

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Т. П. Брусенцова, Т. В. Кишкурно

ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСОВ ПОЛЬЗОВАТЕЛЯ

*Рекомендовано
учебно-методическим объединением
по химико-технологическому образованию в качестве
пособия для студентов учреждений высшего образования
по специальности
1-47 01 02 «Дизайн электронных и веб-изданий»*

Минск 2019

УДК 004.5(075.8)
ББК 32.97я73
Б89

Рецензенты:

кафедра программного обеспечения информационных технологий
УО «Белорусский государственный университет информатики
и радиоэлектроники» (кандидат технических наук, доцент,
заведующий кафедрой *Н. В. Лапицкая*);
кандидат педагогических наук, доцент,
заведующий кафедрой информационных технологий
и моделирования экономических процессов
УО «Белорусский государственный аграрный технический
университет» *О. Л. Сапун*

Все права на данное издание защищены. Воспроизведение всей книги или ее части не может быть осуществлено без разрешения учреждения образования «Белорусский государственный технологический университет».

Брусенцова, Т. П.

Б89 Проектирование интерфейсов пользователя : пособие для студентов специальности 1-47 01 02 «Дизайн электронных и веб-изданий» / Т. П. Брусенцова, Т. В. Кишкурно. – Минск : БГТУ, 2019. – 172 с.
ISBN 978-985-530-799-1.

Пособие содержит сведения о принципах создания удобных и привлекательных с точки зрения пользователя интерфейсов, о требованиях, предъявляемых к дизайну интерфейсов, ориентированному на пользователей, о стадиях их проектирования и критериях качества. Рассматриваются основные способы прототипирования для реализации простых и сложных схем взаимодействия с пользователем и инструменты прототипирования, а также средства реализации на практике концепции юзабилити-тестирования, стратегия и варианты тестирования.

УДК 004.5(075.8)
ББК 32.97я73

ISBN 978-985-530-799-1 ©УО «Белорусский государственный
технологический университет», 2019
© Брусенцова Т. П., Кишкурно Т. В.,
2019

ПРЕДИСЛОВИЕ

Данное пособие содержит материал, предназначенный для изучения дисциплины «Проектирование интерфейсов продуктов мас-смедиа» студентами специальности 1-47 01 02 «Дизайн электронных и веб-изданий».

В настоящем пособии, состоящем из девяти глав, представлены принципы создания удобных и привлекательных с точки зрения пользователя интерфейсов, требования, предъявляемые к дизайну интерфейсов, ориентированному на пользователя, стадии их проектирования, стандарты по юзабилити и дизайну, а также критерии качества интерфейсов. В издании рассмотрены основные способы прототипирования для реализации простых и сложных схем взаимодействия с пользователем и инструменты прототипирования, а также средства реализации на практике концепции юзабилити-тестирования, стратегия и варианты тестирования.

Книга дает будущему специалисту широкий набор практических навыков по определению цели, задач и этапов проектирования интерфейсов, что позволит в дальнейшем эффективно использовать полученные знания в практической работе.

Сведения, получаемые при изучении данного курса, будут также востребованы при изучении специальных дисциплин и станут инструментом для грамотного выполнения и оформления интерфейсов программных продуктов различной направленности.

Данное пособие будет полезным и при выполнении студентами лабораторных работ по одноименной дисциплине.

Глава 1

ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

Проектирование – это один из наиболее важных (если не самый важный) этапов разработки приложения.

Проектирование (от лат. *projectos* – брошенный вперед) – процесс создания проекта-прототипа, прообраза предполагаемого или возможного объекта, состояния, процесса.

Проектирование можно определить как процесс разработки проекта и его фиксации в какой-либо внешне выраженной форме. Таким образом, процесс проектирования представляет собой последовательность этапов, которой следуют проектировщики при поиске и реализации решений.

Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, обеспечивающих взаимодействие пользователя и вычислительной системы.

ГОСТ «Эргономика взаимодействия человек-система», введенный в 2012 г., определяет пользовательский интерфейс (ПИ) как «компоненты интерактивной системы, предоставляющие пользователю информацию и являющиеся инструментами управления для выполнения определенных задач».

Проектирование пользовательского интерфейса – это создание тестовой версии приложения. Это начальный этап разработки пользовательского интерфейса, когда распределяются функции приложения по экранам, определяются макеты экранов, содержимое, элементы управления и их поведение.

Пользователь при обращении с интерфейсом должен представить себе, какая информация о выполняемой задаче у него существует, и в каком состоянии находятся средства, с помощью которых он будет решать данную задачу. Эффективность работы пользователя и его интерес обеспечивает правильно сформулированная методика разработки и проектирования пользовательского интерфейса.

Именно поэтому необходимо большое внимание уделять процессу построения пользовательских интерфейсов (UI) и выстраиванию

ванию пользовательского опыта в целом (UX). Проектирование UI – это не разовая фаза проекта, это непрерывный итерационный процесс, в который вовлечены бизнес-пользователи, UX-инженеры, дизайнеры и программисты.

UI (User Interface – дословно «пользовательский интерфейс») – то, как выглядит интерфейс, и то, какие физические характеристики приобретает. Определяет, какого цвета будет ваше «изделие», удобно ли будет человеку попадать пальцем на кнопки, читабельным ли будет текст и тому подобное.

UI – это дизайн визуальной составляющей интерфейса. UI-дизайнер делает графическую и текстовую информацию приятной и привлекательной, тем самым создает эмоциональную связь пользователя с интерфейсом.

UX (User Experience – дословно: «опыт пользователя») – это то, какой опыт/впечатление получает пользователь от работы с вашим интерфейсом. Удастся ли ему достичь цели и насколько просто или сложно это сделать.

UX-проектирование пользовательского интерфейса – проектирование взаимодействия пользователя с интерфейсом. UX-дизайнер разрабатывает сценарии того, как и каких целей может достигать пользователь при взаимодействии с цифровым продуктом.

UX/UI-дизайн – это проектирование любых пользовательских интерфейсов, в которых удобство использования так же важно, как и внешний вид. UX и UI неразрывно связаны между собой, и являются неотъемлемой частью успешного цифрового продукта.

Визуально привлекательный и удобный пользовательский интерфейс – ключевой показатель качества сайта. В сочетании с грамотной структурой и логичной навигацией по разделам ресурса, он привлекает посетителей и улучшает функциональность сайта. Главная задача в такой работе, как проектирование web-интерфейсов, – максимально упростить жизнь пользователю, сделать так, чтобы он достигал желаемый результат, затрачивая минимум усилий.

Пользовательский интерфейс

Интерфейс пользователя (UI – англ. *user interface* – пользовательский интерфейс) представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными устройствами и аппаратурой.

Иными словами, это тот набор кнопок, ссылок, форм, диалоговых окон, иконок, пиктограмм, баннеров, ползунков и лент прокрутки, с помощью которого пользователь управляет продуктом.

Интерфейс – только половина во взаимодействии с системой, другая половина – человек, пользователь. Для хорошей работы интерфейса нужно точно знать, что именно в любой конкретный момент пользователь воспринимает в интерфейсе, о чем думает, чего хочет добиться.

Интерфейсы являются основой взаимодействия всех современных информационных систем. Если интерфейс какого-либо объекта (персонального компьютера, программы, функции) не изменяется (стабилен, стандартизирован), это дает возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами.

Например, научившись работать с одной программой Microsoft Office, пользователь с легкостью освоит и другие, потому что они имеют одинаковый интерфейс.

Пользовательский интерфейс (ПИ) делится на следующие компоненты:

– **интерфейс командной строки** – инструкции компьютеру даются путем ввода с клавиатуры текстовых строк (команд). В этом виде интерфейса человек подает «команды» компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс реализован в виде пакетной технологии и технологии командной строки;

– **графический интерфейс пользователя** (или **WIMP-интерфейс**: *Window* – окно, *Image* – образ, *Menu* – меню, *Pointer* – указатель) – программные функции представляются графическими элементами экрана. Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов – меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается «опосредственно», через графические образы. Данный вид интерфейса реализован на двух уровнях технологий: простой графический интерфейс и «чистый» WIMP-интерфейс. Отличительные особенности простого графического интерфейса: выделение областей экрана; переопределение клавиш клавиатуры в зависимости от контекста; использование манипуляторов и серых клавиш клавиатуры для управления курсором. Соб-

ственno **WIMP**-интерфейс характеризуется следующими особенностями: вся работа с программами, файлами и документами происходит в окнах; все объекты представляются в виде значков; все действия с объектами осуществляются с помощью меню; применяется широкое использование манипуляторов для указания на объекты;

– *естественно-языковой интерфейс* (или **SILK**-интерфейс: *Speech* – речь, *Image* – образ, *Language* – язык, *Knowledge* – знание) – пользователь «разговаривает» с программой на родном ему языке.

Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный «разговор» человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применение началось для военных целей. Сейчас этот вид интерфейса широко осваивается, особенно для мобильных приложений. Он использует следующие технологии.

Речевая технология. При использовании этой технологии команды подаются голосом путем произнесения специальных зарезервированных слов-команд. Интерфейс часто отождествляется с диалогом, который подобен диалогу или взаимодействию между двумя людьми. Речевые технологии распознают, анализируют и синтезируют голос человека. Имитация речи, восприятие смысла фраз, конвертация речи в текст, работа с голосом как с биометрической характеристикой – все это разные типы речевых технологий.

Биометрическая технология. Здесь человек предстает как совокупность признаков поведения. Картинка считывается с цифровой видеокамеры, а затем с помощью специальных программ распознавания образов из этого изображения выделяются команды.

Семантическая технология. Об этой технологии известно крайне мало. Похоже, что она тесно связана с искусственным интеллектом и сходна со всеми подтипами **SILK** и другими типами тоже. Возможно, что в связи с важным военным значением этих разработок эти направления были засекречены.

Визуально привлекательный и удобный пользовательский интерфейс – ключевой показатель качества сайта. В сочетании с грамотной структурой и логичной навигацией по разделам ресурса, он привлекает посетителей и улучшает функциональность сайта. Главная задача в такой работе, как проектирование web-интерфейсов, – максимально упростить жизнь пользователю, сделать так, чтобы он достигал желаемый результат, затрачивая минимум усилий.

Преимущества хорошего ПИ:

- 1) повышение конкурентоспособности;
- 2) снижение стоимости разработки;
- 3) увеличение аудитории продукта;
- 4) уменьшение затрат на обучение и поддержку пользователей;
- 5) уменьшение потерь производительности работников при внедрении системы и более быстрое восстановление утраченной производительности;
- 6) доступность функциональности системы для максимального количества пользователей;
- 7) снижение риска ошибок.

Отличительные черты качественного интерфейса:

1. *Стилевая гибкость* – возможность использовать различные интерфейсы с одним и тем же приложением, на практике реализуется в виде набора «skins», для web-интерфейсов – с помощью таблицы стилей, в том числе возможность в выборе пользователем собственных установок ПИ (цвет, иконы, подсказки и пр.).

2. *Совместное наращивание функциональности* – возможность развивать приложение без разрушения (т. е. оставаясь в рамках) существующего интерфейса.

3. *Масштабируемость* – возможность легко настраивать и расширять как интерфейс, так и само приложение при увеличении числа пользователей, рабочих мест, объема и характеристик данных.

4. *Адаптивность к действиям пользователя* – приложение должно допускать возможность ввода данных и команд множеством разных способов (клавиатура, мышь, другие устройства) и многовариативность доступа к прикладным функциям (иконы, «горячие клавиши», меню и т. д.). Кроме этого программа должна учитывать возможность перехода и возврат от окна к окну, от режима к режиму, и правильно обрабатывать такие ситуации.

5. Независимость в ресурсах – для создания пользовательского интерфейса должны предоставляться отдельные ресурсы, направленные на хранение и обработку данных, необходимых для поддержки пользователя (пользовательские словари, контекстно-зависимые списки, наборы данных по умолчанию или по последнему запросу, истории запросов и пр.).

6. Кроссплатформенность – при переходе на другую аппаратную (программную) платформу должен осуществляться автоматический перенос и пользовательского интерфейса, и конечного приложения.

7. Мультимедийность – совокупность всех видов информации (графической, звуковой, видео).

Глава 2

ВЕБ-ЭРГОНОМИКА И ЮЗАБИЛИТИ

Часто термины «юзабилити» и «эргоноомика» употребляются в качестве синонимов, однако между ними есть небольшая разница: эргономичность описывает количество затраченных физических сил для работы с сайтом, а юзабилити – общую степень удобства пользования, сумму умственных усилий, требующихся от пользователя для выполнения задачи. Именно поэтому разработчики, как правило, делают акцент на «юзабильности» сетевых проектов.

Эргономика (от греч. *ergon* – работа и *potos* – закон) – научно-прикладная дисциплина, занимающаяся изучением и созданием эффективных систем, управляемых человеком.

Термин в написании «*ergonomia*» (эргономия) впервые был использован польским ученым Войцехом Ястшембовским в 1857 г. в его работе «План эргономики, т. е. науки о труде, основанной на истинах, взятых из естественных наук».

Дальнейшее развитие эргономика получила в 20-х гг. XX в., в связи со значительным усложнением техники, которой должен управлять человек в своей деятельности. Первые исследования в этой области начали проводиться в СССР, Великобритании, США и Японии.

В 1949 г. термин «эргоноомика» был принят в Великобритании, когда группа английских ученых положила начало организации Эргономического исследовательского общества. В СССР в 20 -е гг. XX в. предлагалось название «эргология», в США раньше имелось собственное наименование – *исследование человеческих факторов*, а в ФРГ – *антропотехника*, но в настоящее время наибольшее распространение получил английский термин.

В 1986 г. профессором А. Е. Аствацатуровым был введен термин «инженерная эргономика», а также его методы и методологическая основа.

Более широкое определение эргономики, принятое в 2010 г. Международной ассоциацией эргономики (*IEA (англ.) International Ergonomics Association*), звучит так: «*Научная дисциплина, изучающая взаимодействие человека и других элементов системы, а также сфера деятельности по применению теории, принципов,*

данных и методов этой науки для обеспечения благополучия человека и оптимизации общей производительности системы».

В последнее время эргономика отходит от классического определения и перестает быть напрямую связана с производственной деятельностью.

Эргономика изучает действия человека в процессе работы, скорость освоения им новой техники, затраты его энергии, производительность и интенсивность при конкретных видах деятельности.

Современная эргономика подразделяется на микроэргономику, мидиэргономику и макроэргономику.

Микроэргономика (иногда ее неверно упоминают как миниэргономику) занимается исследованием и проектированием систем «человек – машина». В частности, проектирование интерфейсов программных продуктов находится в ведении микроэргономики.

Мидиэргономика занимается изучением и проектированием систем «человек – коллектив», «коллектив – организация», «коллектив – машина», «человек – сеть». Именно мидиэргономика исследует производственные взаимодействия на уровне рабочих мест и производственных задач. К ведению мидиэргономики, в частности, относится проектирование структуры организации и помещений; планирование и установление расписания работ; гигиена и безопасность труда.

Макроэргономика исследует и проектирует систему в целом, учитывая все факторы: технические, социальные, организационные; как внешние к системе, так и внутренние. Целью макроэргономики является гармоничная, согласованная, надежная работа всей системы и всех элементов системы.

Целью эргономичного дизайна является преодоление проблем в области человеко-компьютерного взаимодействия и оптимизация человека-ориентированного интерфейса, основанная на принципах и методах юзабилити.

Рождение веб-дизайна как части графического дизайна вызвало к жизни появление такого ранее не существовавшего понятия, как юзабилити.

Юзабилити (англ. *usability* – «возможность использования», «способность быть использованным», «полезность») – степень эффективности, продуктивности и удовлетворенности, с которыми

продукт может быть использован определенными пользователями в определенном контексте использования для достижения определенных целей (пункт 3.1 стандарта ISO 9241-11).

Понятие юзабилити в микроэргономике – это эргономическая характеристика степени удобности предмета для применения пользователями при достижении определенных целей в некотором контексте.

Единственное более-менее ощутимое различие между эргономикой и юзабилити – первое ставит больший акцент на технические характеристики, второе же – на процесс взаимодействия. В основе эргономики стоит сам продукт, в основе юзабилити – пользователь.

Эргономичность является количественной характеристикой, описывающей количество затраченных физических сил для работы с сайтом, а юзабилити – качественной, описывающей сумму умственных усилий, требующихся от пользователя для выполнения задач, а также общую степень удобства пользования.

Практикующие веб-дизайнеры говорят о юзабилити в связи с применением методов улучшения функциональности и удобства сайта на стадии его разработки. Для того, чтобы пользователь вернулся на сайт или начал регулярно использовать мобильное приложение, необходимо спроектировать его так, чтобы учесть все потребности аудитории, в случае с журналистикой главная задача пользователя – получить информацию наиболее простым и комфорtnым способом, в отличие от, например, интернет-магазинов, где основной целью остается акт продажи товара или услуги.

Основная цель юзабилити – проконтролировать, спрогнозировать и воздействовать на процессы создания системы с целью повысить конечную эргономичность продукта.

Уже в самом определении юзабилити присутствуют три важных аспекта:

– **определенные пользователи**: интерфейс всегда создается для конкретной группы людей (или нескольких групп), характеризующейся своими особенностями, знаниями, навыками, ожиданиями, сильными и слабыми сторонами. Системы никогда не создаются для «любого» или «среднего» пользователя, даже если позиционируются как «подходящие всем»;

– **определенный контекст использования**: решения, которые подходят для пользователя идеально в одном контексте, могут

быть совершенно неприемлемы в другом. К примеру, управление посредством голосового интерфейса может оказаться удачным решением в условиях работы оператора в изолированном помещении, однако для совместной работы в одной комнате использование такой системы несколькими операторами одновременно будет затруднительным;

– **определенные цели**: ни одно решение в принципе не может быть удачным, если оно не помогает достичь пользователю его целей. А для того, чтобы оно помогало их достичь, нужно узнать, в чем эти цели состоят.

Интерфейс должен быть юзабильным. Но не бывает абсолютной вещи, удобной для всех. Она удобна сейчас для решения конкретной задачи конкретным человеком, но все изменяется, как только здесь изменится какая-то переменная, т. е. другой человек, не тот, на которого рассчитывали (например, рассчитывали на мужчин, а пришла девушка или наоборот, сделали для девушки, а пришел инженер-мужчина – и не соответствует продукт его ментальным моделям). Он пытается не по назначению использовать продукт. Например, сделали веб-сайт, а он его пытается с КПК или мобильного телефона смотреть. Но сайт не рассчитан на это.

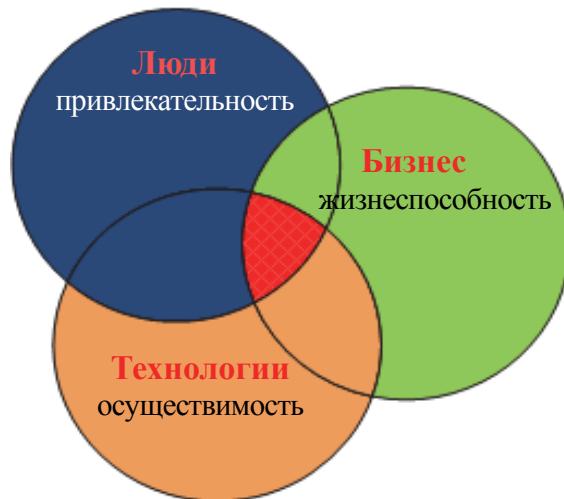
Юзабилисты достигают «золотой середины». Чтобы быть хорошим юзабилистом, необходимо быть хорошим психологом, чего не хватает всегда людям техническим. Они думают о технологии, а не о том, с какой «блондинкой» встретится этот сайт, и как он должен ее завлекать.

Юзабилити – это технология, влияющая на все этапы разработки программной системы. Основывается она на особенностях психологического восприятия информации человеческим мозгом. Основная цель – проконтролировать, спрогнозировать и воздействовать на процессы создания системы с целью повысить конечную эргономичность продукта.

В процессе всего периода разработки интерфейса важно помнить, что именно так, как пользователь видит интерфейс на экране, так он и воспринимает весь продукт в целом. Следовательно, в его понимании, если программа неудобна, значит, она бесполезна во всех своих проявлениях. Тем более это относится к сайтам и играм – конкуренция в этом секторе рынка слишком высока, чтобы позволить себе создавать неудобные продукты. Почему это важно и зачем это нужно знать разработчику?

Основная аксиома пользовательского интерфейса – хороший дизайн, который подразумевает, что программа соответствует ожиданиям пользователей о том, как она должна себя вести. Все остальное – следствия.

Дизайн – это сознательные и интуитивные усилия по созданию значимого порядка. Дизайн интерфейса позволяет придать определенный вид уже существующему поведению системы.



Дизайн, ориентированный на человека

Дизайн интерфейса должен начинаться с людей. Вы разрабатываете его для их потребностей, моделей поведения и желания. Вы будете смотреть на мир сквозь эту призму «привлекательности» в течение всего срока проекта. После того, как вы знаете, что желательно, вы начнете просматривать решения через призму «Технологии» и «Бизнес». Дизайн, ориентированный на человека, должен попасть в перекрытие трех линз (рисунок): решения, которые привлекательны для людей, возможны и жизнеспособны. Он позволяет создавать решения, соответствующие потребностям и целям пользователей с одной стороны, а также бизнес-требованиям и технологическим ограничениям – с другой.

Есть программы, базы данных, сервера, но пользователь работает с интерфейсом. Интерфейс – это то, что видит пользователь, когда он работает с программой. Это наподобие айсберга. Человек видит только надводную часть, т. е. интерфейс. А значит, взаимодействие с ним происходит через интерфейс, который должен подчиняться определенным нормам. Для управления этими нор-

мами и нужно юзабилити. Дизайн же нужен для того, чтобы все это грамотно воплотить в жизнь.

В итоге наша основная цель – доставить пользователю наиболее приятные ощущения при взаимном обмене информацией, во время работы с нашей системой.

Юзабилити и дизайн важны каждому человеку, причастному к разработке. Почему? Потому что целью практически любого разработчика является привлечение потенциального пользователя. Пользователю же в свою очередь важно получать наиболее позитивные чувства. Его не интересуют внутренние процессы, он их не видит.

Главная цель дизайна, ориентированного на юзабилити – быть невидимым. Современная дизайнерская практика показывает – проектировщики часто пренебрегают этим правилом, опираясь лишь на собственную интуицию или рекомендации заказчика, что в конечном счете приводит к снижению функционального компонента дизайна.

По мнению Яакоба Нильсена, удобство использования, или юзабилити, определяется пятью ключевыми компонентами:

1. **Обучаемость.** Насколько быстро пользователи выполняют основные задачи и ориентируются в дизайне, сталкиваясь с ним впервые.

2. **Эффективность.** Скорость выполнения задач и ориентации в проекте, если пользователи ранее ознакомились с дизайном.

3. **Запоминаемость.** Простота и скорость взаимодействия пользователей с дизайном по прошествии длительного времени.

4. **Ошибки.** Количество и качество ошибок пользователей при работе с ресурсом, а также возможность их исправления.

5. **Удовлетворенность.** Оценка эстетических качеств дизайна и степень удовольствия работы с ним.

Все эти вопросы изучает юзабилити (*Usability Engineering*) – научно-прикладная дисциплина, способствующая повышению эффективности, продуктивности и удобству использования инструментов деятельности, в том числе – программного обеспечения.

Когда говорят о научных основах проектирования пользовательских интерфейсов, в первую очередь упоминают термин HCI (англ. *human-computer interaction* – взаимодействие человека и компьютера).

HCI – это научная и прикладная дисциплина, предметом которой является то, как люди используют компьютеры и как следует разрабатывать компьютерные системы для того, чтобы обеспечить более эффективное их применение конечными пользователями. Дисциплина включает элементы психологии, эргономики, информатики, графического дизайна, социологии и антропологии. Составными частями HCI являются: человек (пользователь), компьютер и их взаимодействие.

На Западе HCI – это целая профессия, ей обучают в университетах, выдавая дипломы «Специалист по HCI». Издаётся много журналов по этой теме, существует большое количество web-сайтов.

Также научным фундаментом для юзабилити являются:

– **UCD** (*user-center design, проектирование, ориентированное на нужды пользователей*). Подход к проектированию компьютерных систем (приложений, веб-сайтов), при котором на первый план ставятся пользователи, их нужды и требования. При использовании данного подхода достигается высокий уровень юзабилити, что сказывается, например, на увеличении продаж (в электронном магазине);

– **Human-centred design: ISO 9241–210 [7]** – это существующий стандарт проектирования. Human-centred design учитывает не только процесс взаимодействия пользователя с системой, но и контекст, т. е. каким образом человек будет с ней взаимодействовать (например, взаимодействие с сайтом родителей школьника, который пользуется сайтом и др.);

– **IA** (*information architecture – информационная архитектура*). Совокупность методов и приемов организации информации для облегчения выполнения людьми их информационных нужд по поиску и просмотру информации.

Уже в начале нынешнего века стало понятно, что при разработке цифровых продуктов необходимо учитывать опыт взаимодействия пользователя с этими продуктами. Пользователи устали от «технологии ради технологии». Крупнейшие учебные заведения, такие как Стэнфордский университет и Harvard Business School, признали, что следующее поколение управленцев и технологов должно выделять проектированию взаимодействия определенное место в своих бизнес-планах и графиках разработок – и для этого необходимо изучать его в программах их подготовки. В сен-

тябре 2005 г. официально родилась организация **IxDA Interaction Design Association** (*ассоциация проектирования взаимодействия*). Можно считать, что проектирование взаимодействия наконец становится самостоятельной дисциплиной.

Конечно, проектирование продукта не может концентрироваться только на поведении, нужно учитывать внешний вид (форму) и информационное наполнение (содержание) продукта. Поэтому необходимо сочетать подходы различных дисциплин проектирования: проектирование взаимодействия, где основное внимание направлено на поведение; информационную архитектуру, которая занимается структурированием содержания; промышленный и графический дизайн, который отвечает за форму продуктов и услуг.

Глава 3

КРИТЕРИИ КАЧЕСТВА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Существует четыре основных критерия качества любого интерфейса, а именно: скорость работы пользователей; количество человеческих ошибок; скорость обучения; субъективное удовлетворение пользователей (подразумевается, что соответствие интерфейса задачам пользователя является неотъемлемым свойством интерфейса).

Скорость работы пользователей

Скорость выполнения работы является важным критерием эффективности интерфейса. Любая попытка как-то увеличить производительность труда всегда встречается положительно.

Длительность выполнения работы пользователем состоит из следующих составных частей: длительности восприятия исходной информации; интеллектуальной работы (пользователь думает, что он должен сделать); физических действий пользователя; реакции системы.

Длительность восприятия исходной информации в особых комментариях не нуждается. Пользователь должен представить себе, какая информация о выполняемой задаче у него существует, и в каком состоянии находятся средства, с помощью которых он будет решать данную задачу. Основное время здесь пойдет на считывание показаний системы.

Длительность интеллектуальной работы – оценивается взаимодействие пользователя с системой (не только компьютерной) и состоит из семи шагов:

1. Формирование цели действий.
2. Определение общей направленности действий.
3. Определение конкретных действий.
4. Выполнение действий.
5. Восприятие нового состояния системы.
6. Интерпретация состояния системы.
7. Оценка результата.

Из этого списка становится видно, что процесс размышления занимает почти все время, в течение которого пользователь работает с компьютером, во всяком случае, шесть из семи шагов полностью заняты умственной деятельностью.

К сожалению, существенно повысить скорость собственного мышления пользователей невозможно. Тем не менее, уменьшить влияние факторов, усложняющих и, соответственно, замедляющих процесс мышления, вполне возможно.

Факторы, позволяющие ускорить процесс мышления:

– **непосредственное манипулирование.** Смысл этого метода очень прост. Пользователь не отдает команды системе, а манипулирует объектами. Первым популярным применением данного метода была корзина для удаления файлов в компьютерах *Macintosh* (начиная с *Windows 95*, такая корзина стала стандартом и в *Windows* мире). Если перетащить в нее пиктограмму файла, этот файл будет фактически стерт. На самом деле процесс стирания файла состоит из многих малых, уже неделимых действий (жестов). При этом для ускорения мыслительной работы пользователя необходимо не только сокращать количество этих жестов, но и делать их более простыми;

– **применение в интерфейсе эффективных методов при потере фокуса внимания.** Работая с системой, пользователи постоянно отвлекаются. Таким образом, необходимо максимально облегчать их возвращение к работе и проектировать интерфейс так, чтобы пользователи возможно меньше о нем думали.

Итак, 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.

Предоставлять пользователю всю эту информацию лучше всего визуально.

Длительность физических действий пользователя зависит от степени автоматизации работы и степени необходимой точности работы. Об автоматизации что-либо конкретное сказать сложно. Понятно, что чем больше работы делает компьютер, тем лучше. С точностью все гораздо проще. Любое физическое действие, совершаемое с помощью мускулатуры, может быть или точным,

или быстрым. Вместе точность и быстрота встречаются исключительно редко.

Пользователь, как правило, управляет компьютером двумя способами: мышью и клавиатурой. Клавиатура не требует особой точности движений – неважно, быстро нажали клавишу или медленно, равно как сильно или слабо. Мышь, напротив, инерционна. Именно поэтому оптимизация использования мыши в системе может существенно повысить общую скорость работы.

Мышь не предназначена для очень точных, в 1 или 2 пикселя, манипуляций (попробуйте мышью нарисовать ровный круг). Обычно в графических программах всегда есть возможность перемещать объекты клавишами со стрелками. Именно поэтому любой маленький интерфейсный элемент будет всегда вызывать проблемы у пользователей.

Популярно говоря, лучший способ повысить доступность кнопки заключается в том, чтобы делать ее большой и располагать ближе к курсору. У этого правила есть два не сразу заметных следствия. Чтобы «бесконечно» ускорить нажатие кнопки, ее, во-первых, можно сделать бесконечного размера и, во-вторых, дистанцию до нее можно сделать нулевой.

Кнопка бесконечного размера. При подведении курсора к краю экрана он останавливается, даже если движение мыши продолжается. Это значит, что кнопка, расположенная впритык к верхнему или нижнему краю экрана, имеет бесконечную высоту (равно как кнопка у левого или правого края имеет бесконечную ширину). Таким образом, скорость достижения такой кнопки зависит только от расстояния до нее и точности выбора начального направления движения. Понятно, что кнопка, расположенная в углу экрана, имеет бесконечные размеры (не важно даже, с какой точностью перемещали мышь). Для достижения такой кнопки от пользователя требуется всего лишь дернуть мышь в нужном направлении, не заботясь о ее скорости и не делая попыток остановить ее в нужном месте. Это делает такие кнопки наиболее доступными для пользователя, жалко даже, что у экрана всего четыре угла.

Именно поэтому, например, меню MacOS многократно эффективней меню Windows: если в MacOS меню всегда расположено впритык к верхнему краю экрана, то в Windows меню отделено от края экрана полосой заголовка окна программы (Title Bar).

Нулевая дистанция до кнопки. Рассмотрим контекстное меню, вызываемое по нажатию правой кнопки мыши. Оно всегда открывается под курсором, соответственно, расстояние до любого его элемента всегда минимально. Именно поэтому контекстное меню является чуть ли не самым быстрым и эффективным элементом.

Длительность реакции системы. Часто пользователи надолго прерывают свою работу. Попросту говоря, система делает что-либо длительное. Например, печать документа в сто страниц даже на быстрых принтерах занимает существенное время, соответственно, большинство людей, отправив такой документ в печать, начинают бездельничать. Проблема в том, что сразу после того, как человек отвлекается, система зачастую, во что бы то ни стало, начинает требовать что-либо от человека. Например, появляется диалоговое окно с вопросом «Вы уверены?».

Если процесс предположительно будет длительным, система должна убедиться, что она получила всю информацию от пользователя до начала этого процесса.

Есть другое решение данной проблемы: система может считать, что если пользователь не ответил на вопрос, скажем, в течение пяти минут, то его ответ положительный.

Таким образом, тот же самый сценарий решается по-другому: пользователь отправляет документ на печать и уходит, система спрашивает «Вы уверены?» и ждет пять минут, после истечения этого времени она начинает печать. Этот метод вполне работоспособен, так что им стоит пользоваться всегда, когда невозможен первый метод (разумеется, за исключением случаев, когда ответ «Да» может иметь катастрофические последствия).

Убирайте с экрана все диалоги с вопросами, на которые в течение пяти минут не был дан ответ.

Количество человеческих ошибок

Важным критерием эффективности интерфейса является количество человеческих ошибок.

Человек при работе с компьютером постоянно совершает ошибки.

Часто минимальная ошибка приводит к абсолютно катастрофическим последствиям, например, за одну секунду оператор в банке может сделать кого-то богаче, а банк, в свою очередь, беднее (впрочем, обычно беднее становятся все).

Типы ошибок. Существует великое множество классификаций человеческих ошибок. Среди наибольшего количества человеческих ошибок можно выделить четыре типа:

1) **ошибки, вызванные недостаточным знанием предметной области.** Теоретически эти ошибки методологических проблем не вызывают, сравнительно легко исправляясь обучением пользователей. Практически же, роль этих ошибок чрезвычайно велика – никого не удивляет ситуация, когда оператора радарной установки перед началом работы долго учат, и в то же время все ожидают должного уровня подготовки от пользователей ПО, которых никто никогда ничему целенаправленно не обучал. Еще хуже ситуация с сайтами, у которых даже справочной системы почти никогда не бывает;

2) **опечатки** происходят в двух случаях: когда не все внимание уделяется выполнению текущего действия (этот тип ошибок характерен, прежде всего, для опытных пользователей, не проверяющих каждый свой шаг) и когда в мысленный план выполняемого действия вклинивается фрагмент плана из другого действия (происходит преимущественно в случаях, когда пользователь имеет обдуманное текущее действие и уже обдумывает следующее действие);

3) **ошибки, вызванные несчитыванием показаний системы,** которые одинаково охотно производят как опытные, так и неопытные пользователи. Первые не считывают показаний системы потому, что у них уже сложилось мнение о текущем состоянии, и они считают излишним его проверять, вторые – потому что они либо забывают считывать показания, либо не знают, что это нужно делать и как;

4) **моторные ошибки.** Количество их фактически пренебрежимо мало, но, к сожалению, не так мало, чтобы вовсе их не учитывать. Сущностью этих ошибок являются ситуации, когда пользователь знает, что он должен сделать, знает, как этого добиться, но не может выполнить действие нормально ввиду того, что физические действия, которые нужно выполнить, выполнить трудно. Так, никто не может с первого раза (и со второго тоже) нажать на экранную кнопку размером 1 на 1 пиксель. При увеличении размеров кнопки вероятность ошибки снижается, но почти никогда не достигает нуля. Соответственно, единственным средством из-

бежать этих ошибок является снижение требований к точности движений пользователя.

Для минимизации количества ошибок нужно:

- плавно обучать пользователей в процессе работы;
- повышать разборчивость и заметность индикаторов состояния;
- снижать чувствительность системы к ошибкам.

Для этого есть три основных способа, а именно:

- блокировка потенциально опасных действий пользователя до получения подтверждения правильности действия;
- проверка системой всех действий пользователя перед их принятием;
- самостоятельный выбор системой необходимых команд или параметров, при этом от пользователя требуется только проверка.

Самым эффективным является третий способ. Но он труден в реализации. Рассмотрим эти три способа подробнее.

Блокировка потенциально опасных действий до получения подтверждения. Команда удаления файла в любой операционной системе снабжена требованием подтвердить удаление (рис. 3.1). Эта блокировка приносит пользу только начинающим пользователям, которые проверяют каждый свой шаг.

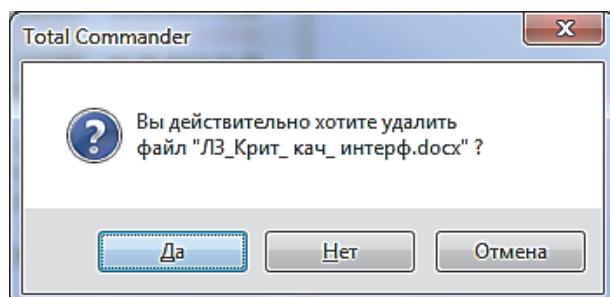


Рис. 3.1. Сообщение операционной системы

Для опытных пользователей это диалоговое окно с требованием подтверждения не работает. Во-первых, оно не защищает нужные файлы. Во-вторых, без надобности отвлекает пользователя и тратит его время.

Надо требовать подтверждения не после команды пользователя, а до нее. Например, сначала – *разблокировать*, после чего – *удалить*. Поскольку эти два действия напрямую не связаны друг с другом – если в одном из них была допущена ошибка, файл удалить не удастся.

Гораздо чаще приходится защищать не отдельные объекты (файлы, окна и т. п.), но отдельные фрагменты данных (например, текст и числа в полях ввода). Понятного и удобного элемента управления для этой цели нет. Единственным выходом служит скрытие потенциально опасных данных от пользователя до тех пор, пока он сам не скомандует системе их показать. Выход же этот отнюдь не идеальный, поскольку некоторым пользователям никогда не удастся понять, что, помимо видимых значений, есть еще и невидимые данные.

Не делайте опасные для пользователя кнопки кнопками по умолчанию.

Проверка действий пользователя перед их принятием. Этот метод гораздо лучше блокировки. Наиболее популярны два универсальных и работающих способа проверки.

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 кнопки, 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 кнопку, но при этом исчезла бы необходимость предупреждать пользователя о грядущей потере данных.

Проверкой всех действий пользователя перед их принятием можно также успешно защищать вводимые пользователем данные, в особенности данные численные. Дело в том, что большинство численных данных имеют 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 ОС есть специальный элемент управления, именуемый «крутилкой». Фактически это обычное поле ввода, снабженное двумя кнопками для модификации его содержимого (в сторону уменьшения и увеличения). Интересен он тем,

что пользователь может не пользоваться клавиатурой для ввода нужного значения, взамен клавиатуры установив нужное значение мышью. Этот элемент имеет то существенное достоинство, что при использовании мыши значение в этом элементе всегда находится в нужном диапазоне и обладает нужным форматом.

Всегда показывайте границы диапазона во всплывающей подсказке.

Но что делать, если пользователь ввел некорректное число с клавиатуры? Ответ прост. Для этого надо индицировать возможную ошибку изменением начертания шрифта на полужирное в обычных программах (иное проблематично), а в случае сайта – заменой цвета фона этого элемента на розовый (благо это нетрудно сделать через таблицу стилей).

В тех же случаях, когда количество возможных значений невелико, лучше использовать другой элемент управления – ползунок. Мало того, что он позволяет устанавливать только определенные значения (с этим справился бы и выпадающий список или комплект переключателей), но он дает возможность пользователю видеть взаимосвязь возможных значений, и при этом применение этого элемента понятно даже новичку.

Самостоятельный выбор команд. Самый эффективный способ. Чем меньше действий требуется совершить пользователю, тем меньше вероятность ошибки.

Система сама должна узнавать 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.

Единственная проблема этого метода заключается в том, что для его использования к проектированию системы нужно подходить значительно более творчески и тщательнее, нежели обычно практикуется.

Уровни ошибок и обратная связь

Помимо классификации человеческих ошибок, приведенной в начале главы, существует еще одна классификация. В ней ошибки расставлены по уровням их негативного эффекта:

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

2. Ошибки, исправляемые после выполнения действия, например, после ошибочного удаления файла его копия переносится из корзины.

3. Ошибки, которые исправить можно, но с трудом, например, реальное стирание файла, при котором никаких его копий не остается.

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

Каждый хороший программист, умеющий мыслить системно, знает, что 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.

Например, межпланетные зонды из-за ошибок в ПО улетают не туда, куда надо; коммерческие договоры, в которых обнаруживаются ошибки, приносят много неприятностей; ошибочные номера телефонов в записной книжке не дают возможности найти абонента – все это примеры неисправляемых ошибок. Разумеется, такие ошибки всегда обнаруживаются, проблема в том, что к моменту их обнаружения становится поздно их исправлять. Именно поэтому данные ошибки гораздо хуже ошибок, которые исправить трудно, но которые, по крайней мере, сразу видны.

Но не каждый хороший дизайнер интерфейса знает, что ошибок из второго пункта нужно всеми силами избегать, поскольку каждая такая ошибка обходится гораздо дороже, чем любая ошибка из первого пункта. Объясняется это просто: дизайн интерфейса гораздо моложе программирования.

Вообще говоря, объяснение данного факта двояко: как субъективное, так и объективное.

Объективное объяснение просто: ошибки, исправляемые после, снижают производительность работы. Как мы уже знаем из предыдущей главы, любое действие пользователя состоит из семи шагов. Всякий раз, когда пользователь обнаруживает, что он совершает ошибку, ему приходится возвращаться назад на несколь-

ко этапов. Более того, чтобы исправить совершенную ошибку, от пользователя требуется:

- понять, что ошибка совершена;
- понять, как ее исправить;
- потратить время на исправление ошибки.

В результате значительный процент 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 вовремя, 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.

Наличие человеческих ошибок, которых нельзя обнаружить и исправить до окончательного совершения действия, всегда свидетельствует о недостаточно хорошем дизайне.

Теперь пора сказать, как избавится от ошибок, исправляемых после. Чтобы дать пользователям исправлять их действия на ходу, надо дать им обратную связь.

Когда пользователь совершает какое-то действие – правильное, неправильное или непонятное, – интерфейс должен сообщать ему об этом. Нужно всегда уведомлять пользователя о произведенных операциях, изменениях текущего состояния, ошибках или исключениях. О том, привели ли действия пользователя к желаемому результату, могут рассказать визуальные либо текстовые сообщения. В интерфейсе Bantam Live (рис. 3.2) для всех действий существуют линейки-индикаторы степени выполнения.

К сожалению, вводить в систему обратную связь получается не всегда. Дело в том, что ее не любят программисты. Мотивируют они свое отношение тем, что она плохо влияет на производительность системы. Так что если вы чувствуете, что программисты правы, вспомните, что производительность связки «система – пользователь» всегда важнее производительности системы просто.



Рис. 3.2. Линейка-индикатор интерфейса Bantam Live

Если же и это не помогает, попробуйте спроектировать обратную связь иначе, более скромно. Например, с помощью ползунков на линейке в MS Word можно менять абзацные отступы, при этом обратная связь есть, но неполная: вместо перманентного переформатирования документа по экрану двигается полоска, показывающая, куда переместится текст.

Скорость обучения

В традиционной науке о человеко-машинном взаимодействии роль обучения операторов чрезвычайно велика. Если человек будет сочтен неподходящим, к системе его просто не допустят.

Напротив, с ПО и сайтами ситуация принципиально иная: как цель ставится возможность работы с системой для любого человека, независимо от его свойств и навыков, при этом целенаправленное обучение пользователей, как правило, не производится. Все это делает проблему обучения пользователей работе с компьютерной системой чрезвычайно важной.

Почему пользователи учатся?

Есть непреложный закон природы: *люди делают что-либо только при наличии стимула, при этом тяжесть действия пропорциональна силе стимула*. Применительно к компьютерным системам этот закон действует без каких-либо исключений.

Обучение есть действие: если обучаться легко, пользователям будет достаточно слабого стимула, если тяжело – стимул придется увеличивать. Но если стимул для пилота самолета получают преимущественно командными методами (если он не научится определенному минимуму, его просто не допустят до работы), то в

случаях компьютерных систем стимул есть вещь почти исключительно добровольная. Это значит, что пользователь обучится пользоваться программой или сайтом только в том случае, если он будет уверен, что это, к примеру, сделает его жизнь легче и приятней.

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 использование жизнь даст ему награду. Одного стимула недостаточно, если пользователь не знает, за что этот стимулдается.

Рассчитывайте на средних пользователей, а не новичков или на профессионалов: средних пользователей, как-никак, абсолютное большинство.

Средства обучения

Обычно считается, что в случае ПО есть два способа повысить эффективность обучения (помимо метода «обучения плаванию посредством выбрасывания из лодки»), а именно: бумажная документация и «оперативная справка». Но существуют более эффективные способы. Рассмотрим некоторые из них: общая «понятность» системы и обучающие материалы.

Понятность системы. Термин «понятность» включает в себя четыре составляющие: ментальную модель, метафору, аффорданс и стандарт.

Ментальная модель. Часто, чтобы успешно пользоваться какой-либо системой, человеку необходимо однозначно понимать, как система работает. При этом не обязательно точно понимать сущность происходящих в системе процессов, более того, не обязательно правильно их понимать. Такое понимание сущности системы называется *ментальной моделью*.

Ментальные модели мы создаем для упрощения картины мира. Наблюдая за событиями, мы их обобщаем и храним в памяти единую картину. Утюгом никогда не сможет воспользоваться человек, который не знает, что провод от него надо воткнуть в розетку. Но, обладая таким знанием, человек может пользоваться утюгом, не зная, сколько энергии тот потребляет (отсутствие точности), равно как сохраняя искреннюю уверенность, что по проводам как вода течет электричество (отсутствие правильности).

Метафора. Разработать пользовательский интерфейс, в котором модель программы соответствует модели пользователя, – задача не из легких. Иногда у пользователей просто нет конкретного представления о том, как работает программа и для чего она предназначена. В таком случае вам придется найти способ подсказать им, как функционирует ваша программа. В графических интерфейсах применяется метод метафор. Он позволяет пользователю не создавать новую модель, а воспользоваться готовой, которую он ранее построил по другому поводу.

Самая известная метафора, применяемая в Windows и Macintosh, – это метафора «десктоп» (рабочий стол). Перед вами маленькие папочки с листочками-файлами внутри, последние можно перемещать из одной папки в другую. Метафора работает, потому что изображения папок напоминают реальные папки, которые мы используем для хранения и сортировки документов в своих кабинетах.

Еще один пример метафоры в интерфейсе – устройство программ для проигрывания звуков на компьютере. Исторически сложилось, что вся аудиотехника имеет почти одинаковый набор кнопок: несколько кнопок со стрелками (назад / вперед), кнопка с треугольником (воспроизведение), кнопка с двумя дощечками (пауза), кнопка с квадратиком (полная остановка) и красный круглый кружок (запись). Про них нельзя сказать, что они совершенно понятны, но выучить их можно без труда. При этом обычно жизнь складывается так, что сначала человек учится пользоваться этими кнопками на материальных устройствах, а уж потом начинает пользоваться компьютером. Соответственно, при проектировании программы аналогичного назначения разумно скопировать существующую систему маркировки кнопок.

Увеличительное стекло – полновесная метафора из реального мира. Интерфейс Word 2003 содержит два маленьких увеличительных стекла (рис. 3.3), одно из которых (по непонятной причине) фигурирует под названием «Предварительный просмотр», второе же названо «Схема документа».

На самом деле, увеличить / уменьшить размер документа можно с помощью выпадающего списка, который на данный момент показывает «100%». Метафоры нет, и потому угадать, где скрывается функция приблизить / отдалить, сложнее.

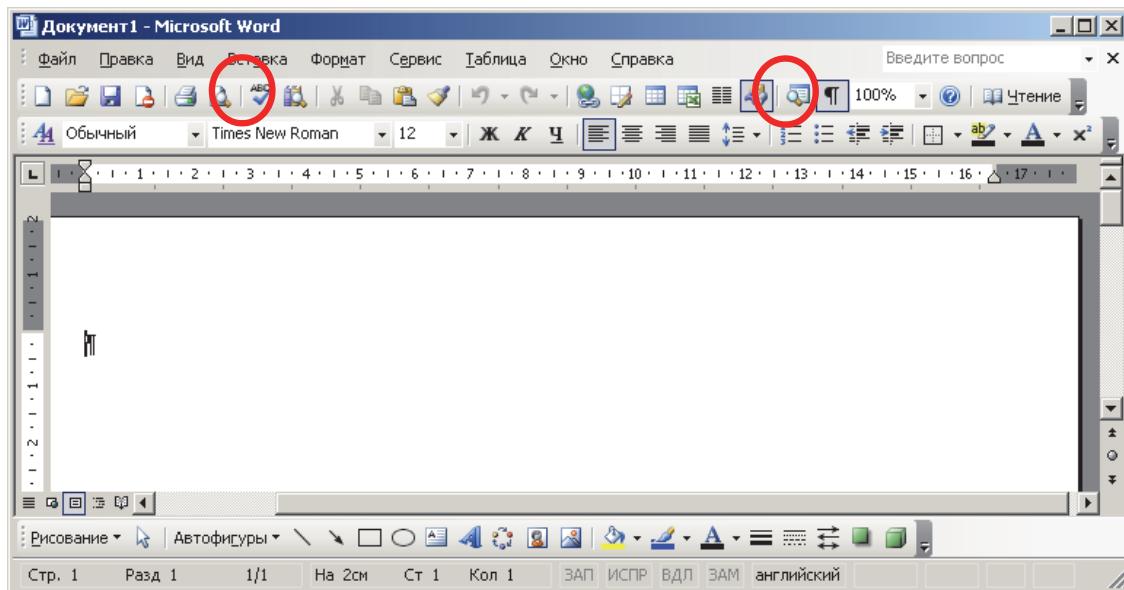


Рис. 3.3. Интерфейс Word 2003

Анализируя опыт применения метафор, можно вывести *следующие правила*:

- опасно полностью копировать метафору, достаточно взять из нее самое лучшее;
- не обязательно брать метафору из реального мира, ее смело 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.

Аффорданс. В современном значении этого термина аффордансом называется ситуация, при которой объект показывает субъекту способ своего использования своими неотъемлемыми свойствами.

Например, у двери с ручкой (рис. 3.4) аффорданс к тому, чтобы ее тянули. Но иногда ручка бывает у двери, которая открывается наружу, соответственно ее надо толкать. Образуется конфликт между аффордансом ручки (тянуть) и функцией двери (открываться толканием). Чтобы нивелировать этот конфликт, зачастую вешают табличку с указанием к действию. Более эффективное и простое решение – убрать ручку. Так образуется аффорданс к толканию.

Надпись «На себя» на двери не является аффордансом, а внешний вид двери, который подсказывает человеку, что она открывается на себя, несет в себе аффорданс.

Еще один пример аффорданса – кирпичи Лего. По дизайну поверхностей Лего можно легко понять, как кирпичики составляются в фигуры.

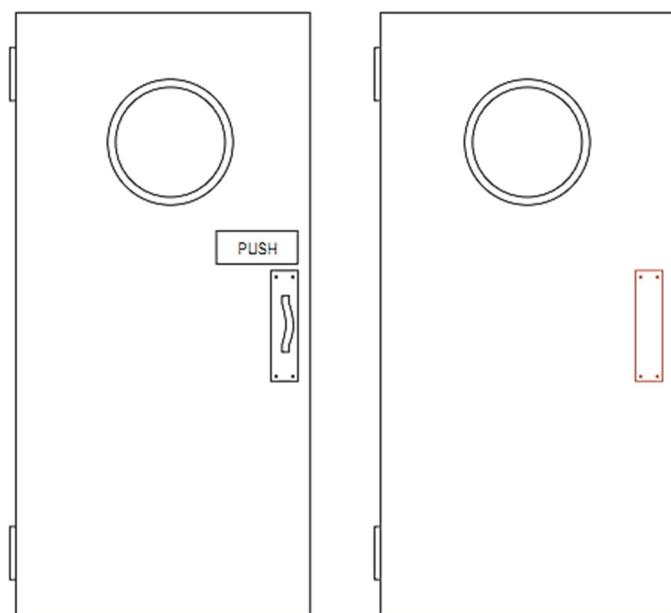


Рис. 3.4. Пример аффорданса

Польза аффорданса заключается в том, что он позволяет пользователям обходиться без какого-либо предварительного обучения, благодаря этому аффорданс является самым эффективным и надежным средством обеспечения понятности.

Принцип аффорданса активно используется в компьютерных интерфейсах. Впервые применительно к разработке интерфейса это понятие начал использовать **Дональд Норман**. Аффорданс в этом случае является подсказкой или указателем на функциональность того или иного элемента.

Самые распространенные примеры – стрелки выпадающего меню или выпуклая кнопка. Дизайн корзины и папок на Рабочем столе, ссылаясь на реальные объекты из жизни, дают понять, как их правильно использовать.

Глядя на них, пользователь понимает, какое действие требуется совершить, и что после этого произойдет. Помимо таких явных аффордансов, существуют *скрытые, метафорические* и так

называемые ***негативные*** указатели, помогающие посетителям сориентироваться на сайте. Все они могут использоваться для упрощения навигации и, как следствие, повышения конверсии сайта.

На одном сайте могут присутствовать несколько видов аффордансов, и каждый из них будет полезен посетителям, если действительно находится на своем месте. В одних случаях требуются прямые указания, в других достаточно скрытых подсказок.

Виды аффордансов: *явные, или прямые, указатели, метафорические, или символические, указатели, скрытые указатели, негативные указатели.*

Явные, или прямые, указатели призваны показать, как посетитель сайта или пользователь приложения может совершить нужное ему (или продавцу) действие. «Выпуклые» кнопки с призывом к действию, текст в полях для заполнения форм, подписи визуальных элементов – явные аффордансы, точно и недвусмысленно указывающие на назначение элемента.

Способы применения очевидны: предложения купить, оформить заказ или подписку, посмотреть товары или перейти в «корзину» необходимы и без аффордансов нереализуемы. Но явные указатели нужны и в других случаях. Если известно, что аудитория сайта слабо знакома даже с самой типичной навигацией, понадобятся подсказки. Без указателей сложно обойтись и на сайтах, где оригинальность дизайна превалирует над интуитивной понятностью интерфейса. И предлагая пользователям новое приложение, стоит объяснить, как оно работает.

Чтобы не перегрузить страницы, можно использовать сочетание текста аффорданса с визуальным элементом. Так опытные пользователи поймут, куда нужно кликать, увидев картинку, а остальные смогут сориентироваться на сайте, прочитав подписи.

Метафорические, или символические, указатели – тоже весьма распространенный вид подсказок. Предполагается, что пользователь поймет назначение элемента, потому что видел его раньше на других сайтах. Пример – изображение конверта в поле подписки на рассылку, которое с первого взгляда ассоциируется с электронной почтой. Или кнопка *call-back* виджета с телефонной трубкой. Чаще всего такие аффордансы используются для оформления главного меню и ссылок на типовые разделы сайта. Поэтому важно правильно подобрать изображение. Допустим, силуэт в платье

может трактоваться как символ и каталога платьев, и раздела с женской одеждой вообще. При этом символические указатели могут одинаково восприниматься независимо от тематики и контента или зависеть от содержания страницы.

Скрытые указатели. Такие подсказки или элементы навигации появляются после совершения какого-либо действия, чаще всего – наведения курсора. Они призваны спрятать дополнительные функции, которые перегрузили бы страницу или отвлекли бы внимание пользователей от главного контента.

Использовать такие аффордансы следует осторожно: если действие, вызывающее появление указателя, совершается редко, пользователи могут так и не узнать о скрытых возможностях.

Негативные указатели. Назначение этих подсказок – показать, что пункт меню или другой элемент в настоящее время неактивен. Как правило, это бледные или «утопленные» кнопки и текст серого цвета. Такое оформление достаточно распространено и привычно для большинства пользователей. Если же человек впервые видит подобный элемент, контраст с другими кнопками и полями должен выступить как дополнительная подсказка. Например, поле ввода пароля серое, потому что в настоящее время оно недоступно для клика и ввода данных. Чтобы разблокировать эти аффордансы, вам необходимо нажать кнопку «Change».

Поэтому не рекомендуется выбирать подобное оформление для кнопок и полей, которые всегда доступны для взаимодействия. Каким бы новаторским ни было дизайнерское решение, пересилить привычку ему вряд ли удастся.

Аффорданс в контексте. При использовании аффордансов важно не допускать двоякого толкования и учитывать опыт целевой аудитории. Конечно, подсказок, распознающихя почти моментально, уже достаточно много, но это не значит, что они знакомы абсолютно всем. И нужно учитывать, как тот или иной символ воспринимается на конкретном сайте.

Например, привычный «конверт», символизирующий электронную почту, может быть как визуальным элементом, привлекающим внимание к форме подписки на рассылку, так и ссылкой для перехода к отправке сообщения сотрудникам компании. Поэтому нужно проверять, правильно ли посетители сайта трактуют подобные элементы.

Но именно контекст позволяет использовать символические аффордансы нетипичным образом, при условии, что у посетителей сайта формируется правильный ассоциативный ряд. Если же аудитория консервативна, лучше выбирать привычные элементы, интуитивно понятные каждому, и не экспериментировать.

Способы передачи аффорданса:

- повторение конфигурации объектов конфигурацией элементов управления (этот способ работает хорошо в реальном мире, но не очень хорошо на экране, поскольку предпочтительней непосредственное манипулирование);
- видимая принадлежность управляющих элементов объекту;
- визуальное совпадение аффордансов экранных объектов с такими же аффордансами объектов реального мира (кнопка в реальном мире предлагает пользователю нажать на нее, псевдотрехмерная кнопка предлагает нажать на нее по аналогии);
- изменение свойств объекта при подведении к нему курсора (бледный аналог тактильного исследования).

Стандарт. Наконец, остался последний, самый мощный, но зато и самый ненадежный способ обучения, а именно стандарт.

Если что-либо нельзя 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 (недаром красный цвет мы называем теплым, а синий – холодным), но главным образом здесь работает привычка.

Основные программы пакета Microsoft Office – Word и Excel – разрабатывались с нуля программистами компаний. Другие же были куплены на стороне: FrontPage, например, у Vermeer, или Visio – у Visio. Что у этих программ общего? Дизайн обеих создавался с самого начала так, чтобы они выглядели и работали как приложения Microsoft Office.

Решение имитировать дизайн приложений Microsoft Office было принято не просто ради того, чтобы угодить Microsoft. И даже не для того, чтобы впоследствии продать свою программу гиганту. На самом деле, создатель FrontPage Чарльз Фергюссон не скрывает своей неприязни к Microsoft, более того, он неоднократно призывал Департамент юстиции США принять хоть

какие-нибудь меры по отношению к «Редмонским бестиям» (до тех пор, пока не продал им свою компанию). Vermeer и Visio скопировали пользовательский интерфейс Microsoft Office, должно быть, просто оттого, что это было выгодно: быстрее и проще, чем изобретать велосипед.

Когда менеджер отдела программирования Microsoft Майк Матье загрузил FrontPage с веб-страницы Vermeer, оказалось, что программа работает во многом так же, как и Word. Поскольку она работала в соответствии с его ожиданиями, пользоваться ею было просто. И эта простота в использовании программы в одночасье произвела самое благоприятное впечатление на господина Матье.

Дизайн элементов управления, выдержаный в едином ключе для различных программ, помогает пользователю обучиться работать с новой программой.

Однако стандарт – штука мощная, но зато и ненадежная. Он очень хорошо работает, если популярен, в противном случае – не работает вовсе.

Популярность стандарта может быть достигнута двумя способами: во-первых, он может быть во всех системах, во-вторых, он может быть популярен внутри отдельной системы.

Например, стандарт интерфейса MS Windows популярен почти во всех программах для Windows, именно поэтому его нужно придерживаться. С другой стороны, этот стандарт оставляет неопределенным очень многое (никто да не обнимет необъятного), и это многое в разных системах трактуется по-разному.

Последовательность в реализации интерфейса есть первое условие качества результата.

Обучающие материалы. Количество подсистем справки, нужных для того, чтобы пользователь научился пользоваться системой, довольно невелико, так что все их можно легко разобрать. Под «подсистема справки» мы будем иметь в виду часть справочной системы, которая выполняет сугубо определенные функции и требует сугубо определенных методов представления.

Базовая справка объясняет пользователю сущность и назначение системы. Обычно должна сработать только один раз. Как правило, не требуется для ПО, зато почти всегда требуется для сайтов.

Обзорная справка рекламирует пользователю функции системы. Также обычно срабатывает один раз. Нужна и ПО, и сай-

там. Поскольку у зрелых систем функциональность обычно очень велика, невозможно добиться того, чтобы пользователи запоминали ее за один раз. В этом случае оптимальным вариантом является слежение за действиями пользователя и показ коротких реклам типа «*A 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 нельзя ожидать знания предметной области, необходимо снабжать их этим знанием на ходу.

Справка предметной области является самой важной подсистемой справки. Но без знания предметной области пользователь никогда не сможет использовать систему правильно и эффективно.

Процедурная справка отвечает на вопрос «*Как это сделать?*». В идеале она должна быть максимально возможно доступна, поскольку если пользователь не найдет нужную информацию быстро, он перестанет искать и так и не научится пользоваться функцией (возможно, никогда). Разработчики чаще всего не привязывают темы справки к интерфейсу: когда пользователям непонятно, как выполнить нужное им действие, им приходится искать в справочной системе нужную тему. Это неправильно, тем более что технических проблем в этом нет.

Контекстная справка отвечает на вопросы «*Что это делает?*» 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, она не может быть вынесена из интерфейса. В целом

это самая непроблематичная для разработчиков система справки, так что в этом пособии разбираться она не будет.

Сpirальность. Поскольку пользователи обращаются к справочной системе при возникновении проблем, можно смело сказать, что применение справочной системы всегда воспринимается негативно. Поэтому следует сокращать объем справочной системы, чтобы сократить длительность неудовольствия. Однако при малом объеме справочной системы возрастает риск того, что пользователи не найдут в ней ответы на свои вопросы.

Эффективный метод решения данной проблемы: так называемые *спиральные тексты*. Идея заключается в следующем. При возникновении вопроса пользователь получает только чрезвычайно сжатый, но ограниченный ответ (1–3 предложения). Если ответ достаточен, пользователь волен 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, но и более объемный ответ. Если и тот недостаточен (что случается, разумеется, весьма редко), пользователь может обратиться к еще более подробному ответу.

Таким образом, при использовании указанного 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. Учитывая тот факт, что разработка спирали в справке непроблематична, рекомендуется делать ее во всех случаях.

Субъективное удовлетворение пользователей

Какие же факторы влияют на субъективное удовлетворение?

Это субъективное ощущение эстетики, времени работы, психологического напряжения, собственной глупости, самовыражения.

Субъективное ощущение эстетики. Все знают, что значительно легче и приятнее пользоваться эстетически привлекательными объектами. Это наблюдение породило весь промышленный дизайн, включая дизайн одежды, интерьеров и т. д.

В то же время в дизайне интерфейсов это наблюдение до сих пор как следует не утвердились: бои между туристами (интерфейс

должен быть, прежде всего, работоспособным) и маньеристами (красота – это страшная сила) никоим образом не затихают.

В то же время «срединный путь» до сих пор не найден, интерфейсы, равно удобные и эстетически привлекательные, до сих пор существуют в единичных экземплярах. Эстетическая привлекательность не измеряет красоту приложения и не характеризует его стиль; скорее, она представляет, насколько хорошо внешний вид и поведение приложения характеризуют его функции.

Программисты заботятся о функциональности приложения, однако внешний вид его может влиять на подсознательное поведение пользователей при работе с приложением. Поэтому в приложениях, которые помогают пользователям выполнить серьезную задачу, можно поставить акцент на задаче, сохраняя декоративные элементы тонкими и ненавязчивыми, и управлять ими с помощью стандартных средств управления предсказуемого поведения. Эти приложения посыпают четкие, унифицированные сообщения о своей цели и своей идентичности, которая помогает людям использовать их.

Если приложение, решая серьезные задачи, выглядит навязчивым и легкомысленным, то пользователи могут подвергнуть сомнению его надежность. С другой стороны, в приложениях, таких как игра, пользователи ожидают увидеть красочный внешний вид, который обещает получение удовольствия и радости от игры. И это вызывает желание открыть такое приложение. Люди не ожидают серьезного внешнего вида от игры. Внешний вид игры и поведение должны соответствовать ее назначению.

Итак, какие их принципы могут быть использованы в дизайне интерфейса? Для того чтобы интерфейс был эстетически привлекательным, необходимо, чтобы он был незаметен в процессе его использования. Во что бы то ни стало добивайтесь того, чтобы интерфейс был неощущаемый. Для этого:

- избегайте развязности в изображении. Лучше, чтобы интерфейс был скромнее;
- избегайте ярких цветов. Существует очень немного цветов, обладающих и яркостью, и мягкостью (т. е. «не бьющих по глазам»). На экране их значительно меньше, поскольку в жизни такие цвета обычно моделируются как собственно цветом, так и текстурой, с чем на экране есть проблемы;
- избегайте острых углов в изображении;
- старайтесь сделать изображение максимально более легким и воздушным;

- добивайтесь контраста не сменой насыщенности элементов, а расположением пустот;
- старайтесь минимизировать количество констант (тем более, что двух констант обычно хватает на все). Разумеется, единожды примененных закономерностей необходимо придерживаться во всей системе.

Стремитесь не столько к красоте интерфейса, сколько к его элегантности.

Красота – понятие относительное. Для одних красивыми могут считаться только живописные закаты, для других – картины художника Кустодиева, а для третьих – комбинация вареных сосисок, зеленого горошка и запотевшей бутылки пива. Это делает красоту вещью не слишком универсальной. Хуже того, любая красота со временем надоедает и перестает восприниматься.

Лучший интерфейс должен соблюдать баланс между ясностью целей и простотой использования. Пытаясь создать оригинальный продукт, проектировщики часто злоупотребляют «украшательством». Например, представьте себе приложение калькулятор (рис. 3.5), которое использует сложный, художественный стиль и образное расположение, чтобы отобразить знакомые элементы калькулятора. Пользователи, конечно, могут понять, как нажать кнопки и прочитать результат, но для людей, которым просто необходим калькулятор для выполнения своей работы, новизна интерфейса стирается быстро – и красивый пользовательский интерфейс становится помехой.



Рис. 3.5. Приложение калькулятор

В отличие от этого, GarageBand мог бы помочь людям создавать музыку без отображения красивых, реалистичных инструментов (рис. 3.6), но это сделало бы приложение менее интуитивным и менее приятным в использовании.

В GarageBand пользовательский интерфейс не только показывает людям, как использовать приложение, но и делает основную задачу (создание музыки) проще. Именно поэтому в интерфейсах обычно нет места красоте.

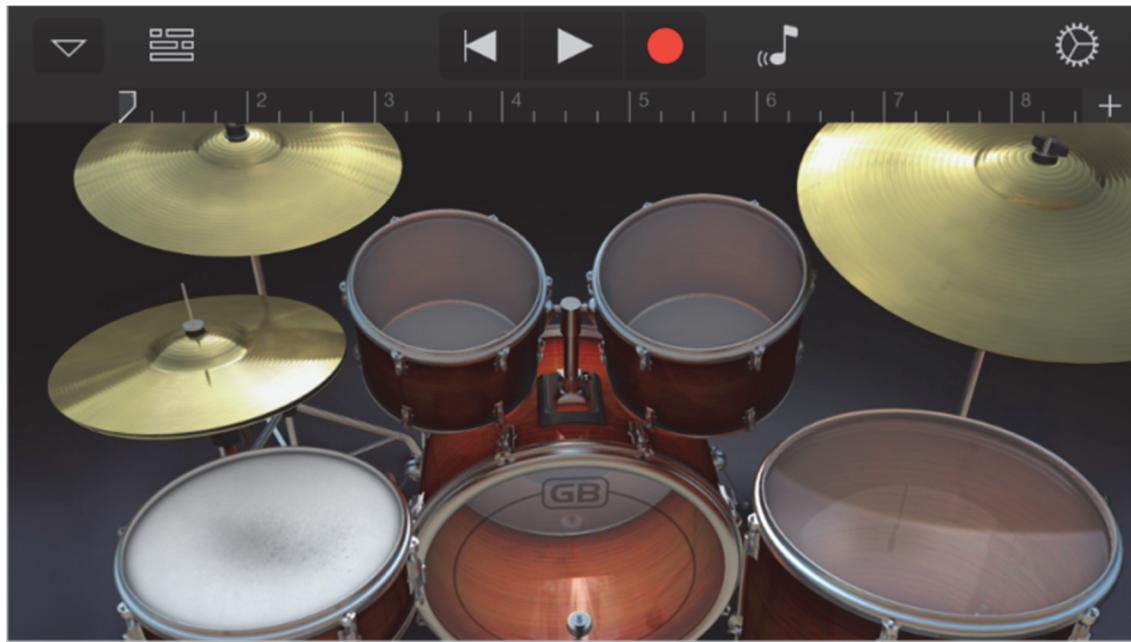


Рис. 3.6. Интерфейс GarageBand

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 древнеримские здания воспринимаются нами красивыми, несмотря на абсолютную разницу культур и времени). В-четвертых, в производстве они гораздо удобнее красоты, поскольку сравнительно легко ставятся на поток.

Итак, как надо действовать, чтобы добиться элегантности:

- старайтесь сделать интерфейс максимально насыщенным визуальными закономерностями (рис. 3.7). Есть универсальное правило – чем больше закономерностей, тем больше гармонии. Даже самые незначительные закономерности все равно воспринимаются. Под закономерностью понимают любое методически выделяемое соответствие свойств у разных объектов, например, высота кнопок может быть равна удвоенному значению полей диалогового окна;
- всемерно старайтесь использовать модульные сетки, т. е. привязывайте все объекты к линиям (лучше узлам) воображаемой сетки, которую выдерживайте во всем интерфейсе;
- старайтесь привязывать все размеры и координаты к золотому сечению ($0,618 \times 0,382$).

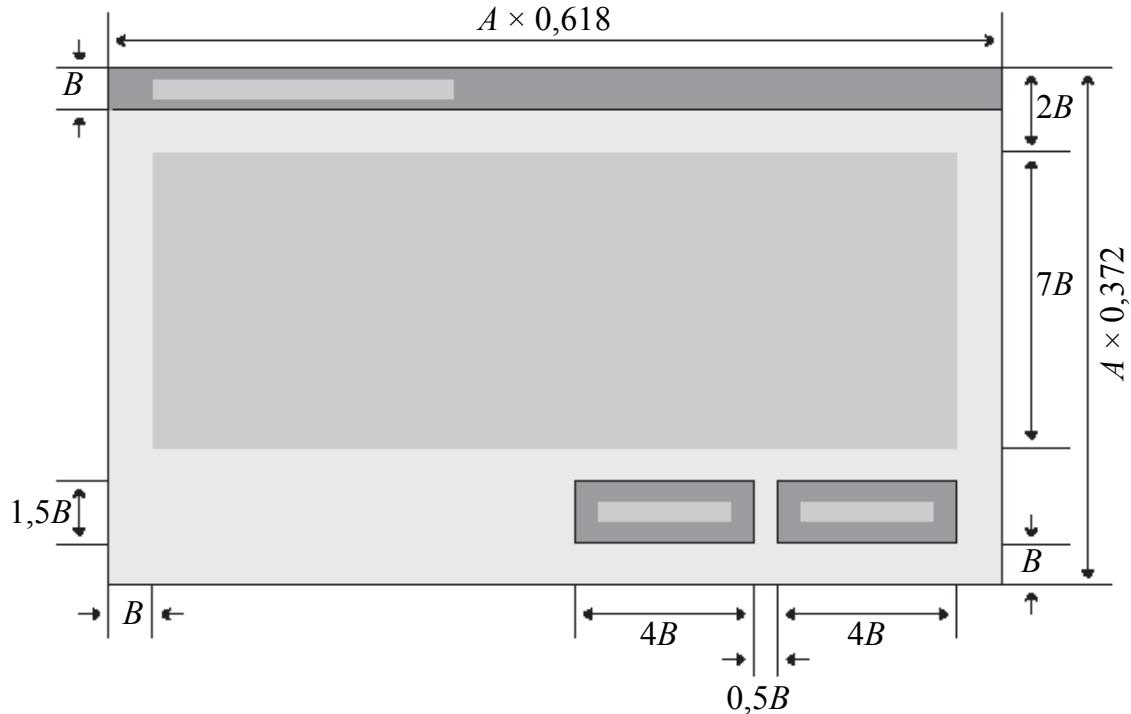


Рис. 3.7. Пример использования правильных размеров

Субъективное ощущение времени. Любой человек хочет работать быстро. Если работу можно выполнить быстро, у человека возникает приятное ощущение. Однако субъективное ощущение времени зачастую сильно отличается от объективного, так что методы повышения реальной скорости работы помогают не всегда.

Воспринимаемая продолжительность действий напрямую зависит от уровня активности пользователя. Известно, что при бездействии (скуке) время течет невыносимо медленно.

Таким образом, субъективную скорость работы можно повысить двумя способами:

- заполнение пауз между событиями. Есть данные о том, что если в периоды ожидания реакции системы пользователям показывается индикатор степени выполнения, субъективная продолжительность паузы существенно снижается. Судя по всему, чем больше информации предъявляется пользователям в паузах, тем меньше субъективное время. С другой стороны, эта информация может вызвать стресс в кратковременной памяти, так что пользоваться этим методом надо осторожно;

- разделение крупных действий пользователей на более мелкие. При этом количество работы увеличивается, но зато субъек-

тивная длительность снижается. Плох этот метод тем, что увеличивает усталость.

С другой стороны, повышение объективной скорости работы зачастую способно повысить и субъективную скорость.

Субъективное ощущение психологического напряжения.

Пользователь знает, что во время работы может что-либо испортить. Он может отформатировать жесткий диск, может стереть или испортить нужный файл. И это вызывает у него психологическое напряжение, иначе говоря – стресс.

Пользователь, знающий, что он не может совершить ошибку, испытывает радость и умиротворение. Чтобы добиться этого, необходимо иметь возможность:

- отменять свои предыдущие действия, без ограничения количества уровней отмены и типа отменяемых действий. Задача эта непростая, но зато результат крайне существенен;

- прятать опасные для пользователя места интерфейса. Проблема заключается в том, что при этом логично прятать все функции, изменяющие данные, например, банальная функция автоматической замены может мгновенно уничтожить текст документа.

Другим фактором, существенно влияющим на субъективное удовлетворение пользователей, является чувство контроля над системой. Таким образом, пользователей нужно всемерно снабжать ощущением, 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 всеми силами внушать пользователям мысль, что только явно выраженное действие приводит к ответному действию системы (это, в частности, главный аргумент против ролловеров – пользователь еще ничего не нажал, а уже что-то произошло).

Субъективное ощущение собственной глупости. Ни один пользователь не может долго и продуктивно работать с системой, которая говорит, что он глуп. Тем не менее такие «скандальные» системы являются нормой. Виной тому сообщения об ошибках. Почему сообщения об ошибках плохи?

Дело в том, что большинство сообщений об ошибках в действительности не являются таковыми. На самом деле они показывают пользователю, что система, которой он пользуется: недостаточно

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

Пользователи ненавидят сообщения об ошибках. 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. Любой сообщение об ошибке говорит пользователю, что он дурак. Именно так пользователи воспринимают любые сообщения об ошибках.

Таким образом, почти все сообщения об ошибках должны быть удалены. Системы изначально надо проектировать так, чтобы в них отсутствовала необходимость в таких сообщениях.

Каким должно быть сообщение об ошибке? Многие пренебрегают подбором формулировок для сообщений об ошибках, считая их чем-то вторичным. Разработчики обычно знают, как избегать неправильных действий в своих продуктах, и потому редко сталкиваются с подобными сообщениями.

Но настоящие пользователи не могут этим похвастаться. Если пользователь плохо понимает, что ему делать после возникновения сообщения об ошибке, то он может вообще забросить ваш продукт. Поэтому текст сообщения может быть вольным, но, как минимум, помогать пользователям быстро разобраться с причиной и избегать сбоев в дальнейшем.

Старайтесь избегать специализированных терминов, пишите сообщения общедоступным, понятным языком.

С помощью сообщений об ошибках можно сразу выделить конкретные поля ввода, на которые нужно обратить внимание пользователя. Иногда сообщения об ошибках такие бестолковые, что не помогают пользователю решить проблему. Всегда старайтесь сразу визуально подсказать, где именно пользователь должен решить возникшее затруднение. На рис. 3.8 представлена форма регистрации Basecamp, в которой не только подсвечиваются нужные поля, но и дается текстовая подсказка по заполнению.

Вероятно, вы считаете, что если в форме всего два-три поля, то все настолько очевидно, что пользователю помогать не нужно. Однако даже в этом случае рекомендуется фокусировать внимание людей на проблемных местах, это куда полезнее, чем просто «вываливать» им список ошибок, как это делается в Amazon (рис. 3.9).



Your full name

Enter your full name

Company or organization

Email

Enter a valid email

Password

Enter a password

Start my two month free trial

Рис. 3.8. Форма регистрации Basecamp

[Your Account](#) | [Help](#)

amazon

✖ There was a problem with your request

Missing e-mail address. Please correct and try again.
 Please enter your password.
 You must provide a name.
 Please check that your passwords match and try again.

Registration

New to Amazon.com? Register Below.

My name is:

My e-mail address is:

Type it again:

My mobile phone number is: (Optional)

[Learn more](#)

Protect your information with a password

This will be your only Amazon.com password.

Enter a new password:

Type it again:

Create account

By creating an account, you agree to Amazon.com's [Conditions of Use](#) and [Privacy Notice](#).

[Conditions of Use](#) [Privacy Notice](#)
 © 1996-2014, Amazon.com, Inc. or its affiliates

Рис. 3.9. Сообщения об ошибках в Amazon

Теперь можно рассказать, каким должно быть сообщение об ошибке, тем более, что ничего сложного в создании идеального сообщения нет. Напротив, все очень просто.

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?

Самовыражение. Страсть к самовыражению является одной из самых сильных черт человеческой натуры. В большинстве случаев люди самовыражаются через вещи (двигают мебель и покупают модную одежду), когда нет вещей – насвистывают или поют изящные песни «лишенным приятности голосом».

Ни один человек не может существовать сколько-нибудь продолжительное время в обстановке, не допускающей самовыражения.

Неудивительно, что пользователи хотят выразить себя и в программах, которыми они пользуются. Соответственно, возможность настроить систему под свои нужды является мощной причиной субъективного удовлетворения. Проблема здесь в том, что это очень дорого стоит. Времени на самовыражение уходит крайне много. По грубым оценкам, на такой процесс самовыражения в среднем уходит около 45 минут в неделю. Получается, что организация всего с сотней работников теряет на этом еженедельно 75 часов, что почти равняется потере недельной работы двух человек.

С другой стороны, настроенный под свои нужды интерфейс, судя по всему, снижает усталость работников и повышает их рабочее настроение. Таким образом, в продуктах, продаваемых пользователям напрямую, возможность настройки под конкретного пользователя обязательна.

Во всех остальных случаях, судя по всему, 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.

Количественный анализ интерфейса

Количественные методы помогают свести спорные вопросы в оценке качества интерфейса к простым вычислениям. Они помогают понять важнейшие аспекты взаимодействия человека с ма-

шиной. Существует множество методов количественного анализа элементов интерфейса. Наиболее известные – это модель GOMS, разработанная Кардом, Мораном и Ньюэллом, а также критерий эффективности Раскина, закон Хика и закон Фиттса.

Закон Фиттса (Fitts's Law) гласит, что «время, требуемое для позиционирования на какой-либо элемент, есть функция от расстояния до этого элемента и от его размера». Ну то есть применительно к интерфесам – чем больше и ближе объект к текущему положению мышки, тем быстрее человек сможет на нем щелкнуть.

Еще в 1954 г. Пол Фиттс (Paul Fitts) сформулировал правило, ставшее известным как Закон Фиттса.

Время достижения цели прямо пропорционально дистанции до цели и обратно пропорционально размеру цели:

$$T_{\text{дост. цели}} = a + b \log_2 \left(\frac{D}{S} + 1 \right) \dots,$$

где a и b устанавливаются опытным путем по параметрам производительности человека (для практического использования можно принять: $a = 50$, $b = 150$); D – дистанция от курсора до цели; S – размер цели по направлению движения курсора.

Закон Хика. Закон Фиттса обычно сопровождается законом Хика, утверждающим, что время реакции при выборе из некоторого числа альтернативных сигналов зависит от их числа. Словами логики – чем больше пунктов в выборе, тем дольше люди будут думать, что выбрать. Поэтому везде слышны рекомендации делать не больше 5–7 пунктов в меню, использовать группировки, выделения и т. п. Закон Хика позволяет количественно определить наблюдение, заключающееся в том, что чем больше количество вариантов заданного типа вы предоставите, тем больше времени требуется на выбор.

Сформулированная еще в середине XIX в. зависимость была экспериментально подтверждена только в 1952 г. психологами Уильямом Хиком (Великобритания) и Реем Хайманом (США) – формула Хика – Хаймана. Ученые вывели формулу, которая описывает логарифмическую зависимость между временем реакции и количеством объектов, из которых нужно выбрать:

$$T = a + b \log_2 (n + 1),$$

где T – общее время реакции; a и b – константы, которые описывают индивидуальные особенности восприятия, такие как задержка

перед выполнением задания и индивидуальный коэффициент скорости принятия решения; n – количество равнозначных альтернативных вариантов, из которых нужно выбрать.

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

При проектировании интерфейсов закон Хика помогает определить оптимальное количество объектов в однородном массиве – например, в меню. Обычно он применяется в связке с законом Фиттса, который помогает определить оптимальный с позиции скорости реакции размер элемента.

При использовании любых положительных и ненулевых значений a и b из закона Хика следует, что предоставление сразу нескольких вариантов одновременно обычно является более эффективным, чем организация тех же вариантов в иерархические группы. Выбор из одного меню, состоящего из 8 элементов, производится быстрее, чем из двух меню, состоящих из 4 элементов каждое.

Закон Хика также тесно связан с другими принципами восприятия и психологическими особенностями принятия решений. Его можно одинаково эффективно рассматривать в контексте теории близости и правила « 7 ± 2 », а также других моделей поведения пользователя на сайте. Вне интернет-среды принципы закона Хика реализованы практически в любом интерфейсе, взаимодействующем с пользователем: начиная от панели управления микроволновой и заканчивая расположением и количеством кнопок на пульте для телевизора.

Особенности комплексного применения законов Хика и Фиттса в UX. Законы описывают действия, которые обычно следуют одно за другим. Вначале нам нужно определиться с выбором (закон Хика), а затем – попасть в нужный элемент (закон Фиттса).

Таким образом, общее время можно вычислить как сумму значений двух формул. В контексте UX это значит следующее.

Одно длинное меню (или расположение однородных элементов в одном блоке) удобнее для пользователя, чем два или несколько отдельных.

При проектировании интерфейса нужно учитывать оба закона и стараться оптимизировать как размеры и положение блоков, так и количество элементов в каждом блоке. Ориентироваться на законы также можно при составлении и оптимизации профилей задач. Особенно показателен закон Хика при анализе процесса локализации и заполнения полей форм.

Закон Хика менее известен, чем закон Фиттса. Однако он его прекрасно дополняет и помогает проектировать взаимодействие с пользователем более осознанно и эффективно.

Выводы и рекомендации:

- чем больше объектов, тем больше времени нужно пользователю, чтобы выбрать из них нужный;
- зависимость между временем реакции и количеством альтернатив выбора описывается логарифмической функцией;
- закон Хика позволяет рассчитать оптимальное количество объектов в блоке;
- применение закона Хика в связке с законом Фиттса позволяет более точно спрогнозировать время реакции пользователя в ходе взаимодействия с интерфейсом.

Модель GOMS. Один из лучших подходов к количественному анализу моделей интерфейсов – классическая модель *GOMS* (*the model of goals, objects, methods and selection rules* – модель целей, объектов, методов и выбор правил).

Моделирование GOMS позволяет предсказать, сколько времени потребуется опытному пользователю на выполнение конкретной операции при использовании данной модели интерфейса.

Эта модель основана на оценке скорости печати. Время, требуемое для выполнения какой-то задачи системой пользователь – компьютер, является суммой всех временных интервалов, которые потребовались системе на выполнение элементарных жестов, составляющих данную задачу. Лабораторным путем установлены стандартные средние интервалы для некоторых жестов, выполняемых различными пользователями: $K = 0,2$ с – нажатие клавиши;

$P = 1,1$ с – указание на какую-то позицию на экране монитора; $H = 0,4$ с – перемещение руки с клавиатуры на «мышь» или обратно; $M = 1,35$ с – ментальная подготовка – мысленный выбор пользователем своего следующего элементарного действия; R – ответ (время ожидания ответа компьютера).

Расчет по модели GOMS. Основные правила, позволяющие определить, в какие моменты будут проходить ментальные операции, представлены в таблице.

Правила модели GOMS

Правила	Ментальные операции
Правило 0 Начальная расстановка операторов M	Оператор M устанавливается перед всеми операторами K (нажатие клавиши) и P , предназначенными для выбора команд. Если P указывает на аргументы этих команд, оператор M не ставится
Правило 1 Удаление ожидаемых операторов M	Если оператор, следующий за оператором M , ожидаемый с точки зрения оператора, предшествующего M , то этот оператор M может быть удален и последовательность PMK превращается в PK
Правило 2 Удаление операторов M внутри когнитивных единиц	Если строка типа $MKMKMK\dots$ принадлежит когнитивной единице, то следует удалить все операторы M , кроме первого. Когнитивной единицей является непрерывная последовательность вводимых символов, которые могут образовывать название команды или аргумент
Правило 3 Удаление M перед последовательными разделителями	Если оператор K означает лишний разделитель, стоящий в конце когнитивной единицы (например, разделитель команды, следующий сразу за разделителем аргумента этой команды), то следует удалить оператор M , стоящий перед ним
Правило 4 Удаление операторов M , которые являются прерывателями команд	Если оператор K является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор M , стоящий перед ним. Но если оператор K является разделителем для строки аргументов или любой другой изменяемой строки, то M следует сохранить перед ним
Правило 5 Удаление перекрывающихся операторов M	Любую часть операторов M , которая перекрывает оператор R , означающий задержку, связанную с ожиданием ответа компьютера, учитывать не следует

Информационная производительность интерфейса E определяется как отношение минимального количества информации, необходимого для выполнения задачи, к количеству информации, которое должен ввести пользователь.

Также как и в отношении физической производительности, параметр E может изменяться в пределах от 0 до 1 .

Если никакой работы для выполнения задачи не требуется или работа просто не производится, то производительность составляет 1 (чтобы избежать деления на 0).

Производительность E может равняться и 0 в случаях, когда пользователь должен ввести информацию, которая бесполезна для выполнения задачи.

В параметре E учитывается только информация, необходимая для задачи, и информация, вводимая пользователем.

Глава 4

ОСОБЕННОСТИ ВОСПРИЯТИЯ ЧЕЛОВЕКОМ ИНФОРМАЦИИ

Любой интерфейс должен учитывать особенности человеческой психологии, физические ограничения человека и его сознания во время работы, способность человека отвлекаться во время работы, неточность его движений, восприятие того что он видит, физическое напряжение во время работы.

Разные люди имеют разные общечеловеческие факторы. Человек имеет пять видов чувств, берущих участие в освоении внешнего мира: зрение, слух, вкус, обоняние (нюх), осязание (ощупь).

Восприятие

Процесс приема и усвоения информации, получаемой при помощи воздействия предметов и явлений на органы чувств человека, называется *восприятие* или *перцепция* (от лат. *perceptio*). Этот процесс построен на взаимосвязи психических и биологических функций организма.

Общеизвестно, что 90% информации об окружающем мире человеку дает зрение, 8% приходится на органы слуха, обоняния и того меньше. Зрение является главным каналом восприятия информации.

Во время прогулки или на экскурсии наши глаза воспринимают информацию и передают ее в мозг, который обрабатывает данные и представляет реалистичную картину того, что нас окружает. Но наши глаза не работают так, как фотоаппарат, объективно фиксирующий мир. Они действуют совместно с мозгом, который определенным образом «истолковывает» видимый мир. Изображение, попадая в мозг, изменяется и интерпретируется. Наш мозг является хранилищем, в котором находятся миллионы «образцов» объектов, и когда мы видим объект, мы сопоставляем его с содержащимися в памяти «образцами».

Представление не несет ничего нового, так как объект восприятия и представления один и тот же. Более того, при восприятии

объект обрисован точнее, нежели чем при представлении. Но преимущества представления есть, и их нужно искать в ином.

Восприятие – это актуальные переживания, т. е. те, которые действуют на человека именно в этот момент. Следовательно, восприятие происходит до тех пор, пока какое-либо воздействие продолжается. Другое дело **представление**. Оно воспроизводит то, что действовало на человека когда-то, т. е. было отражено в виде восприятия.

Отличительный момент в том, что в представлении мы можем пережить то, что совсем не связано с актуальной ситуацией (пример на рис. 4.1). Стало быть, оно позволяет заново пережить ощущения, которые были пережиты в иное время.



Рис. 4.1. Восприятие и представление

На успешное восприятие визуальной информации влияют следующие факторы:

• **размер объекта** (большие объекты воспринимаются острее). Если два аналогичных объекта разного размера показать одновременно, то меньший объект будет восприниматься как расположенный дальше (от наблюдателя), чем более крупный объект. Размер знакомых объектов также можно использовать для указания размеров и глубины незнакомых объектов. Для печатного текста

наиболее оптимальным считается шрифт в 9–12 пунктов. Крупный шрифт используется для аудитории, состоящей из лиц пожилого возраста;

- **цвет объекта** (в некоторых пропорциях цвет способен создавать конкретное настроение и привлекать к себе внимание; и, соответственно, наоборот);
- **интенсивность объекта** (в эмоциональном или физическом значении);
- **контрастность объекта** (выделение объекта из окружающей среды). Использование темного текста на светлом фоне и наоборот. Изображение оптимально в том случае, когда уровень контраста между текстом и фоном составляет 70%.

Когнетика

Человеческий мозг обладает некоторыми особенностями **восприятия** окружающей нас действительности, которые приносят разнообразность нашему мышлению и развитию. Превосходя по всем параметрам и возможностям любой компьютер, наш мозг отличается от машины тем, что умеет сконцентрироваться на определенном объекте и вычитать из своего внимания не интересующие.

Изучение прикладной сферы наших ментальных способностей называется когнитивным проектированием или **когнетикой** (от слова *cognate* – родственный, сходный).

Когнетика учитывает статистическую природу различий между людьми. Она занимается изучением прикладной сферы наших ментальных способностей. Эта наука исследует способности человеческого мозга, его ограничения. Но как это связано с разработкой интерфейсов?

С точки зрения когнетики, существуют **бессознательные** и **сознательные** процессы, происходящие в нашем мозге.

Бессознательными считаются ментальные процессы, которые мы не осознаем в тот момент, когда они происходят. Такие процессы возникают благодаря повторению в привычных ситуациях и имеют огромную производительность.

Сознательные процессы происходят в новых ситуациях или ситуациях, представляющих угрозу. Они, как правило, менее производительны и требуют от человека принятия решений и усилий воли.

Свойства когнитивного сознательного и когнитивного бессознательного сведены в таблицу.

Свойства когнитивного сознательного и когнитивного бессознательного

Свойство	Сознательное	Бессознательное
Инициируется	Чем-то новым, нестандартными ситуациями, опасностью	Повторением, ожидаемыми событиями, безопасностью
Используется	В новых обстоятельствах	В привычных ситуациях
Решает задачи	Принятия решений	Работы с неветвящимися задачами
Принимает	Логические утверждения	Логические или противоречивые утверждения
Функционирует	Последовательно	Одновременно
Управляется	Волей	Привычными действиями
Производительность	Небольшая	Огромная
Период функционирования	Десятки секунд	Десятилетия (всю жизнь)

Интерфейс должен снижать когнитивную нагрузку на пользователя. Она должна быть минимальной. Здесь, прежде всего, следует учитывать нагрузку на основные психические процессы: *память, внимание, воображение*.

Когнитивная нагрузка относится к общему количеству информации, с которой может справиться память человека. Когнитивная перегрузка происходит, когда память получает больше информации, чем она может свободно обработать, что приводит к проблемам при принятии решений.

Привычки

Исходя из того, что бессознательные процессы наиболее производительны, интерфейс должен способствовать формированию привычки – по мере повторения, выполнение того или иного действия должно становиться привычным, чтобы его можно было выполнять автоматически, не задумываясь.

Интерфейс призван формировать у пользователя привычки. Привычки формируются независимо от того, думает ли об этом разработчик. Если одно и то же действие повторяется несколько раз подряд, оно становится привычным. Привычки могут возникать на

умственном и на физическом уровне, постепенно переводя сознательные действия в бессознательные. Способность формировать привычки, если ее использовать правильно, может приносить положительные результаты. Например, унифицированный интерфейс большинства приложений MS Windows позволяет, освоив работу с одним приложением, например MS Word, вполне успешно работать с MS Excel.

Однако, для того чтобы интерфейс формировал привычки, он должен соответствовать следующим требованиям:

- быть достаточно простым. Примитивно устроены современные кофемолки. Нажатием одной кнопки прибор и включается, и выключается. А в стиральной машине нельзя использовать метод «одного выключателя», как в кофемолке, но в то же время многие домохозяйки утверждают, что научить обращаться со стиральной машиной можно даже обезьяну;
- достаточно часто использоваться для формирования привычек. Многократно повторяемые действия ведут к автоматизму и созданию привычки, а перерыв в совершении этих действий ведет к некоторой потере контроля над ситуацией. Недаром пользователи, пусть даже опытные, по вопросам настройки компьютера обращаются к системным администраторам. Если не заниматься подобной настройкой очень часто, то некоторые тонкости забываются.

Привычки высвобождают внимание. Человек с высвобожденным вниманием устает меньше и меньше напрягается. Последствия привычек – так называемые предопределенные действия. Свойство интерфейса формировать привычки может приносить весьма положительные результаты и упрощать работу.

Фокус и локус внимания

Внимание – это то, что позволяет нам обрабатывать информацию об окружающем нас мире. Мы осознаем вещи только тогда, когда внимаем им, обращаем на них свое внимание. Внимание часто сравнивают с фонариком, который мы направляем на объекты для того, чтобы выделить их из бесконечного множества других предметов. После того как мы сфокусировали луч от фонарика на объекте, мы тут же начинаем осознавать и интерпретировать то, что предстало нашему взору.

Фокус внимания человека применительно к компьютерным системам – некоторое место на экране, куда направлен его взгляд и где он сознательно сосредоточен. Фокус внимания может быть только один.

Локус внимания – это некоторое место или область, на которое может быть сосредоточено ваше внимание. В отличие от фокуса, часто обозначающего не только место, но и действие (сфокусировать ваше внимание), локус обозначает только место и переводится с латинского как место положения или область.

Видимый предмет не всегда может быть локусом вашего внимания. Локус внимания может быть только один. Информация, ставшая локусом внимания, перемещается в кратковременную память, где хранится в течение 10 секунд.

С локусом внимания связано как минимум две особенности человеческого восприятия. При смене локуса теряется связанная с ним «оперативная» информация, которая содержится в кратковременной памяти. Соответственно, при возвращении к прежнему локусу эту информацию необходимо каким-то образом восстанавливать. Например, при периодическом переключении внимания с рабочей области документа на уведомления об ошибках эффективность работы снижается. При пристальном сосредоточении внимания все события вне локуса могут игнорироваться или просто оставаться незамеченными.

Мы можем сами управлять своим вниманием. Такой вид внимания называется **произвольным (намеренным)**. Например, когда мы находимся на лекции, мы заставляем себя слушать речь лектора и смотреть на доску или экран; либо когда мы ищем какую-то книжку в библиотеке, мы сознательно обращаем внимание на названия книг, написанные на корешках.

Сигналы из внешнего мира, другие объекты, которые находятся в поле нашего восприятия, также могут управлять нашим вниманием. Например, громкий хлопок вынуждает нас повернуть голову вслед предполагаемому источнику звука, а мигающая иконка в углу экрана моментально приковывает наш взгляд. Этот вид внимания – **спонтанный**, и он управляется внешними стимулами. Психологи называют такой вид внимания **непроизвольным**.

Внимание очень сильно влияет на производительность труда, особенно того труда, который связан со взаимодействием

человека и компьютера. Интерфейс ПО или веб-сайта **должен управлять вниманием пользователя**, помогая тем самым воспринимать ту информацию, которая является значимой «здесь и сейчас».

Существует три вида произвольного внимания:

1. **Избирательное внимание.** Этот вид внимания иногда называют туннельным вниманием. Оно возникает тогда, когда мы обращаем внимание на стимул или задачу так сильно, что начинаем полностью игнорировать все остальные стимулы и объекты. Программист, занятый написанием кода программы, геймер, бороздящий просторы виртуального пространства, или водитель, полностью сконцентрированный на дороге, – все они могут запросто пропустить мимо ушей вопрос, заданный им другим человеком.

При поиске в «зашумленном» интерфейсе некоторой важной информации мы также используем свое избирательное внимание.

2. **Фокусированное внимание.** Его можно назвать более эффективным избирательным вниманием, потому что в данном случае мы целенаправленно перестаем обращать наше внимание на стимулы для того, чтобы завершить задачу. Например, пользователь программы, который сознательно игнорирует уведомление о новом сообщении, мерцающем в углу экрана, для того, чтобы закончить и послать e-mail. В поле фокусированного внимания находится письмо, а остальные стимулы человек намеренно исключил из своего поля зрения. Например, если пользователь пытается сохранить в MS Word документ с именем уже существующего файла, то выводится модальное предупреждение, которое привлекает внимание пользователя и не позволяет ему отвлекаться на что-либо еще.

3. **Распределенное внимание.** Бывают такие ситуации, когда становится невозможным фокусировать свое внимание на одной задаче из-за того, что другие стимулы начинают отвлекать нас. Например, если мы вдруг услышим, как кто-то разговаривает о нас в то время, пока мы ведем беседу с другими людьми, нам станет трудно удерживать свое внимание исключительно на нашем собственном разговоре. Некоторые компьютерные задачи вынуждают пользователей фокусироваться на нескольких вещах. Это значительно понижает эффективность и продуктивность работы пользователя.

Распределенное внимание является особо критическим фактором для задач, при выполнении которых требуется особая бдительность (т. е. те задачи, где пользователь должен отслеживать изменения в интерфейсе в течение длительного времени). Примеры таких задач: контроль над воздушным трафиком, над системой безопасности, управление процессом на атомной электростанции.

Центральное и периферийное зрение

Помимо типов внимания, важно то, куда направлено визуальное внимание в каждый данный момент времени.

Существует два типа зрения:

1) **центральное зрение** – обеспечивается центральным участком сетчатки и центральной ямкой. Дает человеку возможность различать формы и мелкие детали предметов, поэтому его второе название – форменное зрение;

2) **периферийное зрение** – обеспечивает ориентацию человека в пространстве, дает возможность видеть во тьме и полутьме. Кроме объекта, на который вы смотрите, в поле зрения попадает также большое количество различных вещей. Вы видите все эти предметы нечетко, но все же видите, имеете возможность улавливать их движение и реагировать на него.

Когда пользователь применяет избирательное или сфокусированное визуальное внимание, тогда он использует центральную область сетчатки глаза (**центральное зрение**) с самой большой концентрацией фоторецепторов или, другими словами, ту область интерфейса, которая находится в пределах всего поля зрения пользователя. Эту область часто называют UFOV (*useful field of view – полезное поле зрения*). UFOV обычно находится между 1 и 4-м градусами угла зрения, и этот факт нужно учитывать в задачах, в которых информация вне этой области может быть пропущена.

Наше периферийное зрение охватывает зону приблизительно в 207 градусов. Мы используем уши, чтобы следить за звуками и определять, когда нам нужно повернуть голову, чтобы получить зону видимости в 360 градусов. У нас есть зона высокого разрешения (центральное зрение), размер которой составляет немногим меньше, чем размер листа бумаги, находящегося на расстоянии, обычно используемом для чтения.

Наше периферийное зрение использует всю сетчатку, а зона высокого разрешения – только область, которая имеет диаметр всего 0,2 мм.

Проектировщики интерфейса должны понимать принципы работы периферийного зрения и визуальных подсказок, которые можно использовать для привлечения внимания к той области интерфейса, которая находится вне UFOV. Периферийная область нашего зрения самая чувствительная к визуальным подсказкам, основанным на движении, мигании и резким изменениям в контрасте.

Визуальные подсказки

Понимание принципов и законов, по которым работает внимание, позволяет создавать эффективные пользовательские интерфейсы.

Интерфейс должен помогать пользователю фокусировать внимание на важной информации. Можно заострить внимание на текущем объекте в системе с помощью следующих визуальных подсказок.

1. *Положения объекта.* В 2006 г. исследователь Якоб Нильсен обнаружил, что мы просматриваем веб-страницы по определенной схеме. Посетители сайта сканируют информацию на странице по F-образной форме, потому что взгляд пользователя обычно скользит слева направо в направлении к нижней части страницы. Зная, что пользователи будут просматривать веб-страницу по F-образной кривой и потратят на это несколько секунд (не больше 5–8), разместите по такой схеме основные элементы целевой страницы.

Размеры, цвет и контрастность – все эти свойства могут играть роль визуальной подсказки, которая привлечет внимание пользователя к чему-то существенному.

Например, применение синего цвета ссылок на веб-страницах является оптимальным, так как этот цвет, во-первых, контрастный по отношению к цвету фона и не сливаются с основным цветом текста и, во-вторых, у пользователей есть определенный опыт восприятия веб-страниц, они привыкли использовать его каждый раз, попадая на новую страницу или сайт.

2. *Подсветки.* Обычно система подсвечивает текущий объект. То есть при наведении на них курсора их цвет меняется. Например, в браузере Internet Explorer последних версий панель инструментов представляет собой серые пиктограммы, которые становятся цветными. В web частным случаем подсветки является изменение цвета и вида гиперссылки при наведении на нее мыши.

3. *Указания.* Можно указать какую-либо область экрана или объект, если это поддерживается программой. Например, может

быть указана конкретная строка таблицы после «клика» на ней мышью. Указание – это вариант «долговременной» подсветки.

4. *Выделения*. Выделение подразумевает, что пользователь каким-либо образом обозначает для системы объект, над которым нужно произвести некоторое действие. Выделить объект можно, установив флажок в чек-боксе или нажав радио-кнопку. Можно выделить часть текста, изображение или какую-либо еще часть содержимого страницы.

5. *Активация*. Активировать – значит сделать некоторый элемент интерфейса или объект системы доступным для использования или преобразования. Например, выпадающее меню в некоторых случаях может быть использовано только после активации управляющего элемента, т. е. наведения на него курсора. Текущий раздел меню, как правило, идентифицируется при помощи подсветки.

Кроме этого, есть способы, позволяющие привлечь внимание пользователя к ссылкам или функциям, которые призваны обеспечивать цели бизнеса или самого пользователя. Платная регистрация на сайте знакомств – хороший пример функции, которая является ключевой для целей бизнеса. А ссылка «подробнее» – пример ссылки, которая помогает пользователю сориентироваться и получить полный текст статьи, т. е. решить свою задачу.

Очень важно знать, какие элементы нужно сопровождать визуальной подсказкой, когда и каким образом. Например, навигационные ссылки и пункты меню должны всегда сопровождаться визуальными подсказками, иначе посетитель может пропустить целый раздел сайта. Кнопки, которые относятся к важным действиям (например, «зарегистрироваться» или «послать»), должны не только быть похожими на кнопки, но и выделяться среди других элементов.

Кроме того, в опыте каждого пользователя есть целый набор элементов, которые он отфильтровывает, считая их незначимыми и не относящимися к делу. Пользователи уже научились игнорировать баннерную рекламу и избирательно обращать свое внимание лишь на те элементы и функции, которые кажутся им значимыми. Этот эффект получил название «баннерная слепота».

Память

На качество взаимодействия пользователя с системой существенно влияют возможности человеческой памяти. На этот фактор мы не в состоянии повлиять. У каждого конкретного человека

есть свои особенности памяти, т. е. хорошая у него память или плохая, какая память у него более развита (зрительная, слуховая, осязательная), какой объем его кратковременной памяти и т. д.

Для нас актуально знать про две подсистемы памяти, а именно про **кратковременную (КВП) и долговременную (ДВП)**.

Вся информация, воспринимаемая пользователем при работе с системой, хранится в кратковременной памяти, которая является, по сути, не отдельным свойством человеческого мозга, а некоторой составляющей долговременной памяти. Как информация попадает в КВП?

Представьте себе коробку без боковых стенок, в которой один за одним лежат кирпичи. Если с одной стороны попытаться засунуть в эту коробку еще один кирпич, то он туда, безусловно, влезет, но при этом с противоположной стороны вывалится другой, оказавшийся лишним. Таким образом, в коробке всегда находится ограниченное число кирпичей. Так и человеческая память всегда сохраняет текущую информацию, важную в данный момент. Как только потребность в ней исчезает, на ее место приходит новая информация, которая «выталкивает» лишнее из области памяти.

Соответственно, чтобы что-либо попало в КВП пользователя, он должен это заметить (для чего, собственно говоря, и полезно проектировать интерфейс с учетом возможностей человеческого восприятия) и счесть полезным лично для себя. Таким образом, **самое важное в интерфейсе должно быть наиболее заметным** (вот мы и узнали теоретическое обоснование очередного очевидного факта).

Другая интересная особенность КВП заключается в том, что смена содержимого в ней происходит при появлении новых стимулов. Практический смысл этого наблюдения состоит в том, что **нельзя допускать, чтобы пользователь отвлекался, поскольку новые стимулы при отвлечении стирают содержимое КВП**. Необходимо максимально облегчать возвращение пользователя к работе.

Кратковременная память имеет весьма ограниченный объем.

Считается, что человеческая память способна запомнить семь плюс-минус два элемента. Оценивать объем КВП применительно к интерфейсу как всеобъемлющие (7 ± 2) элементов не вполне правомерно.

Во-первых, в КВП информация хранится преимущественно в звуковой форме. Это значит, что вместо смысла запоминаемых элементов в КВП хранится текст, написанный на этих элементах.

Для нас это означает, что ***подвергать ограничению следует преимущественно те элементы, которые содержат текст.***

Во-вторых, известно, что в память помещается гораздо больше, но только в тех случаях, когда элементы сгруппированы. Соответственно, всегда ***можно сгруппировать элементы и поместить в КВП пользователя больше информации.***

Например, большинство людей за 30 с могут запомнить список из пяти слов. Но некоторые могут за это же время запомнить список из десяти слов. Разбиение этого списка на более мелкие фрагменты (например, две или три группы) приравнивает такой процесс запоминания к списку из пяти слов.

В-третьих, существует некоторое количество людей, способных удержать девять значений в КВП, но количество людей, способных удержать в памяти только пять или шесть значений, тоже довольно существенно. Это значит, что с практической точки зрения гораздо удобнее считать, что объем КВП равен ровно семи элементам (или, если ситуация позволяет, шести), поскольку расчитывать нужно не на сильное, а на слабое звено.

Так что значительно эффективнее считать, ***что объем кратковременной памяти равен пяти (шести, из которых один в запасе) элементам.*** Не более, но и не менее.

И запоминание, и извлечение информации из памяти требует усилий. Более того, поскольку содержимое КВП теряется при поступлении новых стимулов, пользователям приходится прилагать усилия, чтобы просто удержать информацию в памяти (вспомните, сколько раз вы повторяли номер телефона, чтобы удержать его в памяти на время, пока вы переходите в другую комнату).

Таким образом, необходимо снижать нагрузку на память пользователей, т. е. избегать ситуаций, когда пользователю приходится получать информацию в одном месте, а использовать ее в другом.

Объем ДВП очень велик, информация, попавшая в ДВП, хранится, судя по всему, вечно. Интерес представляют два вопроса: при каких условиях информация попадает в ДВП и сколько «стоит» вспоминание.

Оба вопроса очень интересны с точки зрения обучения пользователей, второй вопрос, к тому же, интересен еще и с точки зрения улучшения способности пользователей сохранять навыки работы с системой в течение длительного времени (а это одна из основных характеристик хорошего интерфейса).

Считается, что информация попадает в ДВП в трех случаях.

Во-первых, при повторении, т. е. при зубрежке.

Чем больше повторений, тем больше шансов, что информация будет запомнена. С точки зрения дизайна интерфейса это наблюдение вызывает очень простую эвристiku: если системой придется пользоваться часто, пользователи ей обучатся, деваться-то им некуда. Это очень утешительное наблюдение.

Во-вторых, при глубокой семантической обработке (*семантика* – свойство, определяющее смысл информации как соответствие сигнала реальному миру).

Если пользователь долго мучается, стараясь понять, как работает система, он запомнит ее надолго, если не навсегда. Чем больше человек думает о какой-либо информации, чем больше он соотносит ее с другой информацией, уже находящейся в памяти, тем лучше он запомнит то, о чем думает. Несколько помогает понять устройство механизма запоминания его антипода, а именно забывание. Самое простое объяснение имеет затухание: если информация не используется долгое время, она забывается.

В-третьих, при наличии сильного эмоционального шока.

Эмоциональный шок нас интересует слабо – не стоять же, в самом деле, за спиной у пользователя, стреляя время от времени из ружья, чтобы он волновался (тем более, что после шока запоминание прерывается). Достаточно и повторения с обработкой.

Таким образом, повторение можно охарактеризовать как способ мощный, но ненадежный, поскольку трудно рассчитывать на повторение при нечастой работе с системой (существует множество систем, используемых редко или даже однократно). Семантическая же обработка есть способ мощный, но дорогой: без повода пользователи не будут задействовать свой разум, предоставить же им повод сложно.

Лучше всего в качестве повода работает аналогия, неважно как она представлена: как метафора интерфейса или как эпитет в документации.

Считается, что обращение к ДВП стоит довольно дорого. Поспорить с этим невозможно, поскольку в утверждении содержится слово «довольно», обладающее крайне размытым значением.

На самом деле все сложно. Разные понятия вспоминаются с разной скоростью: слова, например, вспоминаются быстрее цифр, а визуальные образы – быстрее слов.

Для того чтобы пользователь запомнил, как работать с системой, ему необходимо либо постоянно «загружать» в кратковременную память одинаковые фрагменты (повторять), либо сознательно запоминать возможные способы работы с системой (обучаться), либо получать постоянные консультации по ходу работы (подсказки и помочь). Так, постепенно получаемая информация переходит из кратковременной памяти в долговременную. Быстрой обработке информации способствует такой интерфейс, который не требует серьезного обучения, обеспечивает поддержку пользователя, понятность и предсказуемость всей системы.

Гештальт-принципы организации восприятия

Обычно разработчики больше сосредотачиваются на деталях веб-дизайна, чем на общем виде. Они заостряют внимание на закругленных уголках, тенях, шрифтах и т. д. Это все хорошо, но может на самом деле не иметь никакого значения, если клиенту с первого взгляда на понравится замысел. Чего многие не понимают, так это того, что мозг сначала видит общие очертания любой модели, а затем начинает фокусировать внимание и видеть частности.

Гештальт (*нем. Gestalt – форма, образ, структура*) – пространственно-наглядная форма воспринимаемых предметов, чьи существенные свойства нельзя понять путем суммирования свойств их частей.

Гештальт-психология возникла в Германии в 20-х гг. XX в. и была основана немецкими мыслителями Максом Вергеймером, Вольфгангом Кёлером и Куртом Коффка, выдвинувшими идею изучения психики с точки зрения целостных структур (гештальтов). Выступая против выдвинутого психологией принципа расчленения сознания на элементы и построения из них сложных психических феноменов, они предлагали идею целостности образа и несводимости его свойств к сумме свойств элементов. По мнению этих теоретиков, предметы, составляющие наше окружение, воспринимаются чувствами не в виде отдельных объектов, а как организованные формы. Восприятие не сводится к сумме ощущений, а свойства фигуры не описываются через свойства частей. В связи с этим часто подчеркивается большая роль гештальт-психологии в становлении системного подхода – не только в психологии, но и в науке в целом.

Вергеймер изложил принципы организации восприятия в своей работе, опубликованной в 1923 г. Он исходил из того, что

мы воспринимаем предметы в той же манере, в какой воспринимаем кажущееся движение – то есть как единое целое, а не как наборы разрозненных ощущений.

Базовая предпосылка этих принципов состоит в том, что организация восприятия происходит мгновенно, в тот же момент, когда мы видим или слышим различные формы или образы. Части перцептивного поля (поля восприятия) становятся связанными, объединяясь между собой, чтобы создать структуру, которая выделялась бы на общем фоне. Организация восприятия происходит самопроизвольно, и ее возникновение неизбежно всякий раз, когда мы смотрим вокруг себя.

На рис. 4.2 мы видим двух пожилых людей, смотрящих друг на друга на фоне вазы. Но если присмотреться, можно заметить, что это девушка и мужчина, сидящие у фонтана.

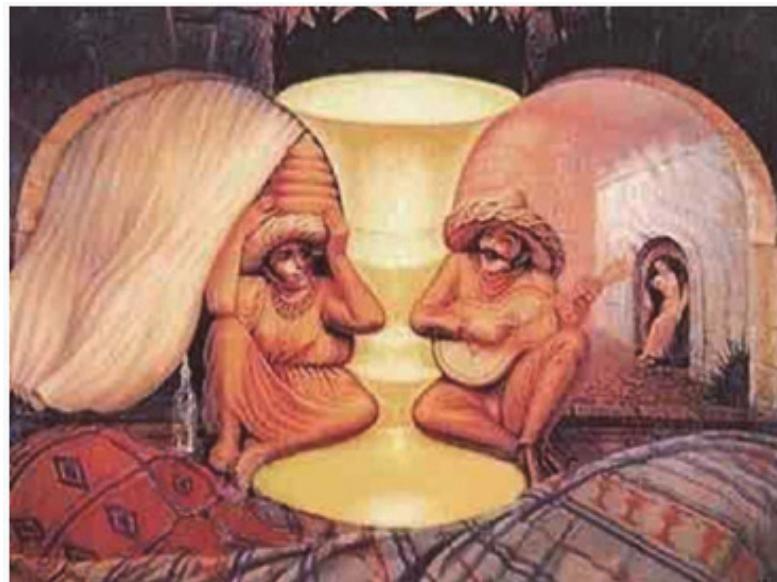


Рис. 4.2. Восприятие целостного образа

Эти принципы восприятия не зависят от высших мыслительных процессов или прошлого опыта; они свойственны самому процессу восприятия.

Согласно гештальт-психологии, свойства целого отличны от свойств его элементов. На основе этого мнения гештальт-психологи разработали несколько принципов, объясняющих особенности организации восприятия.

Все свойства восприятия – константы, фигура, фон – вступают в отношения между собой и являются новое свойство. Целостность

восприятия и его упорядоченность достигаются благодаря следующим принципам.

1. *Близость*. Элементы, которые близки друг к другу в пространстве или во времени, кажутся нам объединенными в группы, и мы стремимся воспринимать их совместно.

2. *Непрерывность*. В нашем восприятии существует тенденция следования в направлении, позволяющем связывать наблюдаемые элементы в непрерывную последовательность или придавать им определенную ориентацию. При этом наш мозг всегда выбирает варианты, более удобные для восприятия и согласующиеся с нашими представлениями о гармонии, устранивая таким образом двусмысленность.

3. *Сходство*. Подобные элементы воспринимаются нами совместно, образуя замкнутые группы.

4. *Замкнутость*. В нашем восприятии существует тенденция завершения незаконченных предметов и заполнения пустых промежутков. Фигуры, изображенные на рисунке, вы воспринимаете как обычные квадраты, хотя на самом деле их контуры не замкнуты.

5. *Простота*. В любых условиях мы стремимся видеть фигуры настолько завершенными, насколько это возможно. В гештальт-психологии это свойство получило название «прегнантная форма». Прегнантный гештальт должен быть симметричным, простым и неизменным и не может быть упрощен или упорядочен каким-либо иным образом. На изображении олимпийских колец мы отчетливо видим именно кольца, а не совокупность фигур более сложной формы. Согласно этому закону, реальность организована наипростейшим образом из всех возможных.

6. *Фигура-фон*. Мы стремимся организовать наше восприятие таким образом, чтобы видеть объект (фигуру) и задний план (фон), на котором она проявляется. При этом фигура представляется наиболее заметной и яснее выделяется на общем фоне изображения. На рисунке фигура и фон являются реверсивными изображениями, в зависимости от того, на что направлено ваше внимание. Фигура всегда выдвинута вперед, фон – отодвинут назад, фигура богаче фона содержанием, ярче фона. И мыслит человек о фигуре, а не о фоне. Однако их роль и место в восприятии определяются личностными, социальными факторами. Поэтому возможно явление обратимой фигуры, когда, например, при длительном восприятии, фигура и фон меняются местами.

Глава 5

ПРОЦЕСС ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Эволюция процесса разработки пользовательского интерфейса

Прежде чем мы будем говорить о проектировании, посмотрим, как развивался процесс разработки программного обеспечения в ИТ-индустрии согласно Аллану Куперу (см. рис. 5.1).

В ранние дни развития ИТ-индустрии процесс разработки сводился к тому, что программисты вынашивали идею продукта, а затем создавали и самостоятельно тестировали его.

В более поздние времена к процессу стали подключаться профессиональные управленцы, их задачи сводились к оценке потребностей рынка и формулированию основных требований к разрабатываемому программному обеспечению.

С развитием ИТ-индустрии в самостоятельную дисциплину выделилось тестирование, а также широкое распространение получили графические интерфейсы пользователя, появилась необходимость разработки различных визуальных элементов, в связи с чем к процессу разработки ПО подключились графические дизайнеры и тестировщики.

В настоящее время используется подход к разработке программного обеспечения, при котором решения о возможностях продукта, его форме и поведении принимаются до начала дорогостоящей и сложной фазы создания продукта. Это обеспечивается включением в процесс разработки этапа проектирования, при этом эффективность проектирования во многом определяется выбранным стилем принятия решений. Можно выделить пять стилей принятия решений.

1. «Непреднамеренное» проектирование – команда сосредоточена на разработке и внедрении приложения, не задумывается об удобстве его использования.

2. Проектирование «для себя» основывается на опыте использования продукта членами команды, имеет большие шансы на успех, чем «непреднамеренное», хорошо, когда члены команды являются главными пользователями разрабатываемого продукта.

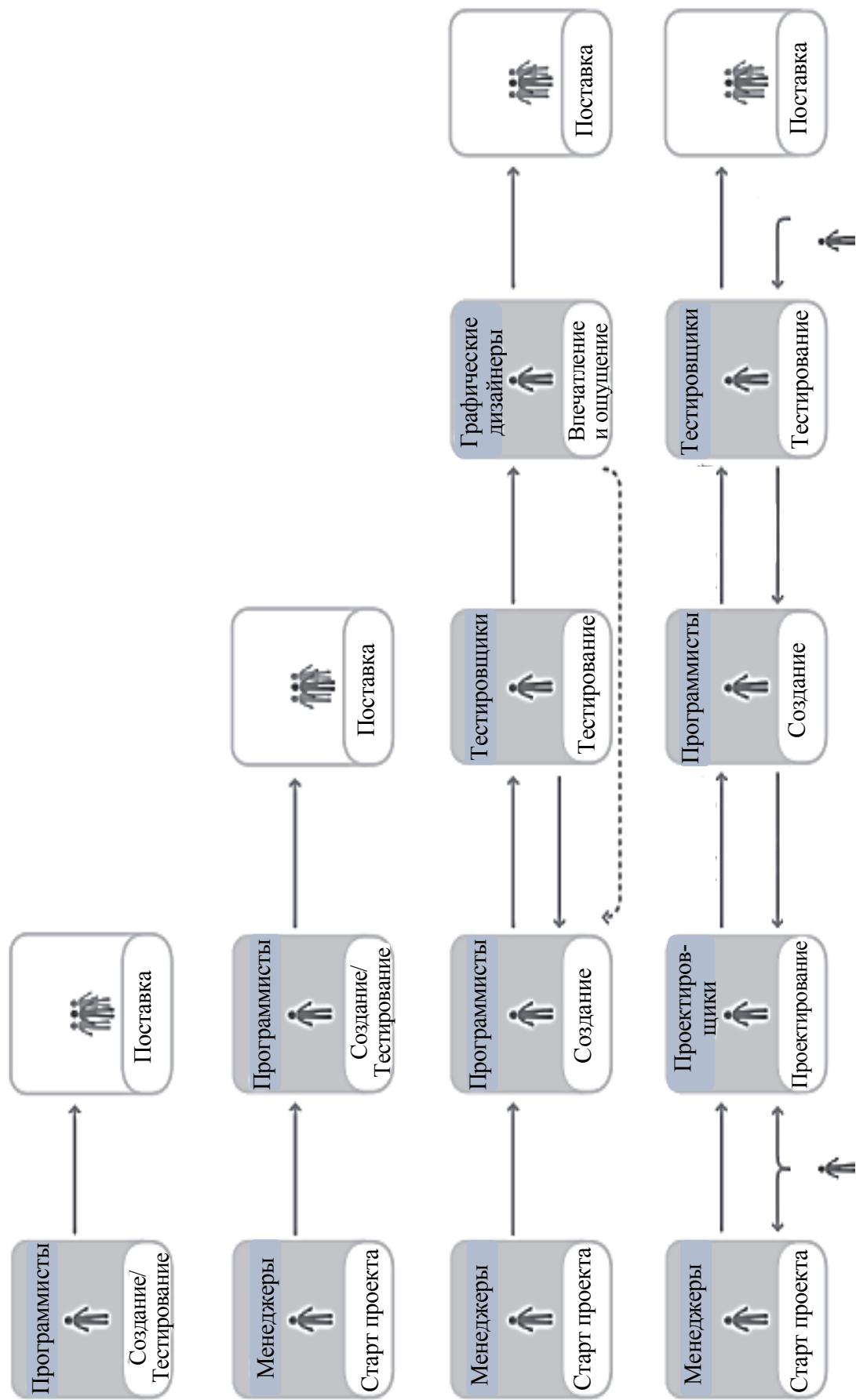


Рис. 5.1. Эволюция процесса разработки программного обеспечения

3. *Genius проектирование* основывается на опыте всех членов команды в проектировании подобных продуктов, хорошо работает, если уже есть опыт проведения предварительных исследований пользователей и сценариев их поведения с последующей проверкой соответствия дизайна ожиданиям пользователей.

4. *Проектирование, ориентированное на деятельность*, основывается на исследовании поведения пользователей. Для исследования часто применяются методики, основанные на деятельности, например, построение диаграмм последовательности операций (work flow diagrams) и ориентированные на задачи тестирования удобства использования.

5. *Проектирование, ориентированное на пользователя*, основывается на глубоком исследовании целей и нужд пользователей, контекста использования, дает возможность принимать детальные решения, которые были бы невозможны при применении других методов.

Стили перечислены в порядке возрастания объема исследований, которые необходимо провести команде проектировщиков для принятия решений. На первый взгляд, должно казаться, что оптимальным будет использование стиля, ориентированного на пользователей, но это не всегда так. Существуют проекты, для которых затраты времени и сил на проведение детального исследования пользователей не оправданы, с другой стороны, существуют проекты, которые без таких исследований просто провалились бы. Умение для каждого проекта выбрать наиболее эффективный стиль проектирования является одной из ключевых характеристик хорошего проектировщика. Наиболее эффективные команды должны владеть всеми пятью методами принятия решений в процессе проектирования и выбирать именно тот, который лучше бы соответствовал нуждам и целям проекта.

Проектирование взаимодействия

Проектирование интерфейса представляет собой довольно сложный и многоэтапный процесс, каждый этап которого состоит в свою очередь из отдельных ступеней. В общем случае весь процесс можно представить в виде четырех этапов (рис. 5.2.), направленных на решение основной задачи – обеспечить оптимальное взаимодействие пользователя с системой.



Рис. 5.2. Этапы эргономического проектирования пользовательского интерфейса

Для успешного проектирования цифровых интерактивных продуктов необходимо особое внимание уделять проектированию поведения. При решении данной задачи нельзя рассматривать составляющие ее части отдельно друг от друга. В системе «человек – машина» главным звеном является человек, а система, являясь подчиненным звеном, должна реагировать на его действия, а не наоборот.

В связи с этим появляется новая дисциплина проектирования, которая получает название проектирование опыта взаимодействия и сосредотачивается в основном на проектировании поведения программного продукта.

Проектирование опыта взаимодействия или проектирование взаимодействия – это новая область научно-практической деятельности, которая в последние годы выделяется как самостоятельная дисциплина, сосредоточенная на проектировании поведения пользователя продукта.

Проектирование взаимодействия – это описание возможного поведения пользователя и определение того, как система будет реагировать на его поведение и приспосабливаться к нему.

Потребовалось немало времени, прежде чем разработчики пришли к мысли, что нужно перейти от разработки программного обеспечения, хорошо работающего с точки зрения машины, к созданию программ, хорошо работающих с точки зрения человека.

Проектирование взаимодействия касается не столько эстетических аспектов, сколько понимания потребностей пользователей и принципов их познавательной деятельности. Форма и эстетическая привлекательность продукта должны работать в гармоничной связке при достижении целей пользователей посредством правильно спроектированного поведения продукта.

Для решения этой задачи эффективно применяется инструментарий, включающий методику персонажей, текстовые сценарии взаимодействия, а также проектирование, ориентированное на цели. Весь этот набор можно объединить под названием «проектирование взаимодействия». При таком подходе к проектированию за отправную точку принимается человек, главная цель – выяснить, чего хочет пользователь.

Проектирование взаимодействия предоставляет описание окончательного варианта продукта, которое содержит предельно ясную и точную информацию о том, кто конкретно будет использовать продукт, каким образом и с какой целью. Имея такое описание, программисты осознают, что именно они создают, руководители могут оценить прогресс в работе программистов, а маркетологи получают понимание источника мотивации покупателя.

Разумеется, проектирование продукта не может концентрироваться только на поведении, необходимо учитывать внешний вид (форму) и информационное наполнение (содержание) разрабатываемого продукта. Поэтому необходимо сочетать подходы различных дисциплин проектирования: проектирование взаимодействия, основное внимание на поведение; информационная архитектура, занимается структурированием содержания; промышленный и графический дизайн, отвечает за форму продуктов и услуг.

Именно грамотная продуманность всех деталей на этих этапах позволит создать армию поклонников вашего продукта. Ярким примером здесь является компания Apple, которая пошла по такому пути и завоевала сердца тысяч и миллионов.

Джесс Гаррет в своей книге «Веб-дизайн. Элементы опыта взаимодействия» представляет проектирование опыта взаимодействия в виде пяти уровней:

- стратегии (или идеи);
- набора возможностей;
- структуры;
- компоновки;
- поверхности (или внешнего вида интерфейса).

Эти пять уровней составляют концептуальную основу для обсуждения связанных с опытом взаимодействия проблем и средств их решения.

Пять наших слоев делятся на две части (рис. 5.3).

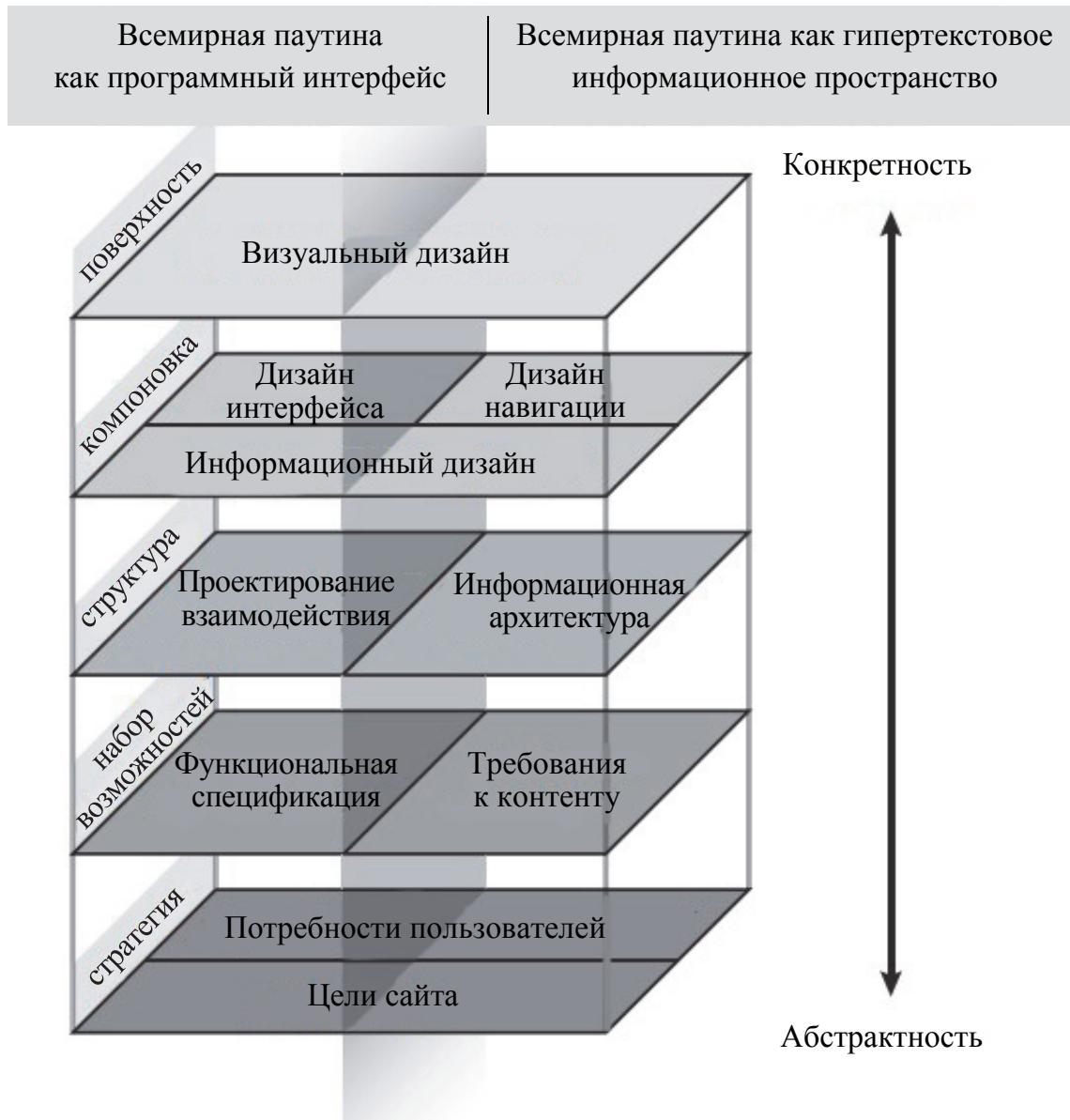


Рис. 5.3. Элементы проектирования опыта взаимодействия

Слева – все, что касается использования **web** как интерфейса программы. **Справа** – все, что связано с информационной структурой. Каждый уровень зависит от уровней, расположенных ниже, и диапазон выбора решений ограничен решениями, принятыми на нижних уровнях.

На каждом из уровней размещаются элементы проектирования.

Уровень стратегии

Цели сайта (Site Objectives) – это бизнес-цели заказчика.

Потребности пользователей (User Needs). Они определяются людьми, которые будут пользоваться нашим проектом. Нужно

понимать, чего хочет аудитория и как эти пожелания согласуются с другими ее потребностями.

Уровень набора возможностей

Функциональная спецификация (Functional Specification) – описание функций и возможностей, которые могут быть представлены пользователю.

Требования к контенту (Content Requirements) – это описание различных элементов содержимого, которые необходимо создать.

Уровень структуры

Определяется, как подчинены друг другу разделы, какова иерархия построения информации, как осуществляются переходы между разделами. Если предыдущие уровни визуальные, то уровень структуры логический.

Проектирование взаимодействия (Interaction Design, IxD) – выяснение, как система будет вести себя в ответ на действия пользователей.

Информационная архитектура (Information Architecture, IA) – организация элементов содержимого в пределах информационного пространства.

Уровень компоновки

Это схематичное разделение страницы на части. Здесь нет информации о внешнем представлении – только о смысле конкретной части страницы, например, блок поиска, блок новостей, блок авторизации. На этом уровне определено, как должны располагаться части страницы.

Информационный дизайн (Information Design) – представление информации в таком виде, который облегчает ее восприятие.

Дизайн интерфейса (Interface Design) – организация элементов интерфейса, позволяющая пользователям взаимодействовать с функциями системы.

Дизайн навигации (Navigation Design) – набор экранных элементов, позволяющих пользователю перемещаться по информационной архитектуре.

Уровень поверхности

Это самый верхний, презентационный уровень. Он представляет собой совокупность шрифтов, таблиц, изображений, цветов, результатов поиска – всего, что видит пользователь. На этом уровне определено, как выглядит для посетителя содержимое сайта.

Визуальный дизайн (Visual Design) – внешний вид конечного продукта. Для получения действительно привлекательных продуктов необходимо, с одной стороны, понимать желания, потребности, мотивации пользователей и контекст, в котором эти пользователи находятся, с другой стороны – понимать возможности, требования и ограничения бизнеса, технологии и предметной области. Использование этих знаний в качестве основы всех планов по созданию цифровых продуктов позволяет сделать их полезными, удобными и желанными, а также экономически жизнеспособными и технически осуществимыми.

Три качества: желанность, жизнеспособность и осуществимость – определяют успешный продукт. Ларри Кили (Larry Keeley) выделил современные задачи разработки, которые как раз предполагают реализацию в продукте всех трех качеств и, если хотя бы один из этих трех столпов значительно слабее двух других, продукт вряд ли выдержит испытание временем.

Если говорить об интерфейсе в целом, то сегодня не существует единой методологии их проектирования, поэтому мы будем использовать сочетание различных подходов.

Этапы работы над пользовательским интерфейсом

В процессе разработки интерфейса можно выделить пять основных этапов:

1. *Сбор функциональных требований* (первоначальное проектирование: уровень стратегии, уровень набора возможностей). Данный этап разработки подразумевает под собой сбор, систематизацию и анализ требований к системе. Также анализируются и систематизируются возможные пользовательские системы (персонажи, актеры). Сбор и анализ требований выполняет бизнес-аналитик.

2. *Информационная архитектура* (уровень структуры). Под информационной архитектурой понимается совокупность методов и приемов структурирования и организации информации. Другими словами, информационная архитектура занимается принципами систематизации, навигации и оптимизации информации, что позволяет облегчить пользователю работу с данными, а именно их поиск и обработку. За проектирование скелетов пользовательского интерфейса и организацию информационных потоков в приложении отвечает информационный архитектор.

3. Прототипирование пользовательского интерфейса. Это этап разработки, который подразумевает создание прототипов экранов системы. Прототипы позволяют обнаружить проблемы функционального характера будущей системы на раннем этапе и устранить их до того, как проект уйдет в разработку к программистам. Прототипы разрабатываются front-end-разработчиком под руководством UI-дизайнера.

4. Юзабилити – тестирование (Ю-тестирование). К тестированию интерфейса привлекают как конечных пользователей, так и специалистов по функциональному тестированию ПО. Ю-тестирование позволяет оценить удобство использования продукта. Информационный архитектор проводит Ю-тестирование и анализирует его результаты.

5. Графический дизайн пользовательского интерфейса. Графический облик интерфейса создает UI-дизайнер. На этом этапе интерфейс системы приобретает необходимый законченный вид. Часто заказчик уже имеет брэнд бук или гайдлайн, задача дизайнера – разработать такой дизайн, который бы соответствовал всем требованиям системы, удовлетворял заказчика и сочетался с задумками информационного архитектора.

На данном этапе может понадобиться помочь смежных специалистов: иллюстратора (художника), 3D-моделлера и др.

Разработанный по всем правилам пользовательский интерфейс значительно повышает эффективность ресурса и дает конкурентные преимущества.

Сбор функциональных требований (первоначальное проектирование)

Важность этого этапа трудно переоценить. На нем закладываются основные концепции системы, влияющие абсолютно на все показатели качества ее интерфейса. Структурные проблемы практически не могут быть обнаружены и решены на остальных этапах (для их обнаружения нужно слишком много везения, а для исправления – денег). Это значит, что чем больше внимания будет уделено проектированию, тем выше будет общее качество.

Прежде чем начать любое проектирование, необходимо выполнить исследование предметной области, которое позволяет проектировщикам лучше понять цели бизнеса, атрибуты бренда и технические ограничения. Необходимо посмотреть, как организо-

вана работа на аналогичных проектах, проанализировать сильные и слабые стороны этих работ, насколько удобно все сделано, насколько грамотно подана информация.

Основной задачей первоначального проектирования является определение необходимой функциональности системы.

Современная наука выдвинула два основных способа определения функциональности, а именно **анализ целей** и **анализ действий пользователей**.

Анализ целей пользователей

Хорошо сформулированная цель должна быть:

– **понятной**. Избегайте использования узкоспециализированной терминологии;

– **ясной**. Избегайте туманных формулировок; подбирайте выражения, которые были бы уместными при определении приоритетов требований;

– **измеримой**. Используйте конкретные утверждения, которые можно проверить независимо, чтобы определить степень успешности проекта.

Идея, лежащая в основе, – «людям не нужны инструменты сами по себе, нужны лишь результаты их работы». Никому не нужен текстовый процессор – нужна возможность с удобством писать тексты.

Очень важно различать задачи и цели. Программисты часто путают их. Они обычно занимаются проектированием, ориентированным на задачи. Программисты начинают проектирование с вопроса: «*Каковы задачи?*». Компьютерное программирование, если добраться до сути, – это создание подробных пошаговых описаний процедур. Процедура есть рецепт решения задачи. Хорошие программисты имеют «процедурный» взгляд на вещи, взгляд, ориентированный на решение задач.

Такой подход дает возможность сделать работу, но не позволяет даже приблизиться к наилучшему решению, а также совершенно не удовлетворяет пользователя.

Цель – это конечное состояние, тогда как **задача** – переходный процесс, необходимый для достижения цели. **Цель** – стабильная сущность. **Задачи** – преходящи.

Вопрос «*Каковы цели пользователя?*» позволяет создавать более качественный и уместный дизайн. Для достижения целей

пользователя, конечно, необходимо решать и задачи, однако существуют разные акценты и различные последовательности выполнения задач.

В случаях, когда для решения поставленных проблем проектировщики взаимодействия анализируют цели, они обычно находят совсем иные, более подходящие решения.

Рассмотрим цели более подробно. Существуют следующие виды целей: личные, практические, корпоративные, ложные.

Личные цели:

- 1) не чувствовать себя глупо;
- 2) не совершать ошибок;
- 3) выполнить адекватный объем работы;
- 4) развлечься (или хотя бы не страдать от скуки).

Личные цели всегда истинны и действительны в определенных рамках для всех людей. Они всегда предшествуют всем другим целям, хотя очень редко становятся предметом обсуждения – как раз потому, что являются личными.

Любая система, идущая вразрез с личными целями, в конечном итоге обречена на неудачу, независимо от того, насколько качественно позволяет достигать целей иного рода.

Корпоративные цели. У каждого делового предприятия свои требования к программному обеспечению, и уровень этих требований достаточно высок:

- 1) увеличить прибыль;
- 2) увеличить рыночную долю;
- 3) победить конкурентов;
- 4) нанять больше сотрудников;
- 5) предложить новые продукты и услуги;
- 6) выпустить акции компании в свободное обращение.

Цель «увеличить прибыль» является преобладающей для совета директоров или держателей акций. Эти цели необходимы для эффективной работы, но сами по себе не мотивирующие. С точки зрения корпорации это все важные цели, однако работу выполняет не корпорация, а люди, а для людей важнее цели личные.

Сущность качественного проектирования взаимодействия состоит в том, чтобы позволить пользователям достигать практических целей, не отказываясь от целей личных.

Программа, которая не позволяет достичь какой-либо корпоративной или личной цели, потерпит неудачу.

Практические цели:

- удовлетворять требованиям клиента;
- сохранять информацию о заказах клиента;
- создавать математические модели бизнеса.

Практическая цель удовлетворения требований клиента соединяет корпоративную цель (более высокие прибыли) с личной целью пользователя (работать продуктивно).

Обычно программисты создают программное обеспечение, которое замечательно помогает достигать практических целей, но совершенно неспособно удовлетворить пользователей. Разумеется, чтобы удовлетворить цели бизнеса, вы должны встроить в программу определенные возможности. Однако, если пользователь не может достичь личных целей, то он не способен эффективно достигать целей компании. Непреложный факт: счастливые и довольные сотрудники наиболее эффективны.

С другой стороны, если программа игнорирует практические цели и служит только целям пользователя, это означает, что вы спроектировали компьютерную игру.

Ложные цели:

- 1) экономия памяти;
- 2) уменьшение потребности в клавиатурном вводе;
- 3) поддержка работы в браузере;
- 4) простота в освоении;
- 5) обеспечение целостности данных;
- 6) ускорение ввода данных;
- 7) увеличение скорости исполнения программы;
- 8) применение супертехнологии или супервозможностей;
- 9) улучшение внешнего вида;
- 10) сохранение единобразия интерфейса на различных платформах.

Повседневные продукты, основанные на программном обеспечении, создаются на основе ложных целей. Многие из этих целей облегчают задачу создания программ, в чем и заключается цель программиста, и потому их приоритет повышается во вред конечному пользователю. Другие ложные цели связаны с задачами, возможностями и инструментами. Все это средства достижения результатов, но еще не результаты, тогда как цели всегда являются результатами.

После того, как истинные цели пользователей установлены (и доказано, что таких пользователей достаточно много, чтобы оправдать создание системы), приходит время выбирать конкретный способ реализации функций, для чего используется второй метод.

Анализ действий пользователей

Достижение почти всех целей требует от пользователей совершения определенных действий. Разумеется, эти действия могут различаться при разных способах достижения.

Единственным же способом проверить, нужна функция или нет, является наблюдение за пользователями и анализ их действий.

Поскольку на этом этапе мы узнаем, какая именно функциональность нужна для каждого варианта, можно избрать верный путь по правилу «*чем меньше действий требуется от пользователя, тем лучше*». Не стоит забывать и про другое правило: *чем меньше функций, тем легче их сделать*.

Создание пользовательских сценариев

Цель – написать словесное описание взаимодействия пользователя с системой, не конкретизируя, как именно проходит взаимодействие, но уделяя возможно большее внимание всем целям пользователей. Количество сценариев может быть произвольным, главное, что они должны включать все типы задач, стоящих перед системой, и быть сколько-нибудь реалистичными. Сценарии очень удобно различать по именам участвующих в них вымышленных персонажей.

Алан Купер предложил эффективный инструмент для анализа действий пользователя: это точное описание пользователя продукта и его целей. Для этого мы выдумываем несуществующих пользователей и проектируем для них.

Таких несуществующих пользователей называют **персонажами** (*personas*), и они представляют собой необходимую базу качественного проектирования взаимодействия.

Краткая справка: «персонаж» (франц. *personnage*, от лат. *persona* – личность, лицо) – действующее лицо пьесы (спектакля), сценария (кинофильма), романа и других художественных произведений.

Персонажи – не реальные люди, но они представляют реальных людей в процессе проектирования. Будучи воображаемыми, они тем не менее определяются достаточно жестко и точно. На практике действительно выдумываются их имена и личные сведения.

Персонажи определяются своими целями. Цели же, разумеется, определяются персонажами.

Персонаж должен быть конкретным. Чем более конкретными мы делаем персонажи, тем более эффективными инструментами проектирования они становятся. Для этого мы выбираем ему имя.

Персонаж должен быть воображаемым. Описание персонажа должно быть подробным, а не идеальным, т. е. важнее определить персонаж как можно подробнее и конкретнее, чем создать абсолютно правильный персонаж.

Изобретенные персонажи уникальны для каждого проекта.

Точно определенный персонаж дает нам:

- определенность относительно уровня владения пользователем компьютером, поэтому мы перестаем терзаться загадкой, для кого проектировать: для дилетанта или специалиста; реалистичный взгляд на уровень подготовленности пользователей;

- возможность объяснять наши решения в области проектирования. Это как прожектор, высвечаивающий для разработчиков, маркетологов, руководителей очевидную правильность наших решений по проектированию.

Программистам свойствен математический подход, и они естественным образом не склонны рассматривать отдельных пользователей, предпочитая обобщение. Ключ к успеху в том, чтобы заставить программистов поверить в существование и реальность созданных персонажей. Примерив на персонаже продукт или задачу, вы сразу можете понять, удастся ли вам его удовлетворить.

Можно выделить шесть типов персонажей, которые назначаются обычно в таком порядке:

- 1) **ключевой** задает основную цель в проектировании интерфейса, выбирается методом исключения: цели каждого персонажа рассматриваются в сравнении с целями остальных. Если не очевидно, какой из персонажей является ключевым, это может означать одно из двух: или продукту требуется несколько интерфейсов, каждый из которых предназначен для своего ключевого персонажа (так часто бывает в корпоративных и технических продуктах), или же объем его функциональности слишком широк;

2) второстепенный в основном оказывается доволен интерфейсом ключевого персонажа, но имеет дополнительные потребности, которые можно включить в продукт, не нарушая его способности служить ключевому персонажу;

3) дополнительный – пользовательский персонаж, не являющийся ни ключевыми, ни второстепенными. Их нужды обычно полностью представлены сочетанием нужд ключевого и второстепенных персонажей и удовлетворяются одним из ключевых интерфейсов;

4) покупатель – персонаж, отражающий потребности покупателей, а не конечных пользователей. Обычно персонажи покупателей используются в качестве второстепенных персонажей. Однако в некоторых корпоративных средах кто-то из таких персонажей может оказаться ключевым, если ему предназначается собственный административный интерфейс;

5) обслуживаемый – не является пользователем продукта, однако его непосредственно затрагивает применение продукта. Обслуживаемые персонажи – это способ отслеживать социальные и физические воздействия второго порядка, оказываемые продуктом. Эти персонажи используются так же, как второстепенные;

6) отвергаемый – используется, чтобы демонстрировать заинтересованным лицам и участникам разработки, что существуют пользователи, для которых продукт не предназначен.

Подбор персонажей

Каждый проект получает собственный набор персонажей в количестве от трех до двенадцати. Мы проектируем не для каждого из них, но все персонажи полезны для выражения пользовательской аудитории. В каждом наборе персонажей есть хотя бы один ключевой персонаж. Эта личность находится в фокусе процесса проектирования.

Персонажи и цели неразделимы, они – как разные стороны одной медали. Персонаж существует потому, что у него есть цели, а цели существуют, чтобы придавать смысл персонажу.

Следует отметить, что набор характеристик, подробно описывающий пользователя, зависит от предметной области и контекста решаемых им задач. Наиболее общий шаблон профиля содержит в себе следующие разделы:

- социальные характеристики (фотография, имя, возраст, место жительства, род занятий и биография);
- навыки и умения работы с компьютером;
- мотивационно-целевая среда;
- рабочая среда;
- особенности взаимодействия с компьютером (специфические требования пользователей, необходимые информационные технологии и др.).

Фотография. Выбирая фотографию, следите за тем, чтобы изображение не казалось постановочным или «глянцевым». Персонаж, снятый в естественной обстановке, создает более яркий и достоверный образ.

Имя. С внешним обликом необходимо связать имя. «Нинель» не только лучше звучит, чем «Блондинка, за 30, работает с детьми», но и проще запоминается и связывается с конкретным персонажем. Не следует использовать имена коллег или заказчиков, хотя такая идея и выглядит соблазнительно.

Возраст. Между моделями поведения 21-летней студентки и 34-летней домохозяйки существуют серьезные различия!

Место жительства. В Италии, например, в разных областях страны говорят на разных диалектах. В Республике Беларусь стоимость потребительской корзины жителя Минска, скорее всего, будет отличаться от стоимости потребительской корзины жителя Крупок.

Род занятий. Представление о том, чем ваш персонаж занимается, поможет вам лучше его понять, так как вы сможете опираться на типичные события и ситуации его повседневной жизни. Персонаж, работающий в поликлинике, ежедневно контактирует со многими людьми, тогда как в жизни оператора подъемного крана вряд ли есть избыток общения.

Биография – убедительная история, которая делает персонаж реалистичным. Она должна быть достоверной, так что наделение персонажа чертами реальных людей – вполне допустимый прием. Сделайте все необходимое, чтобы персонаж выглядел достоверно и как можно более осмысленно для проекта, над которым вы работаете.

Профили пользователей могут по необходимости расширяться за счет добавления других (значимых с точки зрения проектировщика) характеристик пользователей.

При создании персонажей необходимо предоставить достаточно информации, чтобы увлечь людей и заставить их почувствовать человека, описание которого они читают.

Дополнительный контент. Базовые подразделы – своего рода «необходимый минимум» для всех создаваемых вами персонажей. В большинстве случаев вам придется добавить к ним те или иные дополнительные элементы. Ценность ваших персонажей можно повысить, если дополнить их описание следующими подразделами.

1. *Образование.* Человек с аттестатом средней школы может серьезно отличаться своим покупательским поведением и восприятием бренда от человека со степенью магистра; таким образом, эта информация способна повлиять на то, как будет восприниматься ваш персонаж.

2. *Заработка плата.* Эта информация способна привести к значимым открытиям, если вы ориентируетесь на определенные уровни состоятельности.

3. *Интернет-активность.* Учитывая, что сейчас многие проекты имеют интернет-составляющую, при подготовке этого элемента следует руководствоваться здравым смыслом.

4. *Ключевая точка взаимодействия с заказчиком, брендом или проектом.* Персонаж узнал о них от своих знакомых, по телевизору или радио, прочитал в интернет-обзоре, на форуме или во всплывающем рекламном окне? Используйте статистические данные для прояснения этого момента и включите результат в описание персонажа – это поможет заложить основу для привлечения пользователей к проекту.

5. *Техническая подготовка.* На каком компьютере работает ваш персонаж – на РС или на Mac? У него есть собственный компьютер? Использует ли он системы мгновенного обмена сообщениями, Flickr, ведет ли блог? Насколько уверенно чувствует себя при этом? Поможет ли ему очень простое решение, рассчитанное на новичка? Есть ли у него MP3-плеер или другое портативное устройство? Использует ли он DVR, AppleTV или другие устройства для просмотра телепрограмм? Этот список может быть очень длинным. В зависимости от заказчика, бренда или проекта такие мелочи могут сыграть важную роль.

6. *Уровень социального комфорта.* Учитывая бурный рост сетевых сообществ и социальных сетей, точное описание степени

участия персонажа в сетевом пространстве может многое рассказать о нем. Есть ли у него учетная запись Twitter? Если есть, то сколько у него подписчиков? Насколько персонаж активен? Является ли он лидером? Использует ли MySpace, Facebook, LinkedIn, другие агрегаторы или сетевые сообщества?

Разработка сценариев

На данном этапе уточняется, какими должны быть информация и функциональные возможности интерфейса, чтобы пользователь дошел до целевого действия.

Сценарий – это описание действий, выполняемых пользователем в рамках решения конкретной задачи на пути достижения его цели. Очевидно, что достигнуть некоторой цели можно, решая ряд задач. Каждую из них пользователь может решать несколькими способами, следовательно, должно быть сформировано несколько сценариев. Чем больше их будет, тем ниже вероятность того, что некоторые ключевые объекты и операции будут упущены.

Эффективность сценария определяется в большей степени его охватом, чем глубиной. Иначе говоря, важнее, чтобы сценарий описывал процесс от начала до конца, чем чтобы он описывал каждый шаг в исчерпывающих подробностях.

Важно развивать лишь те сценарии, которые позволяют продвигаться вперед в процессе проектирования. Достаточно разработать лишь два вида сценариев, хотя сценариев каждого вида может быть и несколько.

На различных этапах целеориентированного проектирования используются разные типы сценариев, основанных на персонажах, причем на каждой последующей стадии особенностям интерфейса уделяется больше внимания, чем на предыдущей.

Первый тип сценариев – **контекстные сценарии** – используется для высокоуровневого рассмотрения того, как продукт может наилучшим образом послужить потребностям персонажей. (Раньше эти сценарии называли «день из жизни», но в конечном итоге сочли, что этот термин является слишком общим).

Контекстные сценарии создаются на стадии первоначального проектирования, пишутся с точки зрения персонажа и сосредоточены на человеческих действиях, впечатлениях и желаниях. При разработке именно этого вида сценариев проектировщик

располагает наибольшей свободой в представлении идеального опыта пользователя.

После того как команда проектировщиков определила функциональные и информационные элементы, а также создала общую инфраструктуру, необходимо пересмотреть контекстный сценарий. В результате добавления к нему более подробных описаний взаимодействия пользователя с продуктом и применения проектного лексикона он становится **сценарием ключевого пути**.

Сценарии ключевого пути фокусируются на наиболее важных моментах взаимодействия, не теряя из виду того, как персонаж пользуется продуктом при достижения своих целей. По мере уточнения образа продукта эти сценарии параллельно с проектированием проходят итерационную доработку.

В ходе всего процесса команда проектировщиков применяет **проверочные сценарии** для тестирования проектных решений в различных ситуациях. Как правило, эти сценарии менее подробны и обычно принимают форму набора вопросов «а что, если?..», касающихся предложенных решений.

На основе выявленных сценариев работы осуществляется разработка структуры экранов, т. е. определяется количество экранов, функциональность каждого из них, навигационные связи между ними, формируется структура меню и других навигационных элементов.

Сценарии представляют примеры использования как отправную точку для проектирования, а также закладывают основу для юзабилити-тестирования. Сценарии являются реалистичными и детализированными описаниями действий пользователей, но в них не должно быть ссылок на применение каких-либо элементов пользовательского интерфейса.

Основная сложность при использовании этого метода связана с осознанной необходимостью разработки такого количества сценариев, которое покрывало бы наибольшее количество различных ситуаций, а не только самых типичных или, например, интересных разработчикам. Наряду с последовательными, в список стоит включить и нелинейные сценарии, которые будут использованы при тестировании. В дальнейшем, для оценки разрабатываемой системы, должен использоваться полный набор сформированных сценариев.

Для создания сценария проанализируйте следующие вопросы.

1. Кто является главным пользователем в этом сценарии?
2. Посещал ли выбранный пользователь этот сайт ранее (работал ли он ранее с данным программным продуктом)?
3. Какие срочные потребности привели пользователя на сайт (либо заставили обратиться к данной программе)?

Предположим, что необходимо разработать сценарии для будущей почтовой программы. Судя по всему, для этой задачи необходимо три сценария:

– Елизавета Бронеславовна запускает почтовую программу. Она включает процесс скачивания новой почты. Получив почту, она читает все сообщения, затем часть их удаляет, а на одно сообщение отвечает. После чего выключает почтовую программу;

– Еремей Карпович делает активным окно уже открытой почтовой программы и включает процесс скачивания новой почты. Получив почту, он ее читает. Одно сообщение он пересыпает другому адресату, после чего удаляет его, а еще одно печатает. После чего переключается на другую задачу;

– Пришло новое сообщение, и Эмма Валерьевна восприняла соответствующий индикатор. Она делает активным окно почтовой программы и открывает полученное сообщение. Читает его, после чего перемещает в другую папку. Затем переключается на другую задачу.

Дело в том, что на таких сценариях очень хорошо заметны ненужные шаги, например, в третьем сценарии гипотетическая Эмма Валерьевна после получения индикатора не смогла сразу же открыть новое сообщение, но должна была открыть окно системы, найти нужное сообщение, открыть его и только тогда прочесть. Понятно, что от этих ненужных этапов смело можно избавиться уже на этой, весьма ранней, стадии проектирования.

Польза этих сценариев двояка. Во-первых, они будут полезны для последующего тестирования. Во-вторых, сам факт их написания обычно (если не всегда) приводит к лучшему пониманию устройства проектируемой системы, побуждая сразу же оптимизировать будущее взаимодействие.

Проектирование общей структуры

После выделения нескольких профилей пользователей и определения целей и задач, стоящих перед ними, а также пользовательских сценариев переходим к проектированию общей структуры

системы, т. е. необходимо выделить отдельные функциональные блоки и определить, как именно эти блоки связываются между собой. Под отдельным функциональным блоком будем понимать функцию / группу функций, связанных по назначению или области применения в случае программы, и группу функций / фрагментов информационного наполнения в случае сайта.

Типичная структура сайта (слева) и типичная структура программы представлены на рис. 5.4. Если сайты обычно разветвлены, в том смысле, что функции обычно размещаются в отдельных экранах, то программы обычно имеют только один изменяющийся экран, в котором и вызываются почти все функции.

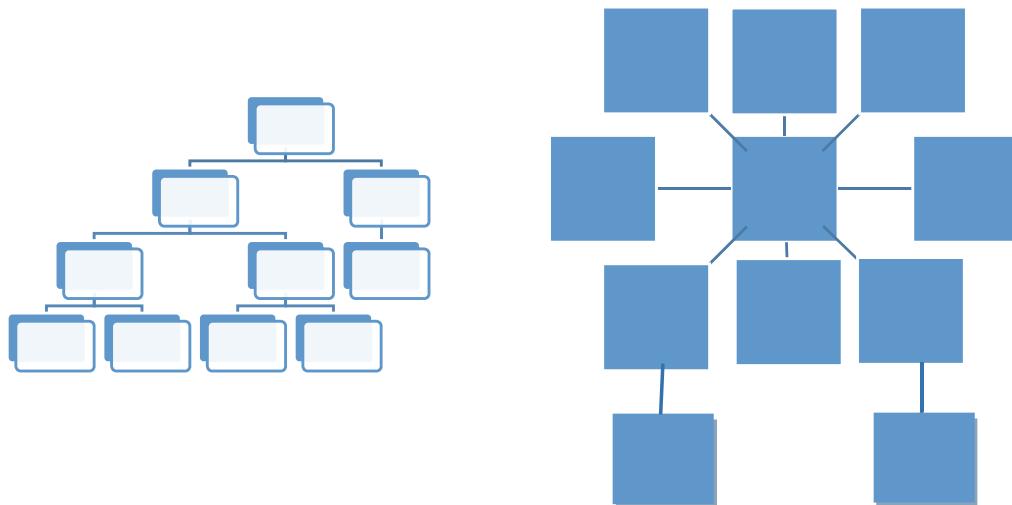


Рис. 5.4. Типичная структура сайта (слева) и типичная структура программы (справа)

Проектирование общей структуры состоит из двух параллельно происходящих процессов: выделения независимых блоков и определения связи между ними. Если проектируется сайт, в завершении необходимо также создать схему навигации.

Выделение независимых блоков

Выделение независимых блоков выполняется на основе выявленного с помощью пользовательских сценариев перечня отдельных функций. В приложении функция представлена функциональным блоком с соответствующей экранной формой (формами). Как правило, несколько функций объединяются в один функциональный блок. Избегайте помещений в один блок более трех

функций, поскольку каждый блок в результирующей системе будет заключен в отдельный экран или группу управляющих элементов. Перегружать же интерфейс опасно. Результатом этой работы должен быть список блоков с необходимыми пояснениями.

Определение связи между блоками

Существует три основных вида смысловой связи между блоками: логическая связь, связь по представлению пользователей, процессуальная.

Логическая связь определяет взаимодействие между фрагментами системы с точки зрения разработчика (суперпользователя). Данная связь очень существенно влияет на навигацию в пределах системы (особенно, когда система многооконная). Чтобы не перегружать интерфейс, стоит избегать блоков, связанных с большим количеством других, – оптимальным числом связей является число три.

Связь по представлению пользователей. В информационных системах, когда необходимо гарантировать, что пользователь найдет всю нужную ему информацию, нужно устанавливать связи между блоками, основываясь не только на точке зрения разработчика, но и на представлениях пользователей.

Например, нужно как-то классифицировать съедобные растения. Помидор, который почти все считают овощем, на самом деле ягода. Не менее тяжело признать ягодой арбуз. Это значит, что классификация, приемлемая для ботаника, не будет работать для всех остальных, причем обратное не менее справедливо.

Процессуальная связь описывает пусть не вполне логичное, но естественное для имеющегося процесса взаимодействие: например, логика напрямую не командует людям сначала приготовить обед, а потом съесть его, но обычно получается именно так.

Жестко заданная процессуальная связь позволяет также уменьшить количество ошибок, поскольку от пользователя при ней не требуется спрашивать себя: «Не забыл ли я чего?».

Замечательным примером жестко заданной процессуальной связи является устройство мастеров (wizards), при котором пользователя заставляют нажимать кнопку «Далее».

Существует любопытная закономерность: чем эстетически привлекательней выглядит схема (без учета цветового кодирования

и интересных шрифтов), тем она эффективней. Всегда надо стараться сделать схему возможно более стройной и ясной.

Навигационная система

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 экранами и объектами страниц.

Процесс навигации должен быть спроектирован таким образом, чтобы помочь пользователям определить, где они находятся, где они находились и куда они могут переместиться в дальнейшем. Для этой работы трудно дать какие-либо конкретные рекомендации, поскольку очень многое зависит от проектируемой системы. Например, рассмотрим систему для сайта новостей.

Незарегистрированный читатель нашел новость – открыл новость – прочитал – поделился с друзьями – прокомментировал – добавил в избранное – вернулся на главную.

Для того, чтобы представить эту схему в графическом виде, нужно каждому этапу присвоить емкое название и придумать блоки, которыми должен быть дополнен каждый этап. Важно не перегрузить пользователя информацией и одновременно включить в схему важные функции. Для этого надо составить полный список всего, что пользователь должен видеть на экране и разделить эту информацию на четыре категории: обязан быть; должен быть; может быть; мог бы быть.

К обязательным элементам нужно отнести функции, решающие задачи пользователей (новости). Кенным – функции, упрощающие достижение этих задач (категории, поиск). К остальным относятся функции, дополняющие решение задач или осуществляющие реализацию других задач (избранное, перепост, комментарий и т. д.).

Навигационные связи между отдельными функциональными блоками отображаются на схеме навигационной системы. Возможности навигации в приложении передаются через навигационные элементы.

Основным навигационным элементом приложения является главное меню. Формирование меню начинается с анализа функций приложения. Для этого в рамках каждой из них выделяют отдель-

ные элементы: операции, выполняемые пользователями, и объекты, над которыми осуществляются эти операции. Следовательно, известно, какие функциональные блоки должны позволять пользователю осуществлять определенные операции над определенными объектами. Выделение операций и объектов удобно проводить на основе пользовательских сценариев и функционала приложения. Выделенные элементы группируются в общие разделы главного меню. Группировка отдельных элементов происходит в соответствии с представлениями об их логической связи. Таким образом, главное меню может иметь каскадные меню, выпадающие при выборе какого-либо раздела. Каскадное меню ставит в соответствие первичному разделу список подразделов.

Интуитивно понятная навигация и положительные ощущения пользователя обусловлены не чем иным, как правильным расположением и представлением информации.

После создания навигационной карты следует еще раз вернуться к портрету своей целевой аудитории и ролям пользователей в приложении. На примерах этих ролей следует снова пройти все этапы навигации внутри приложения, уделив особое внимание функционалу. В конечном итоге, целью пользования любым приложением является получение информации.

Виды структуры web-сайта

Структура сайта разделяется на *внутреннюю* и *внешнюю*. При этом внутренняя структура зачастую значительно влияет на внешнюю.

Внутренняя структура сайта. К ней относятся логические связи между различными страницами ресурса. В данной части необходимо продумать, как пользователь сможет максимально быстро получить доступ к нужной информации. К примеру, позаботиться, чтобы человеку потребовалось не более трех кликов для перехода ко всем важным разделам или интересным страницам. Также иногда внутренней структурой называют особенности размещения директорий и ресурсов на сервере.

Внешняя структура сайта. Она полностью повторяет навигацию ресурса и используется для того, чтобы упростить «путешествие» посетителей по страницам. Благодаря ей человек может получить доступ к основному функционалу сайта с любой страницы.

При этом внешняя структура анализируется поисковыми системами и может повлиять на позицию вашего сайта в выдаче.

Типовые структуры сайтов

Различают четыре основные типовые структуры сайтов. В чистом виде каждая из них используется достаточно редко. Чтобы сделать сайт максимально удобным для пользователя и не навредить его функционалу, следует грамотно комбинировать различные виды структур web-сайтов.

Линейная структура сайта. Данный вид сайта является наиболее простым. Подобная структура последовательна, в ней каждая из страниц ведет на предыдущую и следующую страницы ресурса. В данном случае навигация очень проста и осуществляется с помощью 2–3 ссылок, использование такого сайта чем-то похоже на перелистывание страниц книги.

Решетчатая структура сайта. Разработка структуры веб-сайта подобного типа также не отличается особой сложностью, но она более удобна для использования, чем линейный вариант. В данном случае каждая страница ресурса связана с двумя или тремя страницами одновременно. Обычно каждая из них имеет связи с одной или двумя страницами разделов одного уровня, а также с одной страницей – подразделом. Здесь уже должна прослеживаться иерархическая структура информации на ресурсе.

Иерархическая структура сайта. Основным элементом иерархической структуры является главная страница сайта. Ссылки с нее ведут на разделы второго уровня, а на страницах второго уровня размещены ссылки на материалы/разделы третьего уровня и т. д. В данном случае пользователь, перейдя на главную страницу сайта, должен обязательно посетить страницу определенного раздела, чтобы добраться до подраздела. Зачастую подобная структура web-сайта применяется для каталогов товаров.

Паутинообразная структура сайта. Она предполагает связь страниц «все со всеми», то есть пользователь может попасть с любой страницы ресурса на любую другую страницу, минуя все разделы. Разработка структуры веб-сайта подобного типа достаточно сложна, особенно если на ресурсе расположено много страниц. На практике пользователь просто «потеряется» в огромном количестве ссылок. Стоит отметить, что данная структура web-сайта повторяет в малом масштабе структуру всемирной сети Интернет.

Гибридная структура сайта. Для повышения удобства использования сайта разрабатывают гибридную структуру. Выше описанные структуры web-сайтов практически не пригодны для использования в чистом виде. Они являются конструктором, из которого каждый может взять необходимые ему блоки для реализации своего проекта. К примеру, для каталога товаров интернет-магазина лучше всего использовать иерархическую структуру. При этом на самой странице товара можно расположить ссылки на сопутствующие товары, чтобы пользователь долго их не искал, а это уже элемент паутинообразной структуры.

Главное требование к разработке структуры – это логичность и простота. Пользователь должен с легкостью находить нужные ему материалы. Необходимо отметить, что любой web-сайт не может постоянно поддерживать одну и ту же структуру. Несмотря на то, что разделы и каталоги будут оставаться неизменными, при размещении новых материалов и статей будет возникать внутренняя перелинковка, которая внесет свои корректировки.

Сайты отличаются друг от друга по размеру и предназначению. Интернет-магазины имеют сотни или тысячи страниц, блоги могут обойтись несколькими десятками страниц, а сайтам-визиткам достаточно одной-двух. Поэтому нельзя придумать универсальный шаблон структуры, который подойдет каждому web-сайту.

Создание глоссария

Еще в процессе проектирования полезно зафиксировать все используемые в системе понятия. Для этого нужно просмотреть созданные экраны и выписать из них все уникальные понятия (например, текст с кнопками, названия элементов меню и окон, названия режимов и т. д.). После этого к получившемуся списку необходимо добавить определения всех концепций системы (например, книга или изображение). Затем этот список нужно улучшить:

– уменьшить длину всех получившихся элементов;

– показать этот список любому потенциальному пользователю системы и спросить его, как он понимает каждый элемент. Если текст какого-то элемента воспринимается неправильно, его нужно заменить;

- проверить, что одно и то же понятие не называется 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-инфinitивы (создать, отправить, сохранить).

Нужно стараться не менять список в будущем.

После завершения проектирования структуры интернет-ресурса можно будет перейти к процессу создания страниц, разработки их дизайна и наполнения контентом.

Глава 6

РАЗРАБОТКА ПРОТОТИПА

Результатом выполнения этапа предварительного проектирования является полная функциональная схема, описывающая все взаимодействие пользователя с системой. Известно, сколько экранов (страниц) нужно и что должно находиться на каждом экране. Но даже самые блестящие специалисты допускают ошибки. Это правило особенно очевидно в командной работе. «Умные» команды устраняют все ошибки до того, как продукт попадет в руки пользователя, используя методику, которая называется разработкой прототипов пользовательского интерфейса. В сочетании с тестированием удобства использования продукта (юзабилити тестирование), прототипы не дают команде сбиться с верного курса.

Разработка прототипа – средство, позволяющее проанализировать идеи, прежде чем потратить на них время и деньги. Все опытные мастера и инженеры создают образцы своих изделий до того, как начинают что-либо строить. Архитектор создает модель из бумаги или картона, либо с помощью виртуальных инструментов. Авиаинженеры используют аэродинамическую трубу. Строители мостов разрабатывают модели для исследования нагрузки. Разработчики ПО и веб-дизайнеры создают модели, имитирующие взаимодействие пользователя с их разработками.

Самая веская причина для создания прототипа – экономия времени и ресурсов. По сравнению с реальным продуктом прототипы просты и недороги в разработке. Таким образом, при минимальном вложении средств можно обнаружить ошибки создателей и юзабилити проблемы и улучшить пользовательский интерфейс до того, как сделаны значительные инвестиции в окончательную разработку и технологии.

Прототип страниц – это схематичное представление всех компонентов страницы и их взаимного расположения.

Прототипирование (создание прототипов) и итеративное улучшение проектов интерфейса признаются одними из наиболее мощных методов в проектировании взаимодействия человека и компьютера.

Прототипы используются для достижения одной или нескольких из следующих целей:

- проработать дизайн;
- построить общую коммуникационную платформу;
- увлечь других людей вашими идеями (например, руководство, других проектировщиков и т. д.);
- проверить техническую реализуемость;
- протестировать проектировочные идеи с помощью пользователей (клиентов).

В большинстве случаев прототип после тестирования оказывается неправильным, и его приходится переделывать, причем иногда полностью. Поэтому не следует чрезмерно наводить глянец и стремиться сделать его возможно более похожим на результатирующую систему. Первый прототип стоит делать максимально примитивным. Только после того, как тестирование подтверждает его правильность, стоит делать более детализированный прототип.

В прототипировании выделяют четыре стадии.

Концептуальный дизайн – для исследования различных метафор в интерфейсе и подходов к дизайну.

Дизайн взаимодействия – для организации структуры экранов или страниц и переходов между ними.

Дизайн экранов – разработка внешнего вида каждого из конкретных экранов или страниц.

Тестирование – оценка прототипов для их последующего итерационного улучшения (при помощи небольшого тестирования с представителями реальных пользователей или с использованием «эвристического анализа» экспертами).

Виды прототипов и технологии их создания

Статичные прототипы: включают в себя такую популярную методику, как *бумажное прототипирование*, создание «скелетов» (набросков, эскизов интерфейсов – см. рис. 6.1), создание прототипов при помощи графических редакторов в виде статичных изображений (*раскладовка*).

Бумажный прототип. Необходимо нарисовать на бумаге все экраны и диалоговые окна. Нужно убедиться, что все интерфейсные элементы выглядят единообразно и сколько-нибудь похоже на реальные.

Рукописная схема поможет изучить наиболее важную часть приложения – контент. Понимание возможной схемы взаимодействия пользователя с контентом поможет дать более точную оценку числа страниц / экранов, необходимых в программе.

На первом прототипе вполне можно тестировать восприятие системы пользователем и ее основную логику.

Польза прототипирования на бумаге заключается, во-первых, в исключительной простоте модификации по результатам тестирования, а во-вторых, в возможности безболезненно отлавливать представителей целевой аудитории.

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 пережить множество исправлений и, соответственно, много версий.

Прототип должен создаваться быстро. Его основная функция – показать часть функциональности в простейшем виде без акцентирования на иконках, цветах, шрифтах и тому подобном. Пример бумажного прототипа представлен на рис. 6.1.

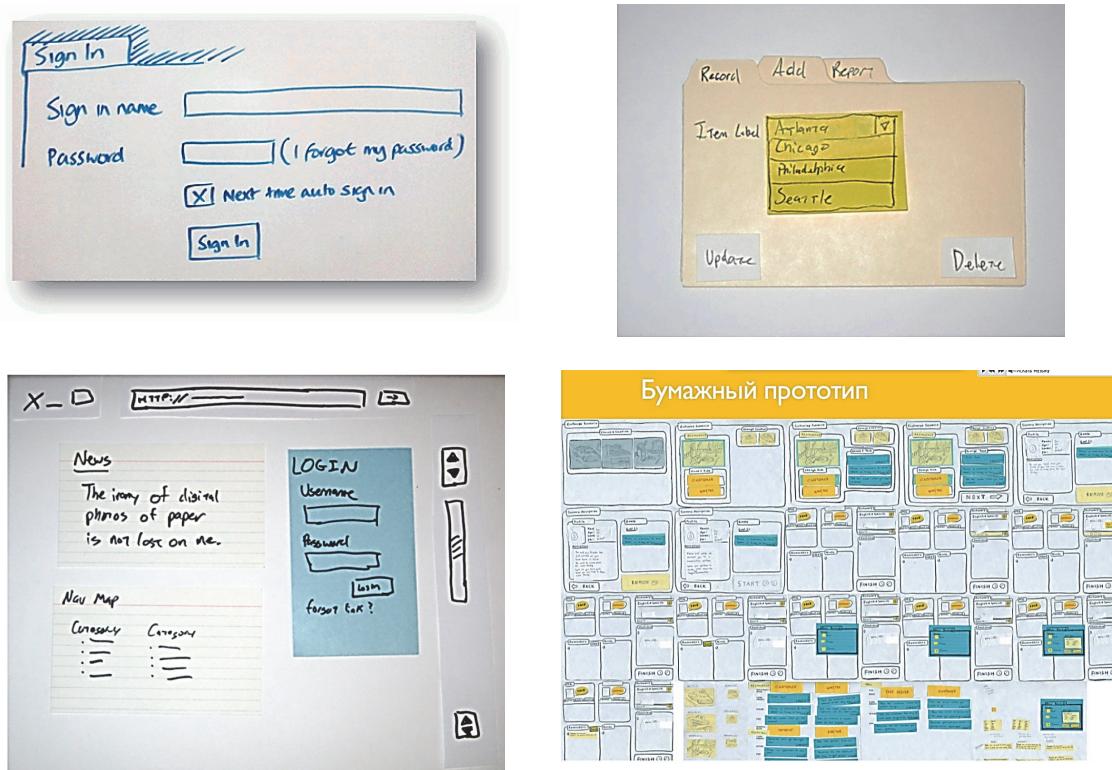


Рис. 6.1. Примеры бумажных прототипов

Раскаровка. Следующий шаг – создание схемы для каждой страницы перехода. И здесь уже можно продолжать итерации, постепенно переходя от бумажных схем к цифровому прототипированию. Схема поможет быстро изучить возможные переходы на страницы / со страниц приложения. Скетчи позволяют «оживить» приложение и понять большее количество деталей и структуры программы.

Это является промежуточным этапом между электронным и бумажным вариантами прототипов. Создаются при помощи средств электронного офиса. Для этого точно так же рисуется интерфейс, но уже не на бумаге, а в какой-либо программе (например, комбинации Microsoft Visio и Microsoft PowerPoint). Данный вид решения определяется как статическая или пассивная раскаровка (рис. 6.2).

Рис. 6.2. Статическая (пассивная) раскаровка

Динамические прототипы. Дальнейшим развитием статической раскаровки является динамическая (интерактивная) раскаровка с применением средств анимации и т. п. При этом каждый экран получает отдельный слайд, а результат нажатия кнопок имитируется переходами между ними. Интерактивная раскаровка представляет собой электронный прототип.

На этом этапе создается действующая модель пользовательского интерфейса. Сделать интерактивный прототип можно быстро и дешево, особенно если использовать инструменты быстрого прототипирования. Получить его можно уже на раннем этапе проектирования, не дожидаясь отрисовки визуального дизайна, а значит, и начать юзабилити-тестирование.

При помощи программных инструментов (вроде Axure RP Pro, Microsoft Expression Blend или плагина к MS Visio Intuitect) вы сможете точно показать, как интерактивные части сайта или приложения будут выглядеть для ваших пользователей (рис. 6.2).

С этой версией прототипа можно тестировать значительно более сложное взаимодействие человека с системой, нежели с бумажной. С другой стороны, исправление найденных ошибок значительно более трудоемко.

Существуют следующие подходы к прототипированию: черно-белый прототип на основе схем страниц (wireframe) и цветной прототип, основанный на принятом заказчиком визуальном дизайне системы.

Однако гораздо проще общаться с заказчиком, пользователями и разработчиками, имея на руках модель интерфейса, которая выглядит максимально похоже на финальный результат (рис. 6.3).

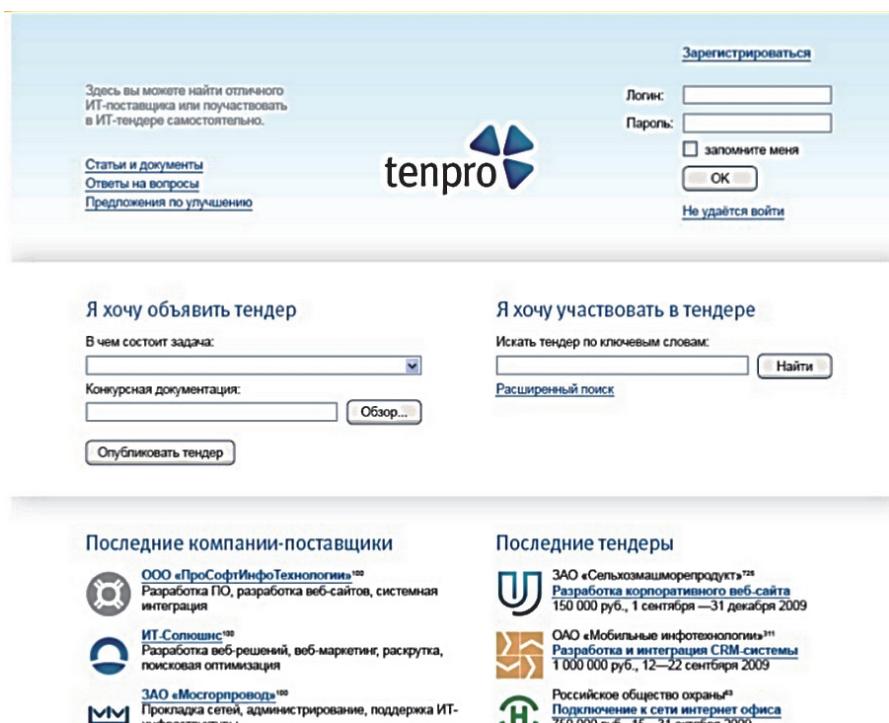


Рис. 6.3. Пример цветного интерактивного прототипа

Основные компоновочные блоки макета страницы

К основным блокам можно отнести навигационные, информационные, сервисные, дизайнерские, рекламные (рис. 6.4).

Навигационный блок. Дизайн навигации кажется простым делом: нужно расставить на каждой странице ссылки, чтобы пользователь смог ориентироваться на ней. Однако если заглянуть чуть глубже, трудности навигационного дизайна станут очевидными.

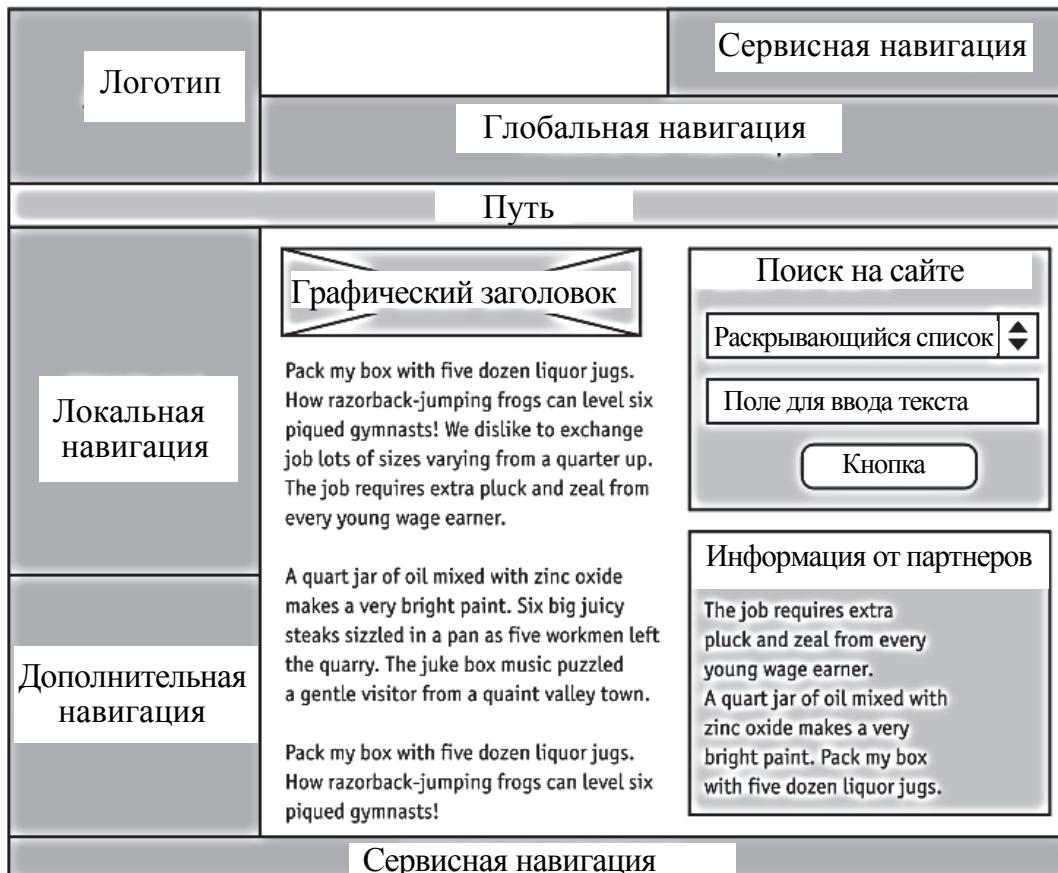


Рис. 6.4. Основные компоновочные блоки макета страницы

Дизайн навигации должен одновременно решать три задачи:

– *предоставлять пользователям способ попасть из одной точки страницы в другую.* Поскольку во многих случаях связать каждую страницу со всеми остальными невозможно (а если и возможно, то из общих соображений неразумно), приходится подбирать навигационные элементы так, чтобы они упрощали реальные передвижения пользователя; в числе прочего это подразумевает, что ссылки должны быть рабочими;

– отражать взаимоотношения между внутренними элементами навигации. Просто предоставить список ссылок недостаточно. Как

эти ссылки соотносятся друг с другом? Являются ли одни более важными, чем другие? Какая между ними разница? Эта информация необходима пользователю для понимания того, какой у него есть выбор;

– отражать связь между содержательной стороной элементов навигации и страницей, которая находится перед глазами пользователя. Какое отношение имеет вся эта куча ссылок к странице, на которую я сейчас смотрю? Эта информация поможет пользователю понять, какой выбор ему следует сделать, чтобы наилучшим образом достичь своей цели или решить стоящую перед ним задачу.

Рассмотрим *навигационные блоки*.

1. «На главную страницу». Данный блок представляет собой гиперссылку, оформленную в виде логотипа либо названия компании. Он чаще всего ведет пользователя на главную страницу. Блок располагается обычно в левой верхней части страниц и должен быть повторен на всех страницах сайта, но на небольших персональных сайтах может и вовсе отсутствовать.

2. Поиск и быстрый переход. Несмотря на то что эти функции разные, их можно объединить в один блок по их внешней схожести и некоторой схожести по сути. Оба варианта представляют собой поле ввода (редактируемое или нет) с кнопкой выполнения введенного запроса. Наиболее часто блоки поиска и быстрого перехода располагаются в верхней правой части страницы. Они важны для навигации, поэтому «прятать» их от посетителя не рекомендуется, особенно если на страницах много информации.

3. Горизонтальное меню. Один из самых главных блоков на странице. Под блоком горизонтального меню понимается список гиперссылок, ведущих к основным разделам сайта. Гиперссылки располагаются на одной горизонтальной линии и могут быть оформлены и как обычный текст (меню), и как текст в виде изображения, и как символ (домик, корзина, конверт), и как вкладки. Они могут быть любого цвета и размера. Горизонтальное меню бывает двухуровневым.

4. Вертикальное меню. Блок вертикального меню по смыслу соответствует горизонтальному. В силу привычности данный блок располагают по левому краю страницы, но он может находиться и справа, и с обеих сторон одновременно. Этот блок бывает как

статичным, так и с выпадающим меню или раскрывающейся древовидной структурой.

5. Вторичная навигация. Визуально представляет собой усеченный вариант горизонтального или вертикального меню. С точки зрения информационного наполнения сайта этот блок не является главным и чаще всего содержит сведения о компании-владельце сайта.

6. Навигация по выборке. Данный блок необходим при работе с выборкой некоторых объектов (изображения, ссылки, результаты поиска), которую невозможно отобразить целиком. Для перемещения между частями выборки используется специальная навигация. В целях удобства пользователя должен быть визуально выделен текущий фрагмент, а также должны отображаться соседние элементы. Если не хватает места для отображения всех порций, то следует использовать стрелки или надписи Следующая / Предыдущая.

7. Авторизация. Этот блок, наверное, самый предсказуемый и понятный. Он располагается в том месте, где пользователь должен идентифицировать себя, чтобы получить доступ к определенной информации, или чтобы система могла распознать его и основывать свою работу на ранее введенных данных (например, для очередного заказа в интернет-магазине).

8. «Подвал» (текстовые гиперссылки). «Подвал» используется для текстовых гиперссылок на основные разделы сайта. Он должен быстро загружаться и быть доступным в момент, когда фокус внимания находится внизу страницы. Достаточно удобно: закончил работу с этой информацией и готов перейти к следующей. Часто под «подвалом» понимают все, что находится внизу страницы, а не только список основных разделов.

9. Навигационная строка. Этот блок – последовательность гиперссылок, определяющая путь посетителя к текущей странице. Навигационная строка показывает, где был пользователь, дает возможность быстро вернуться на один и более шагов назад. Если список элементов становится слишком длинным, то можно отображать только несколько первых и последних ссылок, а промежуточные заменить на многоточие.

Информационный блок. Часто обновляемую информацию удобнее всего размещать в виде информационных блоков. Такой способ публикации облегчает задачу добавления и обновления

информации за счет структурирования данных, а также за счет использования возможностей импорта и экспорта.

Рассмотрим *информационные блоки*.

1. Содержание. Данный блок состоит из заголовка и одного или нескольких блоков с основным содержимым страницы.

2. Текущая информация. Этот блок используется при необходимости сообщить посетителю какие-то краткие сведения, полезные для его работы. Как правило, такие блоки маленьких размеров и располагаются относительно произвольно, часто внутри других блоков.

3. «Раздел». Данный блок достаточно обособленный и содержит информацию, несколько отличающуюся от основного содержимого страницы. Это может быть анонс, новость, краткое описание услуг компании, опрос и т. д. В блоке должно быть название и содержание раздела, а может еще и гиперссылка, позволяющая перейти к полному содержанию, например, смотреть новость полностью. Содержимым блока может быть и текст, и набор гиперссылок, и изображение.

4. Изображения (галерея). Это набор изображений, совмещенный с блоком навигации по выборке, иногда может быть только одно изображение. Изображения не всегда загружаются достаточно быстро. Однако по непонятной причине на многих сайтах этот блок состоит из одного большого изображения, вместо нескольких маленьких, которые можно просмотреть в большем размере дополнительным шагом.

Лучше делать картинки двух размеров, т. е. делать два вида блока – для множества картинок и для одной. Если пользователь не хочет внимательно рассматривать каждую картинку из шести представленных, то он дождется загрузки шести маленьких изображений, а не вынужден будет ждать загрузки страницы с полномасштабным изображением, что в результате вызовет его раздражение.

Информационный дизайн играет важную роль и в тех задачах, где интерфейс должен не только получать какие-то сведения от пользователя, но и передавать ему информацию. Классическая задача информационного дизайна при создании успешных интерфейсов – *сообщения об ошибках*; еще одна – *предоставление инструкций пользователю* (задача непростая уже хотя бы потому, что труднее всего заставить их прочитать эти инструкции).

Каждый раз, когда система должна облегчить пользователю работу с интерфейсом путем предоставления информации (например, когда пользователь только начал работать с сайтом или совершил ошибку), это задача информационного дизайна.

Сервисный блок. Блок данных, в котором отображаются протокольные блоки.

Рассмотрим *сервисные блоки*.

1. «Выбор языка». Этот блок необходим, если ваш сайт поддерживает несколько языков. Часто его располагают рядом со вторичной навигацией, а наименование языка пишут на нем же самом.

2. «Пустой блок». Этот блок представляет собой пустое место между другими и может служить для отделения одного блока от другого, для рекламного баннера. Чаще всего он появляется сам по себе в процессе верстки (например, пустое место под горизонтальным меню, когда меню уже закончилось, а основное содержание страницы располагается ниже).

3. «Версия для печати». Наверное, наименьший по размерам блок, он инициирует вызов текущей страницы и оптимизирует ее для отправки на принтер средствами браузера. Как правило, он располагается вверху или внизу основного содержимого страницы и совмещен с пиктограммой принтера.

Дизайнерский блок. Изображение, созданное дизайнером для украшения сайта и не являющееся одним из основных элементов содержания, может быть использовано и в качестве фона для иных блоков, например, для блока «Название и слоган» или блока текущей информации.

Рекламный блок. Несмотря на то, что такой блок напоминает «Раздел», он может выглядеть как угодно: тут все зависит от фантазии разработчика. Это может быть и мерцающий баннер на половину экрана, и маленькая «гиперссылка-завлекалка», и имитация кнопки, например, «Рейтингуется SpyLog» или «Участник Rambler Top 100». Реклама – она и есть реклама – всегда стремится к разнообразию.

К рекламным блокам можно отнести следующие.

1. «Название и слоган». Этот блок содержит название компании или самого сайта и, возможно, слоган вроде «Уважаем классику, ценим новое». Этот блок оформляется чаще всего крупным

шрифтом, иногда на фоновом рисунке или вместе с логотипом компании. Назначение его, думаю, понятно – это громкое заявление компании о себе.

2. «Копирайт». Блок описания авторских прав компании-владельца сайта, иногда со ссылкой на его создателя, например «Дизайн SuperWeb-Studio».

Пример компоновки. Вариантов компоновки блоков великое множество, что позволяет создавать уникальные и запоминающиеся web-сайты. Прежде всего следует решить, к какой цели вы стремитесь: к выражению собственных творческих способностей или к популярному и удобному информационному ресурсу.

На рис. 6.5 приведен пример сайта инструментов для страховых компаний. В Axure были созданы макеты каждой страницы. Обсуждение их с клиентом помогли понять бухгалтерию страховой компании и разработать лучшее решение в кратчайшие сроки.



Рис. 6.5. Пример сайта инструментов для страховых компаний

На рис. 6.6 показан пример сайта PowerMockup, в котором предусмотрены шаблоны и инструменты для строительных каркасов прямо в PowerPoint. С выпуском версии 3 PowerMockup (Wireframing надстройка для Microsoft PowerPoint) был запущен и обновленный веб-сайт продукта. При разработке сайта сначала произведен мозговой штурм макета идеи на бумаге, прежде чем приступить к выполнению более детальных каркасов в PowerPoint.

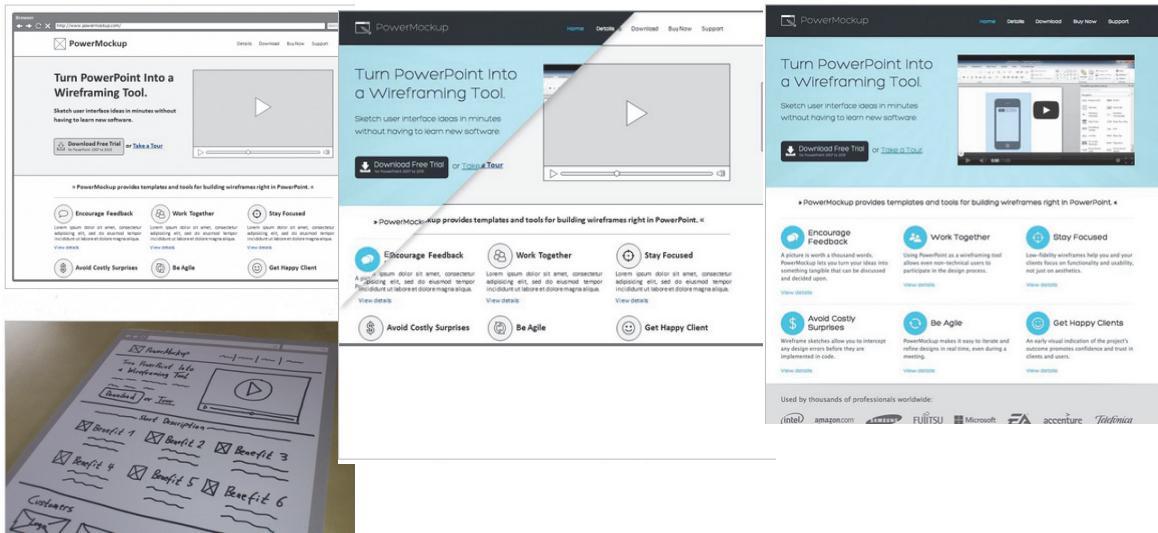


Рис. 6.6. Пример сайта PowerMockup

Проектирование каркасов UI для нового веб-сайта SkyPix Сент-Луис было первым важным шагом (рис. 6.7). Клиент хотел продемонстрировать много фотографий.

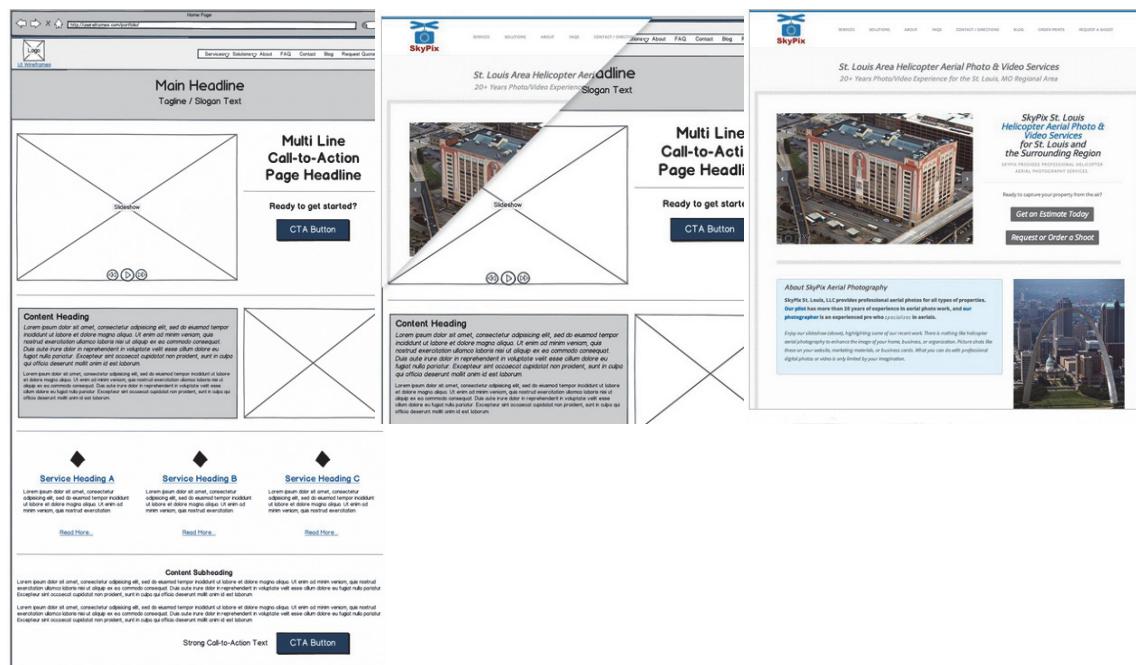


Рис. 6.7. Проектирование каркасов UI для нового веб-сайта SkyPix

Прфессиональные каркасные макеты помогли продемонстрировать расположение и размер этих фотографий, а также другие различные графические элементы. В результате клиент смог лучше представить себе готовый сайт до начала стадии проектирования.

Инструменты для создания прототипов

Существует много инструментов для создания прототипов и дизайна интерфейса (Axure RP Pro, Microsoft Expression Blend или плагин к MS Visio Intuitect). Рассмотрим некоторые из них.

Нужно определиться в самом начале, что в проекте является эталоном интерфейса – схемы страниц (wireframes), макеты дизайна или интерактивный прототип. Для простых задач можно использовать любое средство, чтобы рисовать. Для сложных лучше выбрать специализированный инструмент.

Axure RP Pro – ориентирована на создание веб-сайтов. Генерирует кликабельный HTML-документ и документацию в формате MS Word.

Caretta GUI Design Studio – специализированное средство, позволяющее создавать интерфейсы в разных визуальных стилях, аннотации к ним, раскадровки и т. п. Можно экспортить прототип.

Balsamino Mockups – подходит для быстрого создания макетов интерфейсов. Прототипы выглядят рисованными.

Adobe InDesign – изначально рассчитан для верстки полиграфических материалов, тем не менее подходит и для отрисовки прототипов. На выходе: кликабельный PDF.

Adobe Fireworks – специально предназначен для прототипирования интерфейсов. На выходе – кликабельный PDF или HTML.

Adobe Dreamweaver – предназначен для HTML-верстки. На выходе – кликабельный HTML.

Microsoft Expression Blend. Интегрируется с Visual Studio. Прототип можно преобразовать в конечный продукт. Использует технологии Silverlight или WPF.

Microsoft Silverlight – это программная платформа, включающая в себя модуль для браузера, который позволяет запускать приложения, содержащие анимацию, векторную графику и аудио-видеоролики, что характерно для RIA (Rich Internet application).

WPF (Windows Presentation Foundation) – это система, предназначенная для построения клиентских приложений операционной системы Windows. Она содержит визуально привлекательные возможности для пользователя, среди которых графическая подсистема .Net Framework.

Сейчас лучшей технологией прототипирования является сначала рисование прототипа на бумаге, а затем финализация прототипа в Adobe InDesign (он легче в изучении, чем специализированные

средства разработки, к тому же, хоть и уступая им в скорости со-зания первой версии, лидирует в скорости модификаций). Наконец, InDesign, в отличие от средств разработки вроде Adobe Dreamweaver или Microsoft Visual Studio, универсален – в нем можно запрототипировать любой графический интерфейс, будь то интерфейс программы или сайта.

Для редактирования XAML лучше всего применять специальный пакет, Microsoft Expression Blend или просто Blend. Редактирование внутри него происходит в наглядной, визуальной форме, т. е. проектировщик может создавать элементы, перемещать их по экрану и описывать поведение.

Тестирование и модификация прототипа

Тестирование интерфейса является исключительно важной задачей при проектировании интерфейса. Начальный этап тестирования связан с разработкой прототипа интерфейса. На этом этапе проектировщик использует имеющиеся результаты проектирования: общую схему приложения, планы отдельных экранных форм, глоссарий. Эти результаты сводятся воедино в общую схему, которую необходимо проверить по сформулированным ранее сценариям. Целью такой проверки является выявление несоответствия последовательности действий, описанной в сценарии, и структуры полной схемы. Обнаруженные несоответствия должны быть устранены за счет модификации экранных форм и/или корректировки общей схемы приложения.

Проверка качества интерфейса обычно непроблематична. Все, что для этого нужно, так это несколько пользователей средней квалификации, никогда не видевших тестируемой системы, плюс разработанный прототип.

В литературе часто встречается мнение, что тестированием можно решить чуть ли не все проблемы интерфейса. Утверждение это сомнительно. Тестированием можно определить слабые места интерфейса, но почти невозможно обнаружить сильные, поскольку они пользователями просто не замечаются, и совсем уж невозможно определить новые способы улучшения. Происходит это из-за того, что субъекты тестирования не обладают всей необходимой информацией о системе, ничего не знают о проектировании интерфейсов, их мотивация существенно отличается от необходимой – вместо того, чтобы стремиться сделать хороший интерфейс, они стремятся оставить в этом интерфейсе свой след.

Тем не менее нужно принимать во внимание потребности пользователей во избежание следующих ситуаций:

1) дизайнер интерфейса знает о предметной области меньше, нежели будущие пользователи. В таких случаях система оказывается неспособна решать задачи, о которых дизайнер ничего не знал;

2) уровень «компьютерной грамотности» дизайнера оказывается выше уровня аудитории. В этом случае дизайнер выбирает решения, которые пользователи не могут понять.

Замечено, что опытные пользователи (к которым относятся дизайнеры) создают значительно менее работоспособные иерархии меню, нежели пользователи начинающие. Если переоценка способностей реальных пользователей и незнание предметной области совпадают, результат бывает еще хуже.

Процесс тестирования состоит из *постановки задачи, собственно тестирования, модификации*.

Постановка задачи. Успех тестирования зависит от правильности и корректности постановки задачи тестирования. Тестирование может быть направлено на подтверждение:

- производительности действий при использовании продукта. Оценивается по длительности выполнения задач (тестовых заданий) пользователем. Эффективный продукт позволяет увеличить число пользователей, успешно выполняющих задание в течение ограниченного времени;

- полезности продукта. Продукт является полезным, если позволяет снизить количество человеческих ошибок. Полезный продукт позволяет увеличить число пользователей, способных успешно выполнить задание;

- простоты обучения. Оценивается по времени тренинга, необходимого для достижения пользователем определенного уровня владения продуктом;

- субъективной оценки пользователей. Пользователи оценивают свое отношение к продукту по десятибалльной шкале. Продукт можно считать успешным, если определенная часть пользователей оценила его на 8 и выше баллов.

В начале этого этапа формулируют задачи тестирования. Например, оценить производительность действий при использовании продукта.

Тестовое задание представляет собой задачи для пользователей и включает последовательность действий, записанных в сценарии,

но в отличие от него содержит конкретные значения данных, с которыми оперирует пользователь.

Проверка соответствия тестового задания и последовательности перехода между слайдами выполняется на готовом ролике. Выявленные несоответствия могут потребовать изменения навигационной системы приложения.

Когда сформулированы необходимые тестовые задания, выполнены проверки и требуемые корректизы, приступают к тестированию с привлечением пользователей из целевой аудитории.

Собственно тестирование. Тестирование проводится на представителях пользовательской аудитории, ранее не знакомых с разрабатываемым продуктом. Уровень опыта тестируемых пользователей должен соответствовать уровню, определенному в профилях конечных пользователей. Считается, что тестирование на одном пользователе позволяет выявить примерно 60% ошибок. Поэтому число тестируемых пользователей, необходимых для проведения одного сеанса, зависит от сложности и объема проектируемого продукта. Для «средних» приложений достаточно 4–8 человек. В ходе тестирования категорически запрещено прерывать или смущать пользователя; нельзя внушать testирующему, что testируют его; желательно присутствие разработчиков приложения (программистов), но их роль в тестировании исключительно пассивная.

В качестве методов проведения тестирования могут быть использованы наиболее простые:

1) *наблюдение за пользователем.* Пользователю предъявляется тестовое задание, он его выполняет. Действия пользователя фиксируются. Этот метод эффективен при определении неоднозначности элементов интерфейса: любая неоднозначность, как правило, влечет за собой ошибку пользователя. Поскольку действия пользователя фиксируются, обнаружить ошибки при анализе тестов довольно легко. Кроме того, этот метод подходит для оценки производительности действий пользователя. Для этого необходимо при фиксировании действий замерять время, потребовавшееся пользователю на его выполнение;

2) *комментарии пользователя.* Как и при использовании предыдущего метода тестирования, пользователи выполняют тестовые задания. Действия пользователя также фиксируются, кроме того, фиксируются комментарии им своих действий. В дальней-

шем комментарии позволяют выявить недостатки реализации конкретных элементов интерфейса – неудачное расположение элементов управления, плохая навигация и т. д. Этот метод можно использовать для оценки полезности продукта, простоты обучения работы с ним, степени субъективного удовлетворения. Следует отметить, что метод является «нестабильным»: результаты его использования зависят от личных качеств тестируемого пользователя – его разговорчивости, умения последовательно и внятно излагать свои мысли;

3) качество восприятия. Пользователю предъявляется тестовое задание, а через некоторое время после его выполнения тот должен воспроизвести экранные формы (бумажный вариант), с которыми он работал. Результат воспроизведения сравнивают с оригиналом. Идея теста заключается в следующем: из-за ограничений объема кратковременной памяти количество элементов экранных форм, которые запоминает тестируемый, не может быть выше порога запоминания. Пользователь запоминает только то, что считает наиболее актуальным в процессе работы. Следовательно, при повторном выполнении задания пользователю, знающему расположение необходимых для этого элементов интерфейса, будет проще. Таким образом, данный метод позволяет оценить простоту обучения работе с продуктом, и, кроме того, степень субъективной удовлетворенности пользователей.

Тестирование проводится путем наблюдения за пользователем с фиксированием длительности выполнения действий. Критерий оценки можно сформулировать как выполнение контрольного тестового задания в течение 3 минут 75% тестируемых пользователей после тренинга – выполнения пяти различных сценариев.

Все результаты тестирования обобщаются с тем, чтобы сформулировать рекомендации относительно модификации прототипа интерфейса. Модификации могут быть связаны с изменениями содержимого экранных форм, элементов навигационной системы, терминологии и даже функциональности тех или иных элементов интерфейса.

Модификация. Тестирование само по себе имеет существенный недостаток: если тестирование проблем не выявило, получается, что оно было проведено зря; если выявило, придется проблемы решать, что тоже существенная работа.

Следует отметить, что выявление в ходе тестирования различных ошибок и несоответствий неизбежно. Это является одной из причин того, что тестирование нельзя переносить на окончание проекта, когда вносить модификации уже нет возможности из-за истечения сроков работ.

Разработка пользовательского интерфейса приложения представляет собой итеративный процесс. Каждая итерация связана с отдельным этапом проектирования, созданием прототипа по его результатам, тестированием прототипа и его модификацией. Разработчик должен прилагать особые усилия, чтобы уменьшить число итераций. Тестерам дается задание, они его выполняют, после чего результаты анализируются.

Именно поэтому тестирование бессознательно переносят на самое окончание проекта, когда что-либо исправлять уже поздно. В результате тестирование показывает, что проект сделан плохо, что никому не нравится, включая его создателя, после чего результаты проверки прячутся в дальний ящик, в то время как сама по себе идея тестирования совсем иная. В самом начале работы, когда только создан прототип будущей системы, он тестируется, после чего найденные ошибки исправляются, а затем прототип тестируется опять. При этом опытность дизайнера проявляется исключительно в уменьшении количества итераций. Соответственно, тестирование должно идти параллельно со всеми остальными операциями.

Глава 7

ВИЗУАЛЬНАЯ КУЛЬТУРА ДИЗАЙНА ИНТЕРФЕЙСА

Дизайн – это сознательные и интуитивные усилия по созданию значимого порядка.

Для успешного проектирования цифровых интерактивных продуктов необходимо помнить о сложном поведении, которое эти продукты демонстрируют. Однако силы, вложенные в разработку модели поведения программного продукта, будут потрачены впустую, если вы не сумеете должным образом донести до пользователей принципы этого поведения.

При проектировании интерфейса необходимо учитывать внешний вид (форму) и информационное наполнение (содержание, контент) разрабатываемого интерфейса. Поэтому следует сочетать подходы различных дисциплин проектирования:

- проектирование взаимодействия, основное внимание на поведение;
- информационная архитектура, занимается структурированием содержания;
- графический дизайн, отвечает за внешний вид интерфейса и его функционала.

Визуальный дизайн интерфейсов выполняет две задачи – задачи *графического дизайна* и задачи *визуального информационного дизайна*.

Графические дизайнеры обычно очень хорошо разбираются в визуальных аспектах и хуже представляют себе понятия, лежащие в основе поведения программного продукта и взаимодействия с ним. Они способны создавать красивую и адекватную внешность интерфейсов, а кроме того – привносить фирменный стиль во внешний вид и поведение программного продукта. Для таких специалистов дизайн или проектирование интерфейса есть в первую очередь тон, стиль, композиция, которые являются атрибутами бренда, во вторую очередь – прозрачность и понятность информации и лишь затем – передача информации о поведении посредством ожидаемого назначения.

Визуально-информационные дизайнеры работают над визуализацией данных, содержимого и средств навигации.

Им необходимы некоторые навыки, которые присущи графическим дизайнерам, но они должны еще обладать глубоким пониманием и правильным восприятием роли поведения. Их усилия в значительной степени сосредоточены на организационных аспектах проектирования. В центре их внимания находится соответствие между визуальной структурой интерфейса с одной стороны и логической структурой пользовательской ментальной модели и поведения программы – с другой. Кроме того, их заботит вопрос о том, как сообщать пользователю о состояниях программы и что делать с когнитивными аспектами пользовательского восприятия функции.

Визуальный дизайн интерфейсов – очень нужная и уникальная дисциплина. Она способна серьезно повлиять на эффективность и привлекательность продукта, но для полной реализации этого потенциала нужно не откладывать визуальный дизайн на потом, а сделать его одним из основных инструментов удовлетворения потребностей пользователей.

Визуальный дизайн – это язык, в котором информация передается не только словами, но и образами. Чтобы эффективно общаться на языке визуального дизайна, важно правильно заложить основы. Главное: *мы видим предметы как целое, а не сумму частей этого целого*. То же самое происходит с каждым отдельным дизайном сайта: он никогда не воспринимается путем идентификации его отдельных частей (заголовка, навигации, содержимого, кнопок, таблиц и т. д.), – дизайн с первого взгляда воспринимается как целое. И еще одно простое правило, которое всегда следует держать в уме: «Если вы видите модель, и она вам нравится, а затем вы решаете доработать детали, тогда знайте, что это хороший дизайн. Однако, если вы сначала начинаете *дорабатывать* модель для того, чтобы она вам понравилась, тогда знайте, что это плохой дизайн. Не тратьте зря время на попытки его доделать, просто измените его в целом».

Композиция, вид и ощущение приложения

Все элементы дизайна – от типографики до иконографии – относятся к общей композиции. Каждый экранный интерфейс или веб-страница вызывают у посетителя внутренние эмоции еще до

того, как он успевает заметить мелкие детали. Именно поэтому, разрабатывая дизайн, необходимо уделять особенное внимание композиционным методам проектирования.

Композиция (от лат. *compositio*) – сложение, сопоставление, приведение частей в единство. Под термином понимается объединение отдельных элементов изображения в единое художественное целое, выразительно раскрывающее идею произведения. Это древнее правило, известное архитекторам еще в древней Греции, используется и опытными веб-дизайнерами сегодня. Как сказал кто-то из великих, создать гениальную картину – это положить в нужные места нужные краски.

Элементарные принципы дизайна композиции, цвета и иного точно так же применяются к компьютерному экрану, как к листу бумаги или холсту. Композиция не только влияет на ее эстетическую привлекательность, но также имеет огромное воздействие на применимость разрабатываемого приложения.

Композиция в веб-дизайне – это гармоничное расположение всех элементов сайта на экране таким образом, чтобы максимально раскрывалось их содержание.

Именно композиция является неким связующим звеном между реально существующей действительностью и внутренним отношением автора к ней.

Основные задачи композиции:

- выразительно изображать характерные черты объекта путем создания акцентов, цветовых и световых контрастов;
- достоверно отображать конкретное событие или сюжет;
- заинтересовать зрителя, привлечь его внимание изобразительными приемами.

Цель композиции – сделать так, чтобы все элементы дизайна воспринимались целостно и составляли единую картину. Для этого дизайнер помещает самые важные объекты в центр внимания пользователя. Добиться этого можно с помощью ракурса, цвета, контраста и других техник, о которых подробнее поговорим ниже.

Можно выделить так называемые три «закона» композиции:

- 1) целостность;
- 2) наличие доминанты;
- 3) уравновешенность.

Целостность – изображение или предмет целиком охватываются взглядом как единое целое, не распадаясь на самостоятельные

части. При этом целостность подразумевает не только единство частей изображения и взаимосвязь предметов, но и устранение лишних подробностей, которые отвлекают внимание от наиболее важных элементов изображения.

Наличие доминанты – подчиненность второстепенного главному. Объекты, с наибольшей силой характеризующие отображаемое явление, образуют идейно-композиционный центр, акцент на котором может быть сделан любыми выразительными средствами. Остальные элементы, занимающие периферийное положение и создающие окружающую среду, способствуют усилинию главной идеи. Центр композиции не обязательно связан с геометрическим центром изображения.

Уравновешенность определяется соотношением площадей цветовых и тональных масс, расположенных в правой и левой части изображения по горизонтали или по вертикали.

В целом же, удачная композиция должна быть естественной и служить задачам дизайна: подчеркнуть какой-то блок на экране, подвести пользователя к нужному действию, перевести его внимание с одного блока на другой и т. д. То есть она должна быть не только визуально привлекательной, но и эффективной в функциональном плане. Плохая композиция сразу обращает на себя внимание, режет глаз и вызывает раздражение. Хорошая же условно остается в тени, человек не обращает на дизайн внимания, а просто смотрит на содержание.

Основные элементы композиции в веб-дизайне

Дизайн интерфейсов сводится к вопросу о том, как оформить и расположить визуальные элементы таким образом, чтобы внятно отразить поведение и представить информацию. Каждый элемент визуальной композиции (линия, форма, фигура) имеет ряд свойств, и сочетание этих свойств придает элементу смысла. Пользователь получает возможность разобраться в интерфейсе благодаря различным способам приложения этих свойств к каждому из элементов интерфейса. В тех случаях, когда два объекта обладают общими свойствами, пользователь предположит, что эти объекты связаны или похожи. Когда пользователи видят, что свойства отличаются, они предполагают, что объекты не связаны.

Чтобы создать полезный и привлекательный пользовательский интерфейс, нужно учитывать визуальные свойства каждого элемента или группы элементов и тщательно поработать с каждым из этих свойств.

Цвет

Цвет является одним из наиболее очевидных элементов дизайна, как для пользователя, так и для дизайнера. Он может быть изолирован в качестве фона или применяться к другим элементам, таким как линии, фигуры, или типографике. Каждый цвет говорит что-то свое, и их комбинации могут изменить это впечатление дальше. С помощью цвета можно привлечь внимание к определенной части сайта.

Существуют основные цвета (красный, желтый, синий), вторичные цвета (оранжевый, зеленый, фиолетовый) и дополнительные цвета (противоположные цвета на цветовом круге).

Цвета в интерфейсах обычно подбираются исходя из цветовых схем. Цветовые схемы существуют уже очень давно. Хотя цвета web отличаются от цветов печати, принципы остаются такими же. Одним из инструментов в сети, который позволяет быстро и легко определять цветовые схемы, и даже определить, обеспечивают ли выбранные цвета достаточную контрастность для слабовидящих или не различающих цвета пользователей, является ColorSchemeGenerator II.

Существуют следующие цветовые схемы.

Монохроматические – схема соответствует одному цвету и всем его оттенкам, тональностям и теням. Монохроматические цвета хорошо сочетаются, создавая успокаивающий эффект. Хотя эта схема самая простая для использования, она не вызывает большого энтузиазма при проектировании web у многих дизайнеров.

Аналоговые цветовые схемы (рис. 7.1) создаются из смежных цветов. Один цвет используется как доминирующий, в то время как другие используются для обогащения схемы.

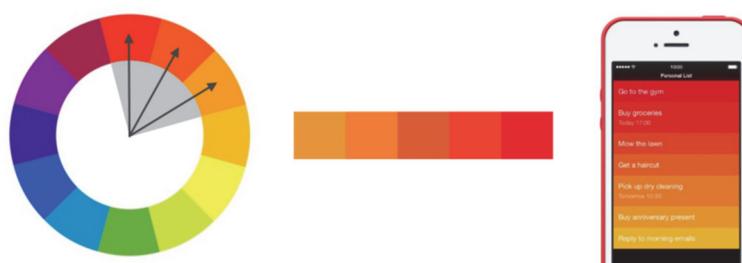


Рис. 7.1. Аналоговая цветовая схема

Дополнительные цветовые схемы. В основе такой схемы лежат только два цвета, которые сильно контрастируют. Эта схема используется для привлечения внимания пользователя. При использовании дополнительной схемы важно выбрать доминирующий цвет, а дополнительный цвет использовать для акцентов. Например, когда человеческий глаз видит объект, полный различных оттенков зелени, немного красного цвета очень хорошо выделяется. При выборе одного цвета и его противоположного используют также все оттенки, тональности и тени обоих цветов (рис. 7.2).

Дополнительные цветовые схемы хорошо подходят для использования на web-сайтах, так как они содержат также *теплые* и *холодные* цвета (рис. 7.3).

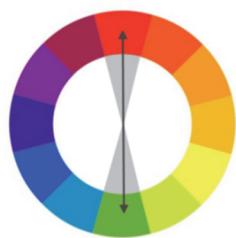


Рис. 7.2. Дополнительная цветовая схема



Рис. 7.3. Тёплые и холодные цвета

Использование этих цветов обеспечивает контраст, и можно легко запомнить, какие цвета являются теплыми, а какие цвета являются холодными.

Теплыми являются те цвета, которые будут напоминать вам о лете, солнце или огне. Они составляют цвета от фиолетовых до желтых.

Холодные цвета могут напоминать весну, лед или воду. Эти цвета проистекают от желто-зеленого и до фиолетового.

Если вы заметили, как цвета работают на круге, то скоро обнаружите, что не можете выбирать один цвет, не выбрав его противоположный по температуре.

Поэтому при выборе горячего красного цвета, противоположным будет холодный зеленый. Или, если вы выбираете холодный сине-зеленый, вы закончите острым красно-оранжевым на другой стороне.

Триадическая цветовая схема создается при выборе одного цвета и добавлении затем двух других цветов, которые должны лежать на одинаковом расстоянии друг от друга на цветовом круге (рис. 7.4).

Триадическая цветовая схема содержит также теплые и холодные цвета, но одна температура будет преобладать. Обычно температура, которая будет преобладать над другими, выбирается для переднего плана.



Рис. 7.4. Сайт Puzzle Pirates – пример хорошей триадической цветовой схемы

Здесь используется основная красно-сине-желтая схема, и эта основная схема отлично подходит для сайта детской игры. Синий цвет является преобладающим, а красные и желтые цвета используются как акценты и направляют движение глаз по странице.

Тетрадические цветовые схемы. Чем больше цветов выбирается, тем более сложной будет цветовая схема.

Однако один из приемов состоит в следующем: выбрать оттенок, тональность или тень и придерживаться выбранного повсеместно, а не смешивать чистые цвета и их оттенки, тональности и тени. Этот метод хорошо работает с четырехцветной тетрадической схемой.

Тетрадическая цветовая схема (рис. 7.5) похожа на дополнительную схему, только используется две пары дополнительных цветов, расположенных на равном расстоянии друг от друга.

Естественная цветовая схема. Лучшие цветовые комбинации можно встретить в природе (рис. 7.6). Почему? Потому что эти схемы кажутся естественными для глаз. Просто сделайте снимок красивого момента и создайте на его основе свою цветовую схему.



Рис. 7.5. Пример тетрадической цветовой схемы – сайт JaneGoodallInstitute

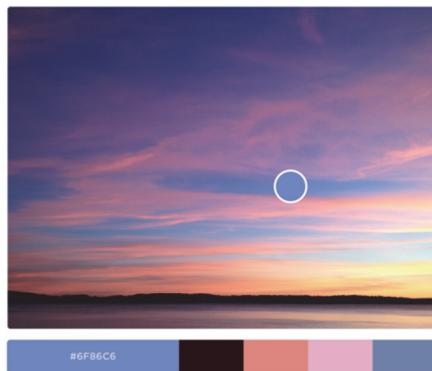


Рис. 7.6. Пример естественной цветовой схемы

Гармония цвета

Цветовая гармония – это сочетание отдельных цветов или цветовых множеств, образующих органическое целое и вызывающих эстетическое переживание. Цветовая гармония в дизайне представляет собой определенное сочетание цветов с учетом всех их основных характеристик: цветового тона, светлоты, насыщенности, формы, размеров, занимаемых этими цветами на плоскости, их взаимного расположения в пространстве, которое приводит к цветовому единству и наиболее благоприятно эстетически воздействует на человека.

При разработке нового продукта часто бывает сложно определить цветовую схему из-за бесконечного количества возможных комбинаций цветов. Существуют определенные правила, которые позволяют выбрать гармоничную цветовую палитру.

Ограничьте количество цветов. Применение цвета к дизайн-проекту имеет много общего с балансом. Чем больше цветов вы используете, тем труднее достичь этого баланса. Вы добьетесь лучших результатов, если будете придерживаться максимум трех основных цветов в вашей цветовой гамме.

Используйте цветовую гамму, наиболее приятную для глаз. Чтобы найти вдохновение, нам нужно всего лишь осмотреться вокруг. Если вы видите особенно красивый или яркий цвет в повседневной жизни, попробуйте создать схему вокруг него.

Пытайтесь следовать правилу 60–30–10. Это вечное правило декорирования, которое может помочь вам легко составить цветовую схему. Пропорции 60% + 30% + 10% предназначены для обеспечения баланса цветов, используемых в любом пространстве. Эта концепция невероятно проста в использовании: 60% – ваш доминирующий оттенок, 30% – вторичный цвет, а 10% – цвет акцента (рис. 7.8).



Рис. 7.7. Правило 60–30–10

Идея состоит в том, что вторичный цвет поддерживает основной цвет, но они достаточно отличаются, чтобы можно было разделять их. Ваши 10% – цвет акцента. Это может быть цвет для призыва к действию или другого элемента, который вы хотите выделить.

Сначала дизайн в оттенках серого. Можно сначала проектировать все в оттенках серого. Проектирование в оттенках серого перед добавлением цвета заставляет вас сосредоточиться на расположении элементов. Цвет добавляется в самом конце, и даже тогда, только с определенной целью – для акцентирования внимания (рис. 7.8).



Рис. 7.8. Макет с оттенками серого

Избегайте использования черного цвета. В реальной жизни чистый черный почти никогда не встречается. Все «черные» объекты вокруг нас имеют некоторое количество света, отражающегося от них, а это значит, что они не совсем черные, они темно-серые. Дороги не черные. Тени не черные. Когда вы помещаете чистый черный цвет рядом с набором тщательно подобранных цветов, черный все пересилит. Он выделяется, потому что это неестественно. В большинстве приложений, которые мы ежедневно используем, есть черные цвета, которые на самом деле не черные, а темные серые. Например, самый темный цвет на верхней панели Asos не # 000000, это # 242424 (рис. 7.9). Поэтому не забудьте добавить немного насыщенности в свой цвет.

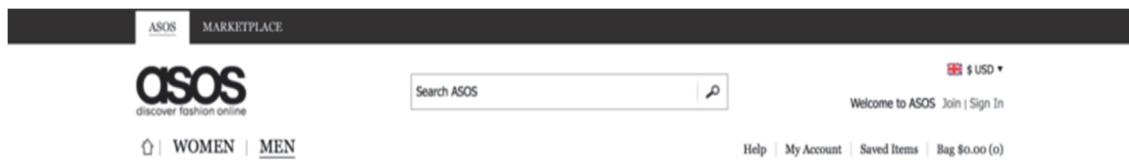


Рис. 7.9. Самый темный цвет у ASOS – не черный

Подчеркните важность, используя цветовой контраст. Цвет – инструмент, который может помочь направить взгляд. Чем больше вы хотите что-то выделить, тем больше вы должны полагаться на контрастные цвета. Как правило, высокий контраст – лучший выбор для важного контента или ключевых элементов.

Если вы хотите, чтобы пользователи увидели или щелкнули что-нибудь, сделайте это выделяющимся.

Используйте цвет для воздействия на эмоции пользователей. Известно, что цвета имеют присущие им значения и могут вызывать эмоции. Эти значения будут иметь прямое влияние на то, как ваши пользователи воспринимают ваш продукт. Когда вы выбираете цветовую палитру для своего приложения / сайта, вы не можете просто думать о том, как все выглядит – вы должны думать о том, как все *ощущается*. Цвета, которые вы выбираете, могут работать за или против идентичности бренда, которую вы пытаешьесь создать.

Типографика

Типографика является одной из самых важных основ дизайна. Текст может многое сказать (в буквальном смысле), а само оформление текста в дизайне способно сказать еще больше.

Любой шрифт сам по себе является законченным объектом дизайна. Практически любой хороший шрифт – результат кропотливой работы опытных дизайнеров. В профессиональной среде существует мнение, что разработка шрифтов – это верх дизайнера ремесла.

Многие исследования показывают, что множество гарнитур шрифта в интерфейсе может путать пользователя. С другой стороны, интерфейс, который использует везде только один шрифт, кажется безвкусным.

Потому при построении дизайна лучше использовать гарнитуру одного-трех шрифтов: *базовый шрифт* – основной шрифт материалов сайта и *акцидентный* – шрифт для заголовков.

В некоторых случаях вводятся дополнительные шрифты для меню и навигации, блоков выделения (важной информации, цитат, выносок), для мелкого текста, с целью повысить читабельность.

Дизайнер должен спланировать единую общую схему размеров отступов/заступов для всех элементов на сайте, иерархию заголовков и навигационных элементов (например, для древовидного меню или облака тегов). Она должна быть цельной и использоваться на всех страницах сайта.

Все последующее оформление информации на сайте должно строится на базе общей схемы (рис. 7.10).



Рис. 7.10. Шрифтовая схема простого корпоративного сайта

Существует четыре следующие основные типы шрифтов.

Шрифт с засечками (serif). Любая гарнитура шрифта, которая содержит завершающие штрихи, расширяющиеся или сужающиеся концы или имеет реально отсеченные окончания (включая прямоугольные засечки).

В ходе истории шрифт с засечками был выбран для печати основного текста, так как его легко читать на печатной странице.

Но интерфейс отличается от печати, и исследования показывают, что шрифты без засечек по удобочитаемости шрифта читать легче, чем с засечками.

Пример типа шрифта с засечками – Times New Roman.

Без засечек (sans-serif). Любая гарнитура шрифта, конечные штрихи которой не имеют никаких расширений, пересекающих штрихов или других украшений.

Его используют для основного текста для web. Пример шрифта без засечек – Verdana.

Рукописный или курсив. Эти шрифты обычно выглядят по большей части как написанные пером или кистью, они будут включать те, которые кажутся рукописными, даже хотя и не являются курсивом.

Одной из причин отказа от использования этого типа шрифта на web-странице, особенно в основном тексте, является трудность чтения в больших отрывках (подумайте о том, как трудно было бы читать написанное от руки письмо или манускрипт двенадцатого века, который можно увидеть в музее). Кроме того, не все браузеры выводят один и тот же шрифт, поэтому если вы решите использовать рукописный и курсивный шрифт, то он может быть

выведен как шрифт с засечками в каком-то другом браузере. Пример рукописного шрифта – Staccato.

Специальные шрифты, включая моноширинный. Единственным критерием моноширинного шрифта является единая фиксированная ширина всех символов.

Некоторые шрифты могут иметь фантастический внешний вид, и эти шрифты используются единственно с декоративной целью. Могут найти применение в небольшой области, такой как заголовки, или в рекламных объявлениях. Пример специального шрифта – Jokewood.

Моноширинные шрифты нашли свое применение на сайте, особенно при выводе программного кода (<http://www.lowing.org/fonts/>):

- TrueType;
- Clear, Dark, slashedzeros;
- AvailablewithfreePovraysoftware.

Существуют также и другие различия. Не все типы шрифта создаются одинаковыми, даже если создаются одного размера в пунктах. Размер в пунктах определяет высоту букв, и некоторые шрифты будут больше при 18 пунктах, чем другие. Расстояния между буквами и словами могут быть разные, или некоторые гарнитуры шрифтов, такие как Jokewood, не имеют букв нижнего регистра. Шрифты могут выглядеть по-разному во всех браузерах, так как различные браузеры остаются по сути несовместимыми. Причина этой проблемы состоит в том, что не все операционные системы поддерживают одни и те же шрифты. И даже если одни и те же шрифты, то вариант, толщина и другие факторы могут представляться по-разному в том или ином браузере.

Тип шрифта является одним из объектов, над которым вы имеете некоторый контроль, но только если вы сохраняете его как можно более простым.

Некоторые правила выбора шрифта

Как сделать типографику на вашем сайте более привлекательной и эффективной, и как сделать текст максимально удобным для чтения? Шрифт должен соответствовать содержанию текста и быть уместным. Также лучше всего выбирать хорошо читаемый и четкий шрифт. Для заголовков можно использовать и декоративные шрифты, но для основного текста лучше всего остановится на *Helvetica*, *Lato* или *Open Sans*.

Следующий пункт, на который стоит обратить внимание – это **размер шрифта**. Как правило, размер шрифта варьируется между 12 и 16 пикселями для основного текста. Но для мобильных устройств этот размер будет маловат. Тут придется сделать размер шрифта максимально разборчивым в зависимости от разрешения экрана. Он должен быть как минимум в два раза больше. Даже на небольшом экране у пользователя не должно быть проблем с чтением текста. Следует, однако, учитывать, что чем крупнее шрифт, тем меньше информации удается разместить на сайте.

Учитывая размер шрифта для основного текста на сайте, можно установить **иерархию размеров** для различных частей текста и соблюдать эту иерархию на всех страницах сайта. Например, основной размер текста 16 пикселей, тогда следующие размеры для блоков с текстом и крупных заголовков будут от 18 до 72.

Часто случается, что надо выделить слово или фразу в тексте, **чтобы акцентировать** на ней внимание. Сделать это можно с помощью курсива, выделения жирным, подчеркиванием, изменением размера шрифта, различными цветами. После того, как стиль выбран, его следует придерживаться на всех страницах сайта.

Если **длина строки** будет слишком короткой – текст становится непонятным, а если слишком длинной – то трудно читаемым. Обычно длина строки варьируется между 40–80 символами. Золотая середина, то что нам нужно, будет равна 65 символам.

Очень важный параметр, влияющий на читаемость текста и общую эстетику сайта – это **межстрочное расстояние**. Выбор размера межстрочного расстояния зависит от используемого шрифта, его размера, стиля и длины строки. Обычно размер межстрочного расстояния на 2–5 пунктов больше, чем размер шрифта.

Хорошее **выравнивание** текста способствует общему приятному впечатлению от сайта. Содержание сайта следует аккуратно разместить согласно логике восприятия текста и визуальному балансу, а страница должна быть разделена на системы столбцов, строк, меню, заголовков, боковые и нижние колонитулы.

Контраст и цвет. Цветовой акцент в тексте может сделать весь дизайн чрезвычайно привлекательным. Размещение двух цветов с низким значением контраста рядом друг с другом может сделать ваш текст очень трудным для чтения (рис. 7.11).



Рис. 7.11. Низкоконтрастный текст (может нарушить удобство использования приложений)

Следует особое внимание уделять соблюдению наивысшего контраста текста: располагать темный текст на светлом фоне, или наоборот – светлый текст на темном фоне (наивысший контраст дает отношение черный / белый). Однако для больших кусков текста лучше остановиться на традиционном сочетании темного цвета для шрифта и светлого для страницы. Экспериментировать с темным фоном и светлым текстом лучше с заголовками и подзаголовками.

Коэффициент контрастности показывает, как отличается цвет от другого цвета (обычно это 1 : 1 или 21 : 1). Чем выше разница между двумя числами в соотношении, тем больше разница в относительной яркости между цветами. W3C рекомендует следующие коэффициенты контрастности для текста и текста на изображении:

- маленький текст должен иметь коэффициент контрастности не менее 4,5 : 1 по отношению к фону;
- большой текст (при 14 pt жирный / 18 pt обычный) должен иметь коэффициент контрастности по крайней мере 3 : 1 по отношению к фону.

Используя инструмент Color Contrast Checker (рис. 7.12), вы можете проверить свои цветовые комбинации всего за несколько кликов (<https://webaim.org/resources/contrastchecker/>).

Color Contrast Checker

[Home](#) > [Resources](#) > Color Contrast Checker

Foreground color: # █ [lighten](#) | [darker](#)

Background color: # █ [lighten](#) | [darker](#)

Contrast Ratio: **8.59:1**

Рис. 7.12. Программа проверки цветового контраста от Webaim

Желательно избегать ярких фоновых заливок, использования графических обоев, затрудняющих восприятие текстовой информации. Если все-таки таковые используются, то следует применять двуцветные шрифты (черные символы с белой окантовкой).

Модульная сетка

Модульная сетка – один из самых мощных инструментов визуального дизайнера. После того как проектировщики взаимодействия определили общую инфраструктуру приложения и элементов его пользовательского интерфейса, дизайнеры интерфейса должны организовать композицию в структуру в виде сетки, которая будет должным образом подчеркивать важные элементы структуры и оставлять жизненное пространство для менее важных элементов и элементов более низкого уровня.

Сетка обеспечивает однородность и последовательность структуры композиции, единый каркас и схему расположения всех основных блоков и элементов.

Использование сетки в визуальном дизайне интерфейсов дает ряд преимуществ.

Удобство применения. Поскольку сетка делает расположение элементов единообразным, пользователи быстро приобретают навыки поиска нужных элементов в интерфейсе. Последовательность в расположении элементов и выборе расстояний между ними облегчает работу механизмов визуальной обработки в мозгу человека. Качественно спроектированная сетка упрощает восприятие экрана.

Эстетическая привлекательность. Аккуратно применяя сетку и выбирая подходящие соотношения между различными областями экрана, дизайнер может создать ощущение порядка, который удобен пользователям и стимулирует их работу с продуктом.

Эффективность. Создание сетки и включение ее в процесс на ранних этапах детализации проектных решений сокращает число итераций и действий по «доводке» интерфейса. Качественная и явно обозначенная сетка закладывает основу для легко модифицируемого и расширяемого дизайна, позволяя разработчикам находить хорошие композиционные решения.

При работе с построением макета дизайна необходимо использовать различные изображения, блоки текста, рекламы и дру-

гих элементов так, чтобы они подходили друг другу и смотрелись как единое целое. Правильным решением будет использование одной структуры сетки для всего шаблона сайта.

В самом простом понимании это решетка из ячеек, где одна из них взята за основную единицу измерения (модуль), а остальные равны или кратны ей. В модульной сетке нет места случайности, все должно быть математически точно.

Сетки бывают простые и сложные, гибкие в использовании и не очень. Как правило, сетка делит экран на несколько крупных горизонтальных и вертикальных областей. Качественно спроектированная сетка задействует понятие шага, то есть минимального расстояния между элементами. К примеру, если шаг сетки составляет четыре пикселя, все расстояния между элементами и группами должны быть кратны четырем.

В идеальном случае сетка должна задавать и пропорции различных областей экрана. Такие отношения обычно выражаются дробями. Среди распространенных дробей – прославленное «золотое сечение» (равное примерно 1,62), которое часто встречается в природе и считается особенно приятным для человеческого глаза; величина, обратная квадратному корню из двух (примерно 1 : 1,41), которая является основой международного стандарта размера бумаги (например, листа А4).

Виды сеток

Самый простой вид сетки – **блочная сетка**. В западной литературе ее также называют «manuscript grid». Представляет собой грубо размеченную область – блок.

Колоночная сетка – состоит только из вертикального членения на колонки. Объекты располагаются, опираясь на эту сетку. Какой-то блок может занять у вас две колонки, какой-то – четыре. Поскольку используется единый модуль и вы сохраняете отступы одинаковыми, дизайн всегда выглядит целостно и логично.

Модульная сетка характеризуется наличием как вертикального членения, так и горизонтального. То, что образуется на пересечениях, есть модуль.

Иерархическая сетка – состоит из блоков, размещенных интуитивно и не поддающихся закономерностям (рис. 7.13).

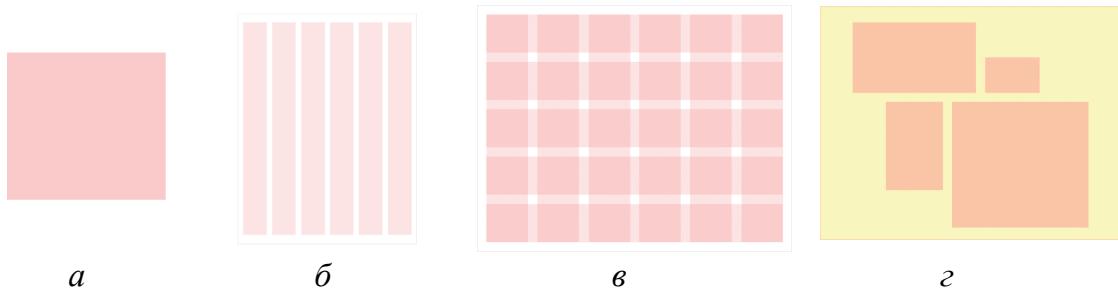


Рис. 7.13. Блочная (*а*), колоночная (*б*), модульная (*в*) и иерархическая (*г*) сетки

Сетки могут быть как простыми – с одинаковыми по размерам модулями, так и сложными, с нелинейными пропорциями у размеров модулей.

К сложным пропорциям относятся следующие.

1. **«Золотое сечение»** – также известное как «Божественная пропорция», «золотая спираль» и т. п. Это соотношение есть во всех аспектах жизни, от строения костей человека до спирального расположения семян подсолнуха и завитков раковин моллюсков, оно лежит в основе всех биологических структур и кажется геометрической схемой самой жизни.

«Золотое сечение» – это деление отрезка на части в таком соотношении, при котором большая часть относится к меньшей, как сумма к большей, т. е. $|AB| / |BC| = |AC| / |AB|$, и оно равно 0,618.

Новый дизайн Твиттера (рис. 7.14) был запущен в конце 2011 г., при этом применялось правило «золотого сечения» для построения макета сайта.

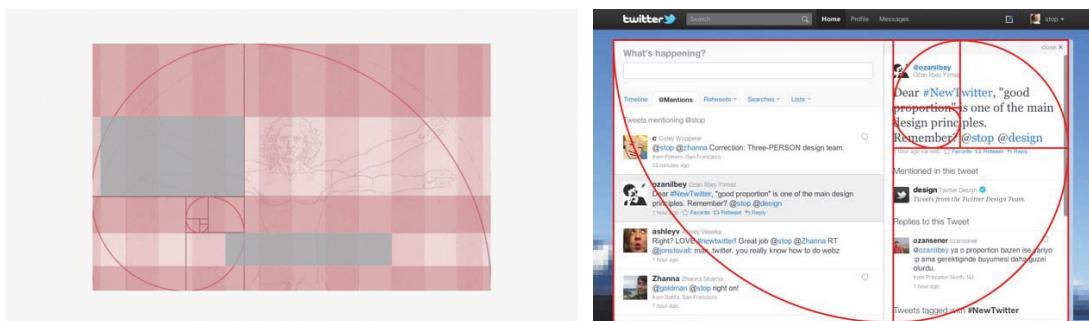


Рис. 7.14. «Золотое сечение»

2. **Ряд Фибоначчи** (каждое последующее число оказывалось равным сумме двух предыдущих: 1, 2, 3, 5, 8, 13, 21 и т. д.).

3. **«Предпочтительные числа»** (ряд чисел геометрической прогрессии, где каждое последующее число образуется умножением предыдущего числа на какую-нибудь постоянную величину).

Разумеется, это не все популярные пропорции.

Шрифтовая сетка обеспечивает вертикальный ритм. Необходимо выбирать высоту строки, единую для всего макета. Все элементы рубрикации с кеглем, отличным от кегля основного текста, должны иметь междустрочный пробел, кратный выбранной высоте строки. Высота каждого такого элемента (в сумме со всеми вертикальными полями) должна содержать целое количество строк шрифтовой сетки.

Таким образом, мы получаем прообраз будущей сетки – «зебру». На этой сетке будет лежать весь текст: абзацы, списки, заголовки, иллюстрации, плашки и прочее.

Чаще всего используются такие параметры для шрифтовой сетки: кегль – 12 пикселей, высота строки – 18 пикселей.

Глава 8

ЮЗАБИЛИТИ-ТЕСТИРОВАНИЕ

Юзабилити-тестирование появилось в крайне бюрократизированных областях – в военной промышленности и сфере рискованного гражданского производства (самолеты, электростанции и т. п.).

Юзабилити-тестированием является любой эксперимент, направленный на измерение качества интерфейса или же поиск конкретных проблем в нем. Его проводят на всех стадиях создания программного продукта: планировании и подготовке, активной разработки, приемки и эксплуатации. Тестируется одно и то же, но с разных сторон.

Юзабилити-тестирование (*англ. usability testing*) – исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект (такой, как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения, основанное на привлечении пользователей в качестве тестировщиков и суммировании полученных от них выводов.

В ходе юзабилити-тестирования изучается, насколько хорошо пользователи выполняют конкретные стандартные задачи и с какими проблемами они при этом сталкиваются. Результаты такого тестирования помогают выявить как проблемы, затрудняющие понимание и использование продукта, так и удачные решения.

Тестирование позволяет:

- понять, насколько плохо или хорошо работает интерфейс, что может либо побудить улучшить его, либо, если он уже достаточно хорош, успокоиться; в любом случае достигается польза;
- сравнить качество старого и нового интерфейсов и тем самым дать обоснование изменениям или внедрению;
- найти и опознать проблематичные фрагменты интерфейса, а при достаточном объеме выборки также и оценить их частотность.

В то же время Ю-тестирование не может сделать из плохого продукта продукт хороший. Оно всего лишь делает продукт лучше.

Юзабилити-тестирование проводится на всех этапах жизненного цикла программного продукта.

Планирование и подготовка. На этом этапе стараются выявить возможные проблемы юзабилити до начала разработки, минимизировать риски при разработке. Здесь используют такие методики, как анализ контекста использования; анализ конкурентов; карточная сортировка; сценарии.

Этап активной разработки программного продукта. На этом этапе разрабатывается полноценное приложение, готовое к релизу. Применяют эвристическую и экспертную оценку; бумажные прототипы; раскадровку; выполняют оценку прототипа.

Приемка. Необходимо дать объективную оценку удобства использования для принятия решения по релизу приложения. Используется метод фокусных групп, эвристическая оценка.

Эксплуатация. На этом этапе необходимо оперативно выявлять и устранять «узкие места» и трудности, с которыми сталкиваются пользователи приложения при эксплуатации и в результате внесения изменений. Применяется проверка посредством наблюдения за пользователем (контрольные списки; опросы пользователей).

В процессе тестирования необходимо соблюдать комплексные методы. Поэтому обычно тестируют все три основные области.

1. Информационную архитектуру, в которой должны быть:

- интуитивно понятная структура информации;
- удобные инструменты навигации и вызова функций приложения;
- наглядный способ представления результатов действия.

2. Рабочие процессы и взаимодействие. Здесь проверяется:

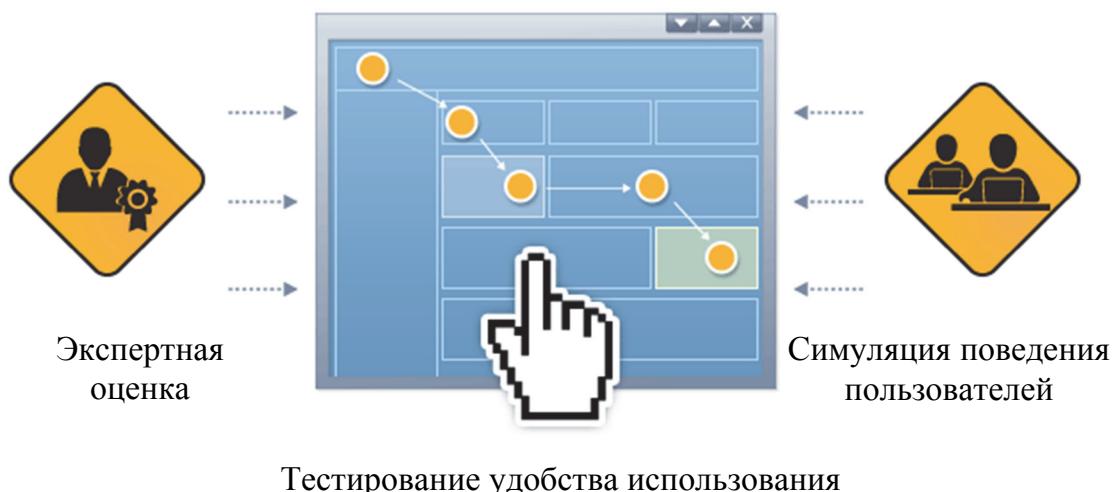
- логика использования и логика рабочих процессов;
- эффективное использование приложения, идентификация и устранение избыточных операций;
- простота использования, интуитивное понимание и скорость адаптации пользователя;
- время отклика на действия пользователя, короткие беспрепятственные пути пользователей (User Journeys);
- плавный процесс выполнения сценариев, корректная обработка ошибок и отказоустойчивость приложения.

3. Графический дизайн. Проверяется:

- простота восприятия и усвоения отображаемой информации;
- целостность и единство представления используемых функциональных и графических элементов в рамках всего приложения.

Для выявления проблем удобства использования приложения, в том числе на ранних этапах планирования и разработки ПО, используется методика двойной проверки (рисунок):

- изучение опыта взаимодействия пользователя с приложением через *имитацию поведения пользователей*;
- проверка соответствия принципам обеспечения удобства пользования и корректного визуального представления в контексте функциональных требований посредством *экспертной оценки*.



Экспертная оценка приложения осуществляется в соответствии с целями проекта, функциональными и нефункциональными требованиями к ПО. Процедуры экспертной оценки включают в себя:

- выявление и изучение возможных сценариев использования и путей пользователя (User Journeys) в контексте бизнес-целей и функционала приложения;
- анализ информационной архитектуры приложения;
- анализ интерфейса и элементов интерфейса;
- анализ функционального соответствия.

Имитация поведения пользователей. Проверка поведения приложения путем имитации поведения пользователей. В результате будет предоставлена полная информация, необходимая для быстрого устранения выявленных дефектов, которые могут негативно сказаться на юзабилити приложения или пользовательском впечатлении в целом.

Сравним достоинства экспертной оценки и юзабилити-тестирования (таблица).

Сравнительная характеристика экспертной оценки и юзабилити-тестирования

Ю-тестирование	Ю-экспертиза
Нужны пользователи	Нужен эксперт
Объективные данные	Субъективные данные
3–4 недели до получения результата	Результат с первого дня
На выходе обозначение проблем	На выходе обозначение проблем + вариант решения

Ю-тестирование находит крупные проблемы интерфейса. Но не может показать, к примеру, некорректные фрагменты интерфейса, недостаточно серьезные, чтобы вызвать человеческие ошибки, но снижающие скорость работы.

Опыт же эксперта позволяет выявить их без труда. Один или несколько экспертов, профессиональных дизайнеров интерфейса или юзабилити-специалист по заданным сценариям или метрикам оценивают удобство системы, фиксируют найденные проблемы и (опционально) дают свои рекомендации.

Тестирование юзабилити – это комплекс действий со стороны пользователя (респондента) по отношению к тестируемому объекту. На основе поведения респондентов юзабилити-специалист (модератор) делает выводы о наличии в интерфейсе юзабилити-проблем и их характере.

В процессе юзабилити-тестирования пользователю предлагаются в «лабораторных» условиях решить основные задачи, для выполнения которых этот продукт разрабатывался, просят высказывать во время выполнения этих тестов свои замечания.

Процесс тестирования фиксируется в протоколе (логе) и/или на аудио- и видеоустройства. Если проверка выявляет какие-либо трудности, то разработчики должны доработать продукт и повторить тестирование.

Методы UX-исследования можно разделить на два лагеря: *количественные и качественные*.

Количественным исследованием является любое исследование, которое может быть измерено численно. Оно отвечает на такие

вопросы, как «сколько людей сюда кликнуло?» или «какой процент пользователей в состоянии найти призыв к действию?». Оно важно для понимания статистических вероятностей и того, что происходит на сайте или в приложении.

Качественное исследование иногда называют «мягким» исследованием. Оно помогает нам понять, почему люди делают то, что они делают, и часто принимает форму интервью или беседы. Обычно задаются вопросы вроде «почему люди не видят призыв к действию» и «что еще люди замечают на странице?». Задача этих методов – помочь понять мотивы поведения и логику пользователей. Количественные исследования направлены на то, чтобы опросить как можно больше респондентов и получить числовой результат.

Виды юзабилити-тестирования

Существуют следующие виды юзабилити-тестирования:

- 1) «коридорное» Ю-тестирование;
- 2) модерируемое удаленное тестирование;
- 3) немодерируемое удаленное тестирование;
- 4) А/В-тестирование.

Коридорное тестирование. Название «коридорное» пошло от начальной идеи ловить 5–6 случайных человек, проходящих мимо по коридору. Респондент выполняет задания, которые дает ему модератор, на компьютере или ином устройстве. Часто используется специальное ПО и оборудование для записи сессий. Вариации: тестирование в оборудованной лаборатории (лабораторное тестирование), выездные сессии, вовлеченнное наблюдение за пользователем на его рабочем месте.

Модерируемое удаленное тестирование. Тестирