введением в грамматику языка формы «поиск информации».

Попытка учесть знания, имеющиеся у пользователя, отражена в модификации описанного подхода, именуемой «грамматика связки задача-действие» (TAG – Task-Action Grammar). Отличие этого подхода – во введении некоторых параметризованных грамматических правил, позволяющих выделить повторяемость и учесть некоторые знания пользователя (например, верх противоположен низу). Для иллюстрации возможности выделение повторяемости (по сути, возможностей обобщения информации) приведем пример описания трех команд ОС UNIX: cp (copy – для копирования файлов), mv (move – для перемещения файлов) и ln (link – для связывания файлов). Каждая из них имеет по два аргумента – имя исходного файла и имя файла назначения либо имя исходного файла и имя директории назначения. Тогда описание в форме Бекуса-Наура (сокращенно BNF) будет выглядеть так:

```
Копировать   ::=   'cp'   +   имя файла   +   имя файла
                 |   'cp'   +   имя файла   +   директория

Переместить  ::=   'mv'   +   имя файла   +   имя файла
                 |   'mv'   +   имя файла   +   директория

Связать      ::=   'ln'   +   имя файла   +   имя файла
                 |   'ln'   +   имя файла   +   директория
```

Средствами BNF нельзя выделить повторяющиеся места (т.е. общее) в этом описании. В TAG-модификации это сделать можно, и аналогичная запись будет выглядеть следующим образом:

```
Файл-опции          ::=   команда + имя файла + имя файла
                        |  команда + имя файла + директория
Команда (Опция = копировать)      ::=   'cp'
Команда (Опция = переместить)     ::=   'mv'
Команда (Опция = связать)         ::=   'ln'
```

Преимущества записи в TAG-модификации очевидны. Таким образом, лингвистико-грамматические модели развивают иерархические описания модели GOMS в направлении повышения их содержательности.

3.4.4. Физические модели

Эта группа моделей рассматривает исключительно моторный, конечный исполнительский уровень деятельности пользователя, что обусловлено их ориентацией на оценку характеристик конкретных действий пользователя. Например, модель клавиатурного уровня рассматривает последовательности простых действий на клавиатуре, не превышающих обычно 20 с в одной серии. Понятно, что такая модель не распространяется на достаточно сложную деятельность. Базовым допущением этих моделей является иерархическая декомпозиция пользователем любой сложной задачи на подзадачи до самого нижнего, моторного уровня (как это делается в методе GOMS – см. разд. 4.4.1). Именно эти моторные действия и являются предметом рассмотрения данных моделей. Для создания физической модели исполнительские действия разбиваются на заранее установленные типы операторов, обычно включающие:

• **K** (Keystroking) – нажатие на клавиши, включая и вспомогательные клавиши (типа Shift, например).

• **B** (Button) – нажатие на клавишу мыши.

• **P** (Pointing) – указание, передвижение мыши (трекбола и т. п.) к цели.

• **H** (Homing) – возврат руки, перемещение руки от мыши к клавиатуре.

• **D** (Drawing) – рисование линий мышью.

• **M** (Mental) – подготовка (планирование) моторных действий в уме.

• **R** (Responce) – ответ системы, который пользователь может и проигнорировать (например, если отправлена копия документа на печать).

Выполнение задачи может включать чередование разных операций. Расчет общего время выполнения операций предполагает аддитивность, т.е. времена каждой операции суммируются. Экспериментально полученные значения времени выполнения отдельных операций приведены в табл. 3.1.

Таблица 3.1[6]

Экспериментально полученное время выполнения отдельных операций

[6] В данном выражении D – расстояние до цели, а S – размер цели.

Операторы	Пояснения	Время, с
K	Нажатие на клавишу клавиатуры:	
	Тренированный (90 слов/мин) Средний (40 слов/мин)	0,12 0,28
	Нетренированный	1,2
B	Нажатие на клавишу мыши: Движение только вниз или вверх	0,1
	«Кликанье»	0,2
P	Указание мышью. Закон Фиттса	$0,1\log_2(D/S+0,5)$[1]
	Среднее движение	1,10
H	Движение рук к клавиатуре и от нее	0,4
D	Рисование – зависит от величины области	–
M	Подготовка (планирование) действия в уме	1,35
R	Ответ системы	–

Конечно, все это время зависит от тренированности пользователя. Оператор «М» есть самое неясное место этой модели, его значение во многом зависит от квалификации пользователя, его восприятия ситуации. Использование данной модели требует внимания. С ее помощью удобно сравнивать пользовательские характеристики, скажем, разных способов ввода (например, либо путем «кликанья» на иконке, либо путем выбора из меню, либо путем нажатия одновременно на какие-то клавиши). При грамотном подходе оценка времени исполнительских действий показывает вполне удовлетворительные результаты, ошибка не превышает 20%. Особенно продуктивно применение этого подхода при часто повторяющихся задачах типа ввода данных или пользования телефоном.

К физическим моделям относится также так называемая модель трех состояний, разработанная специально для сравнения разных устройств ввода данных. Сопоставление мыши, трекбола и светового пера позволило выделить такие состояния, как перемещения без нажатия на клавишу (состояние 1) и перемещение с нажатием на клавишу, что обычно используется для перетаскивания иконки или файла (состояние 2). Однако световое перо имеет еще и третье состояние, когда оно не касается экрана, и это состояние обозначается цифрой 0.

Сенсорный экран аналогичен световому перу без клавиши, поэтому, когда палец не касается экрана, это состояние обозначается тоже как 0, а когда касается – 1. Таким образом, сенсорный экран может находиться в состоянии 0 или 1, а мышь – в состоянии 1 или 2. Учитывая, что состояния 0-2 и 0-1 не различаются между собой, существует всего три варианта. В случае, если устройство имеет несколько клавиш, то одно и то же состояние имеет место при нажатии на любую из них, и тогда обозначать их следует 2левая, 2средняя, 2правая.

Эта модификация физической модели позволяет характеризовать разные состояния устройства ввода через возможные способы ввода. Например, можно сравнить эффективность двух устройств в закрывании окна мышью и трекболом. Способ закрывания – путем перемещения мышью курсора к меню «Файл», затем нажатие на левую клавишу мыши и перетаскивание ее до наведения курсора на опцию «Выход», затем – отпускание клавиши. Используя таблицу коэффициентов закона Фиттса 2, имеем (обозначения операторов сохраняем, как указано выше):

Мышь

$$P\text{ (к строке меню)} = -107 + 223\log_2(11) = 664\text{ мс};$$
$$P\text{ (к опции «Выход»)} = 135 + 249\log_2(5) = 713\text{ мс}.$$

Трекбол

$$P\text{ (к строке меню)} = 75 + 300\log_2(11) = 1113\text{ мс}.$$
$$P\text{ (к опции «Выход»)} = -349 + 688\log_2(5) = 1248\text{ мс}.$$

Из сопоставления времен видно преимущество мыши перед трекболом, во всяком случае для рассматриваемой операции.

Контрольные вопросы

1. Что такое модель пользователя и пользовательский профиль?
2. На какие типы разделяются модели пользователей?
3. Описание моделей, определяемых социальным и организационным окружением USTM, OSTA и ETHICS.
4. Существо методологии разработки программных продуктов, рассматривающих в едином контексте человеческие и организационные аспекты (SSM-методологии).
5. Базовые положения SSM-методологии и примеры их использования.
6. Описание моделей, базирующихся на когнитивных процессах.
7. Общность и различия между тремя типами когнитивных моделей:
а) основанных на иерархической структуре задач и целей пользователя;
б) лингвистико-грамматическими;
в) физическими.
8. Метод GOMS и его описание.
9. Формы Бекуса-Наура, их роль и описание.
10. Технологии декомпозиции действий пользователя и процесса функционирования системы.
11. Правила описания портрета потенциального пользователя (персонажа).

Литература

Основная

1. *Раскин Д.* Интерфейс: новые направления в проектировании компьютерных систем. – М.: Символ-Плюс, 2006.
2. *Солсо Р.* Когнитивная психология: Пер. с англ. – М.: Тривола, 1996.
3. *Мандел Т.* Разработка пользовательского интерфейса. – М.: ДМК, 2001.
4. *Ковальски Р.* Логика в решении проблем. – М.: Наука, 1990.
5. *Dix A., Finlay J.* and others (Eds). Human-Computer Interaction. – Printed in Great Britain, Glasgow. Publisher – Prentice Hall: 3rd edition, 2004.
6. ISO/DIS 9241-13: User guidance.
7. ISO/DIS 9241-14: Menu dialogues.
8. ISO/DIS 9241-15: Command language dialogues.
9. ISO/IEC 10741-1 Dialogue interaction – Cursor control for text editing.

Дополнительная

10. *Тейз, Грибомон, Луи.* Логический подход к искусственному интеллекту: Пер. с фр. – М.: Мир, 1990.
11. *ЛорьерЖ.-Л.* Системы искусственного интеллекта. – М.: Мир, 1991.
12. *MacKenzie S., Sellen A., Buxton W.* A comparison of input devices in elemental pointing and dragging tasks. In: S.P.Robertson, G.M. Olson, and J.S. Olson, editors, Reaching through technology – CHI'91 conference proceedings, pages 161-166. Human Factors in Computing Systems. – N. Y.: ACM Press. April, 1991.
13. *Card S.K., Moran T.P., Newell A.* The Psychology of Human Computer Interaction. –

Lawrence Erlbaum, 1983.

14. *Bovair S., Kieras D.E., Poison P.G.* The acquisition and performance of text-editing skill: a cognitive complexity analysis // Human-Computer Interaction. 1990. Vol. 5. № 1. P 1-48.

15. *Schiele F., Green T.* HCI formalisms and cognitive psychology: the case of task-action grammars // M.D. Harrison and H.W. Thimbleby (Eds). Formal methods in Human-Computer Interaction. – Cambridge University Press, 1990. Ch. 2.

ТЕМА 4. ОЦЕНКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Изучаемые вопросы:

• Цели оценки ПИ и условия, в которых они могут производиться.

• Преимущества и недостатки оценки в лабораторных условиях и в условиях реальной работы.

• Методы оценки, основанные на анализе мнений экспертов.

• Методы оценки, основанные на участии пользователей.

• Основные проблемы оценки, основанной на участии пользователей.

• Классификация аналитических методов оценки.

• Классификация экспериментальных методов оценки и методов на основе анкетирования.

• Классификация методов наблюдений.

• Физиологические параметры, измеряемые при оценке ПИ, и методы их регистрации.

• Критерии оценки ПИ.

• Методы оценки скорости работы и пути увеличения скорости работы.

• Субъективная длительность действий и способы ее уменьшения.

• Факторы, влияющие на скорость и производительность работы, и способы нейтрализации отрицательных факторов.

• Основные направления усилий по снижению числа ошибок и отражение таких усилий в характеристиках ПИ.

• Сообщения об ошибках, стиль, формы предъявления.

• Пути создания адекватной субъективной модели логики работы.

• Роль метафор в ПИ, примеры метафорических дизайнерских решений в ПИ, преимущества и опасности использования метафор.

• Способы передачи свойств объекта, показывающих способ взаимодействия с ним.

• Язык шаблонов в ПИ, их роль, причины широкого распространения, стандартные правила описания шаблона.

• Принципы интуитивной понятности ПИ.

• Подсистемы справки, их виды и содержание.

• Спиральность текста справки, роль и способы реализации.

• Основные составляющие субъективной удовлетворенности пользователя.

• Характерные ошибки в ПИ, вызывающие стресс у пользователей.

• Содержание восьми «золотых правил» Б. Шнейдермана.

• Коллекции хороших и плохих ПИ, имена наиболее известных сайтов.

4.1. Общие положения и структура методов оценки пользовательского интерфейса

Различают три вида оценок юзабилити, функциональности и доступности, целями которых являются:

• оценка функциональных возможностей системы;

• оценка влияния ПИ на пользователя;

• формулирование возникающих проблем.

Все эти оценки могут производиться как в лабораторных условиях, так и в процессе реальной работы.

Оценка в условиях лаборатории. Преимуществами такой оценки являются следующие:

• возможность использования специального оборудования;

• неизменность окружающей обстановки;

• управление выбранными переменными диалогового поведения;

• возможность выбора гипотезы, которая и проверяется;

• возможность исследования множества ситуаций, отличающихся только заданными переменными;

• возможность варьирования методов и единиц измерения действий пользователей или экспертов в зависимости от разных условий.

К недостаткам можно отнести:

• отсутствие контекста;

• трудность реализации кооперации нескольких пользователей (в методах, где предусмотрено их участие);

• необходимость моделирования, прототипирования;

• необходимость соответствия условиям – если система расположена в опасном или неудобном для натурных экспериментов месте либо по иным причинам непосредственные исследования на ней излишне затратные.

Оценка в условиях реальной работы. Преимущества такой оценки следующие:

• естественная среда;

• сохранение контекста (хотя наблюдение может это заменить);

• возможность лонгитюдных, т.е. продолжительных, исследований в течение длительного времени.

К недостаткам можно отнести:

• отвлекающие моменты;

• шум;

• необходимость соответствия условиям – для продолжающихся долговременных исследований важен именно контекст.

Одни группы методов оценки основаны на *анализе мнений экспертов.* Это аналитические методы (чаще всего когнитивный анализ), методы моделирования реальной работы (здесь чаще всего встречается эвристическая оценка), методы анализа самого процесса разработки.

Другие группы методов оценки основаны на участии пользователей. Это экспериментальные методы, методы наблюдения, методы анкетирования. Методы оценки должны тщательно отбираться и быть адекватными выполняемой работе и целям оценки.

Методы, основанные на анализе мнений экспертов

Различают три основных метода оценки: 1) когнитивный анализ; 2) эвристическую оценку; 3) оценку на основе анализа процесса разработки.

Когнитивный анализ позволяет оценивать, насколько хорошо осуществляется поддержка процесса обучения пользователя. Обычно он выполняется специалистом по когнитивной психологии. Эксперт идет шаг за шагом сквозь процесс разработки, используя психологические принципы с тем, чтобы выявить проблемы. Для этого надо иметь описание некоего набора задач и их разбиения на отдельные действия, которые пользователь должен произвести (метод GOMS). Эксперт анализирует каждое действие, отвечая на вопросы:

1. Соответствует ли результат действия ожиданиям пользователя?

2. Видит ли пользователь, что это действие возможно, когда он хочет его осуществить?

3. Понимает ли пользователь, как именно произвести это действие?

4. Достаточно ли понятна пользователю обратная связь о результатах произведенного действия?

Для анализа используются унифицированные формы, что позволяет провести сравнение и последующее обобщение.

Кроме того, в каждой задаче когнитивный анализ дает возможность установить:

• какое воздействие на пользователя будет иметь разработанное интерактивное взаимодействие;

• какие когнитивные процессы задействованы;

• какие проблемы обучения могут возникнуть;

• как анализ фокусируется на целях и знаниях: способствует ли ПИ правильному поведению пользователей.

Эвристическая оценка заключается в том, что устанавливаются критерии (эвристики) оценки ПИ. Эксперты оценивают ПИ на предмет соответствия исходным требованиям. Обычно достаточно 3-5 экспертов. Результат их оценки (по опыту) позволяет решить до 3/4 всех проблем юзабилити.

В качестве примера можно назвать следующие эвристики:

• поведение системы предсказуемо;

• поведение системы согласованно (непротиворечиво);

• информация о состоянии и функционировании системы легко воспринимается;

• обеспечена обратная связь;

• ошибки пользователя эффективно предотвращаются, а совершенные ясно отображаются и легко исправляются.

Оценка на основе анализа процесса разработки производится с использованием данных об аналогичных разработках, при этом какие-то части ПИ либо поддерживаются, либо отвергаются. Оценка производится на основе моделирования реальной работы, и перенос таких данных в новую разработку должен производиться с большой осторожностью. Для расчета (прогнозирования) характеристик действий пользователя возможности ПИ анализируются с помощью когнитивных моделей, например с помощью метода GOMS. Очень полезную информацию для оценки ПИ несет всестороннее обоснование дизайнерских решений.

Методы, основанные на участии пользователей

Указанные методы подразделяются на три группы: 1) экспериментальные; 2) методы наблюдений; 3) физиологические.

Экспериментальные методы. Переменные экспериментальных методов могут быть зависимые и независимые. К независимым переменным относят характеристики, специально изменяемые для создания различных условий, например стиля ПИ, количества пунктов меню и т.п. К зависимым переменным относят характеристики, изменяющиеся в эксперименте, например время тех или иных действий, количество ошибок и т.п.

Экспериментальное исследование обычно имеет целью подтвердить либо опровергнуть какую-то гипотезу. Пример гипотезы: уровень ошибок будет возрастать с увеличением размера фонта. Нулевая же гипотеза прогнозирует отсутствие зависимости от определенных переменных. Пример нулевой гипотезы – размер фонта не влияет на характеристики деятельности пользователя.

Трудно избежать появления неконтролируемых переменных, особенно при экспериментальных исследованиях групп, поэтому такие исследования более сложны, чем эксперименты с одним пользователем. Источником неконтролируемых переменных при этом могут быть состав групп, выбор задач, сбор данных, методы анализа.

Что касается состава групп, то при большом числе участников возрастает время взаимной притирки, возникают трудности с планированием работы, увеличивается стоимость. В результате нередко останавливаются только на 3-4 группах. Выбор задач – это самая сложная часть работы, и он должен отвечать общенаучным требованиям к экспериментальным исследованиям. При исследовании групп людей следует обычно предусматривать возможность нескольких каналов регистрации результатов. При этом практически всегда в исследованиях такого рода целесообразно параллельно вести видеонаблюдение несколькими видеокамерами с обязательной временной разверткой. Следует иметь в виду, что возможны большие разбросы между группами, поэтому необходим качественный анализ и, возможно, микроанализ (например, интервалы в речи) для корректной обработки данных.

Исследования дизайна ПИ нередко сопряжены с групповыми экспериментами. При этом следует учитывать, что внутри группы каждый пользователь выполняет задание во всех исследуемых условиях, вполне вероятен перенос навыков и различия между пользователями, как правило, не приводят к серьезному удорожанию эксперимента и к снижению надежности результатов. Но между группами пользователи выполняют задание только в каком-то одном

варианте исследуемых условий, перенос навыков невозможен, а различия между пользователями, как правило, приводят к снижению надежности результатов.

В целом групповые эксперименты с участием пользователей часто имеют малую отдачу при значительной стоимости.

Методы наблюдений. К ним относят размышления вслух, взаимную оценку сотрудничества, анализ протоколов работы.

Физиологические методы. Сюда относятся методы, регистрирующие определенные физиологические показатели. Такими показателями, как правило, являются: а) время и траектории перемещений взора по ПИ, что отражает когнитивную нагрузку, вызываемую рассматриваемым ПИ; б) уровень потоотделения, что отражает степень эмоционального напряжения; в) артериальное давление, частота сердечных сокращений (особенно ее динамика во времени), наполнение пульса, что отражает сердечно-сосудистую активность; г) электромиограмма (ЭМГ), отражающая мышечное напряжение; д) электроэнцефалограмма (ЭЭГ), отражающая электрическую активность мозга, и др.

Время и траектории перемещений взора по ПИ измеряются с помощью специального оборудования, которое крепится на голове пользователя и регистрирует направление взора, время зрительных фиксаций, их число и локализацию в ПИ. Измерения включают число и длительность фиксаций (отражают степень трудности работы с ПИ), саккады (быстрые перемещения взора от одной точки фиксации к другой), траектории перемещения взора (оптимальным является перемещение взора прямо к нужному месту с короткой фиксацией на нем).

Уровень потоотделения регистрируют с помощью кожно-гальванической реакции (КГР). Интерпретация ЭМГ и ЭЭГ обычно вызывает некоторые трудности и требует опыта, а иногда дополнительных исследований.

Классификации аналитических и экспериментальных методов оценки ПИ представлены соответственно в табл. 4.1 и 4.2, а классификация методов наблюдений – в табл. 4.3.

Таблица 4.1[7]

Классификация аналитических методов оценки

[7] Низкий уровень требующейся информации предполагает большую детальность, а высокий – большую общность. Например, информация «данный фонт лучше читается, чем такой-то» является низкоуровневой, а «данная система более удобна, чем такая-то» – высокоуровневой.

Показатель	Метод оценки на основе			
	когнитивного анализа	эвристического анализа	анализа процесса разработки	моделирования
Стадия разработки	Все стадии		Высокоуровневая разработка	
Оценка лабораторная или полевая (в процессе работы)?	Лабораторная			
Оценка объективна?	Нет		Как источник	Нет
Меры	Качественные		Как источник	Качественные
Уровень требующейся информации[1]	Низкий	Высокий	Как источник	Низкий
Высокая скорость получения результата?	Нет		Как источник	Нет
Влияет ли на результат сама процедура оценки?	Нет			
Требуемое время	Среднее	Мало	От мало до среднего	Среднее
Количество требующегося оборудования	Малое			

Таблица 4.2

Классификация экспериментальных методов оценки и методов на основе анкетирования

Показатель	Метод оценки		
	экспериментальный	интервью	анкетный опрос
1	2	3	4
Стадия разработки	Все стадии		
Оценка лабораторная или полевая (в процессе работы)?	Лабораторная	Лабораторная/Полевая	
Оценка объективна?	Да	Нет	
Меры	Количественные	Качественные / Количественные	
Уровень требующейся информации	Высокий / Низкий	Высокий	
Высокая скорость получения результата?	Да	Нет	
Влияет ли на результат сама процедура оценки?	Да	Нет	
Требуемое время	Много	Мало	
Количество требующегося оборудования	Среднее	Малое	

Таблица 4.3[89]
Классификация методов наблюдений

Показатель	Метод наблюдения		
	Думать «вслух»	Анализ протокола[2]	Апостериорный сквозной анализ
Стадия разработки	Низкоуровневое программирование и отладка		
Оценка лабораторная или полевая (в процессе работы)?	Лабораторная/полевая		
Оценка объективна?	Нет		
Меры	Качественные		
Уровень требующейся информации	Высокий/Низкий		
Высокая скорость получения результата?	Да		
Влияет ли на результат сама процедура оценки?	Да	Да[3]	Нет
Требуемое время	Много		Среднее
Количество требующегося оборудования	Среднее	Много	Малое

Оценка ПИ проводится по следующим критериям:

• скорости работы пользователей;

• количеству ошибок пользователей;

• скорости обучения;

• субъективному удовлетворению (подразумевается, что соответствие интерфейса задачам пользователя является неотъемлемым свойством интерфейса).

Рассмотрению этих критериев и посвящены последующие разделы этой темы.

4.2. Скорость и производительность работы

Скорость работы пользователей является важным показателем качества интерфейса. Учет только этого критерия редко бывает очень важен, но почти всегда он является желательной частью целого. Любая попытка как-то увеличить скорость работы всегда находит одобрение. Длительность выполнения работы пользователем состоит из времени: восприятия исходной информации, интеллектуальной работы (пользователь думает, что он должен сделать), физических действий и реакции системы. Как правило, длительность реакции системы является наименее значимым фактором. Основное время пользователя занимает процесс размышления; соответственно, повышение скорости этих размышлений приводит к существенному увеличению скорости работы. Хороший интерфейс уменьшает влияние факторов, усложняющих (и соответственно, замедляющих) процесс мышления пользователя.

Для определения скорости работы используется чуть ли не единственный в «интерфейсной науке» неэвристический метод GOMS (см. разд. 3.4). Длительность физических действий пользователя зависит прежде всего от количества таких действий в процессе работы и

---

8 Включает видео– и аудиозаписи, а также записи о работе системы.

9 Исключая, конечно, внутрисистемные регистрационные протоколы.

степени их необходимой точности. Любое физическое действие, совершаемое с помощью мускулатуры, может быть или точным, или быстрым. Вместе точность и быстрота встречаются исключительно редко, поскольку при резком движении невозможно быстро остановиться. Для сочетания быстроты и точности нужно выработать автоматизм. Чем точнее движение, тем более плавным и замедленным оно должно быть. Таким образом, чтобы физическое действие пользователя было быстрым, оно не должно быть точным (и наоборот).

Этому соответствуют два наиболее распространенных устройства управления компьютером – клавиатура и мышь. Клавиатура требует сравнительно точных движений. Мышь, напротив, инерционна – есть разница между медленным и быстрым ее перемещением, сильным и слабым приложенным усилием. Поэтому оптимизация использования мыши может существенно повысить общую скорость работы. Мышь не является точным инструментом. Соответственно, мышь не предназначена для очень точных, в 1, 2 или даже 5 пикселей, манипуляций; в графических программах всегда есть возможность перемещать объекты клавишами со стрелками. По этим причинам любой слишком маленький интерфейсный элемент будет вызывать проблемы у пользователей. Еще в 1954 г. Поль Фиттс (Paul Fitts) сформулировал правило, ставшее известным как закон Фиттса: время достижения цели прямо пропорционально дистанции до цели и обратно пропорционально размеру цели. Проще говоря, лучший способ повысить доступность виртуальной кнопки заключается в том, чтобы делать ее большой и располагать ближе к курсору.

Второе требование выполнимо в контекстных меню, которые всегда открываются под курсором, соответственно, расстояние до любого его элемента всегда минимально (еще лучше было бы делать контекстные меню не вертикальными, а круглыми, когда элементы расположены вокруг курсора; к сожалению, на них практически невозможно писать текст элементов, так что распространения они не получили). Именно поэтому контекстное меню является чуть ли не самым быстрым и эффективным элементом. Но не надо думать, что уменьшать расстояния до цели можно только с контекстными меню.

Есть еще диалоговые окна. Они тоже всегда контекстно зависимы, не бывает окон, открывающихся самопроизвольно. По умолчанию они открываются в центре экрана, но это легко можно изменить. Открывать их под курсором гораздо лучше именно потому, что дистанция до их кнопок сокращается, что хорошо не только тем, что перемещать курсор нужно меньше, но также тем, что пользователю сразу становится понятна связь между его действиями и появлением диалогового окна. Есть еще и альтернативное решение. Во многих драйверах для мыши есть возможность автоматически перемещать курсор ко всем появившимся на экране кнопкам, выбранным по умолчанию. Это увеличивает скорость работы, но увеличивает вероятность человеческих ошибок, поскольку пользователь может, не разобравшись, нажать не на ту кнопку. Из сказанного вытекает рекомендация: *открывайте новые диалоговые окна не в центре экрана, а в центре текущего действия пользователя (конечно, если они не будут перекрывать важную информацию на экране).*

Что касается клавиатуры, то, как сказано выше, она требует сравнительно более высокой точности по сравнению с мышью. Но перемещение руки с клавиатуры на мышь и потом обратно занимает почти секунду. Кроме того, работа с клавиатурой подразумевает использование «горячих» клавиш (именно потому, что перемещение по экрану с помощью клавиатуры затруднено). Но хотя «горячие» клавиши существенно увеличивают скорость работы, плохо то, что их трудно запомнить. Таким образом, они являются прерогативой опытных пользователей и для многих людей неприемлемы. Более того, их популярность во многом основывается на субъективных критериях: воспринимаемая пользователем скорость работы с клавиатурой выше, чем скорость работы с мышью (хотя секундомер показывает обратное).

Следует помнить, что субъективно воспринимаемая продолжительность действий напрямую зависит от уровня активности пользователя, так что субъективная длительность действий всегда ниже такой же по времени паузы. Все знают, что при бездействии (скуке) время течет невыносимо медленно. Важно понимать, что действие может быть не обязательно физическим: лежа на диване и просматривая фильм, т.е. не совершая почти никаких физических действий, время можно потратить очень быстро. Таким образом, субъективную

скорость работы можно повысить двумя способами:

• заполнением пауз между событиями. Есть данные о том, что, если в периоды ожидания реакции системы пользователям показывают индикатор степени выполнения, субъективная продолжительность паузы существенно снижается. Судя по всему, чем больше информации предъявляют пользователям в паузах, тем меньше субъективное время. В то же время эта информация может вызвать стресс из-за перегрузки оперативной памяти, так что пользоваться этим методом надо осторожно;

• разделением крупных действий пользователей на более мелкие, при этом количество работы увеличивается, зато субъективная длительность снижается. Плох этот метод тем, что увеличивает усталость.

Первый способ часто иллюстрируют таким примером: служащие одной компании жаловались на то, что они слишком долго ожидают лифта. Реальных возможностей увеличить скорость лифта не было, и тогда было придумано простое и дешевое решение – рядом с лифтовыми кабинами на каждом этаже повесили зеркала, что прекратило жалобы; людям нашлось чем заняться в ожидании лифта. Второй способ можно проиллюстрировать тем, что работа с клавиатурой воспринимается как более быстрая, чем работа с мышью. И наконец, самый верный способ – повысить объективную, реальную скорость пользователей, что приведет, соответственно, к увеличению субъективной скорости.

На скорость работы большое влияние оказывает потеря фокуса внимания. Пользователь довольно часто отвлекается – звонок по телефону, разговоры с коллегами, кофе, срочное задание начальника, совещание и т.д. После каждого такого отвлечения пользователь должен либо вспоминать текущую задачу, либо заново ее ставить перед собой (занимает это несколько секунд, что много). У человека есть только один фокус внимания, так что при любом отвлечении (которое есть не что иное, как переключение на другую задачу) старый фокус внимания теряется. Новые же стимулы заменяют содержимое кратковременной памяти, так что для возвращения к работе требуется заново активизировать в памяти нужную информацию. Необходимо максимально облегчать возвращение к работе и проектировать интерфейс так, чтобы пользователи возможно меньше о нем думали. Таким образом, для продолжения работы пользователь должен знать:

• на каком шаге он остановился;
• какие команды и параметры он уже дал системе;
• что именно он должен сделать на текущем шаге;
• куда было обращено его внимание на момент отвлечения.

Рис. 4.1. Три варианта индикации степени заполнения экранной формы (2-й и 3-й варианты значительно лучше)

Предоставлять пользователю всю эту информацию лучше всего визуально. Разберем это на примере. Чтобы показать пользователю, на каком шаге он остановился, традиционно используют конструкцию типа «Страница N из M». К сожалению, эта конструкция работает не слишком эффективно, поскольку не визуальна. Однако существуют и визуальные способы. Например, когда читатель держит в руках книгу, он может понять, в какой ее части он находится, по толщине левой и правой частей разворота. Можно воспользоваться этой

метафорой и на экране: варьировать толщину левых и правых полей окна (рис. 4.1).

Разумеется, эти методы подходят только для экранных форм. В иных случаях нужно делать так, чтобы все стадии процесса выглядели по-разному, благодаря чему хотя бы опытные пользователи, знающие облик всех состояний, могли бы сразу определять текущий шаг. Показ пользователю ранее отданных им команд чрезвычайно проблематичен. Размеры экрана ограничены, так что почти всегда просто не хватает места для того, чтобы показать все необходимое. Зачастую единственным выходом из этой ситуации является максимальное облегчение перехода к предыдущим экранам, да и то это получается только при работе с экранными формами. Напротив, показывать пользователю, что именно он должен сделать на текущем шаге процедуры, обычно удается легче. В то же время это очень зависит от сущности задачи, так что тут трудно порекомендовать что-либо конкретное.

И наконец, четвертый пункт: показ пользователю, куда было обращено его внимание на момент отвлечения. Тут есть одна тонкость – обычно фокус внимания совпадает с фокусом ввода. Соответственно, нужно делать фокус ввода максимально более заметным. Легче всего добиться этого цветовым кодированием активного элемента. Есть и другой метод – если количество элементов на экране невелико, пользователь быстро находит активный элемент. Таким образом, просто снизив насыщенность экрана элементами, можно значительно облегчить пользователю возвращение к работе.

Производительность работы отражает объем затраченных ресурсов, как вычислительных, так и психофизиологических, при выполнении задачи. Иными словами, этот показатель в части психофизиологической дублирует уровень психической напряженности, обсуждавшийся выше, но содержит еще и «компьютерный» компонент в виде вычислительных ресурсов. Дизайн ПИ должен обеспечивать минимизацию усилий пользователя при выполнении работы и приводить:

• к сокращению длительности чтения, редактирования и поиска информации;

• уменьшению времени навигации и выбора команды;

• повышению общей продуктивности пользователя, заключающейся в объеме обработанных данных за определенный период времени;

• увеличению длительности устойчивой работы пользователя и др. Сокращение непроизводственных затрат и усилий пользователя -

важная составляющая качества программного обеспечения.

4.3. Человеческие ошибки

Пользователь (как и все люди) может ошибаться. Любая работа, тем более на компьютере, невозможна без совершения ошибок. Следовательно, любое программное обеспечение и интерфейс, не способные их предупредить, обнаружить и исправить, не должны создаваться. Любые изменения в интерфейсе влияют на ошибки пользователя, и почти нет такого элемента интерфейса, который бы не сказывался на ошибках. Количество ошибок пользователей является важным критерием качества интерфейса. Иногда одна или две человеческие ошибки погоды не делают, но только тогда, когда эти ошибки легко исправляются. Однако часто, казалось бы, незначительная ошибка приводит к катастрофическим последствиям.

Кроме собственно интерфейса на ошибки влияют и особенности психофизиологического состояния пользователя. Чаще всего эти особенности проявляются в ослаблении или потере внимания. Происходит это и из-за невозможности сохранять напряжение (а внимание – это напряжение) длительное время и отвлечений пользователя на другие дела. Но как только «бдительность» снижается, количество ошибок возрастает в разы. Таким образом, проблема состоит в том, что для успешной работы с любой системой необходима определенная степень бдительности, что пользователям неприятно. В действительности надо стремиться минимизировать количество ошибок, поскольку только это позволяет сберечь время (т.е. повысить производительность) и сделать пользователей более удовлетворенными за счет отсутствия дискомфорта. При борьбе с ошибками нужно направлять усилия на:

• плавное обучение пользователей в процессе работы;

• снижение требований к бдительности;

• повышение разборчивости и заметности индикаторов.

Дополнительно к этим трем направлениям есть и четвертое: снижение чувствительности системы к ошибкам. Для этого существуют четыре способа, а именно:

• блокировка потенциально опасных действий пользователя до получения подтверждения правильности действия;

• проверка системой всех действий пользователя перед их принятием;

• самостоятельный выбор системой команд или параметров, при котором от пользователя требуется только проверка;

• толерантность (терпимость) системы к ошибкам пользователя и определенный стиль сообщений об ошибке.

При этом самым эффективным является третий способ. К сожалению, он наиболее труден в реализации. Разберем эти четыре способа подробнее.

Блокировка потенциально опасных действий пользователя. Указанная блокировка до получения подтверждения правильности действия полезна в основном для начинающих пользователей, которые проверяют каждый свой шаг. Например, команда удаления файла в любой операционной системе сопровождается требованием подтвердить удаление. Но опытные пользователи автоматически жмут клавишу ОК, и если ошибаются, то потому, что выбрали нечаянно не тот файл. Они не читают все диалоговые окна, так как привыкли к ним и воспринимают их как фон (белый шум). Стало быть, для опытных пользователей это диалоговое окно с требованием подтверждения не работает. Во-первых, оно не защищает нужные файлы. Во-вторых, оно без пользы отвлекает пользователя и тратит его время. В то же время некоторую пользу от этого метода получить можно. Для этого только надо требовать подтверждения не после команды пользователя, а до нее.

Предположим, чтобы удалить файл, нужно сначала в контекстном меню выбрать команду, разблокировать, после чего выбрать этот же файл и запустить процесс его удаления (неважно, с клавиатуры или из меню). В этом случае от пользователя действительно требуется, чтобы он подтвердил удаление, поскольку эти два действия напрямую не связаны друг с другом – если в одном из них была допущена ошибка, файл удалить не удастся. К сожалению, этот принцип применять довольно сложно. Дело в том, что ситуации, подобные описанной, встречаются довольно редко. Гораздо чаще приходится защищать не отдельные объекты (файлы, окна и т.п.), а отдельные фрагменты данных (например, текст и числа в полях ввода). Единственным выходом служит скрытие потенциально опасных данных от пользователя до тех пор, пока он сам не скомандует системе их показать. Однако некоторым пользователям никогда не удастся понять, что помимо видимых есть еще и невидимые данные. Отсюда вытекает рекомендация: *не делайте опасные для пользователя кнопки кнопками по умолчанию.*

Проверка системой всех действий пользователя перед их принятием – метод лучше, чем блокировка, но он тоже не без недостатка: трудно проверять команды. Сравнительно универсальных и работающих способов проверки только два. Во-первых, это меню. В случаях, когда пользователь выбирает команду из списка, система может без труда делать так, чтобы в этот список попадали только корректные команды (это вообще достоинство любого меню). Во-вторых, если действие запускается непосредственным манипулированием объектами, можно индицировать возможные действия изменением поведения этих объектов. Например, если бы форматирование диска запускалось не нажатием клавиши, а перенесением пиктограммы диска в область форматирования, можно было бы показывать пользователю, как с выбранного диска исчезают все файлы и папки. При этом не только снизилась бы вероятность ошибочного форматирования диска, поскольку перенести объект в другую область труднее, чем просто нажать на клавишу, но и исчезла бы необходимость предупреждать пользователя о грядущей потере данных с помощью весьма раздражающего сообщения.

Проверка всех действий пользователя перед их принятием. Данной проверкой можно также успешно защищать вводимую пользователем информацию, в особенности числовую. Дело в том, что большинство числовых данных имеет некий диапазон возможных значений, так что даже в ситуациях, когда невозможно проверить корректность данных, можно, по крайней мере, убедиться, что они попадают в нужный диапазон. В большинстве операционных систем есть специальный элемент управления, именуемый «крутилкой» (spinner). Фактически это

обычное поле ввода, снабженное двумя кнопками для модификации его содержимого (в сторону уменьшения и увеличения). Интересен он тем, что пользователь может не использовать клавиатуру для ввода нужного значения, установив его мышью. Этот элемент имеет то существенное достоинство, что при использовании мыши значение в этом элементе всегда находится в нужном диапазоне и обладает нужным форматом. Отсюда рекомендация: всегда показывайте границы диапазона во всплывающей подсказке. Но что делать, если пользователь ввел некорректное число с клавиатуры? Лучше индицировать возможную ошибку заменой цвета фона этого элемента, скажем, на розовый. В тех же случаях, когда количество возможных значений невелико, лучше использовать другой элемент управления – ползунок (рис. 4.2).

Рис. 4.2. Ползунок

Он позволяет устанавливать только определенные значения (с этим справился бы и раскрывающийся список или комплект переключателей) и видеть взаимосвязь возможных значений, и при этом использование этого элемента понятно даже новичку.

Система сама предлагает только те команды и опции, которые возможны. Это самый эффективный способ. Чем меньше действий требуется совершить пользователю, тем меньше вероятность ошибки (при этом пользователь, которого избавили от рутинной работы, уже радуется). Вопрос в том, как системе узнать, что именно нужно пользователю. Проиллюстрировать сферу применения данного метода удобно на примере печати. ОС Windows заставляет использовать один способ что-либо напечатать. Существует команда «Печать» в меню «Файл» и «Параметры печати» в открывшемся диалоговом окне печати (рис. 4.3).

Некоторая проблема заключается в том, что обилие элементов управления замедляет восприятие этих окон и увеличивает вероятность ошибки, хотя даже при небольшом опыте этот недостаток нивелируется. Чем меньше элементов управления, тем меньше вероятность ошибки. Система может уменьшить число элементов, если она знает сама, какими именно параметрами она должна руководствоваться. Главной причиной появления этого диалогового окна является печать нескольких копий. Причем есть простая зависимость – количество выбираемых копий обратно пропорционально частоте печати такого количества копий, т.е. сто копий печатают примерно в сто раз реже, чем печатают одну копию. Стандартное диалоговое окно печати содержит также область выбора принтера из числа установленных в системе принтеров. Большинство же пользователей имеют только один принтер. Так зачем заставлять это большинство каждый раз отвлекаться на совершенно не нужные им элементы интерфейса?

Рис. 4.3. Диалоговое окно печати в MS Word

В заключение можно сказать, что *система сама может узнать большинство из тех*

сведений, которые она запрашивает у пользователя. Главными источниками этих сведений являются:

• здравый смысл разработчика системы;

• предыдущие установленные параметры;

• наиболее часто устанавливаемые параметры.

В то же время, применяя этот метод, надо всегда помнить о том, что цель состоит не в том, чтобы провести пользователя по программе, оберегая его от всего, что может его испугать, а в том, чтобы уменьшить количество ошибок, а стало быть, вероятность возникновения стрессового состояния у пользователя.

Толерантность системы к ошибкам пользователя. Человеческие ошибки разделяют также по уровню их негативного эффекта, например:

1) исправляемые во время совершения действия, например пользователь перетаскивает файл в корзину и во время перетаскивания замечает, что он пытается стереть не тот файл;

2) которые исправить можно, но с трудом, например реальное стирание файла, при котором никаких его копий не остается;

3) исправляемые после выполнения действия, например после ошибочного уничтожения файла его копия переносится из корзины;

4) которые на практике невозможно исправить, т.е. те, которые невозможно обнаружить формальной проверкой (невозможно обнаружить их не случайно). Пример: смысловая ошибка в тексте, удовлетворяющая правилам языка.

Ошибок из четвертого пункта нужно всеми силами избегать, не считаясь с потерями, поскольку каждая такая ошибка обходится гораздо дороже, чем любая ошибка из пункта третьего. То же касается ошибок из второго пункта, поскольку они гораздо дороже, чем любые ошибки из первого пункта. С ошибками из четвертого пункта все ясно. Всякий раз, когда мы теряем возможность проверить валидность данных или самой системы, мы вступаем на скользкий путь. Примеров неисправляемых ошибок из-за ошибок в программном обеспечении множество – межпланетные зонды, улетающие не туда, коммерческие договоры, в которых обнаруживаются ошибки, ошибки обработки различных данных и т.д. Разумеется, такие ошибки, как правило, обнаруживаются, но проблема в том, что к моменту обнаружения их уже не исправить. Именно поэтому эти ошибки гораздо хуже тех, которые исправить трудно, но которые сразу видны. Ошибки первого типа (исправляемые «во время») гораздо лучше ошибок второго типа (исправляемых «после»). Объективное объяснение простое: ошибки, исправляемые после, снижают производительность работы. Всякий раз, когда пользователь обнаруживает, что он совершает ошибку, ему приходится возвращаться назад на несколько этапов. Более того, чтобы исправить совершенную ошибку, от пользователя требуется понять, что ошибка совершена, как ее исправить и как потратить время на исправление ошибки. В результате много времени уходит не на действие (т.е. на продуктивную работу), а на исправление ошибок.

Субъективное объяснение еще проще: ошибки, исправляемые «после», воспринимаются пользователем как ошибки. Ошибки же, исправляемые «во время», как ошибки не воспринимаются просто потому, что для пользователей это не ошибки вообще: все человеческие действия до конца не алгоритмизированы, они формируются внешней средой (так не получилось и так не получилось, а вот так получилось). Ошибка же, не воспринимаемая как таковая, пользователей не раздражает, что весьма положительно действует на их субъективное удовлетворение от системы. Отсюда правило: *наличие человеческих ошибок, которые нельзя обнаружить и исправить до окончательного совершения действия, всегда свидетельствует о недостаточно хорошем дизайне.*

Теперь пора рассказать, как избавиться от ошибок, исправляемых после. Понятно, что исправить что-либо «во время» можно только тогда, когда во время совершения действия видно, что происходит и как это действие повлияет на изменяемый объект. Следовательно, чтобы дать пользователям возможность исправлять на ходу, необходима обратная связь. К сожалению, это простое соображение имеет существенный недостаток: вводить в систему обратную связь получается не всегда. Дело в том, что ее не любят программисты. Мотивируется это тем, что обратная связь плохо влияет на производительность системы. На

самом деле ее реализация просто сопряжена с дополнительной работой. Иногда, впрочем, соображения о производительности системы действительно правомочны. Так что даже если программисты правы, когда утверждают, что система «будет безбожно тормозить», полезно вспомнить, что производительность связки «человек-ПИ» всегда важнее производительности системы как таковой (правильнее сказать, что влияние этой связки на производительность системы значительно серьезнее, чем кажется многим программистам). Можно спроектировать обратную связь весьма скромно. Иногда так получается даже лучше. Например, с помощью ползунков на линейке в MS Word можно менять абзацные отступы, при этом обратная связь есть, но не полная: вместо перманентного переформатирования документа по экрану двигается полоска, показывающая, куда передвинется текст. Это лучше, чем «дрыганье» текста вслед за передвигаемой полоской, которое будет только раздражать.

Стиль сообщений об ошибках

Очень важен стиль сообщений об ошибках. Сам факт ошибки всегда является неприятным для любого человека, тем более важно, чтобы сообщение было, с одной стороны, информативным, кратким и достаточным для корректировочных действий, с другой стороны, предельно тактичным. Однако большинство сообщений об ошибках в действительности не является собственно сообщениями об ошибках. На самом деле они показывают пользователю, что система, которой он пользуется:

- недостаточно гибка, чтобы приспособиться к его действиям;
- недостаточно умна, чтобы показать ему возможные границы его действия;
- самоуверенна и считает, что пользователь дурак, которым можно и нужно помыкать.

Разберем это подробнее. Многие программы искренне «уверены», что пользователь царь и бог, и когда оказывается, что пользователь хочет невозможного (с их точки зрения), они начинают «чувствовать разочарование». Проявляют же они свое чувство сообщениями об ошибках. В действительности множество действий пользователя направлено не на то, чтобы сделать так, а не иначе, а на то, чтобы сделать примерно так, как хочется. Пользователю часто нет дела, можно ли добиться точного результата или нет. Показывать ему в таких случаях диалог об ошибке глупо, поскольку пользователю не нужно ничего знать. Ему нужен результат. Вот что бывает (рис. 4.4), если пользователь попытается ввести значение, которое ему нужно, но которое система не умеет обрабатывать. Тут возможны три решения. Во-первых, при проектировании системы можно более тщательно подходить к выбору ее функциональности. Во-вторых, можно просто игнорировать неправильность значения, округляя его до ближайшего возможного (индицируя, возможно, самостоятельность действий системы однократным миганием соответствующего поля ввода). В-третьих, вместо обычного поля ввода можно использовать «крутилку».

Рис. 4.4. Примеры диалога об ошибке

Во многих случаях пользователь совершает действия, которые воспринимаются программой как неправильные, не потому, что он глуп, но потому, что система не показала ему границ возможного действия. Все такие сообщения порочны, поскольку их можно было бы избежать, просто блокировав возможность совершения некорректных действий или показав пользователю их некорректность до совершения действия (рис. 4.5).

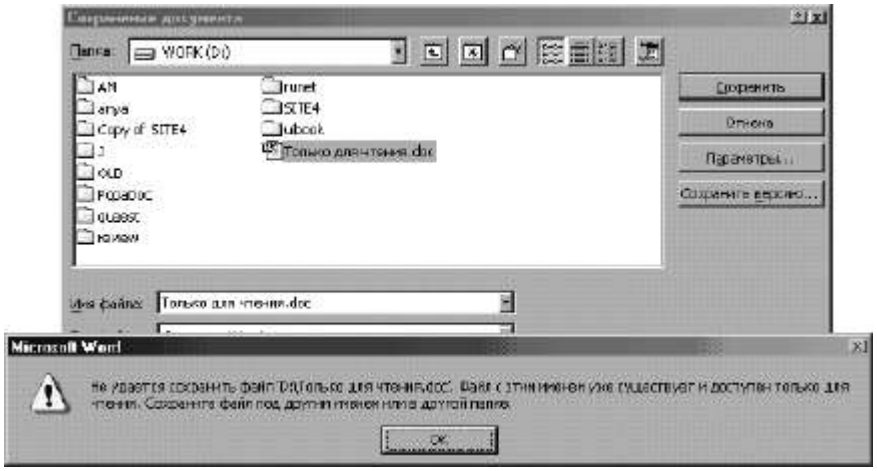

Рис. 4.5. Типичное сообщение об ошибке, вызванное нежеланием системы показать пользователю границы его действий

Нормой также являются случаи, когда система отображает ситуацию так, как будто пользователь идиот, а система, наоборот, есть воплощение безошибочности и правоты (см. рис. 4.4).

Суммируя, можно сказать, что почти любое сообщение об ошибке есть признак того, что система спроектирована плохо. Всегда можно сделать так, чтобы показывать сообщение было не нужно. Вообще говоря, любое сообщение об ошибке говорит пользователю, что он глуп. Поэтому пользователи не любят сообщения об ошибках (рис. 4.6).

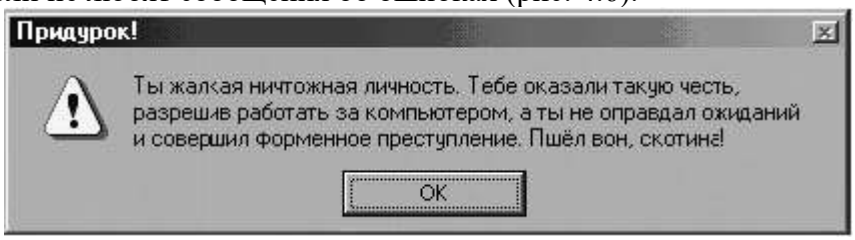

Рис. 4.6. Примерно так пользователи воспринимают любые сообщения об ошибках

Системы изначально надо проектировать так, чтобы в них отсутствовала необходимость в таких сообщениях. Невозможно полноценно работать с системой, которая по нескольку раз за день тебя ругает. Идеальное сообщение об ошибке должно отвечать всего на три вопроса:

1. В чем заключается проблема?
2. Как исправить эту проблему сейчас?
3. Как сделать так, чтобы проблема не повторилась?

При этом отвечать на эти вопросы нужно возможно более вежливым и понятным пользователям языком. В качестве примера идеального сообщения об ошибке удобно взять какое-либо существующее сообщение (из тех, которые точно нельзя просто изъять из системы) и попытаться это сообщение улучшить. Например, попытаемся улучшить уже упоминавшееся сообщение о невозможности перезаписать заблокированный файл. Итак, сообщение об ошибке гласило: «Не удается сохранить файл „B:\Только для чтения.скю“. Файл с этим именем уже существует и доступен только для чтения. Сохраните файл под другим именем или в другой папке». Это довольно неплохое сообщение, во всяком случае, оно гораздо лучше, чем «Указано неверное число». Но и его можно улучшить.

Сначала надо разобраться, в каких случаях оно появляется. Это несложно: оно может появляться, если пользователь пытается сохранить файл на компакт-диске обычным способом (т.е. таким же, как и на дискете) или же пытается сохранить файл, незадолго перед этим скопировав этот файл с компакт-диска. Случаи, когда файл заблокирован сознательно, в жизни редки, так что их можно не учитывать. Главный враг – компакт-диск. Тут существует несколько решений, не противоречащих друг другу.

Во-первых, можно просто блокировать возможность что-либо записать на диске, запись на который невозможна. Собственно говоря, запись и так блокируется, но сообщением об

ошибке. А можно просто не показывать диски, на которые нельзя записывать, в окне записи, что эффективнее, поскольку делает ошибку невозможной.

Во-вторых, как уже говорилось, можно показывать файлы, защищенные от записи, иначе, чем файлы незащищенные. Это будет работать, но тоже неидеально.

Что делать пользователю, который все-таки хочет перезаписать файл? Сейчас в такой ситуации приходится записывать файл под новым именем, потом стирать старый, а новому файлу давать имя старого. Это и потери времени, и ошибочно стертые файлы (лучший способ сделать так, чтобы пользователи не стирали нужные файлы, заключается в том, чтобы лишить пользователей необходимости вообще что-либо стирать в нормальном режиме работы).

Сообщение об ошибке помимо указанной информации должно позволять разблокировать те файлы, которые возможно (т.е. записанные не на компакт-диске). Для этого вносится несколько изменений:

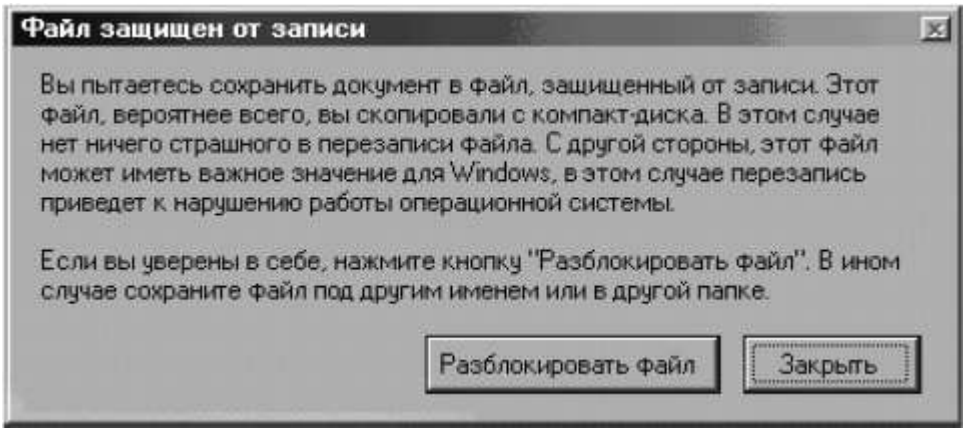

Рис. 4.7. Улучшенное сообщение об ошибке

• диски, на которые ничего нельзя записать, не показываются в диалоговом окне сохранения файлов;

• заблокированные файлы на остальных дисках показываются иначе, чем незаблокированные;

• при попытке записать документ поверх заблокированного появляется сообщение об ошибке (рис. 4.7).

> **Внимание!** Кнопка «Закрыть» выбрана по умолчанию, чтобы снизить вероятность перезаписи важных файлов. Конечно, лучше всего было бы, чтобы ОС сама снимала с копируемых с компакт-диска файлов метку «Read Only». Многие проблемы при этом бы исчезли, поскольку защищенными от записи остались бы только действительно важные для ОС файлы.

Об этом примере осталось сказать немного. Во-первых, текст сообщения должен быть возможно более коротким. Во-вторых, диалоговое окно не самый лучший способ показывать сообщения об ошибках, во всяком случае в ПО. Дело в том, что в Windows имеется элемент управления, который значительно лучше использовать для показа сообщений. Называется этот элемент «пузырь» (balloon) (рис. 4.8).

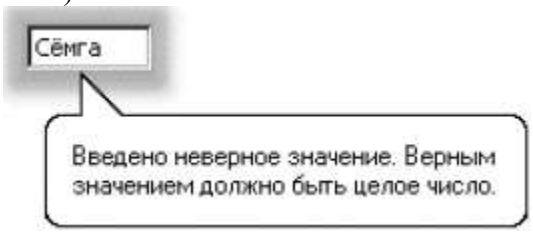

Рис. 4.8. «Пузырь» в его лучшем проявлении (не обращайте внимание на текст)

«Пузырь» по сравнению с диалоговым окном имеет существенные достоинства. Во-первых, он не так сбивает фокус внимания, нежели окно. Во-вторых, чтобы закрыть «пузырь», пользователям не обязательно нажимать на какую-либо кнопку, достаточно

щелкнуть мышью в любом месте экрана. В-третьих, он не перекрывает значимую область системы. В-четвертых (что самое главное), он показывает, в каком именно элементе управления была допущена ошибка. Все это делает «пузырь» вещью незаменимой. К сожалению, «пузыри» имеют и недостатки: первое – в них не может быть никаких элементов управления, так что использовать «пузырь» в предыдущем примере, например, невозможно; второе – «пузырей» практически нет в Интернете.

Для лучшего понимания разных способов сообщения об ошибках полезно просматривать примеры хороших и плохих интерфейсов. К счастью, существуют их коллекции, именуемые «Зал почета интерфейсов» и «Зал позора интерфейсов», первый по адресу http://www.akzhan.midi. ru/iarchitect/mfame.htm, второй – http://www.akzhan.midi.ru/iarchitect/ mshame.htm.

4.4. Обучение работе

Растущее число пользователей делает проблему их обучения работе с компьютерной системой чрезвычайно важной. Вместе с тем цель хорошего ПИ – возможность понять логику его работы неподготовленному пользователю. Минимизация затрат на обучение работе с системой и есть одно из основных направлений человеко-компьютерного взаимодействия.

Учатся же пользователи только при наличии соответствующей *мотивации.* Но мотивация пользователя должна быть адекватна усилиям, затрачиваемым на обучение. Кроме того, пользователь будет учиться какой-либо функции, только если он знает о ее существовании, поскольку, не обладая этим знанием, он не способен узнать, что за ее изучение ему воздастся. Иначе говоря, одной мотивации недостаточно, если пользователь не знает, какое вознаграждение за свои усилия он получит. Так, например, есть ничтожно мало людей, которые бы действительно полностью знали MS Word, подавляющее большинство умеют пользоваться не более чем 5% его возможностей, при этом даже не подозревая об остальных. Плохо это по многим причинам. Во-первых, пользователи работают с системой не слишком эффективно, поскольку вместо методов адекватных они используют методы знакомые. Во-вторых, достаточно часто случается, что пользователи, не зная, что имеющийся продукт делает то, что им нужно, ищут (и находят) продукт конкурента. В-третьих, при таком положении затруднительно продавать новые версии продукта: если пользователь не умеет работать с тем, что есть, убедить его совершить покупку ради новой функции вряд ли удастся. Таким образом, чтобы пользователь начал учиться, ему нужно рассказать о функциональности системы, делая акцент на возможностях, которые пользователь получит, этой функции обучившись. А дальше он сам будет учиться, если, конечно, будет знать чему и за какие преимущества.

Есть два основных направления повышения скорости и качества обучения за счет свойств ПИ – это общая «понятность» системы и наличие обучающих материалов. Рассмотрим их по порядку.

4.4.1. Общая «понятность» системы

Термин «понятность» не очень строг, его следует понимать интуитивно как такой интерфейс, который адекватен представлению пользователя о работе с системой. Основной смысл термина «понятность» – соответствие системы и ее ПИ модели этой системы в голове пользователя, т.е. соответствие объективного и субъективного.

Важность адекватной субъективной модели логики работы. Как правило, чтобы успешно пользоваться какой-либо системой, человеку необходимо понимать, как система работает. При этом не обязательно точно понимать сущность происходящих в системе процессов, важнее понимать то, что можно назвать логикой ее работы. Скажем, утюгом никогда не сможет воспользоваться человек, который не знает, что провод от утюга надо включить в розетку. При этом не обязательно знать, сколько энергии утюг потребляет (отсутствие точности), равно как и то, при сохранении искренней уверенности, что электричество по проводам течет как вода (отсутствие правильности). Неприятности наступают, когда представления человека о логике работы системы не совпадают с реальным ее устройством.

Так, например, человек, никогда не встречавшийся с электричеством, поставит утюг на огонь, отчего прибор непременно сгорит.

Аналогичная ситуация возникает и с ПИ. Трудно бывает объяснить новичку пользу от записи файла после его редактирования. Дело в том, что без понимания сущности происходящих в компьютере процессов понять необходимость записи невозможно. Допустим, пользователь создал новый документ, что-то в нем сделал, после чего попытался выйти из программы. Программа спрашивает его: «Сохранить документ или нет?» Во-первых, в этом вопросе главное значимое слово непонятно. Что такое сохранить? Где сохранить? Куда сохранить? Если же вопрос ставится техническим языком, а именно «Записать документ на диск?», то и здесь возникает недоумение: на какой диск? Во-вторых (что важнее), даже если пользователь понял вопрос, он все равно не может понять, зачем документ сохранять. Ведь документ уже имеется, сохранять его не надо. Ситуация усугубляется тем, что даже начинающий пользователь знает, что, если он нажмет кнопку ОК, начнется какое-то действие. А поскольку пользователь не хочет, чтобы действие, которого он не понимает, начиналось, он недрогнувшей рукой нажимает кнопку «Нет», после чего программа закрывается, а файл не сохраняется. Проблема в том, что пользователь искренне уверен, что, если файл есть, с ним уже ничего не сделается.

Есть два способа научить пользователя сохранять файлы. Сущность первого состоит в следующем: научить пользователя время от времени и перед выходом из программы сохранять файл, иначе у него будут проблемы. Рано или поздно (скорее поздно) пользователь начнет сохранять файл автоматически, по-прежнему не понимая, почему он это делает и только подчиняясь полученному указанию (т.е. совершая действие, по сути, ритуальное). Заметим, что такого рода действия весьма часты среди многих пользователей. Второй способ: пользователю можно объяснить, что в компьютере есть две подсистемы памяти, одна постоянная, а другая временная; при выключении или при зависании компьютера содержимое временной памяти теряется, так что если документ будет нужен и позже, его надо перенести в постоянную память; перенос туда документа называется сохранением.

Проблема в том, что изначально пользователь не имел адекватной субъективной модели логики работы компьютера. Компьютер же не дает ему возможности самому построить модель, так что единственным ее источником может являться обучение. Это один из самых больших недостатков дизайна современных компьютеров. Показательно, что первый компьютер без разделения памяти на постоянную и временную (Palm Pilot) разошелся тиражом в миллионы экземпляров.

Таким образом, без адекватной субъективной модели пользователи фактически не способны научиться пользоваться системой. К сожалению, проектирование системы, для которой модель построить легко, есть дело сложное, так что придумать универсальный алгоритм невозможно. Существует, однако, одно простое правило: поскольку элементы, выполняющие несколько разных функций в зависимости от контекста, существенно усложняют построение субъективной модели, их лучше не создавать. Поэтому слишком много элементов управления обычно делать лучше, нежели слишком мало, во всяком случае, опасно ставить цель «во что бы то ни стало сократить количество элементов.

Метафора как основное средство обеспечения общей «понятности» системы. В большинстве случаев не следует заставлять пользователя создавать свою субъективную модель системы, а хорошо бы воспользоваться субъективными, уже готовыми моделями, которые были построены ранее по другому поводу, но могут использоваться для работы с ПИ. Таковым средством является метафора.

Самым простым примером метафоры в интерфейсе является устройство программ для проигрывания звуков на компьютере. Ведь вся аудио-техника имеет почти одинаковый набор кнопок: со стрелками (назад-вперед), с треугольником (воспроизведение), с двумя дощечками (пауза), с квадратиком (полная остановка) и красный кружок (запись). При этом обычно жизнь складывается так, что сначала человек научается пользоваться этими кнопками на материальных устройствах, а уж потом начинает пользоваться компьютером. Соответственно, при проектировании программы аналогичного назначения разумно скопировать существующую систему маркировки кнопок. Благодаря этому пользователям для работы с этой

программой ничему не приходится учиться (и даже переучиваться). Сказанное выше можно проиллюстрировать примером, представленным на рис. 4.9.

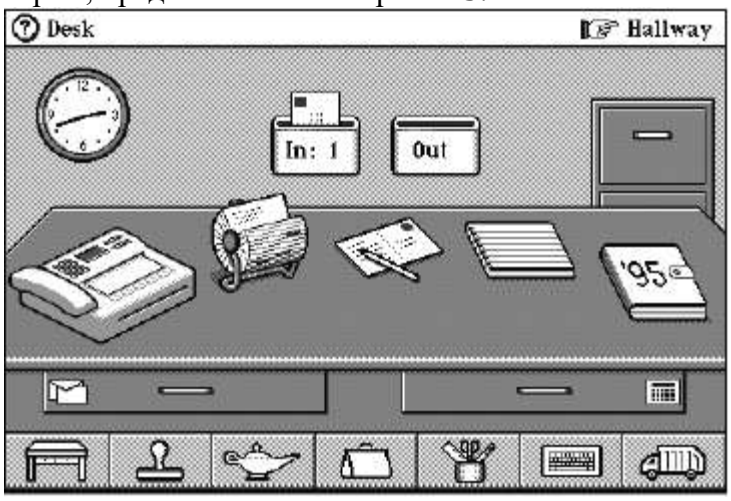

Рис. 4.9. Главный экран ОС «Magic Cap»

Эта система, будучи вся построена на метафорах, приобрела большую популярность среди журналистов компьютерных изданий (они сразу все понимали). В то же время разработчики не смогли добиться такой же любви у других пользователей: они ее понимали, но отказывались с ней работать из-за ее неэффективности.

Другой пример использования метафоры. Сотрудники корпорации «Sony» разработали оригинальную концепцию интерфейса для переноса файлов с одного устройства на другое, получившую название «pick-and-drop» («взять и бросить» (рис. 4.10), по аналогии с «drag-and-drop»). Несмотря на то что сегодня существует множество способов отправки информации, обмен файлами между двумя цифровыми устройствами, находящимися рядом, зачастую затруднен. Применять электронную почту слишком громоздко, а справиться с настройками беспроводного соединения может далеко не каждый пользователь. Созданная система, в свою очередь, обеспечивает интуитивно понятный способ управления и предельно проста в применении.

Работает технология «pick-and-drop» следующим образом. Вначале необходимо выбрать файлы для передачи, ткнув в их ярлыки на экране специальным пером. Это перо имеет уникальную метку и распознается системой при приближении к дисплею. Далее нужно поднести указательное устройство к экрану компьютера-получателя, на котором при этом появится затененное изображение ярлыка. Последующее прикосновение к дисплею завершает операцию отправки файла, автоматически передающегося по локальной сети. При этом создается впечатление того, что файлы переносятся именно пером, в то время как в действительности перо выполняет функцию интуитивно понятного интерфейса. Более того, в настоящее время Sony адаптировала систему для работы с проектором. Документ, «поднятый» с экрана монитора, можно быстро отобразить на большом дисплее, просто щелкнув пером по проектору. Данная технология получила название «pick-and-beam» («взять и отобразить»).

Рис. 4.10. Технология «pick-and-drop» в действии

Вообще, метафора является отправной точкой всякого хорошего интерфейса. Обстановка

на экране и способы взаимодействия с системой должны апеллировать к ситуации, хорошо знакомой пользователю. Так, оконный интерфейс задумывался как метафора рабочего стола с документами. Использованием метафоры достигается сразу несколько целей. Во-первых, пользователю легче понимать и интерпретировать изображение на экране. Во-вторых, ему не нужно каждый раз заглядывать в руководство, чтобы узнать, как выполняется то или иное действие. По крайней мере, некоторые действия должны естественно следовать из метафоры. В-третьих, у пользователя возникает чувство психологического комфорта, характерного для встречи с чем-то хорошо знакомым.

Ярким примером хорошего концептуального дизайна интерфейса (помимо некоторых компьютерных игр) может служить система дорожных знаков. Ее разработка не так проста, как может показаться на первый взгляд. Стоит обратить внимание на сочетание реалистических пиктограмм с абстрактными, на комбинирование многих знаков, висящих вместе, на словарь фонов. Кроме того, удалось решить почти неразрешимую задачу – знаки заметны и не портят красоту окружающей природы там, где эта красота есть. И главное, эта система хорошо работает и не требует от своих пользователей высшего образования. Во многих интерфейсных разработках идея дорожных знаков должна занимать значительное место.

Выбранная метафора может продиктовать все изобразительные решения дизайна интерфейса. Однако следует остерегаться фотографической похожести среды в компьютере с выбранной метафорой (тут есть аналогия с живописью). Все-таки компьютерная среда – искусственна, и полностью повторить все элементы взаимодействия из физического мира не удастся. А фотографическая похожесть может спровоцировать пользователя на то, чтобы пользоваться этой искусственной средой в точности как той, которую она напоминает. В первый раз, когда пользователь натолкнется на различие, он испытает психологический стресс, который может привести к полному отторжению системы.

Сказанное тесно примыкает к важнейшему принципу создания ПИ – балансу между интерактивными возможностями ПИ и сложностью его изобразительного ряда. Так же как при создании игр, главным является баланс между сложностью игры и ее увлекательностью, что занимает основное время, так и в интерфейсе должен обеспечиваться баланс между функциональными возможностями программы, возможностями манипуляции ею и ее изобразительным рядом. Простая программа не должна иметь сложное управление, это очевидно, но для нее исключается и слишком изощренная графика, что типично для многих ПИ, особенно для web-сайтов. Сложная картинка психологически готовит к сложной работе с программой. Из этого не следует, что у сложной программы должны быть изощренная графика и сложные пути взаимодействия. Лучше эту сложность вводить постепенно, подобно наращиванию уровней в компьютерных играх.

Расширенные средства визуализации в новой ОС Windows Vista значительно расширяют возможности метафорических решений ПИ и обеспечивают максимальную понятность информации. Эти средства предоставляют пользователю возможность видеть содержимое файлов, не открывая их (с помощью анимационных иконок и функции «Flip 3D» их увеличения в трехмерном пространстве), мгновенно находить приложения и файлы, эффективно переключаться между открытыми окнами и более уверенно ориентироваться в диалоговых окнах и мастерах. Гарантируется надежное взаимодействие ОС с пользователем: исчезли такие явления, как мерцание и перерисовка экрана, кратковременное прерывание работы, запаздывания и искажение изображения. Усовершенствованные общие элементы окон позволяют сосредоточиться на содержании, не отвлекаясь на оформление интерфейса, а визуальные элементы стали более информативными, интуитивными и полезными. Windows Vista предлагает пользователям выбор из четырех видов интерфейсов: простого, классического, стандартного и Windows Aero.

Отличительной особенностью интерфейса Windows Aero является оригинальный дизайн в виде прозрачного стекла, для которого реализованы такие эффекты, как отражения и плавная анимация. С помощью «стеклянных» окон (блестящее дизайнерское решение!) создается открытая многофункциональная среда, которая помогает сосредоточиться на содержимом, не отвлекаясь на окружающий интерфейс. Две новые функции интерфейса Windows Aero – Flip и Flip 3D – позволяют управлять окнами на рабочем столе, упорядочивая их непривычным (пока),

но удобным способом.

В использовании метафоры есть и несколько подводных камней. Все-таки процесс взаимодействия с пользователем проходит не в реальном мире, а с помощью таких искусственных приспособлений, как экран, мышь и клавиатура. Поэтому где-то приходится метафору «подправлять». Кроме того, возможности мира внутри компьютера обычно шире возможностей физического мира, что может с успехом использоваться для более мощного интерфейса. Наконец, существует сложившаяся практика пользования компьютером у профессионалов, и эта практика кажется естественной создателям новых интерфейсов. Но есть у метафор и довольно серьезные недостатки, а именно:

1. Не для любой функциональности можно подобрать подходящую метафору, причем невозможно заранее узнать, есть ли хорошая метафора или нет, так что можно потратить время и ничего не найти. Это, как минимум, неэффективно.

2. Даже подходящая метафора может оказаться бесполезной, если ее не знает какая-то часть аудитории или ее трудно однозначно передать.

3. Почти всегда метафора будет сковывать функциональные возможности. Что делать, если проектируемая система обладает большим количеством функций, чем копируемый образец? Следование метафоре в таких условиях будет только вредить, поскольку совпадающим функциям будет учиться легче, а несовпадающим – сложнее (они будут иначе устроены). Зачем тогда система, почему бы пользователю не обратиться к исходному образцу?

4. Совершенно не обязательно, что сам по себе копируемый образец работает идеально. Если его копировать, окажется, что система не сможет быть эффективнее своего прародителя. Например, Adobe PageMaker во многом копирует традиционные верстальные гранки, наследуя их известность пользователям вместе с их ограничениями. Благодаря этому он стал чрезвычайно популярен. Однако через некоторое время появился не копирующий гранки QuarkXpress и отвоевал большую часть пользователей лишь потому, что работал более эффективно, не таская на себе груз старой идеи. Пользователи предпочитали потратить больше времени на обучение, зато выиграть в скорости работы. Парадоксальность ситуации в том, что сейчас гранки не используются вовсе, появилось поколение пользователей, которое никогда их не видело. Результат: обучаясь PageMaker, они должны дополнительно обучаться работе с гранками, которая им не нужна. Анализируя опыт успешных случаев применения метафор, можно вывести следующие правила:

• опасно полностью копировать метафору, достаточно взять из нее лучшее;

• не обязательно брать метафору из реального мира, ее можно придумать самому;

• эффективнее всего метафорически объяснять значение отдельных объектов: например, для графической программы слои можно представлять как положенные друг на друга листы стекла (этот пример подходит и для предыдущего пункта);

• если метафора хоть как-то ограничивает систему, от нее необходимо немедленно отказаться;

• почти всегда метафору можно использовать в документации, не перенося ее в интерфейс, при этом с тем же успехом. Достаточно просто написать, что «система во многом напоминает…», и нужный результат будет достигнут.

В заключение можно сказать, что применять метафору надо с большой осторожностью. Полезно помнить, что очень многие системы, базирующиеся на метафоре, проиграли конкурентам. Таков уже упоминавшийся PageMaker, таковой была ОС Magic Cap, оболочка MS Bob, в разработку которой было вложено множество денег, но которая не имела успеха, и т.д. Еще о метафорах будет сказано в разд. 6.5.2 при обсуждении возможностей мультимедиа в ПИ.

Свойства объекта, показывающие способ взаимодействия с ним. Другой составляющей понятности является использование (или придание) таких свойств объекта, которые сами показывают способ своего использования (это свойство в англоязычной литературе называется трудно переводимым словом *affordance).* Например, надпись «На себя» на двери не является таким свойством, а облик двери, который подсказывает человеку, что она открывается на себя, является. Польза заключается в том, что это позволяет пользователям обходиться без какого-либо предварительного обучения, обеспечивая самый эффективный и надежный путь обеспечения понятности (рис. 4.11).

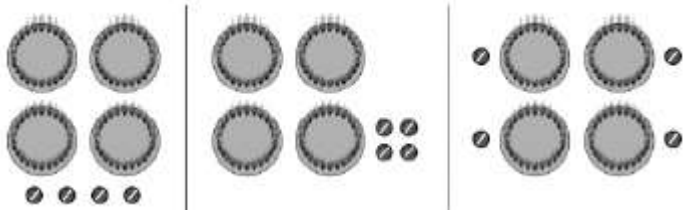

Рис. 4.11. Дизайн кухонной плиты

Слева на рисунке представлен стандартный вариант, недостаток которого заключается в том, что невозможно умозрительно определить, какой диск управляет какой конфоркой. В центре и справа варианты, не имеющие этой проблемы, при этом работающие по-разному. В центральном примере расположение регуляторов повторяет расположение рабочих объектов (конфорок), благодаря чему неоднозначность исчезает. В правом примере каждому объекту соответствует отдельный регулятор. Эти два способа уничтожения неопределенности используются и в экранном дизайне.

Однако, в отличие от мира реального, единственным способом передать какие-то свойства объекта в ПИ является их визуализация. Это хорошо и естественно по отношению к визуальным характеристикам, но крайне сложно (и противоестественно) по отношению к другим (вкусовым, тактильным, обонятельным, проприоцептивным). Это ограничение приводит к тому, что доступными оказываются всего несколько способов передачи таких свойств, из которых основными являются четыре:

1) мэппинг (mapping), или повторение конфигурации объектов конфигурацией элементов управления (этот способ хорошо работает в реальном мире, но не очень хорошо – на экране, поскольку предпочтительнее прямое манипулирование);

2) видимая принадлежность управляющих элементов объекту;

3) визуальное совпадение свойств экранных объектов с такими же свойствами объектов реального мира (реальная кнопка в реальном мире предлагает пользователю нажать на нее, псевдотрехмерная кнопка в интерфейсе, являясь метафорой, предлагает нажать на нее уже по аналогии);

4) изменение свойств объекта при подведении к нему курсора (слабый аналог тактильного ощущения).

Стандарты и язык шаблонов в ПИ. Другой важнейшей составляющей понятности являются стандарты. Дело в том, что, если нельзя что-то сделать «самопроизвольно» понятным, всегда можно сделать это везде одинаково, чтобы пользователи обучались только один раз. Например, кран с горячей водой всегда маркируют красным цветом, а кран с холодной – синим. Частично это соответствует свойствам человеческого восприятия (красный цвет мы называем теплым, а синий – холодным), но в основном здесь работает привычка. Использование стандартов чрезвычайно эффективно, но, к сожалению, не слишком надежно. Дело в том, что стандарт очень хорошо работает, когда популярен, в противном случае не работает вовсе. Популярность же его может быть достигнута двумя способами: во-первых, он может иметь место во всех системах, во-вторых, он может иметь место внутри отдельной системы. Например, стандарт интерфейса MS Windows популярен почти во всех программах для Windows, именно поэтому его следует придерживаться. В то же время этот стандарт оставляет неопределенным очень многое, и это многое в разных системах трактуется по-разному. Бесполезно пытаться найти общий знаменатель во всех системах, гораздо эффективнее установить собственный стандарт, не противоречащий стандарту ОС, но дополняющий его. После этого надо строго придерживаться данного стандарта. Отсюда следует правило: *последовательность в реализации интерфейса есть первое условие качества результата.*

Конечно, разработка собственного стандарта дело довольно трудное. Но сказать, что она совсем уж невозможна, тоже нельзя. Во-первых, самое главное уже стандартизовано. Во-вторых, стандарт может развиваться параллельно процессу разработки, при этом поддержание стандарта не стоит почти ничего. Обширный же стандарт вполне окупает вложенные в него усилия ускорением обучения пользователей, не говоря уже о снижении стоимости разработки.

Развитие идеи стандартов привело к так называемому языку шаблонов. Существуют интерфейсные решения, хорошо работающие почти всегда. Эти решения можно систематизировать с помощью так называемого языка шаблонов. Идея языка шаблонов (pattern language) принадлежит человеку, далекому от разработки ПИ, – известному архитектору XX столетия Кристоферу Александру. Он предложил язык шаблонов в качестве средства для писания архитектурных решений, однако оказалось, что данная концепция может успешно использоваться в дизайне интерфейсов. Более того, на сегодняшний день последователей Александра гораздо больше среди специалистов в области человеко-компьютерного взаимодействия, чем среди архитекторов.

В применении к ПИ идею шаблонов можно описать так: это отнюдь не описание отдельных элементов управления, а описание способа, которым пользователь взаимодействует с системой или с элементом системы. Например, кнопка в интерфейсе Windows – это не шаблон, но неактивная (серая) кнопка является одним из способов реализации шаблона под названием «Отключенные Ненужные Элементы» (автор этого шаблона Дженифер Тидвелл). Причем один и тот же шаблон может быть использован и в высокоуровневых, и в низкоуровневых структурах. Главная сила шаблонов в их универсальности. Определенный шаблон может применяться и в традиционном ПО, и в web-сайте, и в промышленных панелях управления, и в видеоиграх, и в интерфейсе телефона. Есть ряд причин, стимулирующих использование шаблонов в ПИ:

• опытные дизайнеры стремятся систематизировать накопленные знания и опыт для ускорения работы. Такая систематизация выгодна (возможно, даже в большей степени) и начинающим дизайнерам. Использование языка шаблонов может избавить их от повторения ошибок, которые уже кто-то совершал раньше. В данном случае язык шаблонов работает как аккумулятор лучших идей и решений в области дизайна взаимодействия, поэтому важно, чтобы в создании шаблонов принимали участие как можно больше профессионалов;

• язык шаблонов представляет собой отличное средство коммуникации между различными специалистами, работающими над одним дизайнерским проектом: дизайнерами, программистами и др.;

• шаблоны помогают посмотреть на старые проблемы свежим взглядом, преодолеть косность мышления, когда кажется, что все лучшие решения придуманы и остается только копировать чужие (кстати, порой не самые лучшие) интерфейсные решения. Наверное, каждому приходилось произносить фразу «почему я не подумал об этом раньше?»;

• язык шаблонов может служить удобным и универсальным способом документирования интерактивных систем, включая высокоуровневые и низкоуровневые структуры.

Перечисленные достоинства языка шаблонов тесно связаны с настоятельной рекомендацией письменного обоснования (или хотя бы объяснения) всякого дизайнерского решения, о чем подробнее будет сказано в разд. 5.2.3. Язык шаблонов задает структуру для такого обоснования.

На сегодня, к сожалению, всеобщего единого универсального языка шаблонов для дизайна взаимодействия не существует. Тем не менее многие компании, которые занимаются дизайном интерфейсов, создают свои языки шаблонов для внутреннего использования. Создавая язык шаблонов, необходимо определить стандартную структуру описания его элементов (шаблонов). Существует несколько отличных друг от друга структур, разработанных разными авторами. Для примера мы рассмотрим структуру описания, которую использует Дж. Тидвелл [4]. Данная структура хороша как пример именно своей полнотой. Итак, описание каждого шаблона состоит из следующих пунктов:

1. *Примеры* – краткие примеры из различных областей применения дизайна взаимодействия: программного обеспечения, Интернета, панелей управления, бытовой техники и т.д.

2. *Контекст* – контекст, обстоятельства, в которых следует использовать данный шаблон. Например, для шаблона «Контрольная панель» контекст будет следующим: «Изделие должно обеспечить пользователю возможность либо изменить состояние артефакта, либо отдать команду что-то сделать».

3. *Проблема* – проблема, которую при помощи данного шаблона не обходимо решить. Для

того же шаблона «Контрольная панель» примером проблемы является фраза: «Каким образом изделие может наилучшим образом показать, какие действия пользователь может совершить?»

4. *Условия* – дополнительные условия, которые необходимо принимать во внимание при нахождении решения проблемы. Обычно в этом пункте перечисляются 3-4 наиболее важных условия.

5. *Решение* – описание решения проблемы с учетом условий и контекста. Обычно дается краткое, сжатое описание решения, а затем идет объяснение.

6. *Результатный контекст* – обычно ссылки на другие шаблоны, ко-

торые рекомендуется использовать вместе с рассматриваемым. Вообще, описания шаблонов полны перекрестных ссылок друг на друга.

Начало увлечения языками шаблонов в человеко-компьютерном взаимодействии (даже мода на них) пришлось на середину 1990-х годов, и число сторонников этой идеи только растет. Поклонники концепции шаблонов создали целое направление в дизайне интерфейсов (его часто называют «Pattern Movement»). Подробности и литературу об этом направлении можно узнать на сайте http://patterns.usethics.ru/.

Некоторые ***общие принципы интуитивной «понятности» ПИ*** можно свести к следующим.

Полнота – все требования к интерактивному взаимодействию удовлетворяются при решении всех предусмотренных задач.

Управляемость – использованные медийные среды должны управляться пользователем. Должна обеспечиваться возможность всегда остановить процесс, повторить его ход, чтобы сосредоточиться на деталях, либо вывести последовательность процесса для его лучшего понимания. Средства управления должны быть хорошо известны пользователю и спроектированы с учетом зрительной аналогии с реальными физическими средствами.

Последовательность – важно соблюдать последовательность компоновки средств индикации и управления для сохранения хорошей ориентации пользователя. Одинаковые функции или элементы дисплея должны всегда быть на том же самом месте. Одинаковое действие должно всегда иметь тот же самый результат.

Избыточность – параллельное представление важной информации различными средствами мультимедиа, что улучшает восприятие этой информации.

Ориентация – пользователи должны всегда знать, где они находятся и по какому пути следует двигаться для получения нужной информации. Это может достигаться путем использования общего плана, списка разделов, указателей, навигационных диаграмм, круговых обзоров и т.д.

Обратная связь – наличие обратной связи при вводе обеспечивает безопасность пользователей, подтверждая, что система правильно восприняла их действие. Это особенно важно для речевого ввода, поскольку здесь имеет место относительно высокий уровень ошибок по сравнению с обычными способами ввода.

Гибкость – в зависимости от опыта, знания предметной области и намеченных целей пользователь должен иметь возможность выбирать модальность или модальности, с которыми он будет работать в доступных средах.

Реверсивность – пользователь должен всегда иметь возможность вернуться к предыдущей стадии в процессе интерактивного взаимодействия. Каждая среда должна поддерживать возможность «undo», т.е. отмену сделанного действия. Важно также дать пользователю возможность простым способом обращать действие (изменять на противоположное) и в любой момент легко выходить из системы. Например, в web-браузе-рах обычно используется кнопка-ключ «Back», которая доступна всегда и отображается в хорошо заметном и всегда одном и том же месте. Весьма желательна и возможность многоуровневой отмены, позволяющей пользователю возвращаться назад шаг за шагом.

4.4.2. Обучающие материалы

Создание хороших обучающих материалов является сложным делом, требующим значительного опыта и таланта. Здесь не говорится о том, как писать или подавать справочную

систему или документацию, а только о том, как интегрировать их с интерфейсом. Количество подсистем справки, нужных для того, чтобы пользователь научился работать с системой, довольно невелико. Когда говорится «подсистема справки», имеется в виду часть справочной системы (или документации), которая выполняет определенные функции и требует определенных методов представления.

Базовая справка – объясняет пользователю сущность и назначение системы. Обычно должна работать только один раз, объясняя пользователю, зачем система нужна. Как правило, не требуется для ПО, зато почти всегда – для сайтов.

Обзорная справка – рекламирует пользователю функции системы. Также обычно работает один раз. Нужна и ПО, и сайтам, причем тем больше, чем более функциональна система. Поскольку у зрелых систем функциональность обычно очень велика, невозможно добиться того, чтобы пользователи запоминали ее за один раз. В этом случае оптимальным вариантом являются слежение за действиями пользователя и показ коротких реклам типа «А вы знаете, что…» в случае заранее определенных действий пользователей (примером такого подхода являются помощники в последних версиях MS Office).

Справка предметной области – отвечает на вопрос, как сделать хорошо. Поскольку от пользователей зачастую трудно ждать обширных знаний предметной области, необходимо обучать их на ходу. При этом действуют два правила: во-первых, пользователи не любят признавать, что они чего-либо не знают, соответственно, подавать это знание надо как бы между прочим; во-вторых, наличие такой подсказки повышает субъективную оценку справочной системы в целом, т.е. приводит к тому, что пользователи чаще обращаются к справочной системе и от этого эффективнее учатся.

Процедурная справка – отвечает на вопрос, как это сделать. В идеале она должна быть максимально доступна, поскольку, если пользователь не найдет нужную информацию быстро, он перестанет искать и так и не научится пользоваться функцией.

Контекстная справка – отвечает на вопросы, что делать и зачем это нужно. Как правило, наибольший интерес в ПО представляет первый вопрос, поскольку уже по названию элемента должно быть понятно его назначение (в противном случае его лучше вообще выкинуть), а в Интернете – второй (из-за невозможности предугадать, что именно будет на следующей странице). Поскольку пользователи обращаются к контекстной справке во время выполнения какого-либо действия, она ни в коем случае не должна прерывать это действие (чтобы не ломать контекст действий), ее облик должен быть максимально сдержанным, а объем информации в ней – минимальным.

Справка состояния – отвечает на вопрос, что происходит в настоящий момент. Поскольку она требуется именно в настоящий момент, то не может быть вынесена из интерфейса. В целом это самая непроблематичная для разработчиков система справки.

Сообщения об ошибках – это тема весьма многогранная (о ней подробнее см. в разд. 4.3). Поскольку здесь мы описываем только интеграцию справочных систем с интерфейсом, разумно будет свести получившийся список типов обучающих материалов со средами передачи этих материалов. Среды же передачи, имеющиеся в нашем распоряжении, таковы:

1. *Бумажная книга.* На одном листе может быть сосредоточено очень много материала, что позволяет получить большой объем информации за один сеанс; наилучшим образом работает при последовательном чтении. Сравнительно плохой поиск нужных сведений. Объем практически всегда лимитирован.

2. *Справочная карта.* Отдельная краткая бумажная документация, демонстрирующая основные способы взаимодействия с системой (quick reference card). Будучи реализована на одном листе бумаги, позволяет пользователю повесить ее перед собой. Хороша как средство обучения продвинутым способам взаимодействия с системой и устройству навигации в системе.

3. *Структурированная электронная документация.* Не предназначена для чтения больших объемов материала, зато обеспечивает легкий поиск и не имеет лимита объема. Занимает большую часть пространства экрана. Плохо подходит для показа крупных изображений, зато в нее могут быть легко интегрированы видео и звук.

4. *Фрагменты пространства интерфейса, показывающие справочную информацию.*

Занимают ограниченное пространство экрана. Отвлекают внимание, как минимум, один раз, воспринимаются всеми пользователями. Как правило, не способны передавать большой объем информации (рис. 4.12).

Рис. 4.12. Пример пространства интерфейса, выделенного на показ справочной информации (обратите внимание: даже обычная подпись к полю ввода также является таким пространством)

5. *Всплывающие подсказки.* Хорошо справляются с ответом на вопросы, что это такое и зачем это нужно, при условии, что объем ответов сравнительно невелик. Поскольку вызываются пользователями вручную, в обычном режиме не занимают пространства экрана и не отвлекают внимание пользователей. В то же время от них очень легко отвыкнуть – после первой же неудовлетворенности подсказкой пользователь перестает вызывать и все остальные подсказки.

Рассмотрим более детально те виды справочных систем, за которые ответственен разработчик ПИ. Как упоминалось выше, объяснить пользователю сущность и назначение, а также отрекламировать функции системы чрезвычайно важно, поскольку только это создает желание учиться. Эти системы справки обычно реализуются в бумажной документации. Это хорошо, но можно сделать и лучше, поскольку в последнее время появилась возможность интегрировать в справочную систему видео при помощи либо Macromedia Flash, либо Shockwave. Нет сомнений, что реклама, поданная в виде не просто текста с картинками, а анимации, способна повысить как желание ее просмотреть, так и субъективное удовлетворение пользователей от системы. Заниматься этим должен не дизайнер интерфейсов, а графический дизайнер. Наиболее сложно создать более-менее хороший сценарий (его лучше отдать профессиональному техническому писателю). Дизайнер интерфейса в этом случае должен только составить список функций, которые нужно рекламировать.

Справка предметной области – реализуется также в бумажной документации. Однако, как минимум, часть ее можно подавать пользователям в интерфейсе вместе с выдержками из обзорной справки (лучше динамически, типа «помощник» из MS Office). Вообще говоря, справка предметной области является *самой важной подсистемой справки.* Достаточно упрямый или «компьютерно-грамотный» пользователь сможет воспользоваться системой, лишенной всех справочных систем, более того, такой пользователь сможет даже научиться пользоваться такой системой. Но без знания предметной области он никогда не сможет пользоваться системой правильно и эффективно.

Лучшим местом для *процедурной справки* является выделенная справочная система. В нее она чаще всего и помещается. Вызывает, однако, сожаление тот факт, что разработчики чаще всего не привязывают темы справки к интерфейсу: когда пользователям непонятно, как выполнить нужное им действие, им приходится искать в справочной системе необходимую тему. Это неправильно, тем более что технических проблем в этом нет.

Для *контекстной справки* используют всплывающие подсказки (ToolTip) и, в последнее время, «пузыри». Это очень правильное решение. Плохо одно – этот тип справки практически полностью отсутствует в Интернете. Если разработчики ПО уже привыкли писать ко всем объектам и элементам управления подсказки, то для web-дизайнеров это пока новое дело. Интересно, что в Интернете контекстная справочная система, как правило, нужнее, чем в ПО, – просто потому, что большинство сайтов является однократно используемым, при этом их пользователями являются изначально не обученные люди.

Одним из наиболее правильных способов организации справочных материалов является спиральность текста. В отличие от художественной литературы справочные системы не

предназначены для того, чтобы приносить удовольствие. Более того, поскольку пользователи обращаются к справочной системе при возникновении проблем, можно смело сказать, что использование справочной системы всегда воспринимается негативно. Таким образом, следует всемерно сокращать объем справочной системы, чтобы снизить длительность неудовольствия. К сожалению, это имеет свои недостатки – при малом объеме справочной системы возрастает риск того, что пользователи не найдут в ней ответы на свои вопросы.

Есть, однако, весьма эффективный метод решения этой проблемы: так называемые *спиральные тексты.* Идея заключается в следующем.

При возникновении вопроса пользователь получает только чрезвычайно сжатый, ограниченный ответ (1-3 предложения). Если ответ достаточен, пользователь может вернуться к выполнению текущей задачи, тем самым длительность доступа к справочной системе (и неудовольствие) оказывается минимальной. Если ответ не удовлетворяет пользователя, он может запросить более полный и объемный ответ. Если и этот ответ недостаточен (что случается, вероятно, весьма редко), пользователь может обратиться к еще более подробному ответу. Таким образом, при использовании этого метода пользователи получают именно тот объем справочной системы, который им нужен. Спиральность текста считается нормой при разработке документации. Есть веские основания считать, что она необходима вообще в любой справочной системе.

Краткое и в то же время достаточно всестороннее описание инновационных особенностей и функций имеет место, например, в ОС MS Windows Vista, которая является новым поколением клиентских операционных систем Windows и логическим продолжением ОС Windows XP. В руководстве для пользователя представлены сведения о преимуществах, которые Windows Vista предоставляет разным группам пользователей, а также сведения о существующих версиях операционной системы.

4.5. Субъективная удовлетворенность

Субъективная удовлетворенность пользователя от работы тесно связана с комфортностью его взаимодействия с ПИ и способствует снижению текучести кадров на предприятии заказчика за счет привлекательности работы на данном рабочем месте. Повышение такой удовлетворенности при прочих равных условиях всегда приводит к росту производительности пользователя. Требования к удобству и комфортности ПИ ужесточаются с повышением сложности работ и ответственности пользователя за конечный результат.

Вместе с тем интерфейс должен быть прежде всего работоспособным. Частый конфликт между эстетической привлекательностью и функциональностью привел к существованию двух противостоящих позиций – одна считает приоритетным функциональность, другая – эстетику. Возможно, истина лежит, как ей и полагается, ближе к середине (хотя правильнее все же считать, что функциональность важнее эстетики), но «срединный путь» до сих пор не найден, интерфейсы удобные и эстетически привлекательные существуют в единичных экземплярах. Говоря об эстетической привлекательности, надо признать, что принципы многих направлений дизайна вполне применимы к дизайну интерфейсов, он имеет черты, как сближающие его с иными направлениями дизайна, так и разъединяющие. Уровень же психического напряжения (тем более наличие стресса) – важнейший ресурсный показатель работы с ПИ.

Скорость работы пользователя является, с одной стороны, объективным показателем (вспомним метод GOMS, разд. 4.2), с другой – субъективным: скорость работы и вносит свой вклад в субъективное удовлетворение. Дело здесь в том, что субъективное ощущение длительности, как правило, не соответствует ее объективной величине. Это легко проверить с помощью секундомера. Конечно, положительная корреляция между ними присутствует, и повышение объективной скорости работы способно повысить и субъективное ощущение скорости. Способы повышения субъективной скорости работы освещены в разд. 4.2.

Основными составляющими субъективной удовлетворенности считаются эстетическая привлекательность, скорость работы пользователя, уровень психического напряжения (тем более наличие стресса). Скорость работы пользователей достаточно подробно рассмотрена в

разд. 4.2. Эстетическую же привлекательность и уровень психического напряжения рассмотрим далее.

4.5.1. Эстетическая привлекательность

Принципы многих направлений дизайна вполне применимы к дизайну интерфейса, при этом наибольший вклад вносят книжный, коммуникационный и промышленный дизайн (техническая эстетика). Какие принципы могут быть использованы в дизайне интерфейса? Конструируемый предмет должен быть незаметен в процессе его использования. Странно интересоваться, как выглядит стул, на котором сидишь. Когда человек читает книгу, он чаще всего не замечает ее верстку. В то же время предмет должен приятно ощущаться на неосознаваемом уровне. Отсюда важная рекомендация – всегда следует добиваться неощущаемости ПИ. Для этого:

• надо быть умеренным в визуальных эффектах, избегать визуальной «развязности»;

• не использовать без явной необходимости яркие цвета. Существует очень немного цветов, обладающих и яркостью, и мягкостью. На экране их значительно меньше, поскольку в жизни такие цвета обычно воссоздаются как собственно цветом, так и текстурой, а с последней на экране есть проблемы;

• лучше не использовать острые углы в визуальных формах;

• надо стремиться к возможно более легкому и воздушному оформлению ПИ;

• лучше достигать контраста не сменой насыщенности элементов, а расположением пустот.

Красота – понятие относительное. Для одних красивыми могут считаться цветущие розы, для других – картины С. Дали, а для третьих – сосиски с запотевшей бутылкой пива. Это делает красоту «вещью», не слишком универсальной. Хуже того, любая красота со временем надоедает и в лучшем случае перестает восприниматься. Именно поэтому в ПИ обычно не место красоте. Элегантность и гармония гораздо лучше. Во-первых, они не надоедают. Во-вторых, они редко осознаются потребителями, обеспечивая неощущаемость. В-третьих, они приносят эстетическое удовольствие независимо от культурного уровня потребителя (так, древнегреческие и несколько менее древние римские здания воспринимаются нами красивыми, несмотря на огромную разницу культур и времени). В-четвертых, технологически они гораздо удобнее красоты. Есть несколько общих правил, придерживаясь которых можно добиться элегантности:

1. Старайтесь сделать интерфейс максимально насыщенным визуальными закономерностями. Есть универсальное правило – чем больше закономерностей, тем больше гармонии. Даже самые незначительные закономерности все равно воспринимаются. Под закономерностью понимается любое методически выдерживаемое соответствие свойств у разных объектов, например высота кнопок может быть равна удвоенному полю диалогового окна (вспомним пользу стандартов и языка шаблонов – см. разд. 4.4.1).

2. Всемерно старайтесь использовать модульные сетки, т.е. привязывайте все объекты к линиям (лучше узлам) воображаемой сетки, которую выдерживайте во всем интерфейсе.

3. Старайтесь привязывать все размеры и координаты (как минимум, пропорции диалоговых окон) к золотому сечению. Вроде бы несложно, но результат ощутимый.

Пример закономерностей в диалоговом окне показан на рис. 4.13.

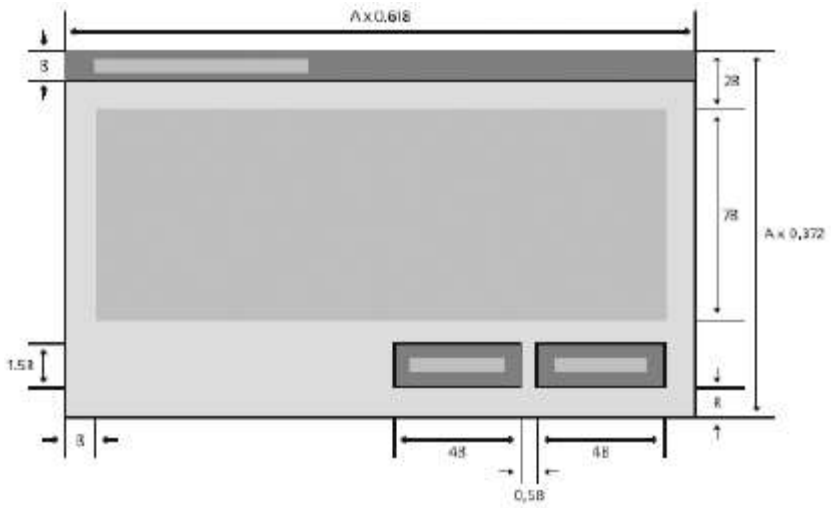

Рис. 4.13. Пример закономерностей в диалоговом окне (разумеется, единожды примененных закономерностей необходимо придерживаться во всем ПИ)

На эту тему можно сказать очень много (странно было бы ожидать, что на нескольких страницах удастся выразить весь опыт дизайнерской культуры). Поэтому полезно прочитать несколько книг по графическому дизайну (лучше книжному). Именно дизайн повседневных вещей является идейной базой для дизайна интерфейса [11].

4.5.2. Уровень психической напряженности

Уровень психической напряженности (наличие стресса) – важнейший ресурсный показатель работы пользователя. В нем отражаются все проблемы ПИ, о которых говорилось выше: скорость работы, ошибки, скорость обучения, эстетическая привлекательность. Поэтому этот показатель является производным от остальных. Конечно, немалое значение имеют и индивидуально-психологические особенности конкретного индивида, но их рассмотрение выходит за рамки рассматриваемой темы. Низкий уровень психической напряженности (именно к такому следует стремиться) достигается в случае:

• прозрачной для пользователя навигации и целевой ориентации в программе. Главное, чтобы было понятно, куда идем и какую операцию программа после этого шага произведет;

• ясности и четкости понимания пользователем текстов и значения икон. В программе должны быть те слова и графические образы, которые пользователь знает или обязан знать по характеру его работы или занимаемой должности;

• быстроты обучения при работе с программой, для чего необходимо использовать преимущественно стандартные элементы взаимодействия, их традиционное или общепринятое расположение;

• наличия вспомогательных средств поддержки пользователя (поисковых, справочных, нормативных), в том числе и для принятия решения в неопределенной ситуации (ввод по умолчанию, обход зависания процессов и др.).

Мы здесь несколько подробнее остановимся лишь на характерных ошибках в ПИ, сравнительно часто вызывающих стресс у большинства пользователей.

Один из наиболее частых источников стресса – страх пользователя что-либо испортить, *чувство неполного контроля над системой.* Конечно, возможность отмены пользователем своих предыдущих действий в значительной мере решает эту проблему, так как человек, знающий, что он *не может* совершить ошибку, значительно реже испытывает стресс. Довольно частым решением этой проблемы является давно существующая практика прятать опасные для пользователя места интерфейса. Формально это хороший и правильный способ. Проблема заключается в том, что при этом логично прятать все функции, изменяющие данные, например банальная функция автоматической замены может мгновенно уничтожить текст документа (достаточно заменить одну букву на любую другую во всем тексте документа и забыть отменить это действие).

Дело в том, что почти все время пользователь может что-либо испортить и знает это. Неудивительно, что пользователь часто боится. А если не боится, то склонен недооценивать свои возможности к разрушению, отчего снижается бдительность. Единственным решением является возможность отмены пользователем своих предыдущих действий, без ограничения количества уровней отмены и типа отменяемых действий (сейчас нормой является, например, невозможность отменить свои действия после записи файла). Задача эта непростая, зато результат крайне существенен. Но надо отметить, что, к сожалению, создание таких систем, не будучи с точки зрения технологии трудным делом, требует смены парадигмы мышления программистов, так что ожидать скорого массового прозрения не приходится.

Очень важно чувство контроля над системой. Существует значительная часть пользователей, для которых работа на компьютере не является действием привычным. Для таких пользователей ощущение того, что они не способны контролировать работу компьютера, является сильнейшим источником стресса. Для остальных пользователей отсутствие чувства контроля не вызывает стресса, но все равно приводит к неудовольствию. Таким образом, нужно пользователю дать ощущение, что ничего не может произойти, пока этого не захочется ему самому. Функции, работающие в автоматическом режиме, но время от времени просыпающиеся и требующие от пользователей реакции, вызывают стресс. В любом случае стоит всегда внушать пользователям мысль, что только явно выраженное действие приводит к ответному действию системы.

Достаточно частый источник стресса – *пароли.* Дело в том, что пароли требуют от пользователей выполнения действий, не приносящих пользы, но уменьшающих *потенциальный* вред. Психологически же потенциальное кажется неважным, а вот действия, которые приходится совершать, важны чрезвычайно. В результате пароли не любит никто. Пароли имеют три принципиальных проблемы. Во-первых, как уже было сказано, пользователи не любят их вводить. Во-вторых, пользователи забывают пароли либо пароли не работают, т.е. ничего не защищают, поскольку пользователи используют те пароли, которые они помнят и которые, соответственно, нетрудно подобрать. В-третьих, в Интернете часто происходит так, что пользователи, не желая вводить пароль (и регистрироваться, кстати), так никогда и не попадают в главную часть сайта. В результате можно утверждать, что пароли есть неизбежное зло, которое необходимо уменьшить. Для этого есть несколько путей. Один из них – отказ от паролей вообще. Этот путь предпочтителен, проблема лишь в том, что он вызывает существенное отторжение у системных администраторов.

Пароли хороши, когда дело касается защиты информации. Однако гораздо чаще, особенно в Интернете, важна не столько защита информации, сколько идентификация пользователя. В этом случае от паролей отказаться невозможно просто потому, что не существует другой, столь же эффективной технологии идентификации. При этом не так важно содержимое пароля, как его отличие от паролей других пользователей (обратите внимание: в этом случае само по себе имя пользователя является паролем). Обычно в таких условиях при регистрации применяют стандартную группу элементов – имя, пароль, подтверждение пароля, после чего уже при нормальной деятельности пользователь получает возможность ввести пароль только в первый раз (рис. 4.14).

Рис. 4.14. Стандартный способ «человеколюбивого» использования паролей: слева – во время регистрации, справа – в дальнейшей работе

Данное решение неплохое, но и оно не без недостатков. Дело в том, что пользователь, однократно введя пароль, потом его забывает, поскольку из-за отсутствия повторения пароль не запоминается. В результате, когда пользователь переходит на другой компьютер или переустанавливает ОС, ему приходится регистрироваться снова. Таким образом, эффективнее всего здраво оценить степень важности защищаемой информации, после чего выбрать адекватную схему защиты, стараясь, чтобы она была максимально легкой для пользователей.

Как правило, их субъективное удовлетворение важнее потенциального вреда от потери информации.

К факторам, существенно влияющим на чувство удовлетворения работой и, соответственно, на уровень психического напряжения, является *возможность самовыражения.* Ни один человек не может существовать сколько-нибудь продолжительное время в обстановке, не допускающей самовыражения. Неудивительно, что пользователи хотят выразить себя и в программах, в которых они работают, поэтому возможность настроить систему под свои вкусы является значимой для субъективного удовлетворения. Проблема здесь в том, что это довольно дорого. Ведь пользователи не склонны удовлетворяться первым получившимся вариантом (творчество, процесс итерационный), и времени на самовыражение уходит много. В то же время настроенный под свои нужды интерфейс снижает усталость работников и улучшает их рабочее настроение. Поэтому в продуктах, продаваемых пользователям напрямую, возможность настройки под конкретного пользователя обязательна. Во всех остальных случаях нужен способ настройки, позволяющий максимально изменить вид системы минимумом команд (чтобы снизить время, затрачиваемое на самовыражение). Хорошим вариантом является банальный выбор варианта готовой настройки из списка без возможности модифицировать встроенные варианты.

И наконец, наиболее часто встречающимся фактором, влияющим на уровень психического напряжения, являются ошибки в ПИ, например *несоответствие ПИ особенностям пользователей.* Характеристики пользователей, к сожалению, редко бывают предметом анализа в начале разработки системы, хотя требования к аппаратной части всегда четко определяются. В результате ПИ может получиться в лучшем случае противоречивым (один модуль рассчитан на опытных пользователей, а другой – на новичков, хотя оба модуля предназначены для одних и тех же людей). Поэтому так важно построение пользовательских профилей и других моделей пользователей с самого начала разработки ПО (разд. 3). Довольно много страниц в сегодняшнем web ярко напоминают Н.В. Гоголя: «А вот есть в этом городе Добчинский!». Кроме факта присутствия в Сети абсолютно неясны мотивы, приведшие к появлению страницы. Вопрос, кто и зачем приходит на мой сайт, задают немногие, а уж про сценарии пользования и пользовательские профили вообще мало кто слыхал. Поэтому такие страницы в основном решают проблемы их авторов, а не проблемы их посетителей. В web-сайтах, например, часто ошибочно считается, что в сайте самое главное – привлекательность, а не содержательность. Конечно, если речь идет о сайте компании «Disney», куда заходят маленькие дети, это может быть и справедливо, но видеть кричащий дизайн в сайте, посвященном научной конференции, как-то странно.

Другой распространенной ошибкой является *неадекватность интерфейса среде использования* системы. Например, скорость работы самой системы иногда оказывается существенным фактором. Так, в некоторых системах есть операции, выполняемые пользователем во время разговора по телефону (таковы многие операции у брокеров, операторов складов и др.). Если такая операция будет занимать долгое время, проблем разработчику не избежать. Или соотношение размера экрана и разрешения мониторов в пикселях. Если физическое разрешение велико, элементы управления становятся слишком малы и неразборчивы. Если система разрабатывается без оглядки на это разрешение, велика вероятность появления так называемой проблемы режима Large Fonts (рис. 4.15).

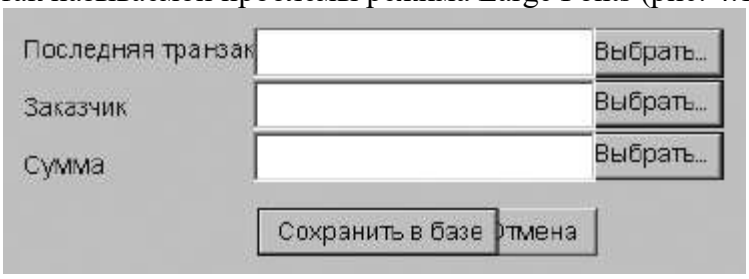

Рис. 4.15. Если интерфейс программы не был разработан в расчете на возможное включение режима Large Fonts, все элементы управления «поплывут»

Другой типичной ошибкой является *неадекватность интерфейса содержательной*

деятельности пользователей. Суть такой ошибки, как правило, в том, что структура ПИ приведена в соответствие с информационными потоками (структурой объектов) внутри самой системы, а не в соответствие со структурой деятельности пользователей. Как следствие интерфейс зашумлен информацией, релевантной объекту, с которым работает пользователь, но либо не нужной ему сейчас, либо не нужной ему вообще, поскольку разным сотрудникам нужна разная информация (рис. 4.16).

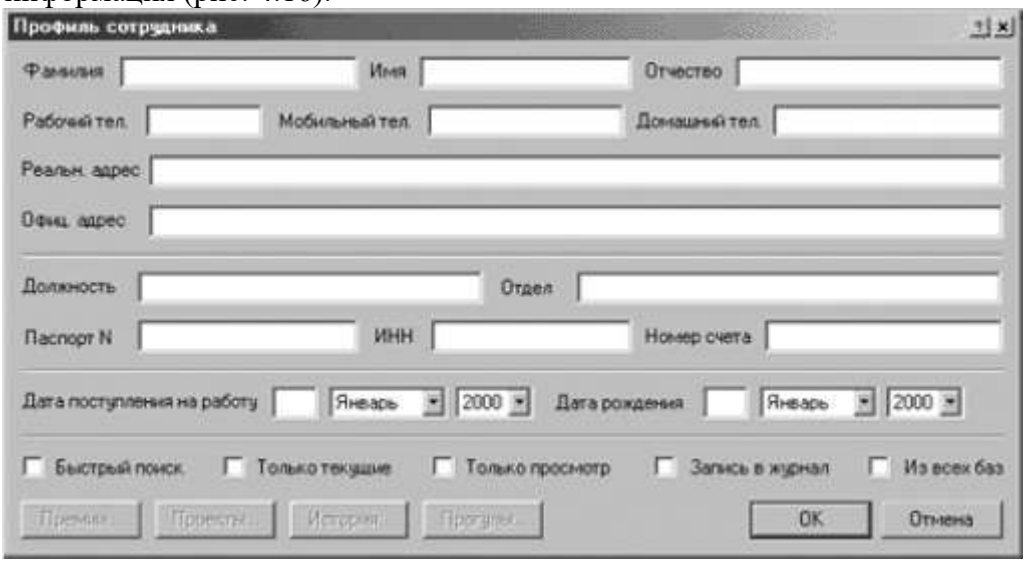

Рис. 4.16. Подобного рода диалоговые окна типичны во многих системах автоматизации

Обратной стороной этой проблемы можно считать универсальность – дешевле разработать единое окно, с которым будут работать несколько разных служащих (бухгалтер, директор, менеджер по продажам и т.п.), чем три окна для каждого из них. Но эта универсальность оборачивается значительной неэффективностью интерфейса (и как следствие системы в целом). Таким образом, ПИ должен в первую очередь определяться структурой деятельности пользователей, а не структурой объектов, с которыми пользователь взаимодействует в процессе работы.

Впрочем, и здесь возможны крайности, когда интерфейсные элементы (выводимая информация или блоки ввода информации) являются атрибутами разных объектов, соответственно, разработчики размещают их на разных экранах, игнорируя то обстоятельство, что для выполнения типовых операций пользователю необходимо взаимодействовать одновременно со всеми этими элементами.

Соответствие стандартам, шаблонам крайне важно для снижения уровня психической напряженности. По этой причине специалистами корпорации Oracle предложен метод количественной оценки *степени соответствия ожиданиям пользователей.* Этот метод используется, в частности, в составе юзабилити-тестирования ПИ. Преимущества этого метода в легкости проведения, простоте анализа результатов и мер, совместимых с другими методами (шкала отношений, коэффициентов). Много неприятностей для пользователей таят и фреймы – недаром появилось общественное движение против фреймов. Фреймы, в частности, нарушают тот единственный стандарт использования браузера, который в остальных случаях работает безотказно: нажатие на кнопку «Назад» возвращает к предыдущему состоянию экрана. При работе же с фреймами результат такого нажатия и само понятие предыдущего состояния экрана становятся совершенно иными.

Говоря о частых ошибках, выражающихся в нарушении принципа соответствия ожиданиям и, соответственно, приводящих к увеличению психической напряженности, следует упомянуть и о баннерах. Перевести слово «banner» на русский язык лучше всего словом «фантик». Предназначение фантиков, с одной стороны,– обернуть полезный товар (сайт) в красивую обертку, а с другой стороны, разместить на них полезную информацию о других (хотелось бы думать, аналогичных) товарах. На самом же деле, «кликанье» по фантику приводит обычно к месту, ничего общего не имеющему с вашими ожиданиями. Фантики в Сети

превратились в своеобразную игру владельцев и участников баннерных сетей, и в этой игре пользователь всегда проигрывает. Но пользователя тоже так просто не проведешь – быстро возникает условный рефлекс игнорировать фантики, и даже если он нечаянно нажал на фантик, быстро нажимает кнопку «Назад», не читая информацию на полученной через фантик странице. Но все же уровень психической напряженности такие фантики увеличивают.

Как было сказано, уровень психической напряженности зависит почти от всех характеристик ПИ. В общем случае можно рекомендовать придерживаться основополагающих правил и принципов при разработке ПИ. Так, например, Б. Шнейдерман [18] указывает восемь «золотых правил» для разработки ПИ.

1. Стремитесь к последовательности (непротиворечивости).
2. Побуждайте тех, кто часто использует данную систему, использовать сокращенные пути.
3. Предлагайте информативную обратную связь.
4. Проектируйте диалоги так, чтобы не прерывался контакт с пользователем.
5. Предлагайте ПИ, предотвращающий ошибки, и простой путь их исправления.
6. Предусматривайте легкую отмену действия и возврат назад.
7. Поддерживайте контроль, расположенный внутри системы.
8. Сокращайте нагрузку на кратковременную (активную) память пользователя.

Особенно много весьма показательных ошибок в ПИ в Интернете, в дизайне web-страниц. Их рассмотрение можно найти по адресам:

http://hci.psychology.ru/Articles/10_Errors.doc
http://www.lboro.ac.uk/eusc/index_r_guides.html
http://www.usethics.ru/lib/krug/index.shtml
http://www.usethics.ru/lib/future_ui/index.shtml
http://www.webclub.ru/
http://citforum.ru/index.html и др.

Контрольные вопросы

1. Каковы цели оценки ПИ и условия, в которых они могут производиться?
2. Каковы достоинства и недостатки оценок в лабораторных условиях и в условиях реальной работы?
3. Состав и содержание методов оценок, основанных на анализе мнений экспертов, и оценок, основанных на участии пользователей.
4. Состав и содержание аналитических методов оценки.
5. Состав и содержание экспериментальных методов оценки и методов на основе анкетирования.
6. Состав и содержание методов наблюдений.
7. Физиологические параметры, измеряемые при оценке ПИ, их значение и методы регистрации.
8. Состав критериев оценки ПИ.
9. Методы оценки скорости работы пользователя и пути увеличения скорости работы.
10. Субъективная длительность действий и способы ее уменьшения.
11. Основные направления усилий по снижению ошибок пользователя и отражение таких усилий в характеристиках ПИ.
12. Стиль и формы сообщений пользователю о его ошибках.
13. Метафоры в ПИ, их роль, преимущества и недостатки, примеры метафорических дизайнерских решений в ПИ.
14. Роль стандартов и языка шаблонов в ПИ, правила описания шаблона.
15. Принципы интуитивной понятности ПИ.
16. Виды обучающих материалов, их классификация.
17. Спиральность текста справки, роль и способы реализации.
18. Основные составляющие субъективной удовлетворенности пользователя. Содержание

восьми «золотых правил» Б. Шнейдермана.

19. Основные принципы и правила создания эстетической привлекательности ПИ.

20. Пути снижения уровня психической напряженности пользователя.

Литература

Основная

1. *Вятчин К.* Язык шаблонов в дизайне взаимодействия. http://www.usethics.ru/lib/ui_templates/index.shtml

2. *Торрес Р. Дж.* Практическое руководство по проектированию и разработке пользовательского интерфейса. – М.: Вильямс, 2002.

3. *Рейнбоу В.* Компьютерная графика. – СПб.: Питер, 2004.

4. *Tidwell J.* Pattern Language for Human-Computer Interface Design, 1999.

5. *О'Квин Д.* Допечатная подготовка. Руководство дизайнера. – М.: Вильямс, 2002.

6. *Robson C.* Experiment, Design and Statistics in Psychology. 3rd ed. – Penguin, 1994.

7. *Wharton C., Rieman J., Lewis C., Poison P.* The cognitive walkthrough: a practitioner's guide // J. Nielsen and R. L. Mack (Eds). Usability Inspection Methods. – John Wiley, 1994.

8. *Nielsen J.* Heuristic evaluation // J. Nielsen and R. L. Mack, (Eds.), Usability Inspection Methods. – John Wiley, 1994.

9. http://www.akzhan.midi.ru/iarchitect/mfame.htm

10. http://www.akzhan.midi.ru/iarchitect/mshame.htm

11. *Norman D.* The Design of Everyday Things. Publ. Currency/Doubleday; Reissue ed., 1990.

Дополнительная

12. *Дедков В.* Adobe Photoshop. Настольная книга мастера. – М.: Компьютерпресс, 2001.

13. *Харовас П., Кундерт-Гиббс Д., Ли П.* Maya complete. Уроки мастерства. – М.: ДМК Пресс, 2001.

14. *Carroll J.M.* HCI Models, Theories, and Frameworks: Toward a Multidiscip-linary Science (Morgan Kaufmann Series in Interactive Technologies). – Virginia Tech. Univ., 2003.

15. http://www.lboro.ac.uk/eusc/index_r_guides.html

16. *Ojakaar E.* Users Decide First; Move Second. Published: Oct 25, 2001. http:// HYPERLINK «http://www.uie.com/articles/users_decide_first/»www.uie.com/articles/users_decide_first/

17. *Duchowski T.* Eye Tracking Methodology: Theory and Practice. – Springer– Verlag, 2003.

18. *Shneiderman B.* Designing the User Interface: Strategies for Effective Human-Computer Interaction. 3rd ed. – Addison-Wesley, 1998.

19. *Rich A., Shores R., McGee M.* (Oracle Corp.) Expected usability magnitude estimation. Proc. of the Human Factors and Ergonomics Society. – 48th Annual Meeting, 2004.

20. http://hci.psychology.ru/Articles/10_Errors.doc

21. http://www.lboro.ac.uk/eusc/index_r_guides.html

22. http://www.usethics.ru/lib/krug/index.shtml

23. http://www.usethics.ru/lib/future_ui/index.shtml

24. http://www.webclub.ru/

25. http://citforum.ru/index.html

ТЕМА 5. ЮЗАБИЛИТИ-ТЕСТИРОВАНИЕ И ПРОТОТИПИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Изучаемые вопросы:

- Понятие юзабилити-тестирования.
- Цель юзабилити-тестирования.
- Базовые характеристики юзабилити-тестирования (согласно стандарту ISO 9241).
- Состав и содержание метрик юзабилити.

• Таксономия юзабилити.
• Подготовка и проведение юзабилити-тестирования.
• Анализ данных юзабилити-тестирования.
• Методики юзабилити-тестирования.
• Контрольные списки, их содержание и место в тестировании ПИ.
• Этапы разработки ПИ и их содержание.
• Принципы разработки ПИ.
• Содержание работ по формированию ПИ на каждом из этапов его разработки.
• Требования к прототипу на разных этапах разработки ПИ.
• Четыре версии прототипа, их характеристики, сравнительный анализ.
• Основные программные средства создания прототипов презентационной, псевдореальной и реальной версий.
• Необходимость письменного объяснения дизайнерских решений ПИ.
• Три основные группы методов объяснения дизайнерских решений, их сравнительный анализ.

5.1. Юзабилити-тестирование ПИ

5.1.1. Понятие и таксономия юзабилити-тестирования ПИ

Понятие «юзабилити-тестирование» объединяет чрезвычайно широкий спектр подходов и методов оценки качества не только интерфейса, но и всего комплекса человеко-компьютерного взаимодействия. Поэтому эта тема должна бы находиться в предыдущем разделе, посвященном именно оценке ПИ. Однако юзабилити-тестирование представляет собой достаточно самостоятельный и весьма объемный комплекс процедур, пронизывающих весь процесс разработки ПИ, и лучше его объединить, по нашему мнению, с описанием процесса итерационной разработки и прототипирования ПИ. Самый адекватный перевод слова «usability», пожалуй, будет «потребительские качества продукта». Необходимо иметь какие-либо способы оценки таких качеств.

Цель юзабилити-тестирования – выявить недостатки потребительских свойств ПИ (удобство, продуктивность, легкости понимания, легкость обучения, устойчивость к ошибкам). Положительный результат юзаби-лити-тестирования говорит о максимально эффективном и экономичном достижении поставленной цели в конкретных условиях работы для конкретной группы пользователей при наличии конкретных ограничений (технических, финансовых, временных и т.д.). Нельзя говорить о положительном или отрицательном результате юзабилити-тестирования в абстрактных условиях. Самый плохой на первый взгляд продукт может быть очень хорошо приспособлен для специфичных пользователей или задач.

Иногда юзабилити-тестирование ошибочно рассматривают как заключительный контроль качества и проводят его на одном из последних этапах разработки проекта. Перечень обнаруженных при этом недостатков, как правило, уже не может быть исправлен за оставшееся время работы над проектом. Поэтому очень важно, чтобы юзабилити-тестирова-ние было органической составной частью разработки ПИ. Это, помимо прочих преимуществ, ведет к снижению затрат и времени.

Юзабилити-тестирование производится на протяжении всего цикла разработки. На ранних этапах разработки тестирование предыдущей версии или конкурирующих продуктов позволяет наметить некоторую цепочку целей, которые необходимо достигнуть в процессе разработки. В середине работы над проектом тестирование предоставляет обратную связь, сообщая места, где ПИ нуждается в улучшении. На заключительных этапах тестирование удостоверяет, что продукт соответствует (или не вполне соответствует) запросам и потребностям тех пользователей, для которых был создан. Можно сказать, что юзабилити-тестирование – важнейшая составная часть разработки ПИ.

Широта трактовок юзабилити-тестирования, постоянное употребление этого термина в текстах по человеко-компьютерному взаимодействию (особенно популярных) нередко приводят к его неоправданной автономизации, отрыву от целостного комплекса работ по

формированию ПИ. Имеются и попытки выделения неправомерно узких специализаций, вроде «юзабилити-инженер» или «специалист по юзабилити». Размытость сферы компетенции и границ ответственности таких специалистов приводит большей частью к их неприятию в среде разработчиков ПО и фактическому устранению из процесса собственно разработки. Эта же причина движет и стремлением более четко определить содержание юзабилити-тестирования.

Стандарт по человеко-компьютерному взаимодействию ISO 9241 включает в себя три базовые характеристики юзабилити:

1) эффективность – *можете ли вы сделать то, что хотите;*

2) требуемые усилия (легкость) – *можете ли вы сделать это без излишних усилий;*

3) субъективная удовлетворенность – *доставляет ли вам удовольствие процесс «делания».*

Этими характеристиками должен быть описан любой конкретный показатель системы, как регламентировано в том же стандарте (табл. 5.1).

Таблица 5.1

Количественные измерения четырех показателей юзабилити по трех базовым характеристикам

Показатель юзабилити	**Базовые характеристики юзабилити**		
	Эффективность	**Требуемые усилия**	**Субъективная удовлетворенность**
Соответствие задаче	Процент достигнутых целей	Время выполнения задачи	Рейтинговая шкала удовлетворенности
Приспособленность для тренированных пользователей	Используемое число возможностей системы (процент ее мощности)	Эффективность работы по сравнению с опытным пользователем (экспертом)	Рейтинговая шкала удовлетворенности возможностями системы (ее мощностью)
Обучаемость	Процент усвоенных функций	Время обучения	Рейтинговая шкала легкости обучения
Терпимость к ошибкам	Процент успешно исправленных ошибок	Время исправления ошибок	Рейтинговая шкала легкости обработки ошибок

Приведем перечень возможных показателей, которые могут использоваться для оценки величины юзабилити и установления лучшего/худшего варианта и также планируемого/текущего его уровня. Перечни такого рода называют метриками юзабилити.

1. Время выполнения задачи.
2. Процент выполненных задач.
3. Процент выполненных задач за определенное время.
4. Отношение успешно выполненных задач (произведенных действий) к неуспешно выполненным (произведенным).
5. Время, потраченное на обнаружение и исправление ошибок.
6. Процент или количество ошибок.
7. Процент или количество участников, показавших более высокие результаты, чем рассматриваемый.
8. Число произведенных действий.
9. Частота использования помощи или документации.
10. Процент благоприятных/неблагоприятных отзывов пользователя.

11. Число повторений неудавшихся действий.

12. Число успешных и неуспешных задач или их составных частей.

13. Сколько раз интерфейс вводил пользователя в заблуждение?

14. Число хороших и плохих элементов интерфейса, которые вспомнил пользователь.

15. Число невостребованных (но доступных) действий.

16. Число возвратов назад в действиях.

17. Число пользователей, отдавших предпочтение вашей системе.

18. Сколько раз пользователи сталкивались с неясной ситуацией, для решения которой они были вынуждены тратить время?

19. Сколько раз пользователи отрывались от работы?

20. Сколько раз пользователи утрачивали контроль над системой?

21. Сколько раз пользователи выражали удовлетворение (либо неудовлетворение, раздражение) от работы с системой?

На основе таких перечней определяются наихудшие (минимально допустимые) и наилучшие (желательные) значения выбранных оценок для конкретной разработки и намечаются направления совершенствования ПИ. Возможные основания для определения значений (наилучших и наихудших) метрик юзабилити следующие:

1) разрабатываемый или предшествующий вариант ПИ;

2) вариант(ы) ПИ в конкурирующей(их) системе(ах);

3) характеристики выполнения задачи без разрабатываемого ПО;

4) абсолютная шкала;

5) ваш собственный прототип;

6) значения метрик юзабилити в предшествующих вариантах и/или аналогах;

7) значения метрик юзабилити при выполнении каждой отдельной задачи;

8) структурирование различия между лучшими и худшими значениями метрики юзабилити на основе анализа деятельности пользователей.

Самая сложная проблема в юзабилити – определение тех показателей (т.е. той метрики), которые будут использоваться для оценки величины юзабилити и установления лучшего/худшего варианта и также планируемого/текущего его уровня, причем в самом начале разработки ПО.

В настоящее время в оценке юзабилити преобладает эмпирический подход, метрика же юзабилити служит инструментом количественного выражения в основном частных и эмпирических оценок. Дело в том, что для окончательной оценки юзабилити необходима целостная деятельность пользователя с реальной системой, но пока она не разработана до конца, такая деятельность невозможна, поэтому использование в процессе разработки частных оценок неизбежно.

Конечно, метрика юзабилити содержит допущения, поэтому процесс разработки как ПО в целом, так и ПИ в частности итеративен. Итеративность предполагает, что каждый шаг вперед в разработке поверяется на предмет юзабилити. Этой цели служат прототипы, о которых подробнее говорится в разд. 5.2.

Попытка создания более полной таксономии показателей, составляющих понятие юзабилити, предпринята специалистами корпорации «Oracle». На основе того же стандарта ISO 9241 и многочисленных публикаций на эту тему выделены 64 показателя, затем методами факторного и кластерного анализа экспертных оценок были оценены вес каждого показателя и величины их взаимных связей. Это позволило ранжировать такие показатели и представить их в виде иерархической структуры, групп разных уровней. При этом степень близости между показателями, входящими в группы нижнего уровня, наибольшая, сами же такие группы объединены как целое также по близости с другими группами и т.д. по иерархии уровней объединения (рис. 5.1).

На рис. 5.1 по горизонтальной оси расположены величины расстояний между показателями, таким образом, чем больше расстояние, тем меньше степень близости. Величины расстояний (linkage distance) от 1 до 3 образуют первую группу; от 4 до 6 – вторую группу; от 7 до 9 – третью группу; от 10 до 14 – четвертую; от 15 до 25 – пятую. Все группы разделены на

две большие области: юзабилити и неюзабилити. В области юзабилити выделены 3 степени близости показателей к этому понятию: основная (или стратегическая) юзабилити – 1-я группа; вторичная (или тактическая) юзабилити – 2-я группа; третичная (общая) юзабилити – 3-я группа.

1. Основная, или стратегическая, юзабилити, которая определяется такими обобщенными понятиями, как согласованность (непротиворечивость), организованность, легкость и интуитивность. Эти показатели субъективно наиболее адекватно отражают содержание понятия юзаби-лити, являются сердцевиной этого понятия. Следовательно, именно они должны обеспечиваться в первую очередь.

2. Вторичная, или тактическая, юзабилити определяется следующими понятиями: эффективностью, степенью знакомства, контролируемостью, завершенностью, полезностью.

3. Третичная, или общая, юзабилити ассоциируется с такими понятиями, как ожидаемость, естественность, гибкость и релевантность, дружественность.

Видно, что понятия третьей группы имеют более обобщенное, размытое содержание, чем основные и второй группы. В то же время большинство используемых понятий отнюдь не являются независимыми, количество и плотность пересечений смыслов между ними столь велики, что принимать это как какую-то шкалу затруднительно.

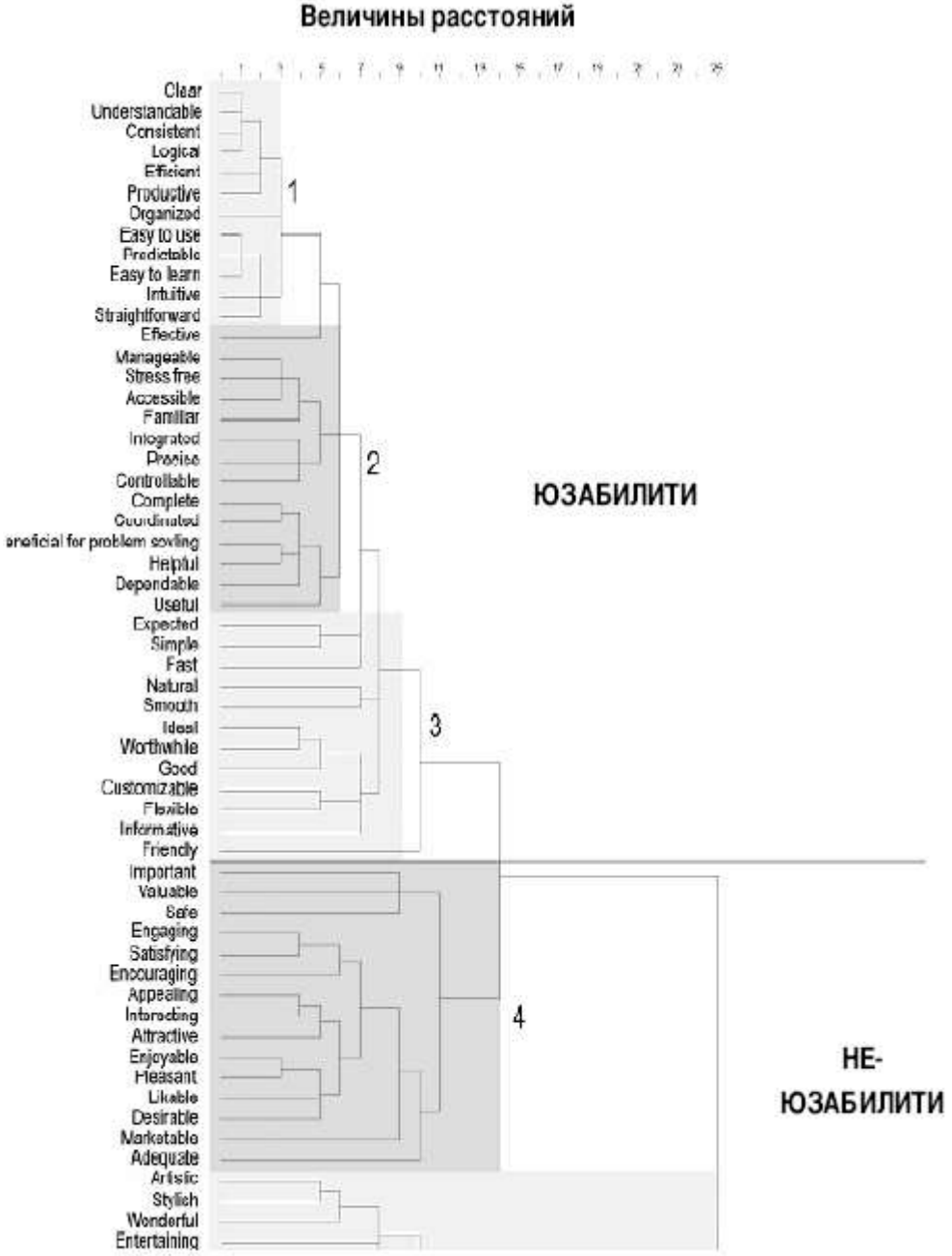

Рис. 5.1. Таксономия юзабилити с величинами ассоциативных связей

К области «неюзабилити» относятся четвертая и пятая группы. В них можно выделить шесть понятий: важность, оцениваемость, удовлетворенность, безопасность, продаваемость, адекватность.

Свойства, входящие в эту группу, связаны с привлекательностью, приятностью, желанностью, удовлетворенностью и т.п. Согласно результатам обработки субъективных оценок они имеют весьма отдаленное отношение к юзабилити и поэтому не могут быть включены в содержание. Как видно из рис. 5.1, понятия, характеризующие стиль и другие свойства, тоже входят в область «неюзабилити».

Результаты исследований позволили предложить следующее определение юзабилити: «Юзабилити есть восприятие того, насколько согласован, эффективен, продуктивен, организован, легок в использовании, интуитивен и прост интерфейс».

В целом провести юзабилити-тестирование несложно. Нужно выяснить, как несколько пользователей работают с исследуемым продуктом. Как правило, в процессе тестирования производятся индивидуальные наблюдения за каждым из пользователей, которые выполняют ряд специально подготовленных заданий.

При этом собираются все возможные данные, касающиеся того, как именно пользователь выполняет задания, – например, сколько времени уходит на выполнение каждого задания или сколько ошибок совершается в процессе работы. Затем производится анализ собранной информации с выявлением тенденций и закономерностей.

Для проведения юзабилити-тестирования нужно иметь несколько пользователей средней квалификации плюс прототип (при наличии основательного бюджета можно, конечно, развернуться, например купить прибор, фиксирующий направление взора пользователя и длительность зрительных фиксаций, различную психофизиологическую аппаратуру для измерения разных видов напряженности работы и т. д.).

Существует распространенное заблуждение, что тестирование, в том числе юзабилити-тестирование, позволяет решить все проблемы интерфейса. С помощью тестирования можно определить только слабые места интерфейса, но почти невозможно обнаружить сильные, поскольку они пользователями просто не замечаются, и совсем уж невозможно определить способы улучшения. Имеется множество скептических оценок возможностей юзабилити-тестирования, особенно когда это касается web-сайтов. Этот скептицизм основан на том, что на сегодняшний день культура тестирования довольно низкая, поэтому тесты редко приносят какую-то пользу. Хуже того, неправильное использование этого важного метода может подорвать и то позитивное, что в нем есть.

В частности, у многих вызывает сомнение, что юзабилити-тестирование требует особого непредвзятого специалиста, т.е. что люди «со стороны» объективнее вас самих оценят то, что делают пользователи. Многие специалисты считают, что тестирование необходимо выполнять с полноценным привлечением всей команды разработчиков. Это даст возможность каждому взглянуть на свою работу с точки зрения пользователя. При этом не стоит ограничиваться лишь командой разработчиков. В тестировании должны принимать участие и другие работники компании. Всякий, кто так или иначе связан с тестируемым продуктом, должен быть привлечён к работе хоть на каком-то из этапов. Если в компании люди увидят, что у реальных пользователей возникают с продуктом реальные проблемы (которые можно решить), они поймут и значение юзабилити-тестирования, и важность выделения времени и ресурсов, которые позволяют создать конкурентоспособный продукт.

Проведение юзабилити-тестирования совместно с командой разработчиков способствует снижению затрат и времени на разработку. Дорогостоящие консультанты больше не нужны, а это позволяет проводить тестирование чаще, согласно принятому в коллективе порядку работ. Такой подход позволяет юзабилити-тестированию превратиться из простого теста, ориентированного на разовый отчет, в многократно повторяемый тест, *реально влияющий на разработку.*

При создании нескольких простых страниц-эскизов дайте паре ваших пользователей опробовать их. Сделали простой прототип? Представьте его группе из нескольких

пользователей. Такой подход к делу позволяет вносить изменения тогда, когда они наименее затратны. Гораздо проще изменить что-то фломастером на белой доске, нежели в готовом эскизе; легче изменить эскиз, чем прототип; и, наконец, легче изменить прототип нежели готовый продукт; по некоторым данным, соотношение затрат как 1:10:100.

Дизайнеры иногда беспокоятся, что изменения в сайте, вызванные результатами тестирования, ограничивают полет их мысли. Однако, когда регулярные тестирования сделаны частью процесса разработки, дизайнеры, наоборот, получают толчок к новым идеям, улучшающим дизайн на основе реальных потребностей реальных людей. Ничто не приносит дизайнеру больше удовлетворения, чем пользователь, довольный его продуктом.

Юзабилити-тестирование очень важно в процессе разработки ПИ, поэтому мы рассмотрим его подробно.

5.1.2. Подготовка к тестированию

Прежде всего при тестировании надо очень ясно поставить вопрос тестирования, причем в терминах, позволяющих однозначно ответить, является ли полученный результат ответом на этот вопрос. Возможно, вас интересует, почему после выхода последней версии продукта участились звонки в службу техподдержки. Или ваше положение на рынке пошатнулось, и вы желаете знать, не вызвано ли это тем, что продукция конкурента более удобна в использовании. Примером плохой постановки задачи может служить такая постановка: «Удобен ли мой продукт?». Хорошая постановка задачи могла бы выглядеть так: «Насколько сложной для новичка является процедура заполнения налоговых форм с помощью данного продукта? Предоставляет ли онлайн-система подсказки достаточное количество информации о налоговом кодексе? Эта информация изложена простым, доступным для понимания языком, а не на юридическом жаргоне?».

Перед тестированием надо отчетливо понимать характеристики потенциальных пользователей, которые и должны стать участниками тестирования. Это необходимо, потому что когда вы займётесь непосредственно подбором участников, очень важно будет знать, кто именно вам нужен: новички, эксперты или опытные пользователи, мужчины, женщины, или же вы нуждаетесь в представителях обоих полов; также важен возраст пользователей. Кто является целевой группой пользователей вашего продукта? Если вы тестируете дисплеи реактивных истребителей, вам ни к чему пропускать через ваши тесты толпы тинэйджеров. Если ваш продукт – автомат для продажи газировки, в таком случае те же ти-нэйджеры придутся как нельзя кстати. Как было сказано в разд. 3, основная работа по выявлению характеристик (профилей) пользователей делается ещё на этапе постановки задачи. Подробнее описание моделей пользователей приведено в разд. 3.

Далее следует определить структуру тестирования, его, так сказать, композицию, которая описывает, как вы будете проводить отдельные тесты, в каком порядке они будут выстроены, чтобы исключить из рассмотрения и дальнейшего анализа все переменные, не представляющие интереса. Предположим, что вы тестируете программный продукт, связанный с налогообложением. Нужны ли вам пользователи, уже использовавшие ваш продукт в работе и уже обладающие некоторыми сведениями и навыками, касающимися именно вашего продукта? Возможно, вы предпочтёте разбить участников тестирования на две группы, разделив их на новичков и пользователей, уже имеющих некоторый опыт работы с продуктом.

После этого разрабатываются задания, которые будут предложены участникам тестирования. Эти задания, несомненно, должны быть основаны на тех задачах, которые пользователи решают с помощью вашего продукта в процессе его нормального использования. Укажите все, что понадобится вам для того, чтобы определить сценарий теста: состояние автомата, машины или компьютера, экраны, документацию, другие средства помощи и подсказки, которые должны присутствовать. Также укажите, каким образом определяется успешное завершение выполнения каждого задания, – например, если пользователь успешно сохранит отредактированный документ или выполнит некоторую производственную операцию, достигнув определенного законченного результата.

Перед тестированием надо также определить, какой вспомогательный инструментарий

вам нужен, и приобрести его. Инструментарий может включать в себя устройства, используемые в процессе проведения тестов, такие, как видеокамеры для записи поведения пользователей, преобразователи развертки для записи того, что происходит на экранах мониторов, диктофоны и записывающая аудиоаппаратура для протоколирования вербального общения и записи вербальных протоколов, односторонние зеркала, позволяющие наблюдателям и экспериментатору оставаться невидимым для участников тестирования, и т.д. Можно собрать много полезной информации, пользуясь простыми любительскими видеокамерами или даже вообще без видеозаписей.

Необходимо составить список пользователей, из которого будут подбираться участники для каждого теста. Количество пользователей должно быть достаточным, чтобы создать выборку в требуемой для теста пропорции характеристик пользователей (скажем, опыта, навыков и демографических характеристик), поскольку в противном случае неучтенные факторы могут повлиять на полученные данные. Профили пользователей, которые вы определили ранее, помогут вам создать модель типового пользователя вашего продукта. Например, приборной доской реактивного истребителя могут пользоваться не только сами пилоты, но и обслуживающие рабочие, механики, инструкторы и диагностирующий персонал. Тем не менее для тех целей, которые преследуете вы (например, установить, позволяет ли использование дисплея радара ближайшего окружения самолета избежать столкновений в процессе полета на спине), интерес представляет рассмотрение только одного сегмента всей выборки – в нашем примере это пилоты.

5.1.3. Проведение тестирования

Многие люди чувствуют себя неуютно, когда попадают в лабораторию, где им приходится выполнять задания, зная, что затраченное на это время замеряется и все их ошибки записываются для дальнейшего анализа, поэтому сделать так, чтобы пользователь чувствовал себя комфортно и спокойно, очень сложно. Важно подчеркнуть, что тестированию подвергается ПИ, а не пользователи и им не стоит чувствовать себя под давлением. Обязательно следует поблагодарить пользователей за участие в тестах. Надо объяснить участникам, что они могут в любой момент остановить тестирование, отлучиться в туалет или сделать перерыв, если это кому-либо потребуется.

Нередко тестирование требует от участников предварительного соглашения о неразглашении информации и подтверждения их согласия на запись хода и результатов тестирования.

Полезно за день до проведения тестирования попробовать выполнить самостоятельно то, что должны будут делать его участники, дабы убедиться, что можно справиться в отведенное время.

В процессе тестирования следует оберегать чувство собственного достоинства пользователей. Надо быть внимательным и тактичным с ними. Если пользователь застрял где-то, его следует ободрить, а по завершении тестирования обязательно поблагодарить (искренне) и дать знать, что его участие было очень ценным и полезным для вас. Вообще, в процессе тестирования экспериментатор должен проявлять доброжелательность и уважение к пользователям. Большинство лабораторий вручает участникам небольшие подарки по завершении тестирования: кофейную кружку, футболку или бесплатное программное обеспечение. В большинстве случаев еще не раз потребуется привлечь этих пользователей для тестирования, поэтому очень важно, чтобы они остались довольны своим участием.

При проведении тестирования главное – наблюдать за ходом мысли пользователей. Когда нет уверенности, что вам понятно, о чем они думают, можно спросить у них об этом. Например, если пользователь смотрит на экран в течение 10 с, можно спросить у него: на что вы смотрите или о чем вы задумались?»

Самая главная задача тестирования – понять, какие на каждом шагу у пользователей возникают ожидания и насколько программа соответствует этим ожиданиям. Если пользователь говорит: «Я не знаю, что делать дальше», вам нужно спросить: «А как вы думаете, что бы вы могли сделать?» или «А что бы вы сделали, если бы были дома?» Когда пользователь уже готов

щелкнуть мышью, можно спросить у него, что он ожидает увидеть. После того как он щелкнет мышью, узнайте, действительно ли результат оправдал его ожидания.

Если вы уже выяснили какой-то вопрос, не тратьте на него время с другими пользователями. Например, если первые два пользователя безнадежно застряли в одном и том же месте и вам ясно, в чем там проблема и как ее решить, то не заставляйте и третьего пользователя без нужды возиться с ней. Как только он дойдет до этого места, объясните ему, что к чему и как продолжить работу дальше.

Следует помнить, что нельзя отвлекать пользователей или оказывать на них давление, но в то же время необходимо выяснить, что они в действительности думают (кстати, они и сами могут не до конца это понимать). Наблюдая за пользователем, который ни во что не «въезжает», можно узнать много полезного. Более опытный пользователь обладает лучшими навыками, позволяющими ему разобраться с проблемой «как получится», и вы можете даже не заметить, что на самом деле он не понимает, как «это» работает.

После каждого тестирования полезно делать короткие записи о том, что вам запомнилось. Если вы не сделаете этого до начала следующего тестирования, то потом вам будет очень трудно вспомнить важные детали.

Пользователи часто подсказывают решение какой-либо проблемы. Такие подсказки могут навести на идеи, о которых вы раньше думали, но по каким-то причинам отклонили их. Иногда случается, что в проект были внесены какие-то изменения (например, было решено применить другую технологию или сместились коммерческие приоритеты) и становится полезным тот метод или подход, который был отклонен вначале.

5.1.4. Анализ полученных данных

После того как задания выполнены и тестирование завершено, следует обсудить результаты с его участником. Это необходимо для того, чтобы получить дополнительную информацию, касающуюся того, о чем думал пользователь в то время. Одним из путей анализа событий является их восстановление и обсуждение с участником тестирования. Кроме этого, можно просто спросить участника, что из случившегося во время тестирования показалось ему заслуживающим внимания.

По окончании тестирования каждый наблюдатель и помощник должен как можно быстрее написать небольшой отчет об основных проблемах, которые были замечены во время тестирования, а также изложить свои мысли по поводу способов их устранения. Не надо писать полных и развернутых отчетов. Пусть это напоминает скорее резюме. В идеале все участники группы по разработке должны прочитать эти отчеты (или хотя бы пробежать их глазами), поэтому они не должны по объему превышать 1-2 страницы.

На последующем собрании рабочей группы следует обсудить два вопроса:

• Каковы проблемы, с которыми сталкивались пользователи, и какие из них должны быть исправлены?

• Каковы возможные решения отобранных проблем?

При анализе полученных данных в первую очередь следует искать крупные проблемы. Найти такие проблемы проще, поскольку они становятся заметны еще при просмотре заметок, сделанных во время наблюдения за пользователями. Если каждый участник сталкивался с проблемами при использовании определенного пункта меню, очевидно, что дизайн этого пункта нуждается в пересмотре.

Данным, касающимся производительности, таким, как частота ошибок и время выполнения заданий, оценка дается с помощью статистического анализа. В основном такой анализ посвящен нахождению среднего значения и стандартного (среднего квадратичного) отклонения, а также проверке достоверности полученных данных.

Наблюдения за действиями пользователей и запись их мнений как во время тестирования (используя метод записи «мыслей вслух» или задавая вопросы), так и до или после него проводятся с использованием анкет и опросных листов. Большинство таких опросников имеет строение, позволяющее количественно оценить мнения, используя численную шкалу, и полученные количественные данные анализировать также с помощью статистических методов.

5.1.5. Методики тестирования

В настоящее время очень многие тестирования проводятся на базе записи «мыслей вслух» в сочетании с каким-либо методом измерения производительности. Измерения производительности в определенной степени полезны для исследований, однако информация, получаемая при записи «мыслей вслух», способна гораздо быстрее отражаться на изменениях в продукте, поскольку она не нуждается в предварительном обобщении и статистическом анализе.

Здесь мы кратко рассмотрим следующие методики тестирования:

- метод фокусных групп;
- проверку посредством наблюдения за пользователем;
- запись «мыслей вслух»;
- проверку качества восприятия;
- измерение производительности;
- карточную сортировку;
- RAFIV-метод.

Метод фокусных групп (или целевых групп – target groups) заключается в опросе специально отобранной группы пользователей. Основное достоинство фокусных групп состоит в том, что они позволяют выявлять спонтанные реакции и идеи и оценивать отношение к этим идеям группы в целом. В исследование, которое обычно продолжается около 2 ч, вовлекаются от 6 до 9 пользователей.

Метод фокусных групп подходит для того, чтобы быстро получить диапазон мнений и оценок пользователей по поводу тех или иных вещей. Он очень полезен для определения потребностей и предпочтений вашей аудитории и подходит для проверки того, насколько актуальна и востребована программа и насколько предлагаемый ПИ привлекателен для пользователей. Кроме того, это хороший способ для подбора названий функции и опций, а также выяснения, что люди думают о ваших конкурентах.

Как правило, участники группы воспринимают происходящее как относительно свободный процесс, но ведущий группы должен иметь предварительный сценарий работы и следить, чтобы групповая дискуссия не выходила за рамки обсуждаемой проблемы. Кроме того, необходимо добиваться равного участия в дискуссии всех членов группы.

Сбор детальной информации при этом методе затруднен из-за относительной стихийности группового процесса, поэтому рекомендуется иметь несколько фокусных групп, состоящих из репрезентативных пользователей.

С помощью этой методики вы можете узнавать те вещи, которые нужно было бы знать на ранних этапах, еще до начала непосредственной разработки. Конечно, этот метод можно применять и позже, например для уточнения деталей при периодическом тестировании.

Данная методика имеет и недостаток, связанный с неполной определенностью того, что следует считать фокус-группой или целевой аудиторией. Честно было бы говорить только об определенном круге людей, образующемся *по факту* приобретения продукта или услуги. Очень часто просто прибегают к опросу коллег или тех, кто оказался под рукой. Но во всех случаях основной недостаток фокус-групп состоит в том, что для оценки совершенно нового продукта они, как правило, не годятся. Потенциальные потребители скажут, например, что на коробке шоколада должны присутствовать чашка чая с золотой каемкой, свеча и роза. Все, что можно узнать, спрашивая мнение людей, – это то, что они знают сегодня, до того, как новый продукт появился на рынке. Поэтому почти все интересные и успешные вещи появляются на свет без участия фокус-групп.

По свидетельству А. Лебедева (www.artlebedev.ru/kovodstvo/126/), во многих известных компаниях не пользуются услугами фокус-групп, например в компании «Apple». Вместе с тем в маркетинге это одна из базовых стратегий, играющая роль универсального обоснования всякого решения.

Проверка посредством наблюдения за пользователем – один из самых простых видов тестирования. Пользователю дается задание, он его выполняет, его действия фиксируются для

дальнейшего анализа на камеру либо какой-нибудь программой записи состояния экрана (например, Lotus ScreenCam). Метод исключительно полезен для выявления неоднозначности элементов интерфейса. Поскольку каждая неоднозначность приводит к пользовательской ошибке, а каждая такая ошибка фиксируется, обнаружить их при просмотре записанного материала очень легко. Впоследствии можно подсчитать количество ошибок и сделать соответствующие выводы. Кроме того, если замерять время выполнения задания, можно оценить и производительность работы пользователей.

Мыслим вслух. Запись или протоколирование «мыслей вслух» – очень распространенная методика, применяемая при юзабилити-тестирова-нии. Во время тестирования, пока участник выполняет то или иное задание в рамках своего сценария, экспериментатор просит его проговаривать мысли, чувства и мнения, возникающие в процессе взаимодействия с продуктом. Комментарии записывают, а затем анализируют. Здесь в первую очередь необходимо предоставить участнику сам тестируемый продукт (или прототип его интерфейса) и сценарий – список заданий, которые ему необходимо выполнить. Попросите участников в процессе выполнения заданий подробно сообщать все, что возникает у них в голове, пока они работают с интерфейсом продукта.

«Мысли вслух» позволяют вам понять, как пользователь подходит к интерфейсу и какими соображениями он руководствуется, используя этот интерфейс. Если последовательность шагов, которую пользователю потребуется пройти для выполнения задания, отличается от той, что он себе представлял, то, возможно, интерфейс излишне вычурный. Может оказаться, например, что терминология, которой участник пользуется для объяснения идеи или функции, должна быть внедрена в дизайн продукта и использована при проектировании или, по меньшей мере, при написании документации.

Эта методика очень дешевая, весьма информативная и может использоваться на любой стадии разработки ПИ.

Впрочем, не следует всецело доверять информации, полученной путем регистрации «мыслей вслух». Дело в том, что далеко не всегда озвучиваются сами реальные трудности, чаще же – представление о том, какими они должны бы быть. Да и навыки, если их требуют проговаривать, могут пострадать (говорят, сороконожка разучилась ходить, когда стала задумываться, как она это делает).

Проверка качества восприятия – тест позволяет определить, насколько легко для пользователя обучиться интерфейсу. Поскольку существует разница между понятиями «видеть» и «смотреть», а запоминается только то, что увидено, необходимо быть уверенным в том, что пользователь увидит если не все, то уж хотя бы все необходимое. А значит, запомнит, благодаря чему в будущем ему не придется обшаривать меню в поисках «чего-то такого, что, я точно знаю, где-то здесь есть».

Методика достаточно проста. Пользователю дается задание, связанное с каким-либо отдельным диалоговым окном. Он его выполняет. Через несколько минут его просят нарисовать (пускай даже грубо и некрасиво) только что виденное им окно. После этого рисунок сравнивается с оригиналом.

Разумеется, пользователь запоминает только то, что ему кажется актуальным в процессе работы с окном (плюс еще что-нибудь из того, что ему показалось интересным, да и то не всегда). Например, пользователь, которому нужно сменить шрифт абзаца на Arial, из всего диалогового окна выбора шрифта в MS Word запоминает только часть элементов управления. При этом он помнит, что помимо них были и другие элементы, но точно вспомнить их, как правило, не может.

Измерение производительности – методика, направленная на получение четких количественных данных. В большинстве случаев последние представлены в форме метрик производительности. Иногда в процессе проектирования ПИ бывают важны такие вопросы: сколько времени занимает выделение блока текста с помощью клавиатуры, мыши или трекбола? как расположение клавиши «Backspace» влияет на частоту появления ошибок? Цели в таком случае могут формулироваться в виде условий, например: «пользователь должен иметь возможность установить соединение с Интернетом без ошибок или по бесплатной линии» или «основная задача должна выполняться за время, не превышающее часа для 75%

пользователей». Эти условия устанавливаются на базе изначального тестирования предыдущей версии или продуктов-конкурентов. Методика предполагает, что должна быть поставлена цель, определены задачи, спроектированы тесты и непосредственно поставлен эксперимент.

При измерениях производительности необходимо учитывать следующие моменты:

1. Критерий выполнения задачи должен быть количественно определен. Дело в том, что для измерения производительности необходимо, чтобы задачи имели четкий критерий в виде численно выраженной величины. К примеру, можно задаться вопросом: что более эффективно – кнопки на управляющей панели или «горячие» клавиши? Ответ на сформулированный таким образом вопрос можно получить при помощи тестирования двух интерфейсов: одного, ориентированного на «горячие» клавиши, а второго – на панели с кнопками. Производительность каждого пользователя определяется при помощи замеров времени, потраченного им на выполнение каждого задания, и допущенных пользователем ошибок.

2. Структура эксперимента действительно важна. Поскольку задачей тестов с измерением производительности является получение корректных количественных данных, структура эксперимента также должна быть корректной. Количественные тесты предполагают, что изменения независимых переменных (таких, как наличие кнопок на панели или доступность «горячих» клавиш) отразятся на зависимых (в нашем случае – на времени, затрачиваемом на запуск команд, использующих одну или две опции). Тем не менее, если дополнительные факторы будут внесены в структуру теста, этот эффект может быть искажен и эксперимент окажется статистически недостоверен, поскольку будет испорчен влиянием неучтенных факторов. Эксперимент должен быть спроектирован с учетом возникновения возможных искажающих факторов, чтобы устранить все, что может повлиять на его результаты.

3. Данные не сообщают всех подробностей. По ряду причин тестирования, ориентированные исключительно на сбор результатов измерений производительности, не так распространены, как это могло бы показаться. Такие тестирования требуют очень скрупулезно спроектированных тестов и значительных ресурсов. У большинства компаний на такие тестирования нет ни времени, ни денег. Кроме этого, полезность получаемых с их помощью результатов зачастую очень невелика. Действительно ли так важно, что использование «горячих» клавиш вместо панелей с кнопками дает выигрыш в полсекунды? Возможно, важно, если вы разрабатываете программное обеспечение для центров обработки заказов и экономия половины секунды на каждом звонке, помноженная на тысячи операторов по всей стране, может сократить годовые расходы на миллионы долларов. Но для большинства офисных приложений экономия в половину секунды не представляет реальной важности.

По всем этим причинам измерение производительности используется на начальных этапах проектирования ПИ для задания контрольных замеров. Также оно используется на протяжении всего проектирования для измерения того, насколько далеко удалось продвинуться относительно этих контрольных замеров.

Измерение скорости работы производится с помощью системы GOMS (Goals, Operators, Methods and Selection Rules – цели, операторы, методы и правила их выбора).

Идея метода проста – все действия пользователя можно разложить на составляющие (например, взять мышь или передвинуть курсор). Ограничив номенклатуру этих составляющих, можно замерить время их выполнения на массе пользователей, после чего получить статистически верные значения длительности этих составляющих. После этого предсказание скорости выполнения какой-либо задачи или, вернее, выбор наиболее эффективного решения становится довольно простым делом – нужно только разложить эту задачу на составляющие, после чего, зная продолжительность каждой составляющей, все сложить и узнать длительность всего процесса.

Отметим, что полноценным российским аналогом метода GOMS является операционно-психофизиологический метод (ОПФ-метод), разработанный Г.М. Зараковским еще в конце 60-х годов прошлого века (подробнее см. в разд. 4.2).

Хотя для различных пользователей время выполнения того или иного действия может сильно отличаться, исследователи обнаружили, что для большей части задач, включающих использование клавиатуры и мыши, вместо измерений для каждого пользователя можно применить набор стандартных величин. Этот набор был получен экспериментально для

типовых действий. В табл. 5.2 приводятся значения времени для некоторых типовых действий.

Таблица 5.2

Среднее время некоторых типовых действий

Тип	Действие	Длительность, с	Комментарии
К	Нажатие на клавишу	0,28	Включая клавиши «Alt», «Ctrl», «Shift»
М	Нажатие на кнопку мыши	0,1	
П	Перемещение курсора мышью	1,1	Время, затрачиваемое на перемещение курсора, зависит как от дистанции, так и от размера цели, поэтому эта величина является компромиссной
В	Перемещение руки с мыши на клавиатуру или наоборот	0,4	
Д	Ментальная подготовка	1,2	Время, необходимое пользователю для того, чтобы подготовиться к следующему шагу
Р	Время реакции системы	От 0,1 до бесконечности	Для базовых операций, таких, как работа с меню, это время можно не засчитывать

На практике указанные значения могут варьировать в широких пределах. Для опытного пользователя, способного печатать со скоростью 135 слов/мин, значение К может составлять 0,08 с, для обычного пользователя, работающего со скоростью 55 слов/мин, – 0,2 с, для среднего неопытного пользователя, работающего со скоростью 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 символа в секунду. Значение К включает в себя и то время, которое необходимо пользователю для исправления сразу замеченных ошибок.

Более сложным является определение точек, в которых пользователь остановится, чтобы выполнить бессознательную операцию, т.е. интервалов М. Вот основные правила, позволяющие определить, в какие моменты будут проходить такие операции:

1. Операторы М следует устанавливать перед всеми операторами К, а также перед всеми операторами Р, предназначенными для выбора команд. Однако перед операторами Р, предназначенными для указания на аргументы этих команд, ставить оператор М не следует.

2. Если оператор, следующий за оператором М, является полностью ожидаемым, то этот оператор М может быть удален. Например, если вы перемещаете мышь с намерением нажать ее кнопку по достижении цели движения, то в соответствии с этим правилом следует удалить оператор М, устанавливаемый по первому правилу. В этом случае последовательность РМК превращается в РК.

3. Если строка вида МКМКМК принадлежит оперативной единице мышления, то следует удалить все операторы М, кроме первого. Оперативной единицей является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент. Например, Y, перемещать, Елена Троянская или 4567,34 являются примерами

когнитивных единиц.

4. Если оператор К означает лишний разделитель, стоящий в конце когнитивной единицы (например, разделитель команды, следующий сразу за разделителем аргумента этой команды), то следует удалить оператор М, стоящий перед ним.

5. Если оператор К является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор М, стоящий перед ним. (Добавление разделителя станет привычным действием, и поэтому разделитель станет частью строки и не будет требовать специального оператора М.) Но если оператор К является разделителем для строки аргументов или любой другой изменяемой строки, то оператор М следует сохранить перед ним.

6. Любую часть оператора М, которая перекрывает оператор Р, учитывать не следует.

Широкая изменчивость каждой из представленных метрик 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 сравнительную оценку каких-то двух интерфейсов (если не целиком, то каких-то их элементов).

Карточная сортировка – это классификационный метод, при котором пользователи сортируют различные элементы разрабатываемого ПИ по нескольким категориям.

Для проведения карточной сортировки создается список параметров, которые предполагается подвергнуть классификации, после чего каждый из указанных параметров выписывается на отдельной карточке.

Карточки предъявляются пользователям, которых просят сгруппировать их наиболее логичным, по их мнению, образом. Полученную в результате карточной сортировки информацию используют для организации пользовательского интерфейса.

RAFIV -метод – это когнитивный метод юзабилити-тестирования, упрощенный и предназначенный для программистов и инженеров, не знакомых с основами когнитивной психологии. Он базируется на том, что большинство проблем ПИ связано с недостатками в представлении пользователю информации, которая должна последовательно помогать решать задачи (двигаться к цели), служить «поводырем», т.е. вести пользователя за собой. Метод предполагает анализ пять когнитивных этапов решения задач, первые буквы названия которых и образуют аббревиатуру RAFIV (Reformulate, Access, Format, Insert, Verify).

Цель анализа на каждом этапе – установить, достаточно ли предъявленных данных для перехода к следующему этапу. Предполагается, что пользователи, приступая к каждой задаче, формулируют (или переформулируют) свои цели в форме определенных характеристик объектов. Далее они должны получить эти характеристики, последовательно пройдя все этапы процесса преобразования данных с помощью меню либо какого-то иного множественного выбора. Если задача предполагает затем ввод требуемых данных, они должны сделать такой ввод в необходимом формате. На заключительном этапе пользователи должны проверить правильность своего ввода и соответствие полученных результатов своим целям.

Таким образом, выделяют пять этапов выполнения задачи и определяют, какая когнитивная нагрузка требуется на каждом этапе. Эти пять этапов таковы:

1. *Формулирование или переформулирование своих намерений* (задачи, которую намеревается выполнить пользователь) в форме определенных характеристик объектов (Reformulate the task in terms of features and data).

2. *Достижение требуемых характеристик объектов* посредством меню, разного рода ссылок либо иных средств (Access the feature that will accomplish the task).

3. *Приведение характеристик к виду, требуемому системой.* Перед вводом данных пользователь должен привести их к нужному виду (Format the data according to the system's requirements).

4. *Ввод характеристик, приведенных к требуемому виду* (Insert the formatted data into the system).

5. *Получение информации о том, что данные были введены и правильно восприняты системой* (Verify and monitor that data were entered correctly).

После того как произведена декомпозиция и классификация всех задач пользователя, подсчитывается для каждой стадии *число потребовавшихся воспроизведений материала из памяти и опознаний.* Затем наиболее внимательно анализируются случаи воспроизведения каких-то данных из памяти и намечаются пути для их уменьшения, что обычно равнозначно улучшению ПИ. К примеру, тот или иной шаг содержит воспроизведение, если при его выполнении необходимо произвести вычисления в уме или помнить, где в интерфейсе спрятана та или иная информация, или если необходимо понимание особенностей алгоритма выполнения каких-то преобразований и т.п. Опознание же предполагает, что интерфейс содержит ясную визуализацию, обеспечивающую правильные действия, или ясную и своевременную информацию о степени продвижения к цели.

Интересно сравнить RAFIV-метод и близкий к нему метод GOMS-анализа. Ведь содержание стадий RAFIV-метода в значительной мере является результатом GOMS-анализа. Различие же состоит в том, что GOMS-анализ предполагает наличие достаточно хорошо сформированных навыков работы у пользователя и представляет лишь способ регистрации того, как пользователь работает с новой системой. RAFIV-метод не опирается на сформированные базовые навыки пользователя, предполагая, что его обучение работе с системой происходит в процессе следования за элементами интерфейса. GOMS-анализ не дает ответа на вопрос, где и почему ошибка могла бы произойти, а RAFIV-метод позволяет выделить действия, которые вызывают наибольшие трудности. Кроме того, RAFIV-метод ориентирован прежде всего на оценку умственной нагрузки, непосредственно «наложенной» на интерфейсные элементы, а GOMS-анализ основан на декомпозиции внешних действий пользователя. Имеются данные об успешном применении RAFIV-метода при тестировании не только ПИ, но и электронной аппаратуры.

В целом при выборе метода и оценке результатов юзабилити-тестиро-вания часто приходится конкретизировать некоторые общие ощущения неудовлетворенности ПИ. Для облегчения этого трудного процесса полезно помнить основные принципы превращения сложных задач в простые, сформулированные Д. Норманном в его знаменитой книге «Дизайн повседневных вещей»:

1. Используйте знания, как находящиеся в голове, так и находящиеся во внешнем мире.
2. Упрощайте структуру задачи.
3. Стремитесь визуализировать задачу, соединяя исполнение и оценку.
4. Отображайте структуру задачи.
5. Используйте мощь ограничений, естественных и искусственных.
6. Всегда предполагайте возможность ошибок.
7. Когда все дополнения исчерпаны и более не проясняют задачу, закрепите то, что осталось.

5.1.6. Контрольные списки

Составление контрольных списков и оценка ПИ с их помощью являются составной частью юзабилити-тестирования, поэтому описание этой методики могло бы находиться в разд. 5.1.5. Однако этот метод несколько отличается от описанных выше. Обычно его используют на заключительном этапе работы в дополнение к другим, чтобы не упустить что-то существенное.

Контрольный список (или checklist – «чек-лист») – это документ с перечнем требований к выполняемой работе. Каждому свойству готовой работы присваивается весовой коэффициент (например, отсутствие грамматических ошибок – 4%); работа считается выполненной, если сумма этих коэффициентов больше заданного числа (например, 97%). Благодаря этому исполнитель может не волноваться по поводу качества работы – он должен только соблюдать все изложенные в списке требования, а заказчик имеет возможность быстро проверить качество выполнения работы выборочно по отдельным пунктам.

Наличие контрольного списка увеличивает производительность работы, так как не требуется тратить время на понимание того, что необходимо вообще сделать и что нужно еще сделать. Ответ на оба вопроса уже содержится в контрольном списке. Контрольный список состоит из вопросов, не требующих субъективных ответов (таких, как «навигация сделана

хорошо»). Благодаря этому проверить интерфейс на соответствие указанному списку может кто угодно: никакая специальная подготовка проверяющему не нужна.

Разумеется, обратной стороной абсолютного списка является его заведомая неполнота. Важно учитывать, что контрольный список не может гарантировать высокое качество интерфейса. В лучшем случае он гарантирует отсутствие грубых ошибок.

Использование контрольных списков является эффективным, быстрым и экономичным средством повышения качества пользовательского интерфейса, позволяющим глобально оценить состояние ПИ, тратя на это очень мало времени.

Требования к контрольным спискам (равно как и многие вопросы че-ловеко-компьютерного взаимодействия) освещены на сайте http://www. usethics.ru/lib/software_checklist/index.shtml в статьях В. Головача и других.

Естественно, что в каждом конкретном случае необходимо разрабатывать свой собственный контрольный список, поскольку он должен учитывать специфику создаваемого ПИ и имеющиеся для этого возможности.

Ниже приведен пример-шаблон контрольного списка для некоего пользовательского интерфейса:

ОКНА

1. *Заголовки*

• Заголовки короткие и адекватные содержимому окна.

• Заголовки соответствуют названиям элементов, при помощи которых окна были вызваны.

• Если окно вызывается элементом, не имеющим явного названия, в заголовке окна отражается название экранной формы.

2. *Дизайн окна*

• Тип окна (модальное, немодальное, возможность минимизации/ максимизации) был выбран осознанно, в соответствии с задачами пользователей.

• При проектировании было учтено, при каком разрешении, а также размере монитора и шрифтов будут работать пользователи.

• На растягивающихся окнах есть индикатор растяжимости.

• Управляющие и информационные элементы расположены достаточно далеко друг от друга.

• Информация в окне адекватно сгруппирована (связанные элементы объединены в группы).

• Кнопки находятся в секции, на которую они оказывают непосредственное воздействие. Терминационные кнопки (управляющие окном) расположены либо снизу в ряд, либо справа в колонку.

• Переход от элемента к элементу внутри окна осуществляется сверху вниз и слева направо.

3. *Диалоговые окна*

• В диалоговых окнах отсутствуют меню или инструментальные панели.

• Диалоговые окна открываются не в центре экрана, а в том месте, на которое в данный момент смотрит пользователь.

4. *Строка статуса*

• В строке статуса выводится только информация о текущем состоянии системы и кнопки (не выглядящие кнопками) для функций, предназначенных только опытным пользователям.

• Индикаторы выполнения выводятся в строке статуса. Исключение – окна-мастера, в них индикаторы выполнения можно выводить внутри самих окон.

МЕНЮ

1. *Пункты главного меню*

• Пункты меню имеют адекватные названия.

• Первая буква в названии пунктов заглавная.

• Пункты меню и кнопки, инициирующие другие действия пользователя, обозначены в конце многоточием (…). Примеры: элемент «Сохранить как…» требует многоточия, поскольку

пользователь должен выбрать название файла, а элемент «О программе» многоточия не требует, так как на открывающемся окне нет самостоятельных интерфейсных элементов.

• Все пункты первого уровня активизируют выпадающее меню.

• Каждому пункту меню назначены ALT-комбинации (общепринятые «горячие» клавиши выделены подчеркиванием). Эти комбинации должны быть по возможности стандартными.

• Недоступные в данный момент интерфейсные элементы заблокированы, а не скрыты.

2. *Раскрывающиеся меню и элементы меню второго уровня*

• Все элементы начинаются с заглавной буквы.

• Если в меню используются пиктограммы, они расположены слева от названия пункта меню.

• Пиктограммами снабжаются только самые часто используемые элементы.

• Высота меню не превышает размер экрана (меню не нужно прокручивать).

• Пункты меню адекватно сгруппированы. Осмысленно использованы разделители в меню.

• Пункты меню расположены в порядке частоты использования, важности, т.е. связанности выполняемых функций.

• Используется не более двух подуровней меню.

• Каждый пункт меню имеет соответствующую «горячую» клавишу.

• Название пункта меню соответствует названию вызываемого окна.

• Пункты меню, открывающие диалоговые окна, обозначены в конце многоточием (.).

• Недоступные пункты меню обозначены серым цветом шрифта.

3. *Всплывающие меню*

• Каждому пункту всплывающего меню соответствует аналогичный пункт в основном меню.

• Элементы, открывающие вложенные меню, выглядят иначе, чем терминальные элементы.

4. *Контекстные меню*

• Для всех объектов интерфейса есть специфичное для каждого объекта контекстное меню.

• В контекстных меню не более 10 элементов.

• В контекстных меню элементы отсортированы по убыванию частоты их использования.

• Все элементы контекстных меню присутствуют и в других фрагментах интерфейса; нет команд, вызываемых только из контекстнъгх меню.

5. *Инструментальные панели*

• Каждому элементу инструментальной панели соответствует всплывающая подсказка.

• Элементы упорядочены и сгруппированы в соответствии с задачами пользователей.

• Для стандартных действий используются общепринятые графические элементы.

УПРАВЛЯЮЩИЕ ЭЛЕМЕНТЫ

1. *Переключатели (Check boxes)*

• В одном окне используется не более 10 переключателей.

• Переключатели сгруппированы, и каждой группе присвоено название.

• Внутри группы переключатели расположены строго вертикально.

• Переключатели не применяются для частого, оперативного использования.

• В названиях используется только позитивная, утвердительная форма.

• Ни один элемент не называется по-разному в разных местах (интерфейсный глоссарий не просто сделан в явной форме, но и выверен).

• В тексте всех подтверждений дается наименование объекта, над которым совершается подтверждаемое действие.

2. *Командные кнопки (клавиши)*

• Все кнопки, запускающие действия, имеют текст в инфинитивной форме глагола (пример: искать), а не другую часть речи либо форму глагола (пример: готово).

• Кликабельный размер кнопок совпадает с их видимым или логическим размером. Между

кнопками, стоящими рядом, должно быть пустое пространство, щелчок по которому не отрабатывается.

• Кнопки, которые выглядят одинаково, не могут быть в разных состояниях.

• Часто используемые кнопки снабжены не только текстом, но и пиктограммами, а редко используемые – только текстом.

• Кнопки имеют краткие и ясные названия.

• В каждом диалоге используется не более шести кнопок.

• Кнопки, выполняющие в разных диалогах идентичные функции, имеют одинаковые названия.

• Типовые кнопки имеют общепринятые названия и общепринятые «горячие» клавиши.

• Кнопки, вызывающие продолжение диалога во вложенных формах, обозначены многоточием (…).

• Недоступные кнопки имеют соответствующие атрибуты (серый цвет шрифта и т.п.). Недоступные команды не исчезают с экрана, а становятся заблокированными.

• Опасные для пользователя кнопки не являются кнопками по умолчанию.

• Обработка формы запускается не только нажатием на термина-ционую кнопку, но и на клавишу «Enter» на последнем поле этой формы.

• Для наиболее частых элементов управления (включая меню) установлены клавиши быстрого вызова.

• Кнопка «Отмена» всегда самая правая.

• Если в форме есть несколько кнопок, одна является кнопкой по умолчанию. Если кнопка в форме только одна, она не может быть кнопкой по умолчанию.

• Если в окне есть свободное место, наиболее часто используемая терминальная кнопка должна быть больше остальных.

• Кнопки находятся в области ПИ, к которой они непосредственно относятся.

• Терминальные кнопки (управляющие окном) расположены либо снизу в ряд, либо справа в колонку.

• Кнопки, относящиеся ко всему блоку вкладок, расположены за пределами блока.

3. *Редактируемые поля со списком (Combo Box)*

• Имеют функцию автовыбора.

4. *Раскрывающиеся списки*

• Высота выводимого на экран списка ограничена 3-8 элементами.

• Если список содержит более 50 элементов, используется фильтр или режим поиска.

• Если все элементы не умещаются в одном фрагменте списка, автоматически появляется полоса прокрутки.

5. *Группы элементов*

• Каждая группа имеет осмысленное название, помимо рамки отделена от других групп и элементов свободным пространством.

6. *Подписи (Labels)*

• Все элементы имеют подписи.

• Учтена возможность увеличения (уменьшения) длины подписей при использовании large fonts (small fonts).

• Подписи выровнены по левому краю поля (если они находятся над полем).

• Подписи расположены посередине высоты поля (если название находится сбоку).

• Если элемент недоступен, подпись отображается серым шрифтом.

7. *Списки*

• Все списки содержат более одного элемента.

• Любому списку предшествует, по меньшей мере, один абзац текста.

• Если список содержит более 50 элементов, используется фильтр или режим поиска. Если в списке более 50 отсортированных по алфавиту элементов, первыми тремя элементами являются наиболее часто используемые элементы. Они также повторяются на своих алфавитных местах.

• Высота ограничена 3-8 элементами.

• Если все элементы не умещаются, автоматически появляется полоса прокрутки.

• В списках уже стоят наиболее вероятные значения.

• Нет часто используемых коротких списков (менее пяти элементов); такие списки представлены как группы радиокнопок или чекбоксов.

• Ширина списков не меньше ширины входящих в них элементов.

• Элементы списка отсортированы либо структурно, т.е. по общим признакам, либо по алфавиту, либо по частотности (только списки меньше 7 элементов).

• Многострочные списки множественного выбора снабжены чекбок-сами возле каждого элемента (списки старого стиля отсутствуют).

• Если есть свободное место, используются расширенные комбобок-сы, а не однострочные.

8. *Кнопки выбора (Option Buttons или Radio Buttons)*

• В одной группе используется не более 6 кнопок.

• В пределах группы кнопки расположены по вертикали.

• Нет состояния, когда ни одна кнопка не выбрана.

• Последовательность расположения кнопок в группе учитывает частоту использования.

• Внутри группы кнопок одна обязательно установлена по умолчанию.

9. *Вкладки (Tabs)*

• Названия вкладок выровнены по центру.

• Каждой вкладке присвоено осмысленное название.

• Количество рядов закладок не превышает двух.

• Все связанные между собой данные находятся внутри одной закладки.

• Если окно или вкладка имеет автоматически пополняемое содержимое, например в нем перечислены приходящие сообщения, в названии элемента интерфейса, который открывает окно или вкладку, выводится число объектов в этом окне и отдельно – число новых объектов. Пример: Документы (8/3).

10. *Текстовые поля ввода (Text Box or Edit Field)*

• В полях ввода уже стоят наиболее вероятные значения.

• Если в поле вводится численное значение, границы диапазона выводятся во всплывающей подсказке.

• Если в поле вводится численное значение из ограниченного диапазона, поле снабжено «крутилкой» (Spinner).

• Длина полей не меньше и по возможности не больше длины вводимых в них данных.

• Если поле предназначено для ввода заметного количества текста, оно многострочное.

• Многострочные поля имеют максимально возможную высоту; нет резервов для их увеличения.

• Для недоступных полей используется серый цвет (название, текст и фон поля).

• В формах ввода нажатие табуляции ведет к правильной последовательности перемещения по форме.

• Содержимое полей выровнено по левому краю, за исключением полей с числовыми значениями (например, для вывода денежных сумм).

• Многостраничные формы организованы так, что пользователь всегда видит количество оставшихся экранов (пример: «Экран *x* из

• Во всех формах, служащих для сбора информации, есть пункты «Другое» и «Неприменимо» или подобные.

• В надписях отсутствуют жаргонизмы.

• В надписях отсутствуют отрицательные формулировки (например, надпись «Не показывать примечания» неприемлема, взамен нее нужно писать «Показывать примечания»).

• Для улучшения удобочитаемости длинные числа разбиваются неразрывным пробелом по три цифры: 1 234 567.

• Точка в конце фразы отсутствует в заголовке (если он отделен от текста), в конце подписи под рисунком и в таблице.

• Все поля, обязательные для заполнения, помечены, и есть соответствующее пояснение.

• Во всех формах, служащих для сбора информации, есть описание целей сбора данных, объясняется, что с этими данными будет сделано и что не будет.

• Подписи к интерфейсным элементам начинаются с прописной буквы и заканчиваются двоеточием.

11. *Порядок табуляции фокуса ввода*

• При открытии окна фокус попадает на элемент внутри окна.

• Схема табуляции соответствует очередности заполнения полей (слева направо, сверху вниз).

• Командные кнопки включены в табуляцию.

• Невидимые и недоступные элементы исключены из схемы табуляции.

• По нажатии клавиши «Tab» переход от элемента к элементу внутри формы осуществляется сверху вниз и слева направо.

• В таблицах все столбцы с цифрами выравниваются по правому краю.

12. *Пиктограммы*

• Направление теней во всех пиктограммах одинаково: снизу справа.

• В группах пиктограмм нет пиктограмм, по цвету и форме сходных между собой. Пиктограммы одного значения, но разных размеров используют одни и те же особенности и/или сюжет.

• Нет пиктограмм со стандартными значениями, но нестандартными сюжетами.

• В пиктограммах нет текста.

ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ

• Система, завершив длительную операцию (больше минуты работы), пищит через встроенный динамик компьютера.

• Если в интерфейсе не используется непосредственное манипулирование, система не имеет своих курсоров. При непосредственном манипулировании свои курсоры применяются только в том случае, если аналогов из ОС не существует.

• Цифры, предназначенные для сравнения либо для копирования в буфер обмена, выводятся непропорциональным шрифтом.

СИСТЕМНЫЕ СООБЩЕНИЯ И ОТРАБОТКА ОШИБОК

• В формах ввода проверка корректности вводимых значений выполняется прямо во время ввода; если вводятся некорректные данные, система сразу сообщает об этом пользователю, не дожидаясь момента, когда пользователь завершит ввод данных во всей форме.

• Сообщения о некорректности введенных данных показываются рядом с элементом управления, данные в котором некорректны.

• Текст сообщений о некорректности введенных данных не говорит, что, дескать, совершена ошибка, напротив, он только информирует пользователя, данные какого типа и формата приемлемы.

• Текст сообщений о проблемах состоит из трех частей: в первой кратко описывается проблема, во второй части – как ее решить, в третьей описывается, как не допускать возникновения этой проблемы в дальнейшем.

• Статусные сообщения («Синхронизация успешно завершена») выводятся только в строке статуса.

5.2. Итерационная разработка и прототипирование ПИ

5.2.1. Общие положения и этапы разработки ПИ

Разработка ПИ является итерационным процессом, так как результаты каждого этапа корректируют работы, проводившиеся на предыдущих этапах. Графически последовательность разработки ПИ представлена на рис. 5.2. Несколько упрощая, можно сказать, что создание хорошего ПИ имеет следующие этапы:

1) придумывание воображаемых пользователей;

2) продумывание видов деятельности пользователей;

3) построение модели пользователя – как он будет осуществлять деятельность;

4) разработка первого наброска дизайна;

5) изменение дизайна в сторону упрощения до тех пор, пока продукт не окажется полностью в рамках возможностей воображаемых пользователей;

6) наблюдение за тем, как реальные пользователи работают с вашим продуктом. Определение областей, в которых они испытывают трудности. Эти области скорее всего и демонстрируют несоответствие моделей ПИ и пользователя.

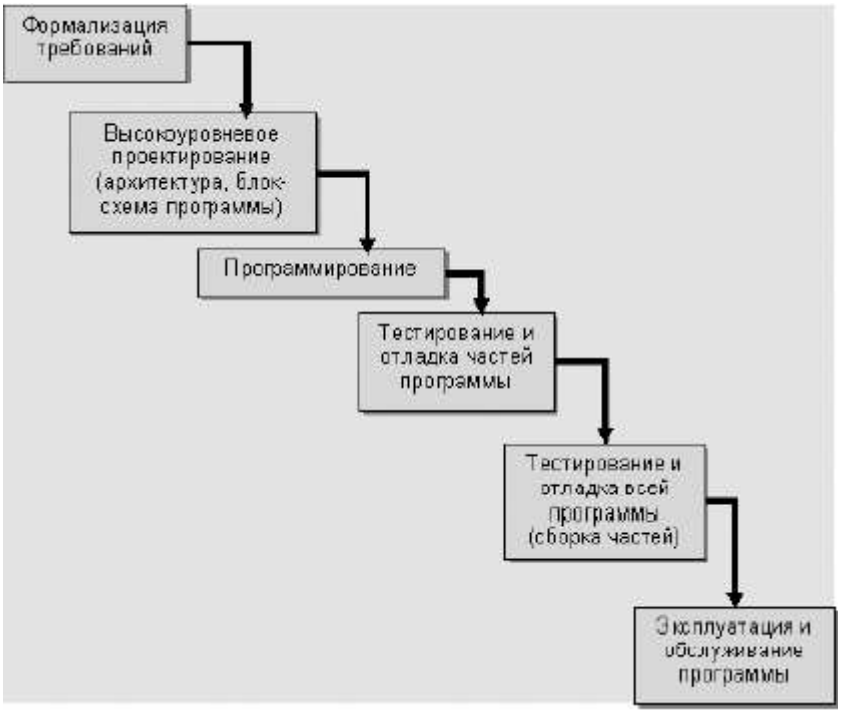

Рис. 5.2. Последовательность разработки ПИ

Разработка ПИ, как и почти любая разработка, не свободна от давления заинтересованных лиц. Это давление влияет на темпы разработки (как правило, требуют все сделать поскорее) и, конечно, на решения по ПИ. Требуется иногда значительное упорство в отстаивании своих позиций. Недаром популярным остается лозунг разработчиков: «Дадим заказчику не то, что он хочет, а то, что ему действительно надо!» На рис. 5.3 показаны наиболее характерные «источники» давления на разработчика ПИ.

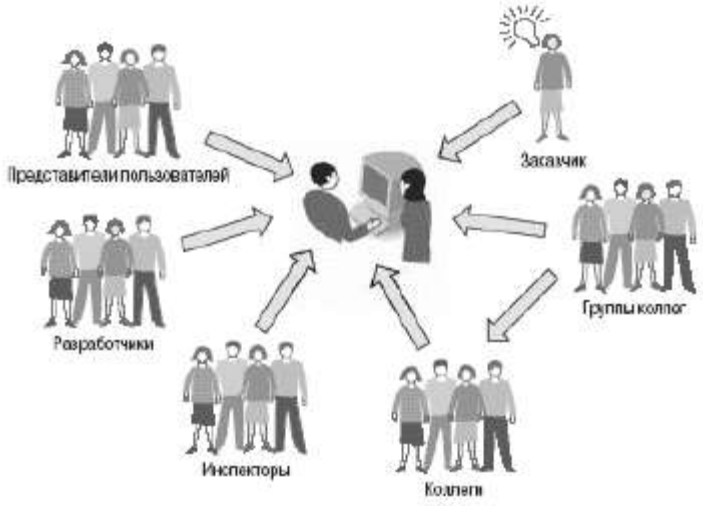

Рис. 5.3. «Источники» давления на разработчика ПИ

В конце 1990-х годов группа исследователей из Государственного университета Северной Каролины в США предложила семь базовых универсальных принципов любой разработки. Они были предназначены для охвата всех областей человеческой деятельности, в том числе и для

интерактивных систем. Эти принципы дают направление для более конкретных разработок.

1. ***Равноправие в использовании.*** Любая техническая или программная разработка должна быть рассчитана на людей с различными способностями, никто не исключается из круга потенциальных пользователей. Везде, где возможно, доступность должна быть одинакова для всех; там, где невозможен одинаковый способ использования, должен быть предусмотрен эквивалентный. Везде, где требуется по обстоятельствам применения, должна обеспечиваться общедоступная защита, секретность и безопасность информации.

2. ***Гибкость в использовании.*** Разработка ориентирована на возможно широкий диапазон пользователей путем возможности выбора методов использования и адаптивности к скорости работы пользователя, к его точности и к другим его индивидуальным особенностям.

3. ***Простота и интуитивность использования системы.*** Функционирование системы должно отвечать ожиданиям пользователя независимо от его знаний, опыта, языка либо внимательности и быть адаптирован к его языку и степени грамотности. Система должна обеспечить корректировку (подсказку) и обратную связь в максимально возможной степени.

4. ***Хорошая воспринимаемость информации.*** Информация должна быть хорошо различима, независимо от окружающих условий или способностей пользователей. Поэтому важна некоторая избыточность представления информации, т.е. ее существование в различных формах (например, графической, текстовой, речевой, сенсорной). Разные формы представления информации должны поддерживаться достаточным набором технических устройств, чтобы обеспечить наилучшую доступность для людей с разными способностями.

5. ***Устойчивость к ошибкам (минимизация ущерба от ошибок и непреднамеренных действий).*** Потенциально опасные ситуации не должны допускаться вообще или быть труднодостижимыми, кроме того, при возможности возникновения они должны сопровождаться заблаговременными предупреждениями.

6. ***Малые физические усилия.*** Системы должны быть удобными в использовании и минимизировать физическую нагрузку и утомление. Рабочее место пользователя должно способствовать поддержанию естественной рабочей позы и разумным операционным усилиям. Повторяющихся или слишком длительных действий нужно избегать.

7. ***Компоновка рабочих мест и геометрия рабочего пространства.*** И то и другое должно соответствовать антропометрическим размерам потенциальных пользователей, рабочим позам и динамике рабочих перемещений. Расположение монитора(ов), средств ввода информации, других физических устройств, их размеры, конфигурация и пр. должны обеспечивать легкую досягаемость и обзорность во всех рабочих положениях. Конструкция средств ввода информации должна предусматривать различия в размерах кисти пользователей и позволять одновременное использование других устройств (например, мыши, трекбола, джойстика и т.п.).

Эти принципы задают определенные рамки для множества более конкретных разработок человеко-компьютерного взаимодействия. Конечно, не все они одинаково применимы ко всем ситуациям. Скажем, 6-й и 7-й принципы жизненно важны для информационных будок, но менее важны для текстовых процессоров. Но все они образуют крайне полезный для разработчиков «чек-лист», своего рода памятку вопросов, ответы на которые необходимы во всех случаях. Говоря очень обобщенно, особенностями дизайна интерфейса являются:

• внимание к деталям. Интерфейс состоит из отдельных деталей, каждая из которых действует сравнительно независимо, поскольку раскрывает различную функциональность. Это сближает дизайн

интерфейса с книжным дизайном, который характеризуется как раз пристальным вниманием к мелочам;

• интерфейс не самоценен. Опять сближение с книжным дизайном (никто ведь не покупает книгу из-за качества ее верстки);

• интерфейс передает информацию своему пользователю. Это опять сближает его с коммуникационным дизайном, и с книжным в частности. Интерфейс обычно предназначен для длительного использования. Это серьезно отличает его от графического дизайна вообще (никто не будет рассматривать журнальный разворот часами), но зато сближает опять с книжным дизайном и дизайном среды обитания;

• интерфейс функционален. Очень часто приходится искать компромисс между эстетикой и функцией. Более того, интерфейс сам по себе зарождается в функциональности, «интерфейс ни к чему» просто не может существовать. Это сближает дизайн интерфейса с промышленным дизайном;

• интерфейс образуется не сам по себе, а в результате практической разработки программного продукта. Дизайнер стульев на мебельной фабрике ограничен не только и не столько своей фантазией, сколько технологией, наличием тех или иных деталей на складе, стоимостью конструируемого стула и многими другими факторами. Подобно ему, дизайнер интерфейса не сам производит интерфейс – за него это делают программисты, имеющие ограничения (стоимость, технология и т.д.).

Разработку ПИ можно разделить на четыре основных этапа:

1) начало работы над проектом;

2) постановка задачи;

3) высокоуровневое проектирование;

4) низкоуровневое проектирование.

Начало работы над проектом. На этом этапе определяется объем работ, планируются затраты и т.п. Длительность этапа, как правило, не превышает 5-8% общей длительности разработки. Для того чтобы адекватно оценить требующиеся ресурсы (время, деньги, количество специалистов), которые необходимо будет потратить на разработку (переработку) интерфейса, нужно хорошо представлять себе объем информации, с которой придется ознакомиться. Таковой является информация о предметной области (литература, эксперты). Результатом указанной работы является количественная оценка ресурсоемкости проекта.

Чтобы предлагать адекватные интерфейсные решения, необходимо иметь очень ясное представление о предметной области системы. Предметная область изучается по литературе, кроме того, весьма полезны беседы с опытными пользователями, другими сотрудниками (экспертами) для выяснения всех деталей и характеристик предметной области. Вместе с жалобами заказчика на текущую версию системы результаты этого этапа составляют основное содержание работы над проектом (экспертная оценка часто обнаруживает проблемы, которые заказчику не видны, так как замаскированы под другие).

Проблемы, выявленные на данном этапе, должны быть решены в новом интерфейсе; наоборот, удачные решения желательно сохранить, чтобы пользователям не пришлось переучиваться (и чтобы сократить затраты на переделку). По сути, этот этап завершается созданием перечня удачных и неудачных интерфейсных решений (основное внимание уделяется решениям неудачным). Если на этом этапе проводилось юзабили-ти-тестирование текущей версии, составляются также краткие протоколы и перечень выводов исследования.

Часто можно упростить разработку, просто переместив процесс проектирования ПИ на этап написания технического задания (ТЗ). Суть этого можно выразить одной фразой – проектирование интерфейса может быть не частью процесса разработки, а частью процесса создания спецификаций на систему, т.е. технического задания. Здесь необходимо сделать уточнение. Во-первых, интерфейс все равно будет разработан, во-вторых, корни многих проблем находятся именно в ТЗ, ибо это основной документ, который при любых противоречиях и конфликтах является отправной точкой. Как говорится, как задумали, так и сделали. Интерфейс может получиться даже плохим, но и в этом случае его внедрение будет облегчено; разумеется, в случае хорошего интерфейса внедрение будет более простым, но такой интерфейс требует больших затрат средств и времени.

Перенос наиболее существенных требований к интерфейсу на стадию ТЗ позволяет устранить различия во взглядах на постановку задачи заказчика и исполнителя и облегчить процесс внедрения системы. Весомая часть проблем внедрения в качественно выполненном программном продукте приходится именно на интерфейс, созданный формально правильно, но неадекватно представлениям заказчика.

Вместе с тем на стадии ТЗ, конечно, невозможно предельно конкретно прописать все интерфейсные решения до мелочей (хотя к этому надо стремиться). Только собственно реальный прототип интерфейса мог бы содержать такого рода требования. К примеру, в любом ТЗ можно прописать: что «в системе есть адресная книга, которая состоит из таких-то данных и

таких-то функций». Но невозможно формализовать уже в ТЗ, как эта адресная книга должна реально работать (какие-то функции нужно «вытащить» наверх, какие-то можно «задвинуть»), как, наконец, эта адресная книга должна выглядеть.

Постановка задачи. На этой стадии собираются и анализируются данные о потенциальных пользователях, формализуется функциональность и определяются критерии успеха проекта. Данные о пользователях и проекте должны содержать:

• характеристики пользователей – их опыт работы с компьютером, знание предметной области, мотивы, размер/важность групп пользователей, образцы (типовые ситуации) использования;

• цели и задачи пользователей;

• задачи проекта – что послужило причиной создания проекта, этапы создания проекта, какие результаты должны быть получены, какая информация необходима и когда;

• технологии разработки и платформа, на которой будут работать пользователи;

• данные о среде, в которой будет создаваться и использоваться проект (физическая, рыночная, организационная и культурная).

Эта работа предполагает доступ к имеющимся и потенциальным пользователям системы, экспертам и проектной документации. На этом этапе разрабатываются пользовательские профили, модели пользователей (разд. 3). Обязательно должны присутствовать показатели субъективных ожиданий пользователей от системы. Без этого трудно или невозможно предугадать отношение пользователей к будущей системе. Поэтому необходимо описать характеристики, которым должен отвечать интерфейс для повышения субъективного удовлетворения, перечень значимых для пользователей характеристик системы. Завершается же эта часть работы описанием среды, в которой используется система, и основных характеристик ПИ.

Характеристики ПИ отражаются в версии прототипа ПИ, которая на данном этапе будет скорее всего бумажной (см. выше). Проводится юза-билити-тестирование (см. разд. 5.1) этой версии прототипа, могут определяться некоторые показатели, например скорость работы, количество человеческих ошибок, скорость обучения, субъективная удовлетворенность пользователей и т.д. Иными словами, на данном этапе конкретизируются действительные цели разработки нового интерфейса. Например:

• группа пользователей постоянно меняет свой состав, и предполагаемый образец используется нечасто. Требуется акцентировать внимание на простоте понимания интерфейса;

• одна и та же задача повторяется многократно, а группа пользователей довольно большая. Следует акцентировать внимание на эффективности использования;

• надо снизить количество человеческих ошибок.

Необходимо также на этом этапе проанализировать интерфейсы конкурирующих систем. Большая часть пользователей любой системой обладают навыками использования нескольких конкурирующих систем; если разрабатываемый интерфейс несхож с интерфейсами конкурентов, пользователям придется переучиваться. Кроме того, конкурирующие системы часто содержат эффективные решения, которые полезно перенять (или учесть при проектировании интерфейса). Как и в случае экспертной оценки текущего интерфейса системы, отчет по выполнению этого этапа работ содержит перечень удачных и неудачных интерфейсных решений, однако отчет более сфокусирован на удачных решениях.

Специальный раздел отчета следует посвятить объяснению (по возможности подробному) всех принятых решений по ПИ, так как на этом этапе решения наиболее принципиальны и должны быть предельно ясно объяснены (см. разд. 5.2.3).

Высокоуровневое проектирование. На данном этапе начинается собственно разработка интерфейса; предыдущие этапы посвящены исключительно сбору данных и прояснению задачи. Этап состоит из трех частей: разработки структуры экранов, навигационной системы и структуры справочной системы.

Разработка структуры экранов основана на выявленных сценариях работы, т.е. определяются количество экранов, функциональность каждого из них, навигационные связи между ними, формируется структура меню и других навигационных элементов. По сути, на этом этапе выделяются отдельные функциональные блоки. Под функциональными блоками

будем понимать функцию или группу функций, связанных по назначению или области применения в случае программы, и группу функций/фрагментов информационного наполнения в случае сайта.

Существуют три основных вида связи между блоками – логическая связь, связь по представлению пользователей и процессуальная связь.

Логическая связь определяет взаимодействие между фрагментами системы с точки зрения разработчика. Полученные связи существенно влияют на навигацию в пределах системы (особенно когда система многооконная). Поэтому, чтобы не перегружать интерфейс, стоит избегать как слишком обособленных блоков (их трудно найти), так и блоков, связанных с большим количеством других.

Связь по представлению пользователей важна, потому что пользователи имеют свое мнение о системе, и это мнение должно быть отражено в интерфейсе. Известно, что самый распространенный способ поиска – по классификации признаков – работает только в том случае, когда пользователи согласны с принципами этой классификации. Большинство же понятий не может быть однозначно классифицировано из-за наличия слишком большого количества значимых признаков. Также проблема состоит в том, что субъективный классификационный признак может отличаться от общепринятого. Из этой ситуации существует простой выход – способ карточной сортировки. Все понятия, которые требуется классифицировать, пишутся на карточках из расчета одно понятие – одна карточка. После этого группе пользователей из целевой аудитории предлагается рассортировать карточки (при этом каждый субъект получает свой набор карточек). Карточки нужно разобрать на составляющие и свести результаты от разных субъектов в один способ классификации.

Процессуальная связь описывает взаимодействие, не обязательно логичное, но естественное для процесса. Установление процессуальной связи обычно довольно трудная задача, поскольку единственным источником информации является наблюдение за пользователем. В то же время установление такой связи дело исключительно полезное. Зачем, например, рисовать на экране сложную систему навигации, если точно известно, к какому блоку пользователь перейдет дальше? В этом смысле зачастую можно навязать пользователю какую-либо процессуальную связь, пожертвовав удобством, зато выиграв в скорости обучения (поскольку пользователю приходится думать меньше).

Жестко заданная связь позволяет также уменьшить количество ошибок. Классическим примером жестко заданной процессуальной связи является устройство мастеров, при котором пользователя заставляют нажимать кнопку «Далее». Результатом данного этапа является полная схема экранов (без описания их содержимого). Пример такой схемы приведен на рис. 5.4.

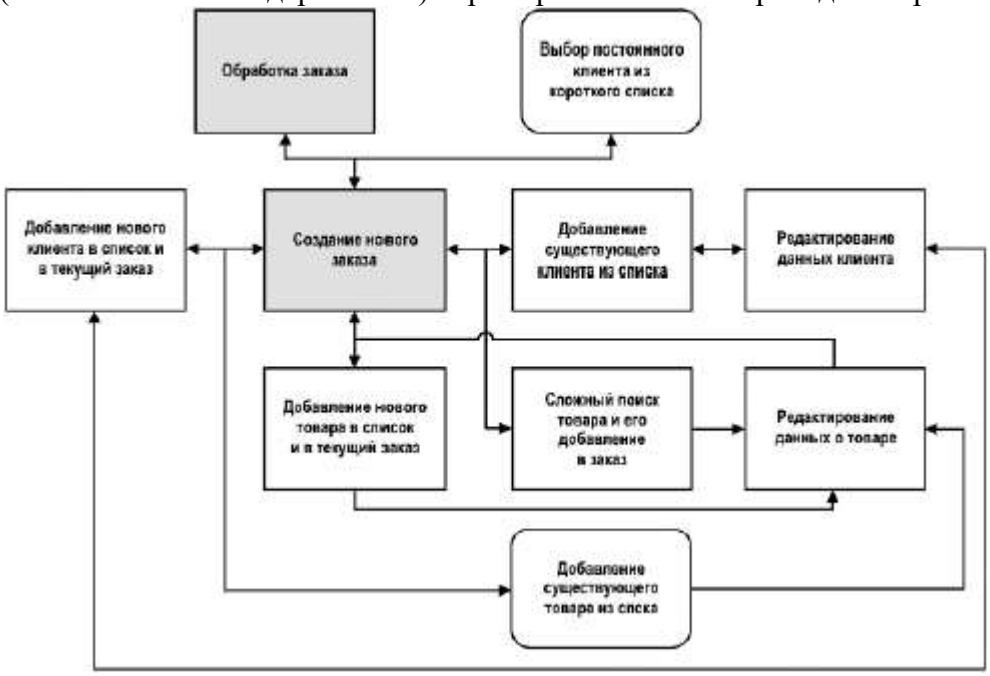

Рис. 5.4. Пример структуры экранов

Различные экраны или страницы сайта могут быть реализованы, скажем, в виде каскадного стиля. Каскадные таблицы стилей (сокращенно – CSS) дают много преимуществ: они являются одним из распространенных шаблонов построения ПИ, позволяя «облегчить» каждую страницу, повысить пропускную способность (уменьшить время загрузки web-стра-ницы) и, главное, серьезно облегчить и ускорить процесс внесения изменений при работе над ПИ, так как можно работать отдельно над каждой страницей. Подробнее об этом можно прочитать у Шмидта Кристофера [11].

На основе разработанной структуры экранов на этом этапе выбирается наиболее адекватная *навигационная система* и разрабатывается ее детальный интерфейс. Навигационная система показывает механизм распределения функций и задач между окнами программы. Навигационная система определяет, каким образом пользователи смогут перемещаться как между различными задачами, так и внутри отдельной задачи. Например, можно ли будет оставить частично завершенную задачу и начать другую.

Результаты данного этапа обязательно отражаются в презентационных, а возможно, и в псевдореальных версиях прототипа, по которым и проводится юзабилити-тестирование (см. разд. 5.1). В соответствии с результатами последнего осуществляются корректировки ПИ.

Структура справочной системы должна не просто описывать интерфейс, но помогать пользователю решать его задачи. После разработки структуры экранов и навигации по ним составление соответствующей справочной системы обычно не вызывает трудностей. На этом этапе начинается также подготовка документации для пользователя. Это очень важная часть работы, часто несправедливо считающаяся второстепенной. Тот, кто составляет документацию для пользователя (будем называть его техническим писателем, по аналогии с установившейся традицией), обычно работает в теснейшем контакте с создателями ПИ, так как, если интерфейс прост и ясен, работы немного, а если сложен и неочевиден – то много. Технический писатель почти всегда знает о конструируемой системе не меньше, чем те, кто создает интерфейс, или, во всяком случае, знает другое. Это делает необходимым тесный контакт с писателем, более того, во многих источниках напрямую рекомендуется ставить писателя выше дизайнера интерфейса, поскольку его вклад в проект более значителен. Дело в том, что скорость обучения работе с системой является важнейшей составляющей качества.

Например, Джеф Раскин, отец Макинтоша, изначально был начальником отдела документации. После того как он обнаружил, что имеющуюся систему невозможно приемлемо описать, он создал новую, описывающуюся хорошо. Побочным свойством новой системы – компьютера Макинтош – было то, что его интерфейс был прост и удобен в работе. Документация есть часть интерфейса, причем в сложных системах – большая часть.

Проблема заключается в том, что роль документации в России недооценивается. Поэтому она почти всегда пишется после того, как система создана, кроме того, и писателей мало ценят. Неудивительно, что средний уровень отечественных специалистов по написанию документации таков, что не позволяет рассчитывать на существенную помощь от нее.

В то же время хорошие специалисты есть. И если вы нашли такого, считайте, что ваша работа будет легче и качественнее.

Одновременно и параллельно с написанием документации для пользователя необходимо также писать документацию для разработчиков ПИ, объясняя каждое свое дизайнерское решение ПИ, большое и малое, приводя все соображения, которые вами двигали при разработке ПИ. Это в высшей степени человеколюбивое дело по отношению к тем, кто будет, отталкиваясь от вашего ПИ, делать какие-то аналоги, усовершенствования, модификации (см. разд. 5.2.3).

Низкоуровневое проектирование. На данном этапе разрабатываются интерфейсы конкретных экранов системы (состав, взаимное расположение и поддерживающие текст интерфейсных элементов). Вся эта работа может быть разделена на четыре части: проектирование основных экранов, тестирование, проектирование второстепенных экранов и, наконец, финальное юзабилити-тестирование.

При *проектировании основных экранов* производятся полное описание их интерфейса (без обработки исключительных ситуаций), организация информации на экранах. К отчету прилагаются макеты экранов с описаниями функциональности каждого интерфейсного элемента. Разрабатывается презентационный или псевдореальный прототип ПИ, а в конце этого

этапа прототип вполне может быть и реальным. При юзабилити-тестировании на основе критериев оценки ПИ и сценариев действий пользователей разрабатываются тестовые задания, которые выполняются пользователями на прототипе с фиксацией всех значимых характеристик деятельности (производительность труда, количество человеческих ошибок). После этого выполняются подсчет соответствующих показателей и сравнение их с заданными (или желаемыми). На основании полученных данных интерфейс либо дорабатывается, либо считается разработанным.

К *второстепенным экранам* относятся диалоговые окна и всевозможные сообщения. Их интерфейс полностью описывается, равно как описываются и исключительные ситуации, влияющие на интерфейс.

При *финальном юзабилити-тестировании* на какой-то версии прототипа разрабатываются и выполняются тестовые задания, оставшиеся после предварительного тестирования. На основании полученных данных интерфейс либо дорабатывается, либо считается разработанным, т.е. это итерационная процедура (как и юзабилити-тестирование в целом). На этом этапе следует помнить о пользе контрольных списков (см. разд.5.1.6).

Здесь также полезно зафиксировать все используемые в системе понятия. Для этого нужно просмотреть созданные экраны и выписать из них уникальные понятия (например, текст с кнопок, названия элементов меню и окон, названия режимов и т.д.). После этого к получившемуся списку нужно добавить определения всех концепций системы (например, книга или изображение). Затем этот список должен быть улучшен, для чего необходимо:

• уменьшить длину всех получившихся элементов;

• показать этот список любому потенциальному пользователю системой и спросить, как он понимает каждый элемент. Если текст какого-то элемента воспринимается неправильно, его нужно заменить;

• проверить, чтобы одно и то же понятие не называлось в разных местах по-разному;

• проверить текст на совпадение стиля с официальным для выбранной платформы (если вы делаете программу, эталоном является текст из MS Windows);

• убедиться, что на всех командных кнопках стоят глаголы-инфинитивы (отменить, удалить, отправить).

5.2.2. Прототипирование ПИ

Создание эффективного прототипа интерфейса является чрезвычайно важной задачей. Прототип должен хорошо выглядеть, чтобы понравиться заказчику и не вызвать вопросов у субъектов тестирования, он должен быть дешев, максимально полон и, что немаловажно, с легкостью обновляться.

Требования к прототипу изменяются со временем. Сначала наиболее актуальными его свойствами являются скорость создания и простота модификации. Эти свойства позволяют быстро разработать и проверить несколько версий интерфейса, исправив значительную часть ошибок. Только затем на первый план выходят функциональность и эстетичность, простота же модификации становится не так важна, поскольку с каждой новой исправленной ошибкой снижается вероятность того, что прототип придется полностью переделывать при обнаружении новой ошибки. Степень сходства прототипа с результирующей системой должна соответствовать стадии разработки. Первый прототип стоит делать максимально примитивным. Только после того как тестирование подтверждает его правильность, стоит делать более детализированный прототип.

При создании прототипа наиболее частой ошибкой является чрезмерное «наведение глянца» и вообще стремление сделать прототип возможно более похожим на результирующую систему. В самом таком подходе нет ничего плохого (все равно определенные части прототипа приходится делать максимально совершенными), проблема в том, что в большинстве случаев прототип после тестирования оказывается неправильным. Его приходится переделывать, причем иногда полностью, при этом все вложенные в прототип ресурсы (в основном время) оказываются выброшенными на ветер. Различают четыре версии прототипа.

Особый интерес представляют первые две, третья и четвертая используются редко, так как

почти не отличаются от готовой программы. Первая версия – *бумажная,* вторая – *презентационная,* третья – *псевдореальная,* четвертая – *реальная.*

Бумажная версия прототипа. Достоинствами ее являются исключительная простота и скорость рисования. Польза начального прототипи-рования на бумаге заключается, во-первых, в исключительной простоте модификации по результатам тестирования, а во-вторых, в относительной простоте привлечения представителей целевой аудитории. Кроме того, бумага помогает не думать о том, как интерфейс выглядит, позволяя сосредоточиться на том, как интерфейс работает. Поэтому не следует делать такой прототип очень реалистичным, он должен просто отражать функциональность.

Бумажная версия прототипа рисуется на этапах постановки задачи и высокоуровневого проектирования (см. выше). Она является предварительной, предстоит еще множество корректировок, поэтому не следует пытаться утвердить ее у заказчика. Не следует также часто использовать ее для тестирования. Тестирование на бумажном прототипе – проявление неуважения к участнику тестирования. На перекладывание бумажек уходит много времени. От скучающего участника ожидать сотрудничества не приходится. Можно в начале работы провести только пробное тестирование восприятия системы пользователем и ее основной логики.

Бумажный прототип рисуется быстро, но медленно перерисовывается. Сложный экран придется перерисовывать много раз. Прототип же, нарисованный на компьютере, изменять легко и просто – а раз это просто, то будет делаться чаще, что самым положительным образом скажется на качестве интерфейса. Бумажный прототип либо быстро рисуется, либо хорошо выглядит. Показывать его заказчику можно только в начале проекта, дальше просто неприлично. Разумеется, обсуждая интерфейс с заказчиком, рисовать экраны на бумаге можно и нужно, но на следующую встречу надо приносить уже компьютерную версию.

Бумажный прототип многое искажает. Можно нарисовать на бумаге прекрасный интерфейс, а потом обнаружить, что он не вмещается в отведенное для него пространство. Поэтому всякий раз, создавая интерфейс на бумаге, нужно помнить, что прототип вам немного врет. В компьютерном прототипе таких проблем нет – его можно нарисовать совпадающим по размерам с реальным. Таким образом, бумажные прототипы не могут полностью заменить прототипы экранные.

Для создания бумажного прототипа необходимо нарисовать на бумаге все экраны и диалоговые окна (или распечатать соответствующие части схемы). Нужно только убедиться, что все интерфейсные элементы выглядят единообразно и сколько-нибудь похожи на реальные. Эта распечатка и является первым прототипом, который используется для пробного тестирования. При рисовании прототипа очень важно не стараться нарисовать его красиво, например не нужно стараться рисовать линии прямыми. На понимание работы интерфейса это не повлияет, зато очень замедлит работу. Так что основным критерием живописности должна стать скорость работы.

Элементы интерфейса, которые нельзя нарисовать однозначно (например, раскрывающиеся списки, у которых значения до поры скрыты), эффективнее всего рисовать неоднозначными, важную же информацию из них лучше всего словами писать на полях. Разумеется, значение слова «версия» весьма условно. В действительности после обнаружения каждой ошибки схема и прототип исправляются, а тестирование продолжается уже на новом прототипе. Так что на этом этапе прототип может иметь множество исправлений и, соответственно, много версий. Примерно в таком виде может выглядеть бумажная версия интерфейса (рис. 5.5).

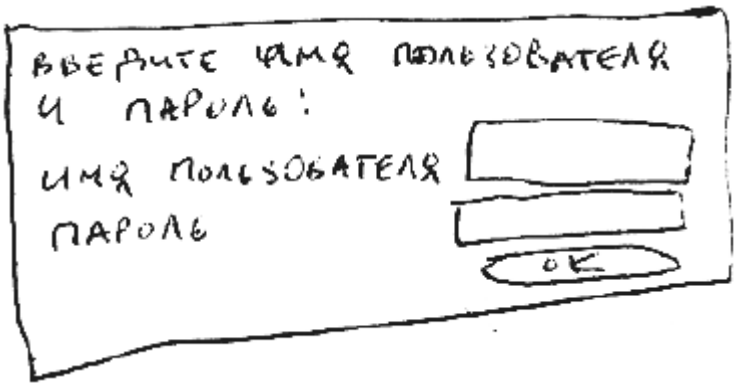

Рис. 5.5. Вариант бумажной версии интерфейса

Презентационная версия прототипа. После того как исчерпаны возможности бумажной версии прототипа, стоит создать новую версию. Для этого точно так же рисуется интерфейс, но уже не на бумаге, а в какой-либо презентационной программе. С этой версией прототипа можно тестировать значительно более сложное взаимодействие человека с системой, нежели с бумажной. В то же время исправлять ошибки значительно труднее. Такая версия прототипа может создаваться уже на этапе высокоуровневого проектирования, но на этапе низкоуровневого проектирования ПИ она обязательна (см. разд. 5.2.1).

Достаточно распространенными инструментами для создания прототипов этого вида являются на сегодняшний день MS PowerPoint и MS Visio. Пакет PowerPoint обладает многими преимуществами для прототи-пирования. Наиболее существенные преимущества – его широкая известность, дешевизна и массовое применение. Он не требует специальных знаний (в HTML или в программировании), достаточно прост и интуитивно понятен. Несмотря на некоторые ограничения, для значительной доли реальных задач он вполне достаточен. Использование возможности гиперсвязей объектов (вариантов интерфейсов) с множеством слайдов позволяет оперативно менять интерфейс, сосредоточиваясь именно на нем, а не на сложных видеоэффектах. Хотя PowerPoint и не предназначен для анимационных эффектов, их можно получить путем предъявления серии последовательно изменяющихся изображений и их яркости и/или величины. Это создает иллюзию движения, особенно при быстрой смене слайдов (максимальная скорость – 4 слайда в секунду). Одним из недостатков этого метода является ограничение количества слайдов, требуемых для прототипа. Некоторые авторы указывают, что, если количество слайдов превышает 300, существенно возрастает риск утраты ранее созданных гиперсвязей (это связано с особенностями программного обеспечения). Кроме того, данный пакет требует весьма тщательного управления конфигурацией дерева слайдов.

Конечно, такие пакеты, как MS FrontPage или Macromedia Dreamweaver, обладают широкими возможностями для создания интерактивных прототипов, они способны, в частности, оперировать не только линейными, но и иерархическими структурами. Пакеты Macromedia Flash и Director имеют возможность для продолжительной и достаточно сложной анимации с большим количеством объектов. Но все такие пакеты намного дороже PowerPoint (Director, например, стоит около тысячи евро) и сложнее в освоении.

MS Visio превосходит по возможностям привычный MS PowerPoint, но уступает Macromedia FreeHand. А вообще, подойдет любой иллюстративный пакет, обладающий способностью работать с несколькими страницами. Возможности MS PowerPoint для такой работы слишком малы, а возможности FreeHand, напротив, слишком велики.

При работе с Visio можно выбрать: либо рисовать все экраны на одном листе, соединяя взаимосвязанные элементы управления и экраны линиями, либо рисовать каждый экран на отдельном листе, связывая экраны ссылками. Первый вариант удобен для разработчика, поскольку позволяет оценить интерфейс в целом, второй – для заказчика и субъектов тестирования, поскольку его легче понять. Как правило, превратить второй вариант в первый оказывается гораздо легче. Если рисовать в PowerPoint, каждый экран удобно создавать как отдельный слайд, а результат нажатия на кнопки имитировать переходами между слайдами.

Большим достоинством MS Visio является возможность записывать результат в

HTML-файл, что позволяет без проблем тестировать интерфейс на территории субъектов тестирования. Раньше это мог только MS PowerPoint, чем во многом и объяснялась его популярность в качестве инструмента для создания прототипов. Сейчас это умеет и Visio (сохранение в формате HTML начало нормально работать в версии Visio 2001). В то же время прототипирование интерфейсов в Visio сталкивается с серьезными трудностями.

1. Результат работы, т.е. собственно прототип, трудно передать заказчику. Visio не является стандартной программой, так что просмотреть прототип заказчик чаще всего не может. Недавно появилась бесплатная программа просмотра, но ее интерфейс неудобен (достаточно сказать, что прототип открывается внутри браузера). Можно экспортировать прототип в другой формат (при этом из прототипа получается нечто вроде сайта), но он будет фактически неработоспособен.

2. В Visio практически невозможно разрабатывать длинные (т.е. высокие) формы. При переходе на такую форму масштаб отображения прототипа изменяется так, чтобы форма вся помещалась на экране; как следствие масштаб становится слишком мелок и его приходится менять вручную. Из-за этого на прототипе почти невозможно проводить никакое юзабилити-тестирование.

3. В Visio присутствуют шаблонные страницы, но гиперссылки на этих страницах не работают. Это значит, что любое навигационное меню приходится размещать на всех страницах, на которых оно используется; при необходимости же изменить один элемент приходится менять его вручную на всех страницах.

Этих недостатков лишен пакет Adobe InDesign. Выходным форматом в InDesign является надежный и общедоступный PDF с гипертекстовыми ссылками. Проблема длинных форм в InDesign решается благодаря тому, что при установке ссылки можно указать масштаб целевой страницы. Третьей проблемы также не существует – гиперссылки на шаблонных страницах работают. InDesign имеет ряд преимуществ.

1. Интерфейс Visio проектировался для спорадической работы неспециалистов – в самом деле, много ли людей, которых нанимают специально под работу с Visio? InDesign, напротив, спроектирован в расчете на то, что с ним будут работать по восемь часов пять дней в неделю. В результате интерфейс InDesign позволяет работать гораздо быстрее – одним только использованием «горячих» клавиш обеспечивается значительное ускорение работы.

2. Работа со слоями в Visio чрезмерно сложна – окно настолько велико, что использовать его почти бесполезно. В InDesign, напротив, работа со слоями реализована прекрасно – так, только теперь появилась возможность готовить два разных прототипа – один специально для разработчиков, а другой (урезанный) – для тестов.

3. Известно, что тени – лучшие друзья web-дизайнера. В прототипах их использовать тоже очень полезно – они прекрасно отделяют примечания к интерфейсу (если они есть) от самого интерфейса. Сделать прототип без примечаний практически невозможно, проблема в том, что эти примечания в прототипе теряются. Использование теней эффективно решает эту проблему, но в Visio они настолько рудиментарны, что пользоваться ими нельзя.

Но есть и проблемы с InDesign. Так, он весьма ресурсоемок: работает медленно, требуя при этом много оперативной памяти и мощного процессора. Правда, экспорт очень быстрый, гораздо быстрее, чем в Visio. В Visio любому объекту можно назначить стиль, и после этого массово менять облик элементов управления легко и просто. В InDesign же есть только стили цветов. Кроме того, работа с гиперссылками в InDesign реализована довольно неудобно. Существуют стили ссылок, т.е. определяемые пользователем точки их назначения, но встроенная в InDesign палитра ссылок почему-то показывает не эти стили, а список уже установленных ссылок. В результате установка ссылки и в особенности поддержание стройной системы наименований страниц превращаются в весьма неочевидную задачу.

Главным преимуществом Visio является наличие библиотеки объектов (stensil) со стандартными элементами управления. Можно не рисовать интерфейс с нуля, а просто расположить элементы управления в правильном порядке, тем самым существенно увеличивая скорость работы. Но имеющиеся ограничения таковы, что эта библиотека на практике бесполезна. Дело в том, что в текущей версии Visio стандартные объекты растровые, поэтому они хорошо выглядят только в одном масштабе. Стоит его изменить, как прототип искажается

до неузнаваемости. Кроме того, из-за чрезмерной детализации стандартных объектов с ними мало что можно сделать. Например, если надо выделить цветом какие-то объекты (скажем, обозначив тем самым в прототипе их кликабельность), то сделать это невозможно. Конечно, можно нарисовать свою библиотеку объектов, но зачем тогда Visio?

Некоторые достаточно продвинутые разработчики не могут избежать соблазна воспользоваться для создания прототипа средствами быстрой разработки приложений (Delphi, Visual Basic и т.д.). В этом случае прототип интерфейса воспринимается лишь как вспомогательная оболочка над программным кодом. Как следствие возникают следующие недостатки.

1. Для тестирования прототипа на пользователях крайне желательно, чтобы он хоть немного работал, т.е. нажатие на кнопку вызывало другое окно или из выпадающего списка можно было выбрать значение. Но в данном случае любое взаимодействие как между отдельными элементами интерфейса, так и между различными формами реализуется только с помощью написания программного кода. Что совершенно не нужно в данном контексте. Зачем усложнять то, что можно сделать проще? Ведь основное достоинство прототипа – это скорость.

2. Чрезвычайно сложно создать принципиально новый элемент интерфейса либо модифицировать уже имеющийся. Подобные задачи часто встречаются на практике, так как для хорошего интерфейса стандартных элементов, как правило, не хватает. Затраты же на создание/модификацию собственного элемента управления в данных средах разработки нельзя назвать адекватными.

3. Каждый элемент управления здесь имеет несколько десятков различных свойств, тогда как для прототипирования требуются лишь настройки внешнего вида – шрифт, цвет, текст, размер.

4. Естественный недостаток: нельзя разрабатывать web-интерфейс. Фактически для большинства случаев этой презентационной версии прототипа оказывается вполне достаточно.

Псевдореальная версия прототипа. В тех случаях, когда в интерфейсе появляются нестандартные элементы или необходимо проверить реальную скорость взаимодействия пользователя с системой, создается еще одна версия прототипа – реально выглядящая, но лишенная каких-либо алгоритмов и, соответственно, не показывающая реальных данных. Делать этот вариант можно как в средах разработки, благо в них есть визуальные инструменты создания интерфейсов, так и в редакторах изображений, что обычно быстрее. Фактически при этом создаются фальшивые снимки экрана, на которых и производят тестирование. Понятно, что как-то модифицировать эти экраны затруднительно, так что лучше не увлекаться такой работой, не получив каких-либо гарантий ее правильности. Эта версия прототипа больше соответствует этапу низкоуровневой разработки ПИ, но может применяться и на этапе высокоуровневой разработки (см. разд. 5.2.1).

Вместе с тем появились средства разработки прототипов, предельно близких к реальности. Сейчас имеется множество вполне доступных инструментальных средств, которые позволяют быстро создавать псевдореальные прототипы и моделировать разные ситуации. Эти средства значительно ускоряют разработку качественного ПО. Для компьютеров Apple Macintosh, например, таковым средством является широко известный и вполне успешный пакет HyperCard. Этот пакет во многом подобен другим инструментальным средствам анимации. В нем пользователь может создать графическое описание некоторой системы, скажем видеомагнитофона, обычными графическими средствами. Графические изображения сохраняются в картах, а создаваемые связи между картами управляют последовательностью предъявления карт, позволяя реа-лизовывать разные анимационные клипы. Помимо этой возможности HyperCard может моделировать достаточно сложные виды интерактивного поведения путем «прикрепления» к любому объекту определенных сценариев, написанных на языке HyperTalk. Так, для видеомагнитофона можно «прикрепить» сценарий к любой кнопке панели управления, чтобы пометить ее или даже сделать слышимым характерный шум видеомагнитофона, когда пользователь «кликает» мышью по его иконке. Тогда область функциональных возможностей этой кнопки отражается в специальном окне видеомагнитофона на экране. Наряду с пакетом HyperCard аналогичные возможности имеют и Macromedia Flash, и Director (см. выше).

Получила развитие и такая форма прототипирования, когда реальный пользователь, хорошо знакомый с предметной областью, но не знакомый с особенностями разрабатываемой системы, вносит наиболее существенные корректировки в ПИ, участвуя в прототипировании. Такого пользователя просят сесть за прототип ПИ и выполнять привычные для него рабочие функции, например для банковского служащего – проверить счета, сопоставляя их с поступающими платежами, не обращая внимания на какую-то специфику системы. Один из разработчиков, сидя в другой комнате, получает информацию о действиях пользователя, воплощая эти действия в язык системы. Так совершенствуется прототип, расширяясь со временем до средства подготовки и тренажа пользователей-новичков.

В последнее время роль прототипов значительно расширилась. Если в своем традиционном качестве они предназначены для текущих корректировок ПИ, то с вынесением многих средств индикации и органов управления в виртуальную область (на дисплей) прототипы превратились из временных имитаторов текущих версий ПИ в реальность непосредственной работы операторов. Для целого ряда систем (особенно в авиони-ке) они стали эффективным средством тренировки, и сейчас уже многие заказчики требуют включить такие средства в комплект поставки ПО.

Реальная версия прототипа. Иногда необходимо тестировать взаимодействие пользователя не только с интерфейсом системы, но и с обрабатываемыми системой данными. Например, работая с графической программой, пользователь не только нажимает на экранные кнопки, но также создает и модифицирует изображения мышью. Область же редактирования данных зачастую вообще не содержит каких-либо визуальных интерфейсных элементов, из чего вовсе не следует, что интерфейса в ней нет, его, наоборот, много. Просто счет в нем идет не на кнопки и переключатели, а на пиксели и миллисекунды.

Понятно, что прототип в таких условиях практически не будет отличаться от готового ПИ. Поэтому лучше всего написать нужные участки программы до создания всего остального и проводить юзабилити-тести-рование на реальной версии прототипа ПИ. Разумеется, прототип такой версии, если он все-таки разрабатывается, возможен только на этапе низкоуровневого проектирования (см. разд. 5.2.1).

5.2.3. Письменное объяснение дизайнерских решений ПИ

Объяснение дизайнерских решений – информация, которая указывает, почему система именно такова, включая ее структурное или архитектурное строение, а также функциональные или поведенческие проявления. Объяснение дизайнерских решений излагается отдельно во всей документации и должно сопровождать весь жизненный цикл разработки ПИ. Объяснение дизайнерских решений целесообразно по целому ряду причин:

• обеспечивает на более поздних стадиях разработки, модернизации и/или эксплуатационного обслуживания критический анализ решений, конкретных альтернатив, которые исследовались, и причин того, почему одна альтернатива была предпочтена другой;

• помогает избежать повторения уже сделанных исследований и сфокусироваться на тех вариантах, которые не исследовались ранее, либо вернуться к отвергнутому ранее варианту, сочтя приведенные соображения несостоятельными;

• дает знания, которые могут быть широко использованы в дальнейшем, ибо то, что работало в одной ситуации, может работать и в другой. Четкое изложение аргументов и контекста при объяснении поможет другой команде дизайнеров лучше понять, правомерно ли распространить это объяснение на новый продукт;

• вынуждает разработчиков более тщательно подходить и к обоснованию своих решений, и к сопоставлению их с другими решениями. Этому может помочь единая методика объяснения, которая наглядно показывает, почему одни доводы принимаются, а другие отвергаются.

В области человеко-компьютерного взаимодействия объяснение дизайнерских решений ПИ особенно важно по следующим причинам:

1. Обычно отсутствует единственный наилучший вариант ПИ. Гораздо чаще имеет место некоторое количество вариантов, у каждого из которых есть свои «за» и «против». Например,

графический интерфейс может требовать определенных действий, которые пользователь может осуществить при помощи мыши. При этом разработчик должен решить, представить ли каждое действие в виде «кнопки» на экране, которая всегда видима, или скрыть все действия в меню, которое следует вызвать прежде, чем действие будет выбрано. Первый выбор максимизирует наглядность действия, но последний занимает меньше места на экране. Разработчик должен решить, какой критерий оценки вариантов более важен, и обосновать свой выбор.

2. Даже если оптимальный вариант для конкретного ПИ действительно существует, область возможных альтернатив настолько велика, что вряд ли разработчик рассмотрел их все. Поэтому важно, чтобы были указаны те конкретные варианты (альтернативы), которые были исследованы. Это может пригодиться позже, когда кто-либо будет совершенствовать данный ПИ и анализировать какие-то иные варианты.

3. Юзабилити интерактивной системы очень зависит от контекста ее использования. Самый прекрасный графический интерфейс бесполезен, если конечный пользователь не имеет высококачественного графического дисплея или устройства прямого управления курсором (мыши, например). Описание всех обстоятельств работы, ее контекста поможет позже, когда будут разрабатываться новые программы. Если контекст остается тем же самым, то старое объяснение может быть принято без пересмотра. Если контекст как-то изменился, старое объяснение может быть проанализировано, и, возможно, это приведет к пересмотру выбора вариантов.

Заказчик, к примеру, совершенно не обязан быть ни ценителем дизайна, ни его знатоком, он, как правило, просит сделать то, что уже у кого-то видел: «Вот у корпорации N хороший интерфейс, сделайте мне так же». Однако, чтобы сделать «как корпорация N», нужно, как минимум, знать, почему N сделал именно так, а не иначе, что он знал, чего хотел и каких ошибок избегал. И если создатели ПИ «корпорации N» оставили письменное объяснение своих решений, ваша задача станет вполне решаемой.

Существуют три основные группы методов объяснения дизайнерских решений. Первая сфокусирована на привязке всех объяснений к определенному моменту на временной оси процесса разработки. Эта группа методов базируется на процессе реальной разработки проекта и предполагает глубокое погружение в этот процесс. Вторая группа ориентирована не столько на временную развертку, сколько на структуру, отношения всех альтернатив к определенным областям проекта. Такие альтернативы, рассматривавшиеся на более ранних этапах разработки, всегда можно восстановить апостериорно при завершении каждого этапа. Третья группа методов опирается прежде всего на психологию пользователя в конкретной интерактивной системе при решении конкретной задачи.

Эти группы методов отличаются, во-первых, степенью влияния на процесс разработки, во-вторых, своей ценой как при непосредственном применении, так и при использовании в последующем. Определенная технология объяснения вполне может изменить и процесс решения, и его результат, а не только служить пассивным отражением в документации. Что касается цены, то объяснение дизайнерских решений в сложной системе может быть чрезмерно объемным, кроме того, пространство решений может изменяться со временем. Обработка и анализ больших объемов такой информации зависят от характера ее представления, который, в свою очередь, зависит от примененного метода объяснения.

Объяснение, ориентированное на последовательность разработки, базируется на предложенной Риттелем (Rittel) еще в 1970-х годах форме представления процесса разработки, названной информационной системой, основанной на результате (Iissue-Based Iinformation System – IBIS). Эта информационная система представляет собой иерархическую структуру объяснения дизайна. Какой-то ключевой вопрос, положение образуют корневой узел структуры, к которому обращены позиции и их аргументы.

Различные позиции отражают варианты решения ключевого положения (корневого узла), к которому они обращены. Каждая позиция поддерживается или опровергается аргументами, которые отображаются как потомки в иерархии IBIS, непосредственно связанные с позициями. Иерархия растет, поскольку появляются вторичные, третичные ключевые вопросы, положения, связанные, в свою очередь, с выше– и нижерасположенными вопросами. Структура объяснения дизайнерских решений в форме IBIS приведена на рис. 5.6. Видно, что ключевые положения,

позиции и аргументы – узлы в диаграмме IBIS, связи же могут быть представлены в виде подкрепляющих аргументов, противоречащих данных, уточняющих ответвлений и т. д. Создается визуализация всех обоснований.

Отметим, что аналогичная визуализация имеет место в широко известной диаграмме причин и воздействующих факторов или диаграмме Исикавы, т.е. схеме, показывающей отношения между показателем качества и воздействующими на него факторами. Диаграмму Исикавы (иногда – Ишикавы) называют еще «рыбий скелет» или «дерево проблем» (разработчик – японский ученый Каору Исикава).

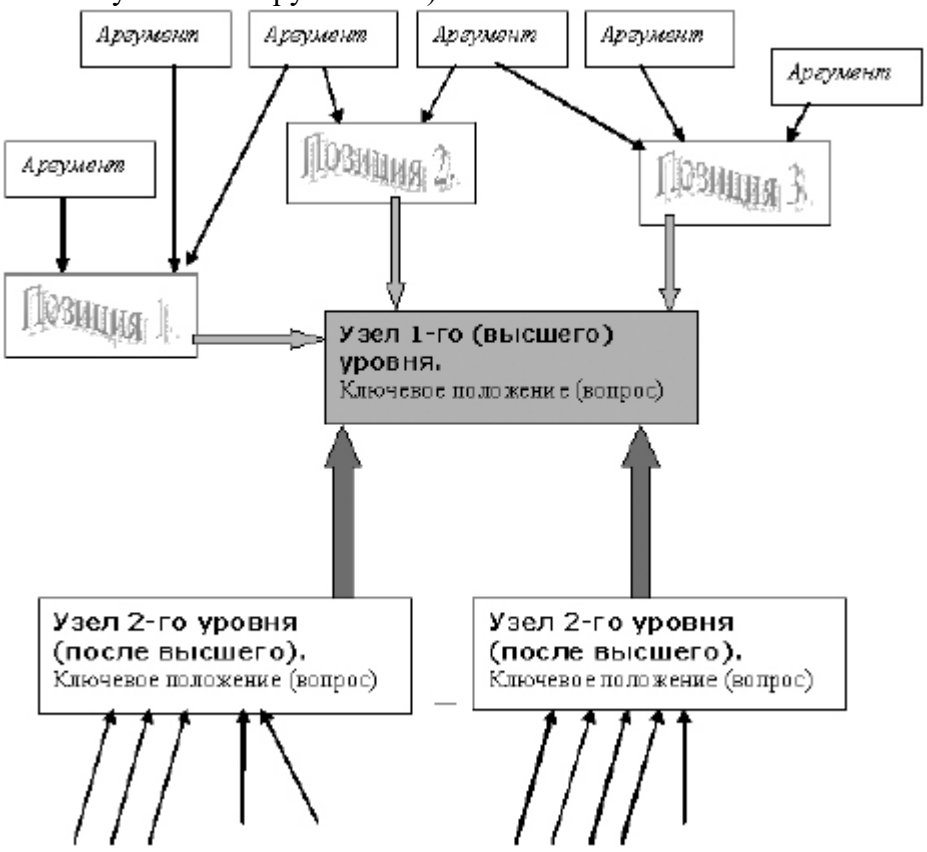

Рис. 5.6. Структура объяснения дизайнерских решений в форме IBIS

В диаграмме Исикавы главные факторы (причины), влияющие на процесс, образуют тоже узлы, вторичные причины воздействуют на эти факторы, а на них действуют факторы, влияющие на вторичные причины. Изображение формируется таким образом, чтобы в «хвосте рыбы» изображались причины, на «туловище» – следствия (проблемы), а в контуре «головы» – варианты решений по устранению причин и, соответственно, проблем, мешающих улучшению качества конкретного продукта. В настоящее время причинно-следственная диаграмма, являясь одним из семи основных инструментов контроля качества, используется во всем мире применительно не только к показателям качества продукции, но и ко многим проблемам, где можно структурировать воздействующие на процесс факторы.

Система IBIS предназначена для использования в процессе разработки ПИ прежде всего как средство регистрации принимавшихся решений. При этом упор делается на последовательности принимаемых решений во времени, учете предыстории каждого решения при разработке конкретного продукта, что позволяет облегчить перенос используемых знаний на другие разработки. Такому подходу противопоставляется структурно-ориентированный подход, к которому мы и переходим.

Объяснение, ориентированное на структуру разработки, основано на апостериорном анализе рассматриваемых вариантов. Такой подход базируется на трех основных составляющих каждого из решений – вопросов, обстоятельств и критериев, сокращенно – QOC (Questions, Options, Criteria). Область решений сначала структурируется некоторым набором ключевых вопросов, ответы на которые содержатся в конкретных решениях. Понятие «ключевой вопрос» здесь аналогично понятию «ключевое положение, вопрос» системы IBIS (рис. 5.7). В данной

системе основное внимание уделено общности критериев. Если, например, надо решить, какой способ отображения – кнопки или меню – более адекватен для пользователя, то такой критерий, как необходимость постоянного отображения всех возможных действий, поможет решить проблему.

Конечно, выбор вопросов, соответствующих им обстоятельств и критериев весьма сложен и в рамках рассматриваемого подхода к объяснению дизайнерских решений недостаточно строго определен. Положительные и отрицательные связи в формате QOC не могут охватить всех обстоятельств, скажем компромиссных решений, отсутствуют средства для учета важности вводимых критериев и пр.

Другой вариант структурно-ориентированного подхода называют «язык представления решений», сокращенно – DRL (Decision Representation Language). Он несколько расширяет средства представления объяснений (и решений) по сравнению с подходом QOC и содержит формальные семантики. Наличие последних позволяет формализовать описание, что очень важно при больших объемах информации для компьютеризации просмотров. С помощью подхода DRL можно, например, отследить зависимости между разными ключевыми проблемами так, что при изменении в объяснении определенного дизайнерского решения это будет автоматически распространено на зависимые решения тоже.

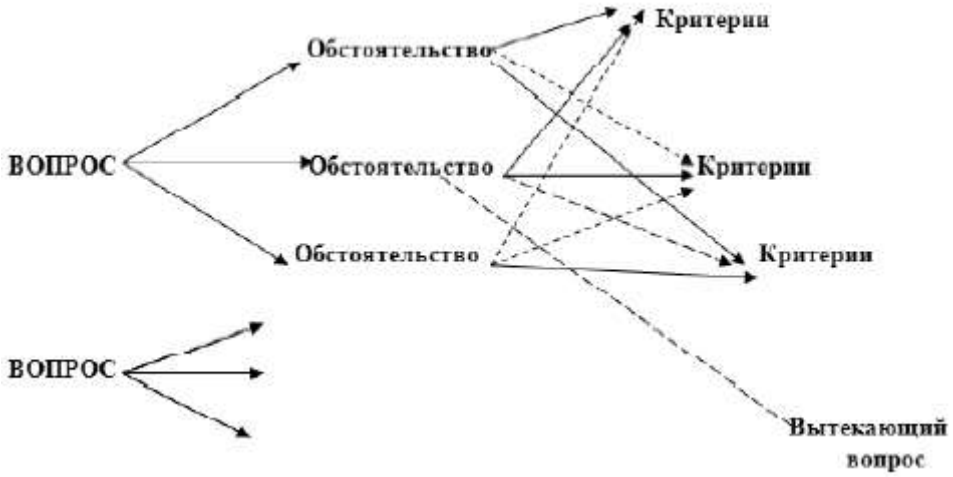

Рис. 5.7. Объяснение в форме QOC

В целом объяснение дизайнерских решений базируется на тезисе: в процессе разработки никогда не рассматриваются все возможные варианты. Поэтому лучшее, что можно сделать, – это четко документировать ту небольшую область возможных решений, которая была реально исследована. Положительные аспекты этого освещены выше. К отрицательным сторонам следует отнести рост накладных расходов и времени на такое документирование. Практика показывает, что при дефиците времени эту часть работы сокращают одной из первых.

Объяснение, ориентированное на психологию пользователя, базируется на том известном положении, что область реального применения любого нового продукта превышает (и чаще всего значительно) те потребности, которые данный продукт изначально был призван удовлетворить. Скажем, одна из первых электронных таблиц VisiCalc разрабатывалась как средство автоматизации табличных вычислений. Но уже через сравнительно короткий период (около 10 лет) электронные таблицы использовались во всех типах финансовой отчетности, в представлении и анализе разных вариантов развития финансовых ситуаций, форматах отчетности и даже как обязательный формат программных языков. С расширением числа реальных областей применения росли и возможности новых версий, призванные удовлетворить потребности, о которых создатели первых VisiCalc даже не подозревали. Другой пример – текстовые процессоры, изначально призванные просто заменить пишущую машинку, а сейчас способные выполнять весьма обширный круг задач. Сегодня области применения электронных таблиц и текстовых процессоров объединены, по сути, в единую область.

Объяснение, ориентированное на психологию пользователя, имеет своей целью выявление этих скрытых вначале потенциальных областей для учета в характеристиках ПИ.

Первый шаг в таком объяснении состоит в фиксации задач, которые призвана решать разрабатываемая система (ПО), и описании этих задач в форме вопросов, на которые пользователь будет пытаться ответить, выполняя их. К примеру, разрабатывается система, призванная помочь программисту изучить среду объектно-ориентированного языка Smalltalk. Изучая программную среду, программист будет стараться ответить на такие вопросы:

• Что я могу делать? Иначе, какие возможные операции или функции позволяет выполнять эта программная среда?

• Как это работает? Конкретнее, что делают различные функции?

• Как я могу это делать? Иначе, если я знаю операции, которые я хочу выполнить, как я могу их запрограммировать?

Для каждого вопроса разрабатывается некоторый набор сценариев возможных действий пользователя, пытающегося найти ответ. Например, по отношению к вопросу, что я могу делать, разработчик может представить сценарий, демонстрирующий работу программы, а затем воплотить этот сценарий в жизнь с помощью какой-то демоверсии, встроенной в обучающую систему. Анализ использования этой демовер-сии позволит понять те трудности, с которыми сталкивается изучающий эту систему программист, и те допущения, которые в явном или в неявном виде он принимает. Например, есть допущение, что содержание экрана связано с возможными действиями: если, скажем, под заголовком «Демоверсия» приводится список программ, то щелчок мыши по имени программы вызывает ее демонстрацию. Объяснение этой части могло бы состоять в том, что обучение происходит в процессе взаимодействия (и это хорошо). Могут, однако, быть отмечены и отрицательные аспекты демоверсии. К примеру, при щелчке мыши по имени программы система часто возвращает пользователя назад, что не позволяет понять функционирование какой-то подпрограммы.

Побуждая разработчика документировать объяснение, опирающееся на психологию пользователей, мы отражаем естественную эволюцию задач пользователя, понимая, как характеристики интерфейса одних задач влияют на поведение пользователя при решении последующих.

Объяснение, ориентированное на психологию пользователя, – это объяснение, опирающееся на последовательность разработки. Содержание вопросов (и запросов) пользователя существенно зависит от стадии разработки, поэтому предыстория таких вопросов очень важна. Документирование такого рода объяснений должно производиться непосредственно во время разработки ПО, а не в конце ее.

Контрольные вопросы

1. Что такое юзабилити-тестирование, когда оно проводится?
2. Каковы цели юзабилити-тестирования на разных этапах разработки ПИ?
3. Основные характеристики юзабилити-тестирования согласно стандарту ISO 9241 и их показатели.
4. Возможные метрики юзабилити и их содержание.
5. Варианты таксономии понятия «юзабилити».
6. Процедура подготовки и проведения юзабилити-тестирования.
7. Основные направления анализа данных юзабилити-тестирования.
8. Методики юзабилити-тестирования, их возможности, сравнительный анализ.
9. Контрольные списки, их содержание и место в разработке ПИ.
10. Последовательность и содержание этапов разработки ПИ.
11. Цели и возможности прототипирования ПИ.
12. Требования к прототипу на разных этапах разработки ПИ.
13. Какие существуют версии прототипа? Их характеристики, сравнительный анализ.
14. Программные средства создания прототипов, их сравнительные возможности, достоинства и недостатки.
15. Причины целесообразности письменного объяснения дизайнерских решений ПИ.
16. Методы письменного объяснения дизайнерских решений ПИ.

Литература

Основная

1. *Нильсен Я.* Элементарные основы юзабилити. Авг. 2003 г. www.i2r.ru/ static/255/out_20415.shtml
2. Usability в России. www.usability.ru/
3. *Сполски Дж.* Руководство по UI дизайну для программистов: Пер. с англ. 2000. http://russian.joelonsoftware.com/uibook/chapters/1.html
4. *Головач В.В.* Дизайнер, проектировщик, юзабилити-специалист. Окт. 2006 г. http://www.usethics.ru/lib/types/index.shtml
5. *Головач В.В.* Прототипирование интерфейсов в Adobe InDesign. Окт. 2004 г. http://www.usethics.ru/lib/types/index.shtml
6. *Борецкий Ф.* Друзья и враги пользователя InDesign. Май 2005 г. http://www. usethics.ru/lib/types/index.shtml
7. *Филатова Е.* Руководство пользователя для пользователя. Июль 2006 г. http://www.usethics.ru/lib/types/index.shtml
8. *Кирсанов Д.* Веб-дизайн. – М.: Символ-Плюс, 2001.
9. *Вукс Т.* Основы web-дизайна. Интерфейс, понятный посетителю сайта. 2005 г. www.i2r.ru/static/255/out_22172.shtml
10. http://www.uie.com/
11. *Кристофер Ш.* Создание web-страниц средствами CSS. – М.: КУДИЦ-ОБРАЗ, 2003.
12. *Norman D.* Design of Everyday Things (formerly – Psychology of Everyday Things). – New York: Basic Books, Inc. Publishers, 1988.

Дополнительная

13. *Porter J.* The Freedom of Fast Iterations: How Netflix Designs a Winning Web Site. Published: Nov. 14. 2006 г. http://www.uie.com/articles/fast_iterations/
14. *Wroblewski L.* Visible Narratives: Understanding Visual Organization. Jan. 20, 2003 г. http://www.uie.com/articles/visible_narratives/
15. *CarrolM., Moran T.P.* (Ed.). Design Rationale: Concepts, Techniques and Use. – Lawrence Erlbaum, 1996.
16. Sommerville. Software Engineering. 6th ed. – Addison-Wesley, 2000.
17. *Lee J., Lai K.-Y.* What's in a design rationale? Human-Computer Interaction, 6(3 4): 251-80, 1991.
18. *MacLean A., Young R.M., Belotti V.M.E., Moran T.P.* Questions, options, and criteria: elements of design space analysis. Human-Computer Interaction, 6 (3 4): 201-50, 1991.
19. *Silvers T.J., Voorheis Ch.M., Anders S.H.* Rapid Prototyping With Microsoft PowerPoint: Page Linking And Animation. Dayton, OH, Proceedings of the Human Factors and Ergonomics Society. 48th Annual Meeting, 2004. P. 1011.
20. *McGee M., Rich A., Dumas J.* (Oracle Corporation). Understanding The Usability Construct: User-Perceived Usability. Proc. of the Human Factors and Ergonomics Society. 48th Annual Meeting, 2004. P. 907.
21. *Whiteside J., Holtzblatt B. and K.* Usability Engineering: Our Experience and Evolution // M. Helander (Ed.). Handbook for Human-Computer Interaction. -North-Holland, 1988.
22. *Diez M., Sherry L., Deborah A., Boehm-Davis.* Rafiv: A Method For Cognitive Usability Analysis. Proc. of the Human Factors and Ergonomics Society. 48th Annual Meeting, 2004. P. 396.
23. *Diez M.* Why A Consumer Electronic Device Is Difficult To Use. Proc. of the Human Factors and Ergonomics Society. 48th Annual Meeting, 2004. P. 927.
24. *Беккер Л.* 90% всех юзабилити-тестов – бесполезны: Пер. с англ. http:// HYPERLINK «http://www.i2r.ru/static/255/out_21891»www.i2r.ru/static/255/out_21891. html
25. http://www.arrtlebedev.ru/kovodstvo/
26. http://www.bio.ru/stat/S5.html
27. http://research.rbc.ru/rev_short/921225.shtml

28. http://www.dist-cons.ru/modules/qualmanage/section3.html

ТЕМА 6. СРЕДСТВА МУЛЬТИМЕДИА ПРИ РАЗРАБОТКЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Изучаемые вопросы:

• Определение и понятие мультимедиа.

• Разные формы представления информации, их целесообразность.

• В каких случаях предпочтительнее использовать текст, чем другие формы представления информации.

• Основные правила текстового изложения материала.

• Правила расстановки ссылок в тексте.

• Преимущества и недостатки цветового кодирования; требования к цветовому кодированию.

• Графическое представление данных, основные типы графических описаний.

• Фотографии (или изображения), видео, анимация – их возможности, целесообразность применения, преимущества и недостатки, основные требования.

• Акустическая форма представления информации – разновидности, возможности, целесообразность применения, преимущества и недостатки, основные требования.

• Правила интеграции звуковой и визуальной информации.

• Роль и основные средства навигации.

• Разновидности и стили метафор для обеспечения навигации.

• Виды навигационных структур, их возможности, области применения.

• Основные правила навигации, пути и средства их выполнения.

• Типичные навигационные решения в web-сайтах.

• Ориентация в процессе поиска, средства и методы достижения.

6.1. Понятие «мультимедиа» и общие принципы разработки мультимедийных приложений

Мультимедиа есть наиболее естественный для человека способ обмена информацией. Поэтому при разработке мультимедийного пользовательского интерфейса предъявляются максимальные требования к всестороннему анализу деятельности пользователя, его стилю и способу работы, сложившимся представлениям о решаемых задачах, субъективным оценкам важности той или иной информации и пр.

Вместе с тем талантливые дизайнерские решения ПИ сами формируют определенное представление у пользователя, и, как часто оказывается, эти представления соответствуют наиболее эффективному стилю работы. В таких случаях подтверждается известная мысль, что реальность копирует искусство в большей степени, чем наоборот.

Мультимедиа могут быть определены как использование в системе различных сред, различных способов представления информации и устройств взаимодействия как зависимых от времени (звук и видео), так и независимых от него (графика, текст). Мультимедийность предполагает, таким образом, комбинацию текста, графики, неподвижных образов, мультипликации, видео, анимации, звуков, музыки и речи. Тем самым достигается преобразование цифрового мира компьютеров в аналоговые средства понимания, значительно более адекватные человеческому способу восприятия мира, следовательно, интуитивно понятные. Взаимодействие с такой информацией пользователь может осуществлять с помощью мыши, трекбола, джойстика, сенсорного экрана, ручки, светового пера, клавиатуры или речи. Сравнительные возможности различных способов ввода информации представлены в разд. 1.4.

Одна и та же информация может быть представлена в различной форме (модальности): текстом, речью, графикой, изображением или видео. Но качество, доступность, полнота и возможность интерактивного управления могут быть при этом разными. Поэтому полезно

руководствоваться некоторыми общими принципами, показанными в табл. 6.1.

Таблица 6.1

Целесообразность представления информации в визуальной либо звуковой форме

Случаи использования звукового представления	Случаи использования визуального представления
Информация простая	Информация сложная
Информация короткая	Информация достаточно длинная
К информации не будут обращаться позднее	К информации будут обращаться позднее
Информация о событиях во времени	Информация о положении в пространстве
Информация требует немедленных действий	Информация не требует немедленных действий
Система визуализации перегружена	Акустическая система перегружена
Требуется речевой ответ	Требуется мануальный (ручной) ответ

Основной критерий выбора формы представления информации – знания и опыт потенциального пользователя. Если пользователь знает вопрос очень хорошо – адекватным и вполне достаточным является текстовое представление. Скажем, наличие перед глазами общей схемы автомобильного двигателя всегда поможет новичку вспомнить расположение обсуждаемой детали, а для опытного специалиста достаточно просто назвать эту деталь. Другой критерий – необходимость запоминания предъявляемой информации. Здесь переходы от наименее к наиболее хорошо запоминаемым формам представления таково: а) слуховое; б) зрительное; в) слуховое + зрительное; г) пространственные, цветовые и другие виды видеопреобразования. В самом общем случае предпочтительна визуальная форма.

Характер предпочтений может меняться в случае нарушений в том или ином анализаторе человека (слуховом, зрительном, тактильном, кинестетическом и пр.). Разработка специализированных интерфейсов для инвалидов с разными типами нарушений – интенсивно развивающаяся, востребованная и весьма интересная область. Подробнее об этом можно прочесть в специальной литературе и на соответствующих сайтах. Информацию, в состав которой входят (и могут быть по-разному связаны) блоки разных типов (текст, иллюстрации, звук, видео), иногда называют гипермедиа, подчеркивая отличие от обыгчного гипертекста.

6.2. Разработка визуальной среды мультимедиа-приложений

Основной эффект мультимедиа состоит в сочетании разных средств и видов как предъявляемой информации, так и возможностей взаимодействия. Однако в этом сочетании важен не просто богатый набор используемых возможностей, но их тщательная проработка. Конечные пользователи чрезвыгчайно чувствительны к качеству мультимедиа-приложений, особенно в сравнении с книжными иллюстрациями или телевидением. Поэтому визуальные характеристики ПИ чрезвычайно важны.

6.2.1. Текст

Большинство ПИ содержит некоторый текст, даже если широко используется графика. Текст – самое гибкое средство передачи информации, что особенно важно при необходимости точного понимания и большом объеме информации. Текст особенно целесообразно использовать, если может понадобиться перечитать информацию.

Замена текстовых элементов изображениями не всегда эффективна. Часто разумнее позаботиться о качественном представлении текста. К примеру, текст, предназначенный для общественных мест, где зрение пользователей может значительно различаться, лучше

представлять ясным, крупным контрастным шрифтом. Текст в этих случаях должен быть кратким, чтобы не утомлять пользователей. К примеру, информация о гостинице для туристов должна содержать небольшое количество заголовков типа местоположения, набора услуг, прайс-листа, но с возможностью детализации информации в каждом заголовке по запросам пользователей. При этом должны употребляться короткие простые слова. Располагать же такую информацию следует в местах, где пользователь имеет достаточно времени для ознакомления с ней.

Регистр, интервал. Обычно должен использоваться и верхний, и нижний регистры (строчные и прописные символы), так как чтение в этом случае быстрее, чем при использовании только верхнего регистра. Надписи одними прописными буквами оправданы, если текст должен привлечь внимание, например в предупреждениях, в сигнальных сообщениях. Интервал между строками должен быть между 1: 2 и 1: 2,7, т.е. расстояние между основанием строки и верхней границей знаков нижележащей строки должно быть равно высоте самих букв или немного больше. Для оптимальной скорости чтения длина строки должна составлять приблизительно 60 знаков, что в русском языке соответствует приблизительно 10 словам. Только треть предъявляемого пространства должна быть заполнена текстом. Пропорциональное распределение пространства и выравнивание по левому краю либо по ширине визуально предпочтительнее.

Шрифт. Отношение толщины линии к высоте в шрифтах должно быть в интервале 1: 6 – 1: 8 для черных букв на белом фоне и 1: 8 – 1: 10 для белых букв на черном фоне. Отношение ширины знака к высоте должно быть около 2: 3. Число различных шрифтов на одном экране не должно превышать трех. Шрифт переменной ширины (MS Sans Serif, например) рекомендуется для ярлыков, маркировки и короткого текста. Шрифты без засечек или с малым их числом должны выбираться для оптимальной разборчивости (например, Helvetica более пригодна, чем Times, хотя с эстетической точки зрения Times более привлекательный). Для выбора типа шрифта важно знать расстояние между пользователем и дисплеем. На практике размер шрифта 12 для экранов современных персональных компьютеров – практический минимум, лучше же использовать размер 14 или больше, особенно при меньшей разрешающей способности дисплея или при расчете на массового потребителя (в публичных местах).

Изложение материала. Предпочтительнее использовать короткие слова, они легче читаются и их удобнее располагать на экране. Предложения должны быть лаконичными и по возможности не переходить на другую страницу. Так как пользователь нуждается в легко понимаемых, явных и неявных «поводырях», сообщения о состоянии системы должны быть соотнесены с текущей задачей, а не с внутренними операциями компьютерной системы, например более правильно выводить сообщение «Документ с именем Sales1.doc уже существует», чем «Имя встречается в файловой структуре». Сообщения должны строиться с точки зрения пользователя, а не программиста или операционной системы компьютера. Используйте технические выражения только в случаях, когда пользователь знаком с ними из ежедневной практики, но всегда, когда возможно, стремитесь быть понятными для большинства. Например, правильнее было бы сообщение «Вы сейчас контактируете с Василием Пряниковым», чем «Контакт с коммутационным блоком».

Последовательность представления информации должна соответствовать последовательности выполнения задачи. Избегайте обвинений в адрес пользователя или того, что может его обидеть; вместо этого сообщите пользователю, почему действие не может быть выполнено. Например, более правильным является выражение «Пожалуйста, измените имя файла, чтобы в нем было не более 24 букв», чем «Имя файла слишком длинное! Оно превышает 24 буквы». Если предложение содержит больше одного слова, на первые места следует ставить слова, несущие основную смысловую нагрузку, например более правильным является сообщение «Откройте папку предпочтений Claris Draw», чем «Откройте Claris Draw и найдите папку предпочтений» (подробнее об этом см. в разд. 4.3).

Что касается текстов в ПИ, то основные требования к ним – хорошая структурированность, высокая информативность и краткость. Особенно это важно при создании web-страниц. Нужно достаточно отчетливо представлять себе аудиторию, которой адресован текст, ее запросы, предпочтения, систему ценностей. Для этого полезно создать

несколько образов типичных представителей этой аудитории, к которым и будет обращено все, что отображено в ПИ.

Каждый текстовый документ должен быть разделен на фрагменты. В web-страницах, в частности, такие фрагменты являются чаще всего составляющими объектами, а сам документ рассматривается как еще один объект, содержащий внутри себя другие объекты, которые, подобно матрешке, содержат другие объекты и так вплоть до бесконечности. Объекты такого рода должны быть информативны, всегда начинаться с указания категории. Они должны обязательно содержать атрибуты, позволяющие вести по ним поиск, и образовывать группы (например, для создания персонализированного контента), классы должны иметь стандартную структуру. В этом случае облегчаются процедуры обнаружения и упорядочения материала сообразно запросам аудитории. Заголовки должны иметь 2-3 уровня и позволять обнаруживать взаимосвязи между фрагментами текста. Ключевые слова, фразы и ссылки должны быть выделены, любые перечисления представлены в виде списка. Основную идею, заключение и выводы лучше помещать в первом абзаце, минимизируя умственные затраты читателя.

Очень важна рациональная расстановка ссылок (и простых, и перекрестных, и гиперссылок), так как они – важнейший аспект сегментирования текста. Сам по себе текст должен быть самодостаточным, а ссылки должны просто помочь получить дополнительную информацию. Ссылки в тексте – это лишь возможность, но никак не необходимость. Если текст в отрыве от браузера (в распечатке, например) непонятен, то такой текст плох. Ссылки надо ставить так, чтобы читатель от их отсутствия ничего не потерял. Например, не следует делать ссылки со слов «тут», «здесь» и т.п. Поэтому лучше для начала подготовить текст, а потом уже расставлять в нем ссылки. При перегруженности ссылками читатель перестает воспринимать текст, поэтому при возможности лучше помещать их в конце материала, когда содержание изучено и надо куда-то двигаться дальше.

Ставить ссылку стоит только в том случае, если она ведет на ресурс, тема которого затронута в тексте, но не раскрыта. Весьма полезны перекрестные ссылки между свежими и архивными материалами; это, во-первых, поможет читателю получить информацию, во-вторых, значительно повысит ценность архива. Интонация и смысл ссылки тесно связаны с целостностью текста. Внешний вид ссылки в тексте (выделение цветом и, как правило, подчеркиванием) уже сам по себе способен влиять на интонацию. Другие способы выделить интонацию – курсив и полужирное начертание. Поэтому стоит избегать совмещения приемов, если ставится задача именно интонационного выделения. Использовать полужирное начертание для ссылки уместно, но именно как способ обратить внимание на ссылку, а не как способ показать, что важное слово является ссылкой.

Аббревиатуры. Количество сокращений должно быть минимальным. Они должны использоваться только там, где сокращения привычны. Сокращения должны быть постоянными во всем приложении. Расшифровка сокращений не должна вызывать трудностей. Легкая ее доступность обеспечивается за счет либо сносок, либо ссылок на соответствующую документацию, либо он-лайновой поддержкой.

6.2.2. Цвет

Цвет – одно из наиболее часто используемых средств кодирования информации. Обычно кодирование цветом средств индикации и управления или элементов ПИ воспринимается быстрее и легче, чем кодирование формой. Цвет может также использоваться для иллюстрации изменений и вариаций как в пространстве (например, географический рельеф на карте), так и во времени (например, новые файлы первоначально окрашены ярко, затем, по мере устаревания, блекнут, наподобие старению бумаги). Цвет очень важен в разработке эффективных ПИ и аппаратных средств. Преимущества цветового кодирования следующие:

• привлекательная компоновка экрана, возможность разделения экрана на области;

• усиление логической организации ПИ;

• высокая эффективность при привлечении внимания пользователя к заданным местам интерфейса;

• выделение определенных элементов из фона, что облегчает поиск;

• возможность привлечения внимания пользователя к ситуациям, требующим срочного действия;

• дополнение к другим видам кодирования (обозначениям, формам, размерам и пр.), что облегчает процесс обработки информации.

Цвет, однако, труден для однозначного опознания. На восприятие цвета человеком влияет окружение, например разное освещение может изменять видимый цвет, нахождение по соседству других цветов также может изменять воспринимаемый цвет, длительная экспозиция насыщенного цвета может вызывать раздражающие послеобразы и т.д. Кроме того, многие люди страдают цветовой слепотой, например около 8% мужчин имеют ту или иную форму нарушений цветоразличения (что, кстати, служит стимулом для выпуска монохромных дисплеев).

Использование цветового кодирования целесообразно совмещать с другими видами кодирования, например с начертанием, формой, размером т.д. Это позволит работать пользователям с ослабленным цветовым зрением, а также использовать монохромные экраны. Недостатки цветового кодирования следующие:

• неприемлемость для малых объектов;

• слабое цветоразличение у части людей;

• возможность неточного опознания из-за эффектов смешения цветов;

• влияние окружения (освещение, цветовые характеристики фона и пр.) на восприятие цвета;

• ограничение области применения преимущественно небольшими или временными элементами (например, пунктами меню) и слабая эффективность для постоянных элементов (например, строк текста).

Ниже приведены основные требования, предъявляемые к цветовому кодированию.

• Не следует использовать более пяти цветов, если предполагается их запоминание. Это касается как различных цветов, так и оттенков одного цвета.

• Не следует использовать цветовой спектр для кодирования порядка событий, так как люди обычно не помнят цветовой спектр. Оттенки и степень насыщения лучше использовать, когда надо выделить элементы, при этом выделяемый элемент должен иметь наиболее интенсивную и насыщенную окраску.

• Если необходимо помнить значения показателя, лучше использовать надписи.

Значения цветов. Опыт показывает, что некоторые цвета ассоциируются у большинства людей с определенным смыслом. Например, красный – с теплом, опасностью или остановкой. Сложившиеся ассоциации для цветов приведены в табл. 6.2. Всегда следует ясно разделять цвета, используемые для оформительских целей (например, цвета обода и подставки монитора), и цвета, используемые на дисплее. Один и тот же цвет должен иметь один и тот же смысл во всем приложении, за исключением черного и белого цветов, которые используются для текста.

Таблица 6.2

Общие рекомендации по использованию цвета

Цвет	Значение	Цвета, с которыми комбинировать	
		следует	не следует
1	2	3	4
Красный	Опасность, тепло, стоп, тревога, финансовые потери	Белый	Зеленый
Желтый	Опасность, предупреждение, риск, ненормальное состояние	Черный,темно-голубой, зеленый	Белый
Зеленый	Безопасно, нормально штатный режим	Белый	Красный
Светло-голубой	Осведомление, прохладно	Черный	Желтый
Темно голубой	Осведомление	Белый	Желтый
Пурпурный	Состояние тревоги	Белый	–
Белый	Осведомление	Зеленый, черный, красный, темно-голубой, пурпурный	Светло-голубой, желтый
Черный	Финансовое прибавление	Белый, светло-голубой, желтый	–

Цветовой контраст и фоновые цвета. Высокий контраст предпочтителен для различения двух элементов (например, двух линий), это лучше, чем использование оттенков серого цвета. При необходимости различать цвета фона и переднего плана контраст между этими цветами должен быть максимален. Черный текст на белом фоне обеспечивает высокий коэффициент контраста, что оптимально для восприятия текста. Использование цветов из противоположных частей спектра может вести к зрительному утомлению; в частности, следует избегать комбинирования красного и синего/голубого цветов. Фон графического дисплея должен быть темным. Во всем приложении пользовательского интерфейса фоновые цвета должны быть одинаковы, но оттенки их могут варьировать.

6.2.3. Графики, диаграммы, шаблоны, пиктограммы

Графические представления особенно наглядны для данных, которые имеют более чем одну численную величину (например, изменения во времени значений различных параметров). Двумерные или даже трехмерные объекты могут быть отображены на дисплее в

соответствующих двух– или трехкоординатных сетках. При этом следует пользоваться неизменными масштабами и координатами, если это возможно; не рекомендуется применять автоматическую адаптацию шкал к диапазонам изменения отображаемых данных, так как это мешает узнаваемости. В случаях же, когда данные одного дисплея одновременно используются разными работниками, такая адаптация должна быть серьезно ограничена. Правильное использование теней на графических изображениях и может осуществляться как на цветных, так и на черно-белых дисплеях.

Графики (или диаграммы) используются для выражения схематических соотношений, определенных характеристик объектов. В случаях плохого качества фотографий (изображений) и необходимости отразить определенные характеристики график может оказаться лучше, чем изображение. Графики (или диаграммы) могут использоваться для показа того, что не существует в видимом мире, они подчеркивают или освещают самое главное. Графическое отражение должно быть максимально простым, ясным и эффективным. Например, для описания электрической схемы используются определенные графические обозначения, цвета линий на карте – границы каких-то административных образований, а линии определенных длин и углы – для представления скорости и направления ветра и т.п. Перенасыщение графическими кодами и неудачное графическое кодирование приводят к обратному эффекту, т.е. к утрате наглядности и информативности.

Основные типы графических описаний

Гистограммы удобны для отображения данных об изменениях числовой величины одного параметра. Они могут использоваться для показа состояния в конкретное время или последнего измеренного значения, т.е. обеспечивают быстрое восприятие количественного значения. С помощью гистограмм могут быть показаны количество и динамика. Наличие шкал позволяет дать точную оценку переменной. Наиболее часто гистограммы применяются в статистическом анализе (рис. 6.1).

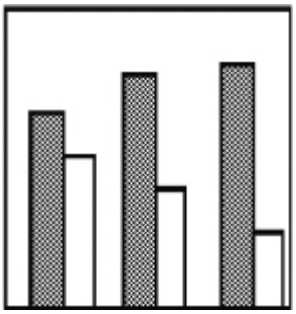

Рис. 6.1. Гистограмма

Графики линейных функций могут показывать, как две непрерывные переменные изменяются по отношению друг к другу, особенно наглядно показывают изменения одной переменной во времени. Часто по оси абсцисс откладывают время. Число переменных, отображаемых в виде линий, не должно превышать 4 на один график. При нескольких переменных каждая должна иметь собственное обозначение точек на графике (рис. 6.2).

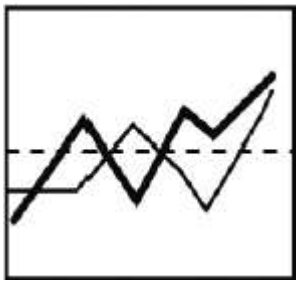

Рис. 6.2. Графики линейных функций

Круговые диаграммы удобны для показа отношения или пропорций частей по отношению к целому. Посредством заполнения или разрыва сегментов круга могут быть показаны части целого (рис. 6.3).

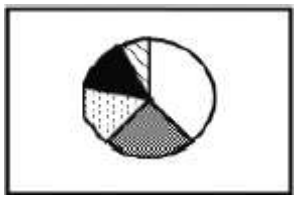

Рис. 6.3. Круговая диаграмма

Диаграммы рассеивания – способ представления двумерной последовательности данных. В диаграмме рассеивания точки значений данных отображаются разными графическими символами. Эти символы должны быть знакомы и привычны для пользователя. Для облегчения восприятия точных значений может наноситься фоновая координатная сетка. В идеале диаграмма рассеивания должна демонстрироваться на цветном экране высокого разрешения во избежание взаимных помех сетки и самих данных (рис. 6.4).

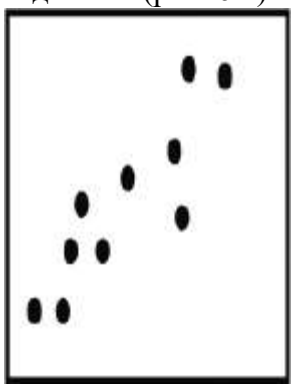

Рис. 6.4. Диаграмма рассеивания

Индикаторы величины могут быть использованы для показа изменяющегося значения по отношению к его предельным величинам (минимальным и максимальным). Для быстрого восприятия количественного значения следует использовать наряду с аналоговой и цифровую индикацию. Индикаторов величины одного параметра достаточно много (от стрелочных разных видов до окошечных) (рис. 6.5).

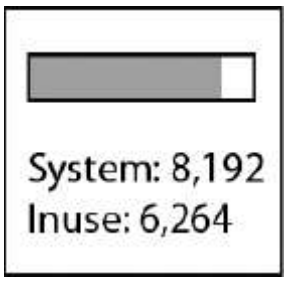

Рис. 6.5. Индикатор величины

Пиктограммы – маленькие символические картинки, используемые в компьютерных меню, окнах и экранах и отражающие какие-то возможности системы. Пиктограммы – разновидность

графики. Их несомненное преимущество – независимость от языка пользователей, малый объем занимаемого пространства и определенная эстетическая привлекательность. Хорошие пиктограммы должны:

• легко ассоциироваться с информацией, которую они несут;

• однозначно трактоваться;

• хорошо отличаться от других символов;

• быть не слишком сложными;

• быть приемлемыми для различных культур (предполагается, что они не вызывают негативных эмоций в различных культурах);

• соответствовать международным или внутренним стандартам.

Использование общих элементов у взаимосвязанных пиктограмм ведет к их лучшей

узнаваемости. Например, большой «понятностью» обладает пара пиктограмм, обозначающих изменение размеров: «+», «-», что значительно эффективнее, чем две различные пиктограммы. Если пиктограммы могут иметь несколько трактовок, то их значение нужно дублировать текстом. Имеются некоторые частые заблуждения относительно пиктограмм, а именно:

- что можно полностью заменить их словами;
- что пиктограммы всегда приводят к облегчению работы пользователей;
- что хорошие пиктограммы всегда должны быть предельно наглядны.

Проиллюстрировать эти заблуждения можно примером, где явная избыточность пиктограмм, причем очень хороших (что может быть яснее и однозначнее дорожных знаков), очевидна. На рис. 6.6 показан один из перекрестков на Невском проспекте Санкт-Петербурга. В кадр попали 45 знаков, висящих над одним перекрестком, плюс реклама. В реальности еще хуже – столько же знаков висит для водителей, едущих в обратную сторону. Очевидны намерения тех, кто знаки вешал, – им хотелось, чтобы над каждым рядом висели все необходимые указания. Но водитель смотрит на мир шире (область обзора несколько больше 180°), и в его поле зрения попадают все знаки плюс реклама.

Рис. 6.6. Пример перегруженности знаками[10]

6.2.4. Фотографии, изображения

Фотографии (или любые изображения) отражают детали реальности, можно сказать, «портрет» реальности, в отличие от графики, которая упрощает сложную информацию, выделяя определенные ее характеристики. Фотографии (или изображения) используются для представления фактографической и документальной информации, т.е. когда необходимо показать что-то с максимальным приближением к действительности. Цветные изображения воспринимаются как более жизненные; например, если изображения показывают людей или ландшафты, естественно использовать цвет. Черно-белые изображения пригодны для показа чего-то схематичного, т.е. сравнительно простого и обобщенного.

Фотографии (или изображения) часто используются для дополнения текста – например, мультимедийная туристическая система может содержать фотографии гостиниц, знаменитых площадей и зданий, блюд местной кухни т.д. Осматривая изображения реальных объектов, а также читая их текстовые описания, пользователь может быстро определить местоположение конкретного объекта. Однако слишком большое число фотографий и изображений существенно снижает эффективность их восприятия.

Отличие фотографий и изображений от пиктограмм состоит в том, что пиктограммы более однозначны: если на пиктограмме изображен стакан, то он и имеется в виду, а если на фотографии или изображении, то можно иметь в виду и стакан, и любой алкоголь, и попойку, и стекольную промышленность, и процент пьяных нарушителей и пр. В целом предпочтительнее,

[10] Фото и комментарии к нему из книги А. Лебедева «Ковodство», § 78. http://www.artlebedev.ru/kovodstvo/

чтобы изображение не выходило за рамки экрана или заданной области, так как иначе пользователь будет вынужден его прокручивать, чтобы видеть целиком. Вместе с тем весьма распространенной ошибкой является неоправданное масштабирование (обычно чтобы вписаться в определенное пространство), что часто приводит к потере информации. Масштабирование, как правило, должно сопровождаться количественным показателем, скажем 2, 4 или 100, 200% и т.д.

6.2.5. Видео. Анимация

Короткие видеоэпизоды, сопровождаемые комментарием либо музыкой, – весьма привлекательный способ представления информации. Видео более эффективно для демонстрации того, что физически движется (балетных сцен, например), чем для показа статических объектов. Видео исключительно удобно для показа каких-то реальных действий и должно использоваться для визуализации реального мира либо его имитации. Информация, не зависимая от механического движения и от какой-либо динамики вообще, экономнее и удобнее может быть представлена в виде изображений.

Видео – чрезвычайно динамическая среда, которая приковывает внимание пользователя (как всякий движущийся объект). Эффективность видео достигается использованием сравнительно коротких видеоэпизодов (клипов), которые представляют целостную информацию и включают обычно помимо видео текст, графику, цвет, звуковой фон (скажем, музыку) или звуковые сообщения (скажем, речь) и т.д. Это отличает их от фильмов, где развитие действия и составляет собственно содержание, т.е., кроме видео, там ничего нет. Рациональное использование видео предполагает выполнение ряда требований:

• время одной последовательности видео не должно превышать 40– 45 с, при большей длительности снижается концентрация внимания;

• видео должно сопровождаться звуком, что отвечает ожиданиям пользователей и позволяет дублировать видеоинформацию, либо передать дополнительную, которую передавать звуком удобнее;

• видео должно сопровождаться индикацией того, какая часть времени его демонстрации уже прошла и какая часть осталась;

• видео лучше показывать в небольшом окне, что позволяет ограничиться малым разрешением, сохраняя сравнительно хорошее качество изображения;

• следует предусмотреть клавиши (виртуальные) управления видео, которые должны быть подобны таковым в видеомагнитофонах: «вперед», «быстро вперед», «проигрывать», «повторно проигрывать», «проигрывать в обратном направлении», «быстрая перемотка», «пауза», «останов», «поиск». Однако наиболее важны клавиши «проигрывать», «пауза», «повторное проигрывание». Также следует предусмотреть клавиши регулировки качества изображения: яркости, контрастности, цвета и оттенков.

Анимация охватывает широкий круг медиатехнологий и средств – от объемных физических тел, «оживляемых» в фильмах, до образов искусственных, т.е. частично или целиком генерируемых с помощью компьютерных технологий. Анимация характеризуется быстрым и непрерывным изменением изображений, рисунков или графических образов. Как и видео, анимация нуждается в некотором смыслообразующем сюжете. Динамика анимации может использоваться, чтобы привлечь внимание пользователя к определенному объекту.

С помощью анимации могут быть визуализированы даже время или невидимые процессы, например работа алгоритма или программы, могут быть показаны опасные процессы или такие, которые трудны для непосредственного представления (в физике, химии, биологии, геометрии). Анимация весьма полезна при исследовании сложного окружения и является хорошим способом стимуляции интереса к обучению у молодых пользователей, особенно для образовательного программного обеспечения. Вместе с тем анимация может отвлекать внимание, если постоянно сопровождает фрагменты текста, которые пользователь должен прочитать. Она может быть также раздражающей, если ее темп превышает способность человека воспринять и понять смысл.

Продолжительность анимации должна быть небольшой: максимум 20-30 с. Анимация

может быть подкреплена звуковым сопровождением. Например, звук хорошо дополняет такие характеристики, как положение объекта, направление, скорость и продолжительность его движения.

Существует множество анимационных методов для передачи общих изменений (к примеру, вырезание, постепенные переходы цветов и форм, поглощение, изменения фактуры, изменение углов обзора и др.) и изменений, относящихся к определенной области экрана (к примеру, показать зеркально, вращать, выпадать вниз, вверх, вставить либо вытащить и пр.). Но польза всех таких технологий может оказаться весьма сомнительной (даже отрицательной), если они будут отвлекать пользователя от содержательных аспектов.

Пользователь должен иметь возможность в любое время останавливать анимацию и повторять ее части, также контролировать скорость анимации, при необходимости прокручивая кадры быстро вперед или назад. Также весьма желательно обеспечить возможность прямой манипуляции объектами, например возможность протаскивать тот или иной объект по экрану, сохраняя индивидуальные особенности его движений и характер взаимодействия с другими объектами на экране.

Во всех центрах, известных разработкой новых интерфейсов (XEROX PARC, MIT Media Lab, Apple Computer, Carnegie Mellon Univ.), идут разработки разных концепций дизайна интерфейсов, опирающихся на возможности анимации. Здесь основной проблемой является синхронизация точки внимания пользователя и точки активности системы.

С одной стороны, пользователь должен уметь сказать системе, где и что он хочет изменить (обыгчно это делается щелчком мыши в нужном месте). С другой стороны, система должна уметь привлечь внимание пользователя к месту наиболее актуальных изменений. Это привлечение внимания достигается какой-то динамикой (изменением цвета, местоположения, яркости и т.д.). На этом же основаны многие способы привлечения внимания с помощью движущегося предмета. Созданы, например, две анимированные среды интерфейса (в той самой фирме XEROX PARC, которой мы обязаны появлением идеи «оконного» интерфейса).

Одна – «Конические деревья» – является визуализацией файловой системы компьютера и похожа на систему детских пирамидок, каждый уровень которой соответствует уровню файлового каталога. Сами файлы из каталога отображаются в виде трехмерной карусели под своим каталогом. Идея в том, что нужный файл можно приблизить поворотом карусели (может быть, не одной), идущим в режиме анимации.

Анимация за счет увеличения времени перехода от одной картинки к другой (а именно времени анимированного преобразования картинок) существенно сокращает время осознания новой картинки. В психологическом смысле новой картинки не существует, есть преобразованная старая, а так как все преобразования шли на глазах у изумленных зрителей, то пользователь практически немедленно готов к взаимодействию.

Существует еще одно свойство анимационного ПИ, которое заметно улучшает его полезность по сравнению с графическим интерфейсом, а именно – динамические визуальные сигналы, т.е. изменение изображения на экране с целью дать пользователю дополнительную информацию. Уже в стандартном «оконном» интерфейсе мы можем видеть примеры таких сигналов. При выполнении программой длительных действий курсор мыши приобретает форму песочных часов. Это сигнал о том, что на действия пользователя система временно реагировать не будет. Второй пример – изменение изображения кнопки при нажатии на нее мышью. Создавая анимационный интерфейс, надо закладывать систему динамических визуальных сигналов с самого начала, поскольку они являются столь же естественной, сколь и необходимой частью анимационного интерфейса.

В последние годы возможности 3D-анимации (3Dimension, т.е. трехмерной) и спецэффектов значительно расширились за счет создания очень мощного пакета Maya, в который были интегрированы лучшие характеристики высококачественных пакетов Explore, Advanced Visualizer, Power Animator, Dynamation и Kinemation. Программа, изначально разработанная для Silicon Graphics, перенесена на новые платформы и может функционировать в различных ОС, в том числе в Windows NT, Windows 2000 и Linux. Пакет Maya оброс хорошей инфраструктурой, включая встраиваемые модули, сценарии на языке MEL, книги и интернет-поддержку. Именно в Maya появились многие решения, которые определили развитие

современной компьютерной графики. К ним в первую очередь относятся:

• уникальная технология Artisan, позволяющая модифицировать поверхности виртуальных объектов почти так же, как скульптор работает над глиняной моделью. Выбрав инструмент нужной формы и размера, дизайнер может производить операции выдавливания, вытягивания, сглаживания и т.д. Artisan позволяет также управлять анимационными свойствами поверхностей (например, жесткостью), просто раскрашивая объекты различными оттенками серого цвета;

• сплайн-моделирование (с использованием кривых до седьмой степени), полигональное моделирование и моделирование при помощи поверхностей разбиения (SubDivision Surfaces);

• мощный модуль для работы с динамикой твердых и мягких тел, частицами и спецэффектами, который представляет собой полностью интегрированный в Maya и расширенный пакет Dynamation;

• технология IPR (Interactive Photorealistic Render), которая позволяет обновлять единожды визуализированную сцену при изменении параметров материалов, текстур или освещения. При этом программа сама отслеживает связи между каждым элементом изображения и всеми параметрами, влияющими на его конечный вид;

• модуль Maya Fur для имитации шерсти и меха;

• модуль Maya Cloth для моделирования и анимации одежды и тканей;

• модуль Maya Live, который помогает автоматизировать и ускорять чрезвычайно сложный процесс совмещения движений камеры и объектов из видеоматериала, отснятого реальной камерой, со сценой или анимацией, полученной средствами компьютерной графики.

Кроме того, в Maya полностью интегрирована изящная парадигма нелинейной анимации (Non Linear Animation). В программе она реализована в виде инструмента TraX Editor, позволяющего монтировать и многократно использовать однажды сделанную анимацию, создавать библиотеки движений для существующих и будущих персонажей. Однако из-за разнообразия средств и своей мощности Maya очень сложна в освоении. Огромный объем хорошо спланированной документации облегчает решение конкретных проблем, но не может служить учебным пособием.

Хороший пример грамотного применения анимационных эффектов – новая ОС Windows Vista. Во многих панелях этой ОС используются «живые» анимированные иконки, отображающие содержание файла. Такую иконку можно увеличить или уменьшить. Скриншот контрольной панели Windows Vista показан на рис. 6.7.

Окна отображаются в трехмерном пространстве, что также облегчает их обзор. Функции «Flip» и «Flip 3D» позволяют переключаться между открытыми окнами – первая с помощью сочетания клавиш «Alt+Tab» с отображением активного эскиза каждого окна, вторая с помощью колеса прокрутки мыши (рис. 6.8, 6.9).

Рис. 6.7. Скриншот Windows Vista – контрольная панель

Визуализация документов в Windows Vista также существенно улучшена, что позволяет их просмотреть без раскрытия документа (рис. 6.10, 6.11).

Рис. 6.8. «Живые» иконки Windows Vista – функция «Flip»

Рис. 6.9. Функция «Flip 3D» Windows Vista

Рис. 6.10. Скриншот Windows Vista – визуализация документов

Решая многие проблемы для пользователя, анимационный интерфейс, как это часто бывает, ставит проблемы перед программистом и дизайнером. Для использования анимационного интерфейса придется переходить к программам, управляемым временем. При этом, естественно, такая программа должна быть всегда доступной для взаимодействия, но, в отличие от многих сегодняшних мультимедиа-программ, не прерывать отображаемый поток, а плавно изменять его в соответствии с воздействием пользователя.

Рис. 6.11. Скриншот Windows Vista – визуализация документов

После выработки сквозного визуального решения необходимо прорисовать картинки, которые аниматоры называют фонами. Точнее было бы назвать их неподвижной составляющей подвижного изображения. На каждом фоне надо расположить анимированные элементы взаимодействия. И наконец, самое трудное – надо спроектировать визуальные переходы между существенно отличающимися состояниями. И все это сохраняя выбранный стиль!

Анимационный интерфейс – орудие очень мощное и поэтому требует особой осторожности. Попытки потрясти мир могут привести к быстрой утомляемости пользователя и как следствие к отторжению системы. Основной задачей дизайнера становится организация не неподвижного пространства, а целой серии пространств, неразрывно связанных между собой. Аналогия с созданием фильмов представляется здесь очень уместной.

6.3. Разработка акустической среды мультимедиа-приложений

6.3.1. Общие вопросы разработки акустических интерфейсов

В мультимедийных приложениях визуальная информация часто сопровождается акустической. Визуальная и аудиосреды дополняют друг друга и составляют единое целое. Термины «акустический» или «слуховой» применяются как обобщающие для речи, пения, музыки или любых иных звуков. Таким образом, можно говорить об акустическом интерфейсе пользователя (AUI), подобно тому как мы говорим о графическом интерфейсе пользователя (GUI). Информация, представляемая в акустической форме, используется для:

- обратной связи в форме музыки или каких-то звуков;
- сообщений о тех или иных событиях в системе;
- разного рода подсказок системы;
- речевой инструкции по конкретной задаче или ее компонентам;
- речевого отчета или комментариев одного человека или более;
- речевого диалога или обсуждения действий пользователя при решении задачи;
- речевой связи для задач, решаемых совместно в интерактивном режиме.

Акустическая форма представления информации особенно эффективна:

• для общей активизации внимания;

• для привлечения внимания пользователя к определенному месту;

• при одновременном завершении или старте ряда событий или процессов;

• при перегрузке визуального канала;

• для людей с нарушениями зрения или моторики, которые в ином случае были бы не способны к работе на компьютере.

Акустический интерфейс обеспечивает следующие функциональные возможности:

• ускоряет взаимодействие с пользователем, с помощью речевого ввода команд вместо ручного или речевого выхода информации вместо диалогового окна; голос особенно полезен, если пользователь по роду работы должен отходить от компьютера;

• минимизирует промахи и ошибки пользователя путем акустической инструкции, предупреждений и положительной обратной связи;

• уменьшает физическую и умственную нагрузку, добавляя речь и звук, где необходимо, или заменяет им ручное управление и визуальную информацию;

• улучшает обучаемость и память, используя устные объяснения (по запросу пользователя), слуховую навигацию и звуковую обратную связь;

• предлагает альтернативные средства взаимодействия при изменениях в окружении или требований задачи, предпочтений или ожиданий пользователей;

• приближает человеко-компьютерное взаимодействие к реальному миру.

Состав пользователей акустическим мультимедийным приложением может включать одного человека, нескольких известных лиц или большое количество неизвестных пользователей. В зависимости от профиля потенциальных пользователей формируются требования к такому интерфейсу. Типичные вопросы, возникающие при этом, могут быть следующими:

• Известны ли пользователю язык и словарь интерфейса?

• Требуется ли некоторая тренировка, пусть даже и небольшая?

• Может ли система быть мобильной?

Физическое пространство, где будет происходить взаимодействие, чрезвычайно важно для акустических характеристик интерфейса: будет ли это маленькая комната или рабочее место в большом зале; будет ли часто меняться местоположение интерфейса, предназначен ли интерфейс для личного либо общественного использования.

Пользователи должны иметь возможность включать и отключать звук, а также регулировать уровень громкости. Кроме того, необходимо предусмотреть органы управления характеристиками звука, такими, как баланс, басы и высокие звуковые частоты. Пользователь должен также иметь возможность вызвать повторение предыдущего сообщения, прерывать или останавливать текущее сообщение, временно отключать звук, а затем продолжать при прежних его характеристиках (mute-функ-ция). В целом разработка аудиокомпонентов в мультимедийном приложении должна проводиться весьма тщательно, с учетом социальных, организационных и физических аспектов среды, в которой это будет реализовано.

В настоящее время довольно мощным звуковым редактором является GoldWave. Он предназначен для проигрывания, редактирования, смешивания и анализа звуковых файлов, работает с большими файлами быстрее других программ, имеет много разных эффектов, понимает множество форматов, также умеет конвертировать из одного формата в другой, умеет записывать файлы с аудио-CD.

6.3.2. Речевой ввод

Речевой ввод позволяет пользователю непосредственно говорить компьютеру, без промежуточных клавиатурных или рукописных действий. Основные цели систем распознавания речи следующие: независимость от того, кто говорит, распознавание непрерывной речи, большие словари и способность к обработке естественных языков. Независимость от говорящего означает, что система может принимать и узнавать речь большого количества различных пользователей, включая голоса, которые не были частью

обучающей последовательности. Такие системы не требуют предварительного обучения под конкретного пользователя. Система же, зависящая от говорящего, требует образцы речи конкретного пользователя для обучения и настройки.

Непрерывная речь означает, что система может обрабатывать слова так, как их обычно произносят в беглой речи. Другие разновидности систем распознавания речи – распознание отдельных слов и словосочетаний. Опознавательные устройства отдельных слов – наиболее простое (и дешевое) решение, но оно требует хотя бы короткой паузы между каждым словом. Опознавательные устройства определенных последовательностей слов находятся между опознанием отдельных слов и опознанием непрерывной речи. Они узнают слова при условии, что они не изменяются, бегут вместе, но это требует отчетливого произношения.

Система, способная распознавать большое количество различных слов, не всегда является оптимальным решением. Часто достаточным является набор небольшого, фиксированного количества слов. Однако такого рода системы, например система медицинской диагностики, могут потребовать достаточно много слов, иногда до 50 000.

Надежное распознавание произносимых слов, не зависящее от голоса и дикции конкретного человека и содержащее небольшой, ограниченный набор слов, является несложной технической задачей. Однако с увеличением количества распознаваемых слов и их многообразия существенно возрастают технические требования и, соответственно, стоимость распознающей системы. Распознавание свободной естественной речи ограничено пока четко определенной областью задач. Как следствие в процессе разработки системы следует оценить, какие компоненты задачи должны выполняться с использованием аудиовыхода и какие не должны. Речевой ввод наиболее предпочтителен в следующих ситуациях:

• при вводе простых команд (например – проигрывание, стоп, быстрая перемотка вперед и т.д.);

• для зачитывания простых, но хорошо структурированных данных;

• как альтернатива клавиатурному вводу для людей с физическими недостатками;

• для обнаружения голосовых звуков либо иной активности с использованием акустических датчиков;

• когда определенная функция требует использования акустических клавиш;

• для объектов или выбора опций, использующих акустическое меню;

• для речевого ввода данных или задач с речевым заполнением специальных форм;

• для речевых комментариев со стороны системы либо самого пользователя по поводу объектов задачи.

При использовании речевого ввода следует:

• структурировать словарь, чтобы ограничить число вводимых слов на каждой стадии (т.е. использовать по возможности малый коэффициент ветвления). Это улучшит точность распознавания;

• выбирать вводимые слова так, чтобы их акустические характеристики значительно разнились. Это упростит обучение и обеспечит более прочное узнавание;

• помнить, что слова, которые ясно различимы в текстовом виде или человеческим ухом, не обязательно столь же отчетливо различимы опознавательным устройством речи;

• дать пользователю возможность включать и отключать речевое распознавание и возвращаться к более традиционным видам ввода и вообще взаимодействия с компьютером. Запрограммируйте ключевое выражение, например «Звук, проснись», чтобы включить режим голосового ввода. Аналогично фраза «Звук, спи» могла бы тогда использоваться для остановки речевого ввода;

• запрограммировать ключевое слово, чтобы с его вводом можно было остановить или отменить неправильные действия;

• обеспечить адекватную обратную связь, информирующую вас о том, что система интерпретировала речевой ввод, можно со звуковым либо с визуальным сигналом;

• обеспечить пользователю, если необходимо, возможность исправлять ошибки перед утверждением ввода либо отменять предыдущий ввод. Однако даже низкий процент ошибок может оказаться неприемлемым в некоторых ситуациях, например в стрессовых или критических по отношению к безопасности.

Если мультимедийный продукт предназначается для разноязычной аудитории, язык взаимодействия должен выбираться пользователем. Пользователи должны иметь определенную степень контроля над размером и содержанием используемого словаря – например, он должен иметь возможность добавлять синонимы и имена, определяемые им самим. Иногда опознавательное устройство не может осуществить выбор одного из двух или более слов-кандидатов. В таких случаях они должны предоставляться пользователю для выбора и утверждения ввода. Чтобы улучшить точность распознавания, необходимо обеспечить пользователя ручным микрофоном (возможно, включаемым в малоприметном месте пульта). Слова речевого ввода должны быть тщательно отфильтрованы, с тем чтобы на каждой стадии требовались одна-две команды. Однако иногда может быть полезным включение одной-двух более длинных фраз, которые будут более четко отделять рабочий ввод от свободной речи во время отдыха, поскольку содержат больше информации.

6.3.3. Речевой выход

Разговорный язык особенно хорош для улучшения понимания, потому что слово произнесенное имеет множество дополнительных признаков, таких, как интонация, тембр, нюансы произношения, настроение – в общем, все то, что недоступно для слова написанного. Речь может использоваться для выделения, дополнения или объяснения текста и картинок на экране.

Текст может также быть выделенным и на экране, например полужирное начертание или обведение рамкой во время его громкого произношения, что полезно для каких-то важных определений либо предупреждений. Речь можно также использовать для объяснений, поскольку при этом можно передать более подробную информацию, чем позволяет визуализация, что облегчает понимание и запоминание. Чтение слова вслух может использоваться для демонстрации произношения, например при вводе иностранных слов, технических терминов и акронимов.

Речь может также использоваться как средство запроса пользователя о вводе, обеспечивать инструкции по вводу либо объяснять образы, предъявляемые на экране, скажем изображения Тадж-Махала может сопровождать речевой комментарий. Чтобы помочь пользователям различать виды речевого взаимодействия (например, просто информационный блок, запрос ввода и т.д.), полезно иметь набор различных голосов.

Говорение не должно превышать 45 с, если при этом на экране не происходят какие-либо события. Однако при этом требуются все же 3-4 предложения минимальной длины, чтобы избежать ощущения разрывов речи. Синтетическую речь следует использовать, если текст генерируется в процессе выполнения каких-то операций, тогда как оцифрованная речь (наговоренная профессиональными дикторами) должна предназначаться для текста, который известен уже в процессе создания программы. Использование различных голосов целесообразно для создания иллюзии реальности либо для дополнительного обозначения различных аспектов информации. Например, сообщение-предупреждение и сообщение-помощь должны передаваться различными голосами, значение которых должно распознаваться сразу. Для достижения подлинности полезно использовать записи настоящих звуков из нужного контекста. Например, можно воспользоваться промышленными звуками завода как фоном при интервью с рабочими на заводе. Целесообразно показывать текущую позицию и общую длину последовательности речи на шкале времени.

Пользователи предпочитают и лучше понимают аналоговую или цифровую форму воспроизведения речи, чем синтетически сгенерированную. Это, в частности, предполагает желательность введения в синтезированную речь интонации, ритма и др., если это технически осуществимо и экономически не слишком обременительно. В качестве альтернативы сейчас можно генерировать речь, используя ранее записанные компоненты живой человеческой речи. Такая технология обеспечивает лучшее понимание и удовлетворенность.

Речевой выход предпочтительнее визуального в тех случаях, когда:

• сообщение срочное или требует немедленного действия;

• пользователь не может оставаться в положении, где он (она) держит в поле зрения

монитор компьютера;

• визуальный канал поврежден вследствие действия факторов окружающей среды или имеет дефекты;

• визуальный канал перегружен, например, множеством задач, отображаемых на экране.

Визуальный вывод предпочтительнее речевого в тех случаях, когда:

• сообщения повторяются;

• к сообщению могут обращаться более одного раза (например, инструкция по последовательности операций);

• сообщение частное или конфиденциальное для пользователя;

• вокруг слишком шумно.

Другие недостатки речевого вывода заключаются в более медленном слушании речи по сравнению с чтением текста, что может повышать когнитивную нагрузку на пользователя.

6.3.4. Музыка

Музыка может давать существенную информацию. Например, в мультимедийном представлении о Моцарте очень хорошо было бы дополнить изображения и текст отрывками из его произведений. Музыка, используемая в данном случае в качестве фона, может стимулировать (или ослаблять) внимание пользователей, добавляя эмоциональную окраску происходящего на экране.

При любых аудиовключениях (тем более музыкальных) важно точно указывать текущее положение каждого звука (его соответствие определенному тексту и изображению) и показывать длину музыкальной последовательности на шкале времени. При использовании музыкальных фрагментов следует помнить о соблюдении авторских прав. Программные средства аудиосинтеза и, в частности, синтезаторы музыки, музыкальные библиотеки, средства обработки музыки и создания модульной музыки можно найти на сайте http://citkit.ru/section/music/1.html

6.3.5. Звуки

Для улучшения интерфейса могут использоваться не только речь и музыка, но и другие звуки – тоны, разные акустические сигналы, звуки живой природы. Тоны и акустические сигналы обычно содержат информацию о состоянии системы. Акустическими могут, например, быть предупреждающие сигналы: сигнал ошибки или случайные шумы типа звука дисковода. Тоновыми могут быть сигналы: «вызов» (звонок), «занято», «ожидания вызова» и др.

Звуковые эффекты могут также использоваться для создания ощущения реальности мультимедийной сцены, например объединяя изображение леса с его звуками. Преимущества использования звуков следующие:

• вносят ощущение реальности в визуально представляемую сцену;

• привлекают внимание к визуально предъявляемой информации (не требующей постоянного внимания);

• эффективны, если информируют о необходимости немедленного действия;

• являются простым средством передачи информации о состоянии системы;

• могут быть полезным дополнением визуальной информации либо использоваться, если применение визуальной информации невозможно.

Недостатки использования звуков следующие:

• отсутствие определенных звуков, связанных с конкретным объектом;

• очень низкая способность передачи больших объемов информации;

• трудность их запоминания;

• отсутствие интуитивного значения;

• низкая способность к представлению описательной информации;

• низкое информационное содержание, что требует дополнения и пояснения другими формами информации (текстом, образами).

При использовании звуков требуется придерживаться некоторых общих рекомендаций.

Количество различных сигналов и тонов должно быть минимальным, кроме того, нужно, чтобы их частотные характеристики были предметом внимательного изучения. Например, более высокие тоны хороши для передачи срочных сообщений, но пожилые пользователи плохо их различают. Установлено, что большинство людей могут различить не более пяти тонов разной высоты. Рекомендуемая частота сигналов и тонов должна быть между 500 и 3000 Гц.

Для сигналов следует использовать сложные звуки (например, с многократно меняющимися частотами или комбинацию звуков), так как их легче идентифицировать, чем простые. Временная задержка между каким-то изменением в системе и соответствующим звуком (сигналом или тоном) не должна превышать 1/2 с.

Необходимо обеспечивать регулирование громкости сигналов и тонов, чтобы пользователи могли устанавливать ее соответственно своим предпочтениям и уровню помех. И конечно, пользователь должен иметь возможность выключить звуки, скажем, чтобы не доставлять неудобства другим людям. Сигналы и тоны целесообразно дублировать визуальной информацией там, где возможно. Например, удобство использования телефона можно повысить, продублировав его акустический сигнал (звонок) загорающейся лампочкой, что особенно важно для людей со сниженным слухом; добавление тактильного сигнала (виброзвонок) еще больше повышает юзабилити такого телефона. Периферийные звуки, например звук дисковода, играют важную роль привычной обратной связи; если они исчезают благодаря, скажем, техническим усовершенствованиям, следует рассмотреть возможность сохранения обратной связи, хотя, возможно, и в иной форме. Интенсивность, или громкость, фоновых окружающих шумов должна быть минимально низкой и не превышать существующих санитарных норм.

Обычно легче распознаются звуки, отличающиеся от других такими признаками, как тембр или шаг ритма. Если в основе распознавания лежит последовательность тонов, следует проиграть эту последовательность не быстрее чем четыре тона в секунду, иначе порядок тонов не будет воспринят. Широкополосный звук (0,1-7 кГц) обеспечивает естественность и ясность человеческой речи. Высокая же точность воспроизведения звука, обеспечиваемая полосой пропускания 40-12,5 кГц, не является необходимой (кроме специальных приложений типа музыкальной передачи).

Если в приложении используется больше одного-двух акустических фрагментов, следует достичь их максимальной различимости путем изменений не одной, а нескольких характеристик. Другими словами, изменять полезно и тембр, и ритм, и другие характеристики. Предупреждающие сигналы и сигналы тревоги должны значительно отличаться от других, чтобы привлечь внимание.

Если в мультимедийном интерфейсе звуки используются для навигации, то более низкие звуки должны соответствовать более высоким уровням иерархических меню. Если тоны используются при работе с массивами данных, то более низкие тоны должны отражать большие значения, при этом следует стремиться к высокой корреляции между звуковым и графическим представлениями данных.

6.4. Интеграция сред мультимедиа

6.4.1. Общие положения

Успех мультимедиа больше зависит от правильного сочетания различных сред и способов их реализации, чем от многообразия сред. Например, если качество изображения весьма высокое, но аудиосопровож-дение плохое, может наблюдаться раздражающий дисбаланс восприятия. В целом если на человека одновременно воздействует более чем одна среда, это улучшает понимание информации. Очень внимательно, однако, следует относиться к интеграции медиасред.

В общем случае вся важная визуальная информация должна дублироваться, т.е. представляться еще и в аудио– и/или тактильной форме. Особое внимание следует обращать на

характер обратной связи, например при нажатии на виртуальную клавишу или при выборе какого-то элемента дисплея. Пользователь всегда должен иметь возможность убрать или добавить дублирующую информационную среду (скажем, звук). Также необходимо предоставить пользователю возможность регулировать визуальные характеристики изображения (контрастность, яркость, цветовая гамма, вид и частота мигания курсора и др.).

Аналогично важную звуковую информацию полезно при возможности дублировать визуальной и/или тактильной. Это включает любой речевой вывод, а также звуковые сигналы, особенно сигналы тревоги. Поскольку люди с дефектами слуха полагаются больше на зрение, важно, чтобы качество отображения было высоким, а освещение окружающей среды – соответствующим. Шум от вентилятора, дисковода или принтера должен быть минимизирован, так как это ухудшает качество звукового выхода.

Интеграция разных модальностей требует определенной стратегии. Например, одновременно следует использовать только две зависящие от времени модальности (скажем, визуальную и звуковую).

Следует стремиться к тому, чтобы наибольшее количество важной информации дублировалось в разных модальностях. Например, заголовки (или подзаголовки), выделенные части текста, ключевые слова и др., наряду с визуальным представлением, могли бы дублироваться и аудиовыходом (речевым), чтобы пользователь мог выбрать предпочтительный для него вариант.

6.4.2. Сочетание звуковой и визуальной информации

При объединении звуковой информации со зрительной (типа текста или изображения) важно, чтобы эта связь была достаточно ясной. Например, система могла бы предъявить иконку динамика рядом с соответствующей частью текста. Пользователь в этом случае мог бы нажать на эту иконку, если он хочет выбрать речь. Обычно лучше, если речь точно сопровождает текст на экране, за исключением случаев, когда требуется составить некое резюме текста. Весьма полезно также подсвечивать текущие участки текста в момент их произношения. Это особенно важно в программных продуктах, направленных на изучение иностранного языка, когда надо связать начертание слова и его произношение. В этих случаях необходимо соблюдение синхронности подсветки текста и его озвучивания.

В случае параллельного выхода нескольких звуковых последовательностей они могут мешать друг другу. Например, речевой выход может сопровождаться фоновой музыкой, и в случаях особенного внимания к произношению (как, скажем, при изучении иностранных языков) такой фон может быть помехой. В таких случаях на время говорения гораздо предпочтительнее приглушить фоновый звук, чем выключить его совсем. Полное выключение фонового звука явится менее естественным и будет служить дополнительным раздражителем.

Нужна ли визуализация говорящего во время речевого вывода? Это зависит от предмета и цели сообщения, как и от того, включено ли оно в передний план либо является фоном. Например, если речевой вывод состоит только из краткого сообщения, инициированного системой, или краткого пояснения, данного по запросу пользователя, визуализация диктора не требуется. Аналогично если речевое сообщение логически связано с основным визуальным изображением, например диаграммой, нет нужды показывать диктора, поскольку это будет скорее раздражителем для пользователя. В то же время если проводится спонтанное интервью в процессе какого-то действия или если устный отчет является частью мультимедийной продукции, говорящий должен обычно быть видимым, так как визуальная сцена является важной фоновой информацией. Наконец, при наличии текущих комментариев, монологов или группового обсуждения без обильного иллюстративного материала визуальная демонстрация говорящих людей очень желательна, поскольку пользователь сможет следить за ходом мысли и поведением участников – один только звук не дает в полной мере такой возможности.

Для правильного сочетания аудио-, видео– и анимационной составляющих мультимедиа чрезвычайно важна их синхронизация. Необходимо иметь возможность взаимной регуляции всех таких составляющих для представления осмысленного хода информации без перегрузки пользователя. Связи между составляющими можно определить как тройственные: временные

(чистая синхронизация), объектные (порядок следования) и комбинации их. Представляя изображение говорящего человека, важно соблюсти синхронизацию движений губ со звуками. Плохая синхронизация более заметна, если звук опережает видимые движения.

6.5. Навигация, ориентация и доступ

6.5.1. Общие положения

Легкость, с которой пользователь перемещается в системе, всегда хорошо представляя свой путь, важна для общего уровня юзабилити ПИ. Одна из наиболее распространенных ошибок в ПИ мультимедийных систем состоит в демонстрации на экране привлекательного, многоцветного, графически богатого объекта, но без полной ясности, где у него область управления, область ввода, область вывода и где просто фон. Создание общего стиля для областей управления, ввода и вывода минимизирует усилия пользователя по овладению системой, к чему, собственно, и следует стремиться.

Всегда целесообразно давать краткие, ясные пояснения, которые всегда легкодостижимы. Такие пояснения могут иногда предусматривать и некоторые практические упражнения до начала основной работы. Это полезно и для новичка, и для эпизодических пользователей, которые могут не помнить всех возможностей системы. Структура должна быть представлена таким образом, чтобы пользователи легко могли вернуться назад, к последней развилке и попробовать иной путь. Например, система может предполагать действия пользователя по нанесению пометок на те или иные области или объекты для их дальнейшего использования.

Карты, схемы или другие формы навигации должны предусматривать какие-то указатели того, в каком месте сейчас находится пользователь и какие места еще остались неотработанными. Этой цели часто служат цвета или флажки. Учет индивидуальной скорости обучения, как правило, предпочтительнее установления временных ограничений, поэтому полезно отражать не абсолютные, а относительные длины каждой единицы системы, чтобы пользователь мог оценить, сколько уже сделано и сколько осталось. Для этого используют разные скользящие шкалы. Однако слишком большое количество метафорических конструкций или необычных ассоциаций, искусственно связанных с привычными объектами, может сбивать с толку, как, впрочем, и наличие очень близких по смыслу метафор.

6.5.2. Использование метафор

Возможности мультимедиа многократно увеличивают эффект использования метафор для навигации. Визуальные метафоры, подобные книгам, помещениям и домам, картам или подъемникам, полезны для демонстрации отношений. Предполагается, что пользователям хорошо знакомы реальные предметы или явления, соответствующие этим метафорам, и они, опираясь на это знание, могут быстрее понять значение, скажем, каких-то средств управления ПИ.

Технические метафоры, подобные средствам управления технического оборудования, например видеомагнитофона, также обеспечивают пользователю хорошо знакомую среду.

Структурные метафоры можно разделить на три группы: последовательности, деревья и сети. Линейные структуры или последовательности показывают порядок следования каких-то информационных фрагментов; обычное оглавление, к примеру, позволяет сразу увидеть такого рода последовательность. Древовидная или ветвящаяся структура часто представляет иерархические меню; для начинающих пользователей предпочтительнее, например, небольшая ширина меню и значительная глубина, а для опытных – наоборот. И наконец, правильное представление связей в сетевой структуре гипертекста обеспечивает мощное и гибкое средство для доступа к нужной информации.

Выделяют следующие общие стили метафор, которые с успехом используются в ряде ПИ:

• метафора рабочего стола: представление предметов и действий с ними подобно таковым

за обычным рабочим столом. Но этой метафоры оказывается недостаточно из-за роста разнообразия процедур ввода/вывода, и тогда требуются дополнительные обозначения описания;

• метафора помещений: расширяет метафору рабочего стола до расположения комнат в доме. Множество комнат, в которых пользователь мог бы разместить соответствующие работы, существенно облегчает переход от одной работы к другой. Изображение двери, например, предлагает переход в другую часть приложения;

• метафора чередования (метафора стеков): рассматривает приложение как последовательность некоторых единиц, которые соединены посредством определенных связей. Пользователь перемещается от одной единицы к другой посредством связей, отраженных в метафоре чередования.

Другой способ обеспечения навигации – дать ясную и детальную картину текущего информационного окружения и одновременно с этим – общую картину, позволяющую представить информацию, которая отсутствует в данный момент. Для отражения такой общей картины подходящими являются метафоры типа фотообъектива «рыбий глаз», увеличительного стекла, стены в перспективе.

Метафоры прежде всего являются визуальными подсказками, такими же, как пиктограммы или физические объекты; например, изображение телефона используется для обозначения речевого сообщения. Однако и текстовые, и аудиоподсказки полезны, если надо прояснить имя или какую-то функцию при отсутствии непосредственной видимости на экране. Такие подсказки могут активизироваться, если пользователем выбраны соответствующие пиктограммы или объекты. Когда же пользователи хорошо знакомы с приложением, его метафоры быстро становятся практически невидимыми, а метки и подсказки могут оказаться излишне назойливыми.

Использование пиктограмм или объектов внутри метафорической структуры должно соответствовать некоторым единым базовым установкам. Например, в графических интерфейсах при выборе тех или иных опций их начертание должно подсвечиваться, изменять цвет и/или мерцать. Активные объекты должны подсвечиваться, в то время как неактивные должны иметь меньшую освещенность.

6.5.3. Навигационные структуры

Перед человеком, путешествующим по какой-то области Всемирной сети, стоит задача построения ментальной модели огромного пространства, подобно тому как перед любым из нас, оказавшимся в незнакомом городе, стоит проблема построения модели «как этот город устроен». И если в последнем случае есть привычное и удобное решение – карта города (первое, что обычно делают в незнакомом городе – покупают его карту), то пользователь Всемирной сети лишен такой возможности.

Навигация в сети должна информировать пользователей о том, где они располагаются, где находятся объекты, которые они ищут, и как добраться до объектов поиска.

Частный, но от этого не менее важный случай – как найти на сайте нужную информацию. Как правило, известно, что она на сайте есть, но через какие ссылки дойти до нее с главной страницы, знают только создатели. Знают, но не могут сообщить об этом пользователям в первую очередь потому, что не обеспечили адекватных средств. Довольно часто наблюдаются попытки решить эту проблему только за счет локальных поисковых систем, что выглядит, как минимум, неуклюже. Это в лучшем случае напоминает алфавитный список улиц и не привносит ясности.

Существуют различные схемы сайтов, и навигационные структуры должны отражать их. Схему сайта целесообразно выработать в самом начале работы над его архитектурой. Она может быть широкой или глубокой. Широкая схема сайта используется место при наличии множества элементов на одном уровне, т.е. все расположено на расстоянии одного щелчка мыши. Глубокая же схема включает в себя уровни, подуровни и, возможно, подподуровни. Чтобы добраться до некоторых элементов такого сайта нужно сделать иногда 7-9 щелчков мыши.

При небольшом количестве страниц сайта лучше использовать простую и ясную широкую схему. При большом же количестве страниц, разделов, больше, наверное, подойдет многоуровневая древовидная схема.

В целом же создание навигационной структуры – процесс творческий, и схемы сайтов, а тем более их реализация, меняются от дизайнера к дизайнеру. Но без знания накопленного опыта создание действительно удобного и привлекательного сайта сомнительно.

Различают три основных вида навигации:

• глобальную – это набор инструментов навигации, которые обеспечивают переход к ключевым областям, категориям сайта и которые присутствуют и не меняются во всех разделах или на всех страницах сайта;

• локальную – это навигация к подразделам внутри разделов или страниц сайта;

• разбивка на страницы – последовательный переход при просмотре перечня каких-то предметов, не разделенных на группы, категории.

Глобальная навигация обычно реализуется с помощью гиперссылок на вкладках поперек страницы, т.е. перечня мест, куда можно перейти с каждой из них (см. например, сайты Ошибка! Недопустимый объект гиперссылки., Ошибка! Недопустимый объект гиперссылки. и др.). Такие вкладки чаще располагают в верхней части страницы, но бывает, что и в нижней (см. сайт «Big Healeys»). С ними соседствуют наиболее важные сервисы, такие, как, например, строка поиска (товара, услуги, какой-либо информации). Вместо вкладок некоторые дизайнеры используют для глобальной навигации стрелки, на которых как бы «нанизаны» все разделы сайта (см., например, сайт «E*Trade»).

Локальная навигация позволяет перемещаться по подразделам внутри сайта. Это может быть реализовано с помощью перехода от подраздела к подразделу только через родительскую категорию (название способа – «pogosticking», т.е. перепрыгивание), либо переходом к любому подразделу из любого другого подраздела того же уровня (название способа – «crabwalking», т.е. прогулка краба). Соответственно, и дизайн локальной навигации может иметь форму в виде, скажем, перечня ссылок на начальной странице с цепочками ссылок на последующих (очень яркий пример – стиль сайта «Yahoo!») в первом случае или в виде боковой панели навигации на всех страницах – во втором.

Разбивка на страницы – достаточно простой инструмент, позволяющий перебирать похожие предметы в разделе, читать текст в режиме он-лайна. Средствами навигации при этом могут быть виртуальные клавиши типа «Предыдущая„ „Главная“, „Следующая“ или номера страниц, позволяющие сразу переходить к любой странице. Некоторые сайты, представляющие определенные товары, обеспечивают их последовательный просмотр, а также возможность просмотра всего сразу на одном экране (например, сайт компании «The Gap“ и др.).

Из основных средств навигации можно выделить следующие.

Путеводители. Это навигационная структура, обеспечивающая заранее установленные траектории перемещения. Следует предусмотреть возможность различных траекторий перемещения (вперед, назад, возврат из любой точки траектории перемещения в исходную точку) внутри одной и той же системы для удовлетворения интересов различных пользователей. При этом всегда должен сохраняться общий вид, т.е. указание на те части, которые отсутствуют в текущей индикации.

Нередко у пользователей возникает неопределенность – какой путь на путеводителе выбрать. Пункты меню высокого уровня, как правило, недостаточно детальны для описания всех опций нижних уровней, и поэтому всегда найдутся опции, которые трудно сразу понять. Наличие возможности для пользователя перемещаться назад – один из способов преодолеть эту трудность. Другой способ – предусмотреть возможность «обзора вперед» или «предварительного просмотра», для чего пользователь может кликнуть на пункте меню (и удерживать нажатой клавишу мыши), чтобы увидеть следующий уровень меню (показанный в окне рядом) перед тем, как этот пункт будет выбран. Использование «прохода сквозь систему» или каскадных меню позволит пользователю исследовать структуру системы до выбора какой-то опции. Здесь пользователь двигает мышь вниз по главному меню и затем от конкретного пункта главного меню к пункту меню нижележащего уровня. Аналогично он может перемещаться вверх, от пункта меню нижнего уровня к вышележащему меню. В целом

тщательная разработка пунктов меню – простейший и наиболее эффективный способ облегчить навигацию.

Примером очень хорошего путеводителя как средства навигации может служить схема метро. Интересно, что схема лондонского метро в 2006 г. признана верхом дизайнерского искусства, победив на конкурсе, организованным лондонским Музеем дизайна, в котором определялись три высших достижения дизайна XX в. Эта схема разработана еще в 1931 г. Гарри Беком, который первым окрасил каждую линию в особый цвет, а также изобразил станции пересадок в виде кружков. Следом расположился истребитель «Спитфайр» времен Второй мировой войны, также в тройку лучших попал сверхзвуковой пассажирский лайнер «Конкорд». Примечательно, что годом раньше схема лондонского метро также опередила всех своих британских конкурентов, заняв 3-е место после шариковой ручки и канцелярской скрепки, которые были изобретены умельцами из других стран. Схемы метро других городов (и Москвы в том числе) повторяют тот же дизайн.

Блок-схемы. Структура узловых пунктов и отношения между ними могут быть наглядно отражены с помощью блок-схемы. Примеры – блок-схемы структур проблем, стратегий решения, аргументов. Они могут эффективно использоваться для документации процессов решения и развития, обеспечивать доступ к дополнительной информации.

Кластеры. Для структурирования информации и наглядного представления отдельные элементы должны быть правильно сгруппированы. Отношения между единицами информации внутри группы должны быть близкими, в то время как между различными группами они должны быть существенно менее близкими, тогда эффект наглядности группировки будет наилучшим.

Координатная сетка представляет данные в двух измерениях и, таким образом, отражает динамику данных с изменением двух переменных. Регулярная структура координатной сетки облегчает ориентацию для пользователя.

Основные задачи навигации можно свести к передаче информации о том, где вы сейчас, что вы можете делать, куда вы движетесь по сайту или что произойдет далее, где вы уже были или что вы уже сделали.

Один из самых существенных вопросов, возникающих у пользователя при повторном посещении web-страницы: что нового появилось на ней с момента предыдущего посещения и когда она в последний раз менялась? Поэтому на страницах обязательно должны быть отображены дата последнего изменения либо значок «New» для указания на новую информацию. На рис. 6.12 показан распространенный способ отображения пути в иерархии сайта, облегчающий навигацию.

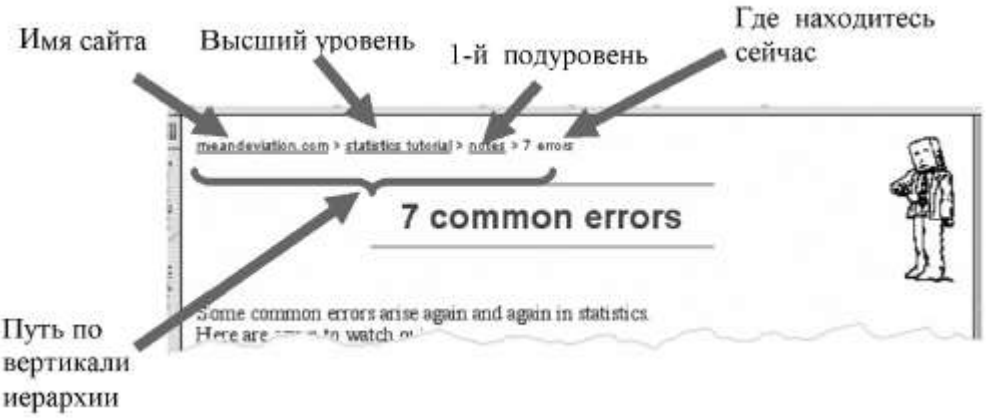

Рис. 6.12. Отображение пути в иерархии сайта, облегчающее навигацию

Предлагается множество интересных интерфейсных решений по обеспечению наглядности навигации. Например, предложенное А. Бе-лышкиным и др. [13] новое решение полосы прокрутки. Они развили идею Б. Шнейдермана, предложив использовать пространство полосы прокрутки не только для стандартной демонстрации размера документа, но и одновременно для символического отображения его содержимого. Это позволяет быстро идентифицировать фрагменты (например, страницы документа) и сразу переходить к нужному

месту. Кроме того, изменять полосу прокрутки предложено в зависимости от типа приложения, в котором она используется. В разных контекстах этот усовершенствованный интерфейсный элемент может решать ряд дополнительных задач, необходимых пользователю. В текстовых редакторах он может содержать эскизы страниц (это особенно важно, когда страницы кроме текста содержат рисунки, таблицы и т.д.), в поисковых системах на полосе прокрутки можно отображать местоположение найденных слов в тексте (это удобно при больших объемах информации).

В каталогах файлов полоса прокрутки может отображать принцип сортировки файлов и тем самым помочь быстро находить нужные. В графических файлах (например, в такой популярной системе, как фотошоп) на полосе прокрутки можно отображать отмасштабированную схему картинки, дающую о ней общее впечатление (это удобно, если в процессе работы приходится сильно увеличивать изображение и появляются горизонтальные и вертикальные прокрутки). В длинных web-формах отображение преварительного просмотра страниц на полосе прокрутки также принесет свои выгоды для пользователя: можно быстро определить местонахождение нужной страницы, увидеть превью всей страницы целиком и составить о ней представление; и так как превью отображает разбивку страницы, сразу будет видно, где заголовок, где низ, где подзаголовки, иллюстрации и т.д. К сожалению, эти предложения пока не реализованы.

Другой пример очень удачного навигационного решения для просмотра изображений разработан в лаборатории человеко-компьютерно-го взаимодействия Университета штата Мэриленд в США (www.cs.umd. edu/hcil/photomesa). Программа эта называется «PhotoMesa». В ней использована концепция так называемого zoomable interface, т.е. интерфейса с возможностью увеличения. На практике это выглядит следующим образом. В рабочем окне «PhotoMesa» вы видите сразу все картинки, находящиеся в заданной области (это может быть целый диск, одна или несколько вложенных папок и т.д.). Чем больше исходных картинок, тем меньше отображающие их иконки. Впрочем, картинка под курсором сразу же увеличивается так, чтобы ее можно было подробно рассмотреть. Если же вы хотите вблизи посмотреть сразу на группу картинок, то достаточно щелкнуть по ней мышью, и эта часть окна приблизится к вам, вытесняя все остальное за пределы экрана. По двойному клику оригинал картинки выведется в полноэкранном режиме. Таким образом, можно одним взглядом охватить любую фотоколлекцию, а в следующий момент уже рассмотреть заинтересовавшее вас изображение во всех подробностях. И при этом не надо возиться с разными мелкими кнопками, выпадающими меню, полосами прокрутки и списками директорий. Интерфейс интуитивный и очень удобный.

Пользователь должен иметь возможность выбрать различные опции предлагаемых средств навигации. Должно быть показано, к какой информации обеспечивают доступ те или иные навигационные шаги, в какой форме эта информация представлена – видео, графики, речи, карты и т.д. Необходимо обеспечить пользователя дополнительной информацией о возможных связях, т.е. какая еще информация может быть вызвана, что достигается с помощью изменения формы курсора (обычно на кисть руки с поднятым указательным пальцем).

Важна также и поддержка поиска. Обычно пользователь вводит имя или имена объектов, которые надо найти. Поисковая система ищет соответствия этим именам в базе данных и выдает сообщение, сколько найдено таких соответствий. Это помогает пользователю решить, отображать ли их все либо производить очередные поиски, сужая их, чтобы последовательно уменьшать число найденных объектов, т.е. сократить область выбора.

Другая технология поддержки поиска состоит в том, что система динамически реагирует на действия пользователя при вводе информации. Например, когда вводится первая буква поисковой строки, сразу показываются все объекты, начинающиеся с этой буквы. После ввода второй буквы область выбора сужается, и из объектов, показанных на предыдущем шаге, остаются те, две первые буквы которых начинаются с введенных двух. Пользователь может в любое время нажать клавишу «ENTER» для выбора текущего элемента, либо перемещать вручную курсор по списку. Эта технология часто избавляет пользователя от необходимости печатать больше нескольких букв имени, которое он хочет найти. Однако требуется тщательное эмпирическое исследование для подтверждения того, оказывает ли такая технология реальную поддержку пользователю.

Как и в остальных вопросах навигации, для поиска важна единая логика (и единство технологии) для разных приложений и продуктов. Весьма наглядный пример этого – Windows Vista, где реализована единая технология поиска – на уровне ядра ОС, клиентских приложений и серверов, с едиными механизмами расширения во всех продуктах. В окнах обозревателя доступна функция быстрого поиска. Есть инструмент сохранения поисковых запросов в XML.

Контрольные вопросы

1. Определение и понятие мультимедиа. Области применения, возможности.
2. Формы представления информации, их сравнительная оценка.
3. Преимущества текстовой формы представления информации.
4. Основные требования к текстовому изложению материала.
5. Правила расстановки ссылок в тексте, примеры.
6. Оптимальные размеры символов, строк и интервалов в тексте.
7. Требования к цветовому кодированию, преимущества и недостатки цветового кодирования.
8. Основные типы графических описаний. Преимущества и недостатки графического представления информации.
9. Определение видео, анимации, изображения (фотографии). Их сравнительные возможности, целесообразность применения, преимущества и недостатки, основные требования.
10. Разновидности акустической формы представления информации. Их сравнительные возможности, целесообразность применения, преимущества и недостатки, основные требования.
11. Правила интеграции звуковой и визуальной информации.
12. Основные средства мультимедиа в обеспечении навигации.
13. Виды навигационных структур, их возможности, области применения.
14. Основные правила навигации и средства их выполнения.
15. Типичные навигационные решения в web-сайтах, их сравнительная оценка.
16. Средства и методы достижения стабильной ориентации в процессе поиска.

Литература

Основная

1. Dix A., Finlay J. and others (Eds). Human-Computer Interaction. – Printed in Great Britain, Glasgow. Publisher – Prentice Hall: 3rd ed., 2004. Ch.21.
2. Уодтке Кр. Информационная архитектура: чертежи для сайта. – М.: КУ-ДИЦ-ОБРАЗ, 2004.
3. Прайс Дж., Прайс Л. Текст для web. Доступность и привлекательность. Изд-во Вильямс, 2003.
4. Shneiderman B. Designing of the User Interface. 3rd ed. – Addison Wesley, 1998.
5. Кирсанов Д. Веб-дизайн: книга Дмитрия Кирсанова. – СПб: Символ-Плюс, 2006.
6. РаскинДж. Интерфейс: новые направления в проектировании компьютерных систем. – СПб: Символ-Плюс, 2005.
7. Vossen P., Maguire M., Graham R., Heim J. Design Guide for Multimedia. 2nd ed. (Version 2.1). Information Engineering Usability Support Centers, HUSAT Research Inst., 1997. Telematics Applications Project.
8. Черкасский В. Т. Эффективная анимация во Flash. – М.: КУДИЦ-ОБРАЗ,2002.
9. Сеймур-Коэн Луэнн. Секреты дизайнера. Профессиональные приемы в Adobe Photoshop 7 и Adobe Illustrator 10. – М.: КУДИЦ-ОБРАЗ, 2003.
10. www.artlebedev.ru/kovodstvo/83/

Дополнительная

11. Bearne M., Jones S. and Sapsford-Francis J. Towards usability guidelines for multimedia

systems // ACM Multimedia 94. Proc. of the 2nd ACM Intern. Conf. on Multimedia. – San Francisco, Oct. 1994. P. 105-110.

12. Харовас П., Кундерт-Гиббс Дж., Ли П. Maya complete. Уроки мастерства. – ДМК Пресс, 2001.

13. Белышкин А., Бойко И., Филатова Е. Новая жизнь полосы прокрутки. www.usethics.ru/lib/scroll_bar/index.shtml

14. Гото К., Котлер Э. Веб-редизайн: книга Келли Гото и Эмили Котлер. 2-е изд. – СПб: Символ-Плюс, 2006.

15. Bevan N, Bowden R. Usability Context Analysis – a practical guide. V.4.02. NPL Usability Services. National Physical Lab., Queens Rd, Teddington, Middlesex, 1996.

16. Clarke A. M., Allison G., (Eds.). Human Factors Guidelines for Designers of Telecommunication Services for Non-Expert Users. Vol. 1. P.112; Vol. 2. P.302. HUSAT Research Inst., 1996. Loughborough Univ., Elms Grove, Loughborough, Leics, UK.

17. Poulson D., Ashby M., Richardson S. (Eds.). USERfit A practical handbook on user-centred design for assistive technology. Handbook produced within the European Commission TIDE programme USER project. HUSAT Research Inst., 1996. Loughborough Univ., Elms Grove, Loughborough, Leicestershire, UK.

ПРИМЕРНАЯ УЧЕБНАЯ ПРОГРАММА ДИСЦИПЛИНЫ «ЧЕЛОВЕКО-КОМПЬЮТЕРНОЕ ВЗАИМОДЕЙСТВИЕ»

ЧЕЛОВЕКО-КОМПЬЮТЕРНОЕ ВЗАИМОДЕЙСТВИЕ

Для направления подготовки дипломированного специалиста…

ВИД УЧЕБНЫХ РАБОТ	ОБЪЕМ РАБОТ, Ч		
	Всего	7-й семестр	8-й семестр
		... недель	... недель
Выделено на дисциплину	153	...	...
Аудиторная работа	102	...	...
Лекции	102	...	...
Семинары	34	...	...
Лабораторные работы	–	...	...
Самостоятельная работа	17	...	...
Домашнее задание	8	...	...
Курсовая работа	–	...	...
Курсовой проект	–	...	...
Сроки выполнения контрольных мероприятий			
Домашнее задание	–	–	... неделя
Рубежный контроль	1	–	... неделя
Контрольная работа	–	–	–
Оценка знаний			
Экзамен	1	–	Экзамен
Зачет	–	–	–

Город, 20.... г.

I. Цель и задачи дисциплины

Основной целью дисциплины «Человеко-компьютерное взаимодействие» является получение студентами специальных знаний и представлений о способах и средствах разработки пользовательского интерфейса, компоновке и размерах рабочего места пользователя, требованиях к средствам отображения информации и ввода данных, методах и процедурах разработки и оценки взаимодействия «человек-компьютер». Рассматриваются практические примеры.

Полученные знания могут быть использованы в практической работе для более полного учета человеческого фактора при разработке пользовательских интерфейсов, web-сайтов, в оценке дизайнерских решений пользовательского интерфейса, для сокращения ошибок и времени освоения программных продуктов, повышения эффективности использования программного обеспечения.

Основными задачами курса

• Изучение основных факторов, влияющих на качество человеко-компью-терного взаимодействия; содержания работ, направленных на формирование качественного пользовательского интерфейса; путей, методов и инструментария оценки и создания качественного пользовательского интерфейса.

• Изучение основных направлений учета человеческого фактора в процессе разработки пользовательского интерфейса, а также наиболее характерных ошибок и путей их предотвращения.

• Изучение и освоение наиболее распространенных программно-инструментальных средств создания и оценки качественного человеко-компью-терного взаимодействия.

II. Содержание дисциплины

Лекции

№ п/п	Основные разделы	Объем, ч
1	Компоновка рабочего места при работе на компьютере. Следствия нарушения правил компоновки. Планировка рабочего помещения. Требования и нормативные документы. Средства моделирования	4
2	Понятие активного и пассивного комфорта. Характеристики рабочего кресла, обеспечивающие активный и пассивный комфорт для человека	2
3	Освещенность рабочего помещения и рабочего места. Типы освещенности, разновидности источников освещения. Требования к характеристикам монитора (яркость, контрастность, размеры символов и пр.)	2

№ п/п	Основные разделы	Объем, ч
4	Устройства ввода информации. Разновидности и характеристики клавиатур. Устройства координатного ввода (мыши, трекболы, джойстики и пр.). Преимущества и недостатки различных устройств ввода информации. Требования и нормативные документы	4
5	Виды интерфейсов, их функциональные возможности. Преимущества трехмерного интерфейса. Возможности интерфейсов прямой манипуляции. WIMP-интерфейс, его составные части, функции каждой из частей. Принцип WYSIWYG и функции, которые он обеспечивает	4
6	Виртуальная реальность, основные признаки. Характеристики пользовательского интерфейса, обеспечивающие виртуальную реальность	4
7	Принципы создания объемных изображений, их сравнительная оценка, варианты технической реализации. Возможности объемных изображений для создания пользовательского интерфейса	4
8	Эволюция человеко-компьютерного взаимодействия (изменения форматов данных, мерности представления пространства, способов и средств взаимодействия)	4
9	Модели пользователя и пользовательские профили. Типы, примеры использования, сравнительный анализ	8
10	Технологии декомпозиции действий пользователя и процесса функционирования системы. Метод GOMS, описание, возможности, применение	4
11	Портрет потенциального пользователя (персоны), варианты и правила описания, примеры использования	2
12	Оценка пользовательского интерфейса. Виды оценок, классификация, описание, сравнительный анализ, примеры. Измеряемые параметры	6

№ п/п	Основные разделы	Объем, ч
13	Критерии качества и методы оценки пользовательского интерфейса	8
14	Принципы интуитивной понятности пользовательского интерфейса. Роль метафор, их преимущества и недостатки, примеры метафорических дизайнерских решений. Стандарты и язык шаблонов, правила описания шаблона, библиотеки шаблонов	8
15	Виды обучающих материалов, их классификация. Спиральность текста справки, роль и способы реализации	4
16	Основные принципы и правила создания эстетической привлекательности пользовательского интерфейса. Пути снижения уровня психической напряженности пользователя	8
17	Юзабилити-тестирование пользовательского интерфейса. Цели, содержание на разных этапах разработки, характеристики, метрики, таксономия	4
18	Методики юзабилити-тестирования, их возможности, сравнительный анализ	4
19	Прототипирование пользовательского интерфейса. Версии прототипов. Программные средства создания прототипов, их сравнительные возможности, достоинства и недостатки	4
20	Письменное объяснение дизайнерских решений, целесообразность, методы	2
21	Мультимедиа в пользовательских интерфейсах, возможности, формы представления информации, правила интеграции сред, возможности для навигации, программные средства	8
22	Навигационные структуры, их виды, пути и средства реализации	4
ИТОГО		102

Семинары

№ п/п	Перечень тем	Объем, ч
1	Освоение и решение конкретных учебных задач по оптимизации рабочего места «человек–компьютер» с помощью средств моделирования, визуализации и анализа данных о движениях людей, размерах, позах, требуемых усилиях и т.д. (например, с помощью программ «Observer» и/или «CSM-анализ»)	4
2	Практическое освоение процедуры декомпозиции действий пользователя в конкретных учебных задачах. Применение метода GOMS, расчет характеристик действий, анализ результатов	4
3	Оценка пользовательского интерфейса конкретного клиентского приложения, разработка предложений по его модификации, обоснование программных средств модификации и прогноз результатов	6
4	Освоение работы с шаблонами из библиотеки в Интернете, использование шаблонов для решения учебных задач. Отбор, возможные модификация и использование разных шаблонов-метафор	4
5	Обоснование методики, показателей и проведение юзабилити-тестирования пользовательского интерфейса на разных стадиях разработки в конкретных учебных задачах	6
6	Создание презентационной и псевдореальной версии прототипа конкретного пользовательского интерфейса на разных стадиях его разработки. Использование при этом пакетов MS PowerPoint, MS Visio и Adobe InDesign	6
7	Разработка мультимедийного средства навигации разрабатываемого пользовательского интерфейса в учебной задаче	4
ИТОГО		34

Лабораторный практикум

№ п/п	Перечень тем	Объем, ч
Не предусмотрены		

Темы домашних заданий

№ п/п	Перечень тем домашних заданий. Сроки (недели) их выполнения	Объем, ч
Темы домашних заданий выдаются преподавателем в соответствии с тематикой лекций. Срок сдачи – ... неделя		8

Темы рубежного контроля

№ п/п	Перечень тем, входящих в рубежный контроль. Сроки (недели) их проведения	Объем, ч
1	Рубежный контроль № 1 – темы № 1 – 10	10

III. Учебно-методические материалы

Литература

Основная

1. Dix A., Finlay J. (Eds). Human-Computer Interaction. Printed in Great Britain, Glasgow. Prentice Hall. 3rd ed., 2004.

2. Handbook of Human Factors and Ergonomics (Ed. by Gavriel Salvendy) 2nd ed. Purdue Univ. John Wiley Sons, Inc. – N.Y., 1997. 3. ISO/DIS 9241

4. Головач В. В. Дизайнер, проектировщик, юзабилити-специалист. Окт. 2006 г. www.usethics.ru/lib/types/index.shtml

5. Головач В. В. Прототипирование интерфейсов в Adobe InDesign. Окт.2004г. www.usethics.ru/lib/types/index.shtml

6. Гото К., Котлер Э. Веб-редизайн: книга Келли Гото и Эмили Котлер. 2-е изд. – СПб: Символ-Плюс, 2006.

7. Торрес Р. Дж. Практическое руководство по проектированию и разработке пользовательского интерфейса. Изд.-во Вильямс, 2002.

8. Мандел Т. Разработка пользовательского интерфейса. -М.: ДМК Пресс, 2001.

9. Кирсанов Д. Веб-дизайн: книга Дмитрия Кирсанова. – СПб, Символ-Плюс. Сер.: Библиотека дизайнера, 2001.

10. Вукс Т. Основы web-дизайна. Интерфейс, понятный посетителю сайта. 2005 г. www.i2r.ru/static/255/out_22172.shtml

11. Прайс Дж., Прайс Л. Текст для web. Доступность и привлекательность. Изд-во. Вильямс, 2003.

12. Человеческий фактор: В 6 т. / Под ред. Г. Салвенди. – М.: Мир, 1991.

13. Божко А. Н. Dreamweaver MX. Базовый курс. – М.: КУДИЦ-ОБРАЗ, 2003.

14. Shneiderman B. Designing of the User Interface. 3rd ed. Addison Wesley, 1998

15. Черкасский В.Т. Эффективная анимация во Flash. – М.: КУДИЦ-ОБРАЗ, 2002.

16. Вятчин К. Язык шаблонов в дизайне взаимодействия. www.usethics.ru/lib/ui_templates/index.shtml

17. Нильсен Я. Элементарные основы юзабилити. Авг. 2003 г. www.i2r.ru/static/255/out_20415.shtml

18. www.artlebedev.ru/kovodstvo/

19. Usability в России. www.usability.ru/

Дополнительная

20. www.io2technology.com/salesinquiry

21. www.seereal.com/

22. www.grundig.com/

23. www.optics.com/

24. www1.cs.columbia.edu/graphics/projects/mars/mars.html

25. www.epindustries.com/4Dvideo/index.html

26. Wharton C., Rieman J., Lewis C. and Poison P. The cognitive walkthrough: a practitioner's guide. // Nielsen and R. L. Mack (Eds). Usability Inspection Methods. John Wiley, 1994.

27. Nielsen J. Heuristic evaluation. // Nielsen and R. L. Mack (Eds). Usability Inspection Methods, John Wiley, 1994.

28. Дедков В. Adobe Photoshop. Настольная книга мастера. Изд. Компьютерпресс, 2001.

29. Сеймур-Коэн Луэнн. Секреты дизайнера. Профессиональные приемы в Adobe Photoshop 7 и Adobe Illustrator 10. – М.: КУДИЦ-ОБРАЗ, 2003

30. Харовас П., Кундерт-Гиббс Дж., Ли П. Maya complete. Уроки мастерства. – ДМК

Пресс, 2001.

31. Carroll J. M. HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science (Morgan Kaufmann Series in Interactive Technologies) – Virginia Tech. USA, 2003.

32. Борецкий Ф. Друзья и враги пользователя InDesign. Май 2005 г. www.usethics.ru/lib/types/index.shtml

33. Филатова Е. Руководство пользователя для пользователя. Июль 2006. www.usethics.ru/lib/types/index.shtml

34. Diez M., Sherry L., and Boehm-Davis D. A. Rafiv: A Method For Cognitive Usability Analysis. Proc. of the Human Factors and Ergonomics Society. 48th Annual

Meeting, 2004. P. 396.

35. Diez M. Why A Consumer Electronic Device Is Difficult To Use. Proc. of the Human Factors and Ergonomics Society. 48th Annual Meeting, 2004. P. 927.

36. Беккер Л. 90% всех юзабилити-тестов – бесполезны. www.i2r.ru/ static/255/out_21891.shtml

37. Vossen P., Maguire M., Graham R., Heim J. Design Guide for Multimedia. 2nd ed.(Version 2.1), Information Engineering Usability Support Centers. HUSAT Research Inst. Telematics Applications Project IE 2016, 1997.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1.

Основные понятия и термины, часто встречающиеся при разработке ПИ

Анализ задач (Task Analysis)

Метод, предназначенный для выделения операций, которые пользователь предпринимает, взаимодействуя с системой. Этот метод позволяет дизайнеру взаимодействия определить, какие операции непосредственно необходимы для решения задач, а какие избыточны и являются следствием несовершенства системы.

Анализ требований (Requirements Analysis)

Процесс, осуществляемый с целью описания функций и возможностей, которыми должна обладать разрабатываемая система. Исчерпывающий анализ требований включает различные аспекты, такие, как бизнес-требования, требования, вызванные особенностями целевой аудитории, а также технологические требования.

Анимация. Мультипликация (Animation)[11]

Анимация – технология мультимедиа; воспроизведение последовательности картинок, создающее впечатление движущегося изображения. Восприятие изображения как движущегося возникает при частоте смены видеокадров более 16 в секунду.

Аппаратное обеспечение (Hardware)

Комплекс электронных, электрических и механических устройств, входящих в состав системы или сети. Включает в себя:

• компьютеры и логические устройства;

• внешние устройства и диагностическую аппаратуру;

• энергетическое оборудование, батареи и аккумуляторы.

Аппаратные средства мультимедиа

Подразделяются на основные и специальные.

• Основные средства – компьютер с высокопроизводительным процессором и памятью большого объема, манипуляторами и мультимедиа монитором со встроенными стереодинамиками;

[11] От лат. multiplicatio – умножение.

• Специальные средства – приводы CD-ROM, TV-тюнеры и фрейм-грабберы, графические ускорители, платы видеовоспроизведения, звуковые платы, акустические системы и др.

Аудиовидеосистема (Multimedia System)

Обеспечивает обработку текстовой, графической, звуковой информации, а также видеоданных.

Векторная графика (Vector Graphics)

Метод графического представления объекта в виде отрезков прямых (векторов). В полиграфии векторная графика обычно используется для подготовки макетов.

Видеозахват

Технология мультимедиа; захват и сохранение в цифровом виде отдельных видеокадров.

Видеотехнология (Video Technology)

Технология разработки и демонстрации движущихся изображений. Виртуальная реальность (Virtual Reality – VR)

Высокоразвитая форма компьютерного моделирования, которая позволяет пользователю погрузиться в искусственный мир и непосредственно действовать в нем с помощью специальных сенсорных устройств, которые связывают его движения с аудиовизуальными эффектами. При этом зрительные, слуховые, осязательные и моторные ощущения пользователя заменяются их имитацией, генерируемой компьютером.

Характерными признаками виртуальной реальности являются:

• моделирование в реальном масштабе времени;

• имитация окружающей обстановки с высокой степенью реализма;

• возможность воздействовать на окружающую обстановку и иметь при этом обратную связь.

Виртуальный мир (Virtual World)

Наглядная компьютерная модель реальных процессов и явлений.

Всемирная паутина (Веб), Служба глобального соединения

World Wide Web (WWW, Web); World-Wide-Web service (WWW service)

Всемирная паутина – основная служба в Интернете, позволяющая получать доступ к информации на любых серверах, подключенных к сети. Всемирная паутина организована на принципах гиперсреды.

Веб-сервер (Web -сервер; WWW -сервер – Web Server)

Сервер, обеспечивающий предоставление информации в службе глобального соединения, хранит и предоставляет во внешнюю сеть данные, организованные в виде веб-страниц, отвечает за обработку запросов клиентов к веб-сайту и исполнение CGI-, JSP-, ASP-, PHP– и других приложений.

Воксел (Voxel)

Наименьшая составляющая цифровых трехмерных изображений. Воксел характеризуется значениями, определяющими яркость, цвет, плотность и другие характеристики в соответствующей точке объекта.

Высокоуровневая архитектура интерфейса

Основополагающая часть интерфейса, включающая навигационную систему, структуру экранов и окон, терминологическую базу.

Гиперсреда (Hypermedia)

Технология представления информации в виде относительно небольших блоков, ассоциативно связанных друг с другом.

Гиперссылка Гиперсвязь (Hyperlink)

Гиперссылка – фрагмент HTML-документа, указывающий на другой файл (который может быть расположен в Интернете) и содержащий полный путь (URL) к этому файлу.

Гиперссылка для пользователя – графическое изображение или текст на сайтеили в письме электронной почты, устанавливающие связь и позволяющие переходить к другим объектам Интернета.

Гипертекст (Hypertext)

Принцип организации информационных массивов, при котором отдельные информационные элементы связаны между собой ассоциативными отношениями,

обеспечивающими быстрый поиск необходимой информации и/или просмотр взаимосвязанных данных.

GIF -анимация

Последовательное отображение с заданной частотой растровых изображений, хранящихся в одном GIF-файле.

Графика (Graphic art)[12]

Искусство изображения предметов контурными линиями и штрихами. Иногда в графике допускается применение цветных пятен.

К графике относятся рисунок и различные виды его печатных воспроизведений: гравюра, литография, монотипия и др.

В зависимости от содержания и назначения графика бывает:

• станковая;

• подготовительная – эскиз, набросок, зарисовки;

• книжная и журнально-газетная;

• прикладная – промышленная, грамоты, марки, этикетки, реклама и т. д.;

• плакатная.

Графика трехмерная (30-графика)

Технология мультимедиа; компьютерная графика, создаваемая с помощью изображений, имеющих длину, ширину и глубину.

Графика компьютерная, машинная (Computer graphics)

Технология создания и обработки графических изображений средствами вычислительной техники.

Компьютерная графика изучает методы получения изображений, созданных на основании невизуальных данных или данных, представленных непосредственно пользователем.

Графика деловая (Graphics for Managers)

Технология создания изображений с сопровождающим текстом для нужд коммерции.

Детальный дизайн интерфейса.

Интерфейс низкого уровня (и процесс разработки этого интерфейса), включающий использование и расположение конкретных элементов управления. Основывается на высокоуровневой архитектуре интерфейса.

Дизайн взаимодействия (Interaction Design)

Систематический итеративный процесс разработки высокоинтерактивных пользовательских интерфейсов. Методология включает исследовательские техники, такие, как анализ требований, анализ задач, анализ потенциальных пользователей, а также различные методы прототипирования, экспертизы и оценки. Дизайн, ориентированный на нужды пользователя (User Centered Design) Подход к дизайну, при котором во главу угла ставятся пользователи, их нужды и требования.

Дизайн пользовательского интерфейса (User Interface Design)

Полный процесс планирования и проектирования того, как пользователь будет взаимодействовать с системой. Дизайн пользовательского интерфейса связан с многими этапами разработки продукта, включая анализ требований, информационную архитектуру, дизайн взаимодействия, пользовательское тестирование, создание документации и справочной системы.

Домен (Domain)

Самая крупная структурная единица Интернета. Обычно домен соответствует стране или другой большой структуре. Домены могут подразделяться на поддо-мены, отражающие различные области интересов или ответственности. Организовать группы компьютеров в Интернете с помощью иерархии доменов позволяет служба имен доменов DNS.

Звуковые эффекты

Технология мультимедиа; сохранение в цифровом виде звучания музыкальных инструментов, звуков природы или музыкальных фрагментов, созданных на компьютере либо

[12] Греч. graphike; фр. art graphique; нем. graphic; лат. grapho – рисую.

записанных и оцифрованных.

Инженерная психология

Отрасль психологии, изучающая психологические особенности труда человека при взаимодействии его с техническими средствами.

Информационная архитектура (Information Architecture)

Наука и искусство организации информации с целью облегчения выполнения людьми их информационных нужд (например, поиск и использование информации). Тесно связана с дизайном пользовательского интерфейса.

Информационная технология (information technology)[13]

Совокупность методов, производственных и программно-технологических средств, объединенных в технологическую цепочку, обеспечивающую сбор, хранение, обработку, вывод и распространение информации. Информационные технологии предназначены для снижения трудоемкости процессов использования информационных ресурсов.

Информационный рынок (Information Market)

Система экономических, правовых и организационных отношений по торговле информационными технологиями, информационными продуктами и услугами.

Итеративный дизайн (Iterative Design)

Концепция, суть которой состоит в том, что процесс разработки системы должен представлять собой повторяющийся цикл. На каждом этапе дизайн системы оценивается, улучшается и тестируется. Результаты тестирования на каждом этапе являются источником для внесения улучшений в систему в следующем цикле.

Когнитивная перегрузка (Cognitive Overload)

Результат чрезмерных требований системы к когнитивным (познавательным) процессам пользователя, в особенности к памяти. Проявляется в высоком уровне ошибок, быстром утомлении, низкой удовлетворенности пользователей.

Компьютерная графика, машинная графика (Computer Graphics)

Технология создания и обработки графических изображений средствами вычислительной техники. Компьютерная графика изучает методы получения изображений, созданных на основании невизуальных данных или данных, предоставленных непосредственно пользователем.

Компьютерная игра (Computer Game)

Игра, построенная с использованием мультимедийных возможностей компьютера, определяется алгоритмом, описывающим процесс ее прохождения. Компьютерные игры подразделяются на деловые, развивающие, обучающие и развлекательные.

Конвергенция информационных технологий (Convergence)

Процесс сближения разнородных электронных технологий в результате их быстрого развития и взаимодействия.

Контекст использования (Context of Use)

Описание реальных условий, в которых система используется в обычных повседневных ситуациях. Примерами таких условий являются характеристики пользователей, решаемые задачи, оборудование, а также физическое, социальное и организационное окружение, в котором используется система.

Контрольный список (Checklist)

Документ со списком требований к системе или отдельному ее фрагменту.

Ментальная или концептуальная модель (Mental or Conceptual Model)

Собственные представления человека о взаимодействии с окружающим миром (в частности, с системой). Эти представления помогают нам решать проблемы и использовать артефакты, такие, как, например, компьютерные системы и другие интерактивные системы. Несовпадение ментальной модели с реальным устройством системы затрудняет использование этой системы.

Модель пространственных данных, представление пространственных данных (Spatial Data

---

[13] Фр. technologie d’information.

Representation; Geospatial Data Model; Spatial Data Model)

Представление пространственных данных – способ цифрового описания пространственных объектов, тип структуры пространственных данных. Наиболее употребительными представлениями являются векторное, растровое, регулярно-ячеистое и квадротомическое.

Мультимедиа (Multimedia)

Совокупность компьютерных технологий, одновременно использующих несколько информационных сред: графику, текст, видео, фотографию, анимацию, звуковые эффекты, высококачественное звуковое сопровождение. Технологию мультимедиа составляют специальные аппаратные и программные средства.

Мультимедийная продукция

Сочетание звука, текста и образов в цифровой форме, пригодное для воспроизведения на ЭВМ.

Мультимедийные приложения

Энциклопедии, интерактивные курсы обучения, игры, интернет-приложения, тренажеры, средства торговой рекламы, электронные презентации и др.

Мультимедийные функции (Multimedia Functions)

• Цифровая фильтрация и масштабирование видео.
• Аппаратная цифровая компрессия и декомпрессия видео.
• Ускорение графических операций, связанных с трехмерной графикой.
• Развертка живого видео на мониторе.
• Наличие композитного видеовыхода.
• Вызов телевизионного сигнала на монитор.

«Мысли вслух» (Think aloud Protocol)

Название техники, применяемой в пользовательском тестировании. Суть ее в следующем: пользователей просят вербализировать свои мысли, ощущения и мнения в процессе взаимодействия с системой и выполнения тестовых заданий. В то время как основное внимание при пользовательском тестировании уделяется эффективности выполнения заданий, вербализация довольно полезна для понимания ошибок, допущенных пользователем, и получения представления о том, что эти ошибки повлекло и как интерфейс может быть улучшен для избежания подобных проблем.

Ознакомленность (Familiarity)

Соотношение между существующими знаниями пользователя и теми знаниями, которые необходимы для эффективного взаимодействия с системой.

Открытая система (Open system)

Вычислительная среда, состоящая из аппаратных и программных продуктов и технологий, разработанных в соответствии с общедоступными и общепринятыми (международными) стандартами. Обязательные свойства открытых систем:

• переносимость;
• интероперабельность;
• масштабируемость;
• доступность программного и аппаратного обеспечения для развития и реструктуризации.

Персона, персонаж (Person)

Описание фиктивного пользователя будущей системой, помогающее разработчику интерфейса эмоционально слиться с пользователями его продукта и тем самым избежать создания интерфейса «ни для кого». Метод и термин введены в обиход Аланом Купером (http://www.cooper.com/)

Пластиковая карточка (Plastic Card)

Пластина стандартного размерова 85,6 53,9 0,76 мм, изготовленная из устойчивой к механическим и термическим воздействиям пластмассы. Пластиковая карточка является носителем информации.

Пользователи (Users)

Группа людей, которые работают с определенной системой. Для разработки качественного интерфейса всегда необходимо знать характеристики и особенности этих людей

(эти данные входят в контекст использования системы).

Пользовательский интерфейс (User Interface)

Все аспекты системы, с которыми пользователь взаимодействует, включая способы передачи информации от системы к пользователю и от пользователя к системе, а также шаги, которые необходимо сделать для выполнения задач, и элементы управления.

Протокол передачи гипертекста (HyperText Transfer Protocol – HTTP)

Базируется на TCP/IP, обеспечивает доступ к документам на web-узлах. Основная задача протокола состоит в установлении связи с web-сервером и обеспечении доставки HTML-страниц web-браузеру клиента. Протокол HTTP:

• определяет взаимодействие партнеров на прикладном уровне;
• предназначен для передачи сообщений, являющихся блоками гипертекста;
• используется в службе глобального соединения.

Прототип (Prototype)

Экспериментальный вариант дизайна целой системы или ее части, который используется в целях тестирования или пояснения.

Прототипирование (Prototyping)

Разработка неполной репрезентации системы, необходимая для проведения пользовательского тестирования. Прототипирование является важным элементом итеративного дизайна, при котором дизайн создается, оценивается и улучшается до тех пор, пока не будет достигнута необходимая эффективность системы. Прототипы могут быть как очень простыми (например, бумажный прототип), так и сложными (выполняющие практически все функции финальной системы).

Растровая графика

Метод графического представления объекта в виде множества точек. Сервисы Интернета (Internet Servise)

Сервисы, предоставляемые Интернетом пользователям, программам, системам, уровням, функциональным блокам. В Интернете сервисы предоставляют сетевые службы. Наиболее распространенными интернет-сервисами являются:

• хранение данных;
• передача сообщений и блоков данных;
• электронная и речевая почта;
• организация и управление диалогом партнеров;
• предоставление соединений;
• проведение сеансов;
• видеосервис.

Совместный дизайн (Participatory Design)

Подход к разработке, при котором в процесс дизайна системы вовлекается широкий круг участников, таких, как дизайнеры, разработчики, менеджеры, потребители, потенциальные пользователи и т.д. При использовании этого подхода пользователи не просто являются участниками пользовательского тестирования, но фактически вовлекаются в процесс разработки.

Субъект тестирования (Participant)

Человек, принимающий участие в оценке или тестировании системы. Субъект тестирования может и не являться представителем целевой аудитории системы, но на таком субъекте можно протестировать только ограниченное число характеристик системы.

Телевизионный прием

Технология мультимедиа; вывод телевизионных сигналов на монитор компьютера параллельно с работой других программ.

Трехмерное изображение (Three Dimensional Image; 3D)

Изображение объемного предмета, выполненное на плоскости. В компьютерной графике трехмерная поверхность предмета аппроксимируется большим количеством малых плоских фигур (треугольников).

Трехмерный интерфейс (Three Dimensional Interface)

Стандартный интерфейс прикладных программ, описывающих трехмерные изображения.

В соответствии с трехмерным интерфейсом:

• устанавливаются тени в зависимости от расположения источников света;

• удаляются скрытые детали изображения;

• выполняются другие функции.

Человеко-компьютерное взаимодействие (Human-Computer Interaction, HCI)

Широкая научная и прикладная дисциплина, предметом изучения которой является то, как люди используют компьютеры и как следует разрабатывать компьютерные системы, чтобы обеспечить более эффективное их использование. Дисциплина включает элементы информатики, графического дизайна, социологии и антропологии, психологии, эргономики.

Экспертная (эвристическая) оценка (Expert Review)

Один из неэкспериментальных методов оценки юзабилити системы. Несколько экспертов порознь проверяют интерфейс на соответствие определенным юзаби-лити-принципам (эвристикам). Затем они комбинируют свои результаты и определяют степень серьезности каждой проблемы. Экспертная оценка более дешева и оперативна, чем полноценное пользовательское тестирование, но ее валид-ность менее высока.

Эргономика (Human Factors, Ergonomics)

Научно-прикладная дисциплина, занимающаяся изучением и созданием эффективных систем, управляемых человеком, а также среды, оптимальной для человека.

Юзабилити-тестирование (Usability Testing, Usability Evaluation)

Любой из экспериментальных методов определения легкости овладения и удобства использования системы. Например, при простом пользовательском тестировании приглашаются несколько потенциальных пользователей систем, каждый из которых выполняет при помощи системы серию заранее разработанных заданий. За процессом выполнения заданий следит наблюдатель, который отмечает, с какими сложностями сталкивается каждый из пользователей. В некоторых случаях требуется проводить тестирование в привычной (естественной) для пользователя обстановке или комбинировать тестирование с другими методами сбора информации (например, интервью).

ТЕРМИНЫ НА АНГЛИЙСКОМ ЯЗЫКЕ

Activity (Работа, операции)

Операция или совокупность смежных операций, объединенных по некоторому признаку. Работа выполняется в течение определенного времени и завершается определенным результатом. Работа – основной структурный элемент проекта.

Activity duration (Продолжительность работы)

Оценка времени (в часах, днях, неделях, месяцах, годах), требуемого для выполнения работы с учетом ее характера и необходимых ресурсов.

Activity Status (Статус работы)

Состояние работы с точки зрения выполнения: планируемая (работа еще не началась), в процессе (работа началась, но не закончилась), завершена (выполненная работа).

Activity-oriented (Ориентированный на работу)

Способ построения сетевой модели, в которой вершины изображают работы, а дуги – зависимости между ними.

Actual Cost (Фактические затраты)

Сумма стоимостей ресурсов за фактически выполненный объем работ и фиксированных затрат.

Actual Cost of Work Performed (ACWP) (Фактическая стоимость выполненных работ, ФСВР)

Фактические затраты на работу, часть проекта, весь проект. Actual Finish Date (Фактическая дата окончания)

Календарная дата фактического завершения работы в рамках проекта.

Actual Start Date (Фактическая дата начала)

Календарная дата фактического начала работы в рамках проекта.

Actual Time (Фактическое время)

1. Фактическое время события – время, когда это событие действительно произошло, т.е. когда все работы, предшествующие ему, закончены.

2. Фактическое время начала или окончания работы.

Actual % Complete (Физический процент выполненного)

Процент выполненного от объема работы. Используется для вычисления плановой стоимости выполненных работ.

Actual Work (Фактический объем работ)

Полный фактический объем работы, выполненной с помощью всех ресурсов.

Aims and Tasks Tree (Дерево целей и задач)

Один из типов структурной модели проекта, охватывающей его элементы по уровням (цели, задачи, мероприятия) и связи между ними (включение или подчиненность). Используется при программно-целевом управлении.

Arrow (Дуга (стрелка))

1. Графическое изображение работы. Хвост стрелки означает начало работы, острие – окончание.

2. Графическое изображение зависимости между работами. Дуга направлена от предшествующей работы к последующей.

Arrow Diagram, Arrow Diagrammed Method (ADM) (Диаграмма «работа-дуги», метод дуг)

Способ построения сетевой модели, в которой вершины изображают события, а стрелки (дуги) – работы.

As-built Schedule (Фактический график)

Фактический график (законченного) проекта, где указаны фактические даты выполнения работ.

As Late As Possible (ALAP) (Как можно позже (КМП))

Тип работы, который планируется на максимально поздние сроки, не допуская, однако, задержек в ранних этапах.

As Soon As Possible (ASAP) (Как можно раньше (КМР))

Тип работы, для которой устанавливаются ранние даты на самые ранние допустимые сроки. Тип работ по умолчанию.

Available Resource (Наличные ресурсы)

Ресурсы, которые можно использовать для выполнения проекта. Backward Pass (Обратный проход)

Расчет поздних сроков выполнения работ (совершения событий) сетевой модели. Производится в обратном порядке по сравнению с прямым ходом (см.: forward pass).

Bar Chart (Диаграмма, график)

Графическое отображение работ в виде линии на временной шкале (иногда ее называют диаграммой Гантта) (см.: Gantt chart).

Baseline concept (Базисная концепция)

Основные положения концепции, плана или какого-либо иного основополагающего документа проекта.

Baseline Cost of Work Performed (BCWP) (Плановая стоимость выполненных работ, ПСВР)

Параметр, используемый при стоимостном анализе, позволяющий количественно оценить прогресс в денежном выражении. Называется также «фактическая выработка на дату».

Baseline Cost of Work Scheduled (BCWS) (Плановая стоимость запланированных работ, ПСЗР)

Плановая стоимость, умноженная на процент выполненного, который должен быть достигнут к текущему числу согласно исходному плану проекта.

Baseline Duration (Плановая длительность)

Плановая длительность задачи на момент принятия исходного плана.

Baseline Work (Плановый объем работ)

Общий запланированный объем работ.

Basic Project Package (Задание на проектирование)

Формулируемые и(или) утверждаемые заказчиком проекта краткие указания его разработчикам, определяющие цели проекта, подлежащие достижению результаты и основные

ограничения и требования по содержанию проектных решений.

Bidding (Торги)

Способ закупки товаров, размещения заказов и выдачи подрядов, при котором выбор подрядчика (поставщика) производится на конкурсной основе.

Brainstorming (Метод мозгового штурма)

Метод сбора информации, цель которого – привлечь экспертов для генерации всех возможных идей по поводу обсуждаемого вопроса.

Budget At Completion (BAC) (Бюджет на завершение)

Плановая стоимость на завершение проекта.

Budget Cost of Work Scheduled (BCWS) (Смета проекта)

Сумма согласованных затрат по плану на работы, предназначенные к выполнению в течение рассматриваемого периода.

Business Plan (Бизнес-план проекта)

Основной документ, представляемый инвестору по инвестиционному проекту, в котором в краткой форме, в общепринятой последовательности разделов излагаются главные характеристики проекта. Бизнес-план призван убедить инвестора в эффективности намечаемых вложений.

Calendar (Календарь)

Действующий в данной местности календарь, используемый для составления графика выполнения проекта с учетом принятых выходных, праздников и других нерабочих дней.

Calendar Date (Календарная дата проекта)

Дата, привязанная к календарю.

Calendar Range (Календарный интервал)

Интервал времени, в котором выполняется проект или его часть. Calendar Unit (Единица времени по календарю)

Наименьшая единица времени в составленном календаре. Как правило, такой единицей бывают час, день, неделя, месяц.

Change (Изменения)

Корректировки, вносимые в содержание проекта. Корректировке могут быть, в частности, подвергнуты смета, график работ, требуемые технические характеристики и т.д.

Commissioning (Пуск объекта)

Рабочие процессы, направленные на обеспечение выхода производственных мощностей на проектный уровень.

Completion (Завершение)

Завершение проекта, работ.

Completion Phase (Фаза завершения проекта)

Обычно на этой фазе производятся сдача проекта заказчику, свертывание сил, участвующих в реализации проекта, подведение итогов и окончательные расчеты по проекту, анализ эффективности реализации проекта.

Communications Planning (Планирование коммуникаций)

Определение информационных и коммуникационных потребностей участников проекта и заинтересованных лиц, разработка мероприятий по их удовлетворению.

Concept (Концепция проекта)

Формулировка целей проекта и выбор пути его реализации. Concept Phase (Концептуальная фаза)

Первая из четырех последовательных фаз в цикле реализации проекта как целого. В этой фазе формулируется концепция проекта, производится проверка его осуществимости.

Conceptual Development (Концептуальная проработка)

Процесс выбора и документального оформления наилучшего подхода к достижению целей проекта.

Conceptual Project Planning (Концептуальное планирование проекта)

Процесс составления документации, охватывающей все основные аспекты реализации проекта, из которой вытекают затем технические требования, сметы, графики работ, процедуры контроля и управления проектом.

Conceptual Schedule (Концептуальный график)

Укрупненный график, составленный в концептуальной фазе проекта.

Constraint (Ограничение, зависимость)

Ограничение на время или условия выполнения работы. Иногда зависимость между работами проекта.

Constraint Dates (Целевая дата)

Дата, дополнительно назначенная пользователем для позиционирования работы во времени. Существуют два типа целевых дат – начала и конца.

Constraint Finish (Целевой конец)

Дата конца работы или проекта, назначенная пользователем.

Constraint Start (Целевое начало)

Дата начала работы или проекта, назначенная пользователем.

Consumable Resources (Расходуемые ресурсы)

Тип ресурса, сохраняющегося в неизменном количестве до тех пор, пока он не будет израсходован (например, материалы).

Construction Management (Управление строительством)

Процесс, который обеспечивает координацию, информационный обмен и управление всем процессом строительства, начиная с проектирования, закупок и кончая до вводом сооружений в эксплуатацию.

Contingencies (Резерв на непредвиденные затраты)

Резерв ресурсов или финансовых средств, рассчитанный на возникновение непредвиденных затрат в пределах определенного объема работ по проекту. Резерв на непредвиденные затраты может быть включен в смету отдельной строкой.

Contingency Planning (Планирование непредвиденных затрат)

Разработка планов организационной деятельности, приводимых в действие в случае определенных событий, связанных с риском.

Contingency Rundown (Снижение неопределенности)

Систематическое уменьшение неточности, например, при прогнозе стоимости, отражающее накопление изменений и снижение неопределенности.

Corporation Project Strategy (Стратегия корпорации в отношении проекта)

Общие принципы и направления реализации проектов в данной корпорации.

Corrective Action (Управляющее воздействие)

Изменения, предпринимаемые с целью выполнения проекта в заданные сроки, с заданным качеством, в соответствии с бюджетом и т.д.

Cost (Стоимость)

Стоимость проекта или его элемента.

Cost Control (Контроль затрат)

Процессы сбора, накопления и анализа данных по затратам, составления отчета по ним и регулирования их по состоянию на текущий момент.

Cost Curve (График затрат)

График планируемых, прогнозируемых или фактических затрат. Cost Optimization (Оптимизация стоимости)

Построение такого графика осуществления проекта, при котором обеспечивается его минимальная стоимость при заданных сроках окончания проекта.

Costs (Затраты, издержки)

Сумма расходов, необходимых для производства определенного вида товаров, оказания услуг, выполнения работы по проекту.

Cost Plus Contract (Контракт «цена плюс»)

Подрядчик обязуется за дополнительное, установленное контрактом вознаграждение выполнить обязательства по контракту в рамках определенной суммы.

Cost Varience (CV) (Отклонение по стоимости, ОСТ)

Разница между плановой и фактической стоимостью выполненных работ.

Critical activity (Критическая работа)

Работа, выполнение которой нельзя отложить без изменения общей продолжительности

или даты окончания проекта. Работа, принадлежащая критическому пути.

Critical path (Критический путь)

Последовательность взаимосвязанных критических работ, определяющих общую продолжительность проекта.

Critical Path Method (CPM) (Метод критического пути)

Метод составления графика работ с использованием сетевой модели. Используется для определения продолжительности проекта, критических работ и резервов времени.

Current Finish Date (Текущая дата окончания работы)

Текущая оценка календарной даты завершения работы.

Current Start Date (Текущая дата начала (работы))

Текущая оценка календарной даты начала работы.

Data Date (Датаразделения данных – текущая дата)

Календарная дата, отделяющая фактические (ретроспективные) данные от плановых.

Definitive Schedule (Полный график)

График, в котором полно и детально изображены все работы проекта.

Delegation (Делегирование)

Процесс, при котором полномочия руководителя передаются подчиненным. Delphi Technique (Метод Дельфи)

Метод, позволяющий получить оценку специалистов-экспертов путем проведения анонимного анкетирования.

Design (Проект как результат проектирования)

Документально оформленный план сооружения, конструкции (не путать с терминомproject, имеющим более широкий смысл).

Design Control (Контроль на этапе проектирования)

Система контроля за показателями объема работ по проекту, графиками работ и затратами на стадии проектирования.

Detail Schedule (Детальный график, детальная (рабочая) сетевая модель)

График, используемый для осуществления оперативного управления – обычно на уровне ответственных исполнителей.

Detailed Project Package (Технический проект)

Совокупность документов, примерно соответствующая по составу техническому проекту.

Development Phase (Фаза разработки)

Вторая из четырех последовательных фаз в цикле осуществления проекта. Определение структуры проекта, построение графика работ, определение затрат, ресурсов, подготовка всей необходимой документации, подбор исполнителей.

Dummy Activity (Мнимая операция (работа))

Мнимая операция (работа), фиктивная операция. Используется в сетевой модели для обозначения логической зависимости между реальными работами.

Duration (Длительность работы)

Количество рабочего времени, требуемого для выполнения работы. Early Date (Ранняя дата)

Вычисляемая в процессе прямого прохода дата, когда работа может начаться или закончиться.

Early Finish (EF) (Раннее окончание (РО))

Дата самого возможно раннего завершения некоторой работы. Определяется условиями сетевой модели.

Early Start (ES) (Раннее начало(РН))

Дата самого возможно раннего начала некоторой работы. Определяется условиями сетевой модели.

Environment (Среда)

Совокупность внутренних и внешних сил, которые способствуют или мешают достижению целей проекта. Они сопряжены как с внешними условиями: политическими, социально-экономическими, технологическими, административными, климатическими, так и с внутренними: участниками проекта, условиями контракта, командой проекта, местом

реализации и т.д.

Earned Value Analysis (Анализ стоимости проекта с учетом освоенного объема)

Анализ хода выполнения проекта, при котором фактические денежные средства, трудозатраты (или другие количественные показатели), предусмотренные в бюджете проекта и фактически израсходованные, сравниваются со стоимостью выполненных работ.

EV (Earned Value)

Физический объем выполненных работ или же утвержденный бюджет на этот объем работ. Сумма утвержденных оценок затрат (может включать накладные расходы) на работы (или части работ), завершенные за определенный период времени. Раньше назывался сметной стоимостью выполненных работ (BCWP).

Event (Событие)

Момент времени, связанный с изменением состояния проекта. Часто определяется началом или окончанием работы или группы работ. В сетевой модели, ориентированной на события, изображается узлом.

Event Oriented (Ориентация на событие)

Способ построения сетевой модели, в которой вершины изображают события (см.: Arrow diagram).

Exception Reporting (Отчетность по отклонениям)

Процесс документального оформления ситуаций, в которых происходят значимые отклонения от требований. Когда процесс выходит за установленные границы, составляется отчет с указанием причин, по которым это отклонение произошло.

Execution (Выполнение)

Выполнение проекта, работ.

Fast Track (Быстрый путь)

Метод управления проектом, при котором различные процессы совмещены друг с другом, например проектные и строительные работы.

Final Completion (Полное завершение)

Проект закончен на уровне требований контракта, за исключением работ, оговоренных по условиям приемочных испытаний.

Financing (Финансовое обеспечение проекта)

Это понятие включает технику и методику поиска финансовых источников и определение объемов финансирования.

Finish Activity (Конечная работа)

Работа без продолжения.

Finish Date (Дата окончания)

Момент времени, связанный с завершением работы.

Finish Milestone (Конечная веха)

Работа-веха для отслеживания завершения фазы проекта. Finish Slack (Конечный резерв)

Количество рабочего времени между ранней датой конца и поздней датой конца. Finish to Finish Relationship (Зависимость «конец-конец»)

Взаимосвязь между работами сетевой модели, при которой окончание последующей работы зависит от окончания предшествующей.

Finish to Start Relationship (Зависимость «конец-начало»)

Взаимосвязь между работами сетевой модели, при которой начало последующей работы зависит от окончания предшествующей.

Fixed Cost (Фиксированные затраты)

Любые затраты на выполнение задачи, не связанные с оплатой работы назначенных ресурсов.

Fixed Price Contract (Контракт с фиксированной ценой)

Подрядчик обязуется выполнить работу или оказать услугу по установленной в контракте цене.

Float (Резерв)

Величина, в пределах которой можно задержать выполнение работы.

Floating Activity (Свободная работа)

Работа, имеющая резерв времени.

Formative Quality Evaluation (Формирующая оценка качества)

Анализ данных в процессе цикла осуществления проекта относительно заранее разработанных спецификаций качества с тем, чтобы можно было своевременно внести изменения, необходимые для обеспечения заданного уровня качества проекта.

Free Float (Свободный резерв)

Величина, на которую можно задержать выполнение работы, не изменяя раннего начала всех последующих работ.

Function Chart Structure (Построение функциональной схемы)

Построение функциональной схемы руководства проектом на основе необходимой совокупности знаний.

Function Quality Integration (Интеграция функций обеспечения качества)

Процесс активной интеграции организаций-партнеров с взаимным согласованием планов и программ, необходимых и достаточных для обеспечения заданного качества, создаваемого командой, работающей над проектом.

Functional Organization (Функциональная организационная структура управления)

Тип организационной структуры, в которой персонал группируется по своей специализации.

Gantt Chart (Диаграмма Гантта)

То же, что и сетевой график (см.: Bar chart).

Graphical Evaluation and Review Technique (GERT) (Метод графического анализа и оценки программы)

Метод сетевого анализа, который дает оценки вероятности реализации событий, основанные на статистических данных, получаемых в результате моделирования.

Hammock (Гамак)

Работа, обобщающая совокупность более детальных работ для отображения в сводных документах проекта.

Historical Records (Архивные записи)

Данные о фактическом ходе реализации проекта, которые можно использовать для анализа и прогноза.

Impact Analysis (Анализ влияния (факторов при прогнозировании))

Математический анализ природы рисков проекта, а также возможных сочетаний взаимосвязанных рисков.

Implementation Phase (Фаза осуществления, реализации)

Третья из четырех последовательных фаз цикла реализации проекта. Фаза выполнения, реализации проекта.

Imposed Data (Навязанная дата)

Календарная дата, определяемая внешними условиями безотносительно к логике сетевой модели. В отечественной литературе используются термины «внешняя дата», «внешнее ограничение».

Imposed Finish Date (Предельная дата окончания)

Предопределенная дата, установленная вне связи с сетевым графиком и определяющая конец работы и всех остальных работ, предшествующих этому концевому узлу.

Information System of Project Management (Информационная система управления проектами (ИСУП))

Организационно-технологический комплекс методических, технических, программных и информационных средств, направленных на поддержку и повышение эффективности процессов планирования и управления проектом.

Initiating Project (Инициация проекта)

Процесс управления проектом, результатом которого является санкционирование начала проекта или очередной фазы его жизненного цикла.

Inprogress activity (Выполняемая работа)

Работа, начатая, но не завершенная к текущему моменту.

Input Limits (Заданные ограничения)

Input Milestones (Заданные узловые события (контрольные точки))

Задаваемые узловые даты или события, которые определяют принцип.

СЛОВАРИК ПРОФЕССИОНАЛЬНОГО ЖАРГОНА ДЛЯ НАЧИНАЮЩЕГО РАЗРАБОТЧИКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Апгрейд	Upgrade	Улучшение (системы, программы, компьютера)
Апгрейд	Update	Улучшение (системы, программы)
Апдейт репорт	Update report	Отчет о состоянии продукта
Бага	Bug	Ошибка в программе
Багобаза	Bug base	База данных с описанием ошибок в программном продукте
Батон	Button	Кнопка (в диалоговом окне, на клавиатуре)
Браузер	Browser	Программа для просмотра (как правило, HTML-страниц)
Бук	Book	Портативный компьютер
Бэкап	Backup	Резервное копирование данных
Винда	Windows	Операционная система Windows
Вьюер	Viewer	Программа для просмотра (как правило, графики)
Гайдлайнсы	Guidelines	Руководство (пользователя)
Девелопер	Developer	Программист. Компания по разработке программного обеспечения
Диалог	Dialog	Диалоговое окно программы
Дока	Documentation	Документация
Дропдаун	Dropdown	Элемент пользовательского интерфейса
Зафиксить	Fix	Исправить (ошибку)
Зафризить	Freeze	Заморозить (код)
Имплементить (заимплементить)	Implement	Внедрить, разработать
Инстолить (заинстолить)	Install	Установить (программу, операционную систему)
Исходники	Source code	Исходный код
Кастомер	Customer	Заказчик
Клипборд	Clipboard	Буфер обмена данными
Кликнуть (дабл кликнуть)	Click (double-click)	Нажать (дважды нажать). Нажать кнопку
Контрол	Control	Элемент диалогового окна, панели инструмента

Мануал	Manual	Руководство пользователя
Операционка	Operating system	Операционная система
Отрепортить	Report	Отчитаться
Пейстить (запейстить)	Paste	Вставить объект из буфера обмена данными
Постить (запостить)	Post	Выкладывать (на FTP)
Префы	Preferences	Предпочтения, опции
Прога	Program	Программа
Радиобатон	Radiobutton	Элемент пользовательского интерфейса
Риквест	Request	Запрос (исправить ошибку, на внедрение)
Рилиз	Release	Выпуск (продукта)
Рилиз нота	Release note	Письмо, которое сопровождает выпуск продукта
Сервак	Server	Сервер
Серцы	Sources	Исходный код
Сетапить (засетапить)	Set up	Установить (программу, операционную систему)
Скролер	Scroller	Элемент пользовательского интерфейса
Спека	Specification	Спецификация
Тайпить (натайпить)	Type	Набрать текст на клавиатуре
Тулбар	Toolbar	Элемент пользовательского интерфейса. Панель инструментов
Тулзы	Tools	Инструменты
Фича	Feature	Функциональность. Функция. Элемент интерфейса. Отличительная черта программы
Чекать (прочекать)	Check	Проверить (ошибку, программу)
Чекбокс	Checkbox	Элемент пользовательского интерфейса
Шара	Shared folder	Папка, доступная группе пользователей
Шарить (зашарить)	Share	Сделать доступным (папку, файл); поделить (ресурс)
Юзать	Use	Пользоваться
Юзер	User	Пользователь

ПРИЛОЖЕНИЕ 2.

Перечень стандартов ISO в области человеко-компьютерного взаимодействия

ISO 9241-1:1997

Ergonomic requirements for office work with visual display terminals (VDTs). Part1: General introduction.

Эргономические требования к видеодисплейным терминалам при работе в учреждениях. Часть 1: Общие положения

ISO 9241-2:1992

Ergonomic requirements for office work with visual display terminals (VDTs). Part2: Guidance

on task requirements.

Эргономические требования к видеодисплейным терминалам для офисной работы. Часть 2: Руководство по требованиям в зависимости от задач

ISO 9241-3:1992

Ergonomic requirements for office work with visual display terminals (VDTs). Part3: Visual display requirements.

Эргономические требования к видеодисплейным терминалам при работе в учреждениях. Часть 3: Требования к устройствам отображения

ISO 9241-4:1998

Ergonomic requirements for office work with visual display terminals (VDTs). Part4: Keyboard requirements.

Эргономические требования к видеодисплейным терминалам при работе в учреждениях. Часть 4: Требования к клавиатуре

ISO 9241-5:1998

Ergonomic requirements for office work with visual display terminals (VDTs). Part5: Workstation layout and postural requirements.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 5: Размещение рабочей станции и положение тела

ISO 9241-6:1999

Ergonomic requirements for office work with visual display terminals (VDTs). Part6: Guidance on the work environment.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 6: Требования к окружающей среде

ISO 9241-7:1998

Ergonomic requirements for office work with visual display terminals (VDTs). Part7: Requirements for display with reflections.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 7: Требования к дисплеям с отражением

ISO 9241-8:1997

Ergonomic requirements for office work with visual display terminals (VDTs). Part8: Requirements for displayed colours.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 8: Требования к отображаемым цветам.

ISO 9241-9:2000

Ergonomic requirements for office work with visual display terminals (VDTs). Part9: Requirements for non-keyboard input devices.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 9: Требования к неклавиатурным устройствам ввода.

ISO 9241-10:1996

Ergonomic requirements for office work with visual display terminals (VDTs). PartlO: Dialogue principles.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 1О: Принципы диалога.

ISO 9241-11:1998

Ergonomic requirements for office work with visual display terminals (VDTs). Part11: Guidance on usability.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 11: Требования к юзабилити.

ISO 9241-12:1998

Ergonomic requirements for office work with visual display terminals (VDTs). Part12: Presentation of information.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 12: Представление информации.

ISO 9241-13:1998

Ergonomic requirements for office work with visual display terminals (VDTs). Part13: User guidance.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 13: Руководство пользователем.

ISO 9241-14:1997

Ergonomic requirements for office work with visual display terminals (VDTs). Part14: Menu dialogues.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 14: Диалог с использованием меню.

ISO 9241-15:1997

Ergonomic requirements for office work with visual display terminals (VDTs). Part15: Command dialogues.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 15: Командный диалог.

ISO 9241-16:1999

Ergonomic requirements for office work with visual display terminals (VDTs). Part16: Direct manipulation dialogues.

Эргономические требования к видеодисплейным терминалам при работе в офисе. Часть 16: Диалог с прямой манипуляцией.

ISO 9241-17:1998

Ergonomic requirements for office work with visual display terminals (VDTs).

Part17: Form filling dialogues.

Эргономические требования к видеодисплейным терминалам при работе в учреждениях. Часть 17: Диалог с заполнением форм.

ISO/AWI 9241-18

Ergonomics of human system interaction – Physical input devices – Ergonomic principles.

Эргономика взаимодействия человека с системой. Устройства физического ввода. Эргономические принципы.

ISO/AWI 9241-19

Ergonomics of human system interaction – Physical input devices – Design criteria for products.

Эргономика взаимодействия человека с системой. Устройства физического ввода. Критерии дизайна.

ISO/AWI 9241-20

Ergonomics of human system interaction – Physical input devices. Part 2O: Ergonomic selection procedures.

Эргономика взаимодействия человека с системой. Устройства физического ввода. Часть 2O: Процедура эргономического выбора.

ISO 14915-1:2002

Software ergonomics for multimedia user interfaces. Part 1: Design principles and framework.

Эргономика программного обеспечения мультимедийных пользовательских интерфейсов. Часть 1: Структура и принципы проектирования.

ISO 14915-2:2003

Software ergonomics for multimedia user interfaces. Part 2: Multimedia navigation and control.

Эргономика программного обеспечения мультимедийных пользовательских интерфейсов. Часть 2: Мультимедийное управление и навигация.

ISO 14915-3:2002

Software ergonomics for multimedia user interfaces. Part 3: Media selection and combination.

Эргономика программного обеспечения мультимедийных пользовательских интерфейсов. Часть 3: Выбор и комбинирование сред представления.

ISO 14915-4

Software ergonomics for multimedia user interfaces. Part 4: Domain specific aspects.

Эргономика программного обеспечения мультимедийных пользовательских интерфейсов. Часть 4: Особенности применения мультимедиа в различных областях.

ISO 11064-1:2000

Ergonomic design of control centers. Part 1: Principles for the design of control centres.

Эргономическое проектирование центров управления. Часть 1: Принципы проектирования центров управления.

ISO 11064-2:2000

Ergonomic design of control centers. Part 2: Principles for the arrangement of control suites.

Эргономическое проектирование центров управления. Часть 2: Принципы компоновки управляющих систем.

ISO 11064-3:1999

Ergonomic design of control centers. Part 3: Control room layout. Эргономическое проектирование центров управления. Часть 3: Обстановка рабочего помещения.

ISO/FDIS11064-4

Ergonomic design of control centers. Part 4: Layout and dimensions of workstations.

Эргономическое проектирование центров управления. Часть 4: Размещение и габариты рабочих станций.

ISO 10075:1991

Ergonomic principles related to mental work-load – General terms and definitions. Эргономические принципы, относящиеся к умственной загруженности. Общие понятия и определения.

ISO/IEC 10741-1

Dialogue interaction – Cursor control for text editing.

Взаимодействие человека и компьютера. Управление курсором при редактировании текста.

ОБ АВТОРЕ

Магазанник Валерий Дмитриевич – известный ученый в области создания и функционирования сложных человеко-машинных систем, доктор психологических наук, профессор, действительный член Академии качества.

Автор более 100 научных работ в области «человеческого фактора», эргономики технических систем и потребительских товаров. Основные работы посвящены процессам принятия решения операторами и их математическому моделированию, структуре памяти человека, процессам обобщения и классификации, структурированию информации человеком при принятии решений, закономерностям поведения в условиях стресса и др. Прикладные работы охватывают весь спектр эргономических задач при разработке, эксплуатации и реинжинирингу технических систем и потребительских товаров.

В последние годы активно занимается человеко-компьютерным взаимодействием, оптимизацией организационно-управленческих структур, вопросами функционирования современных социотехнических систем, особенностями работы и управления в виртуальных коллективах, в том числе с учетом растущей социально-экономической роли последних.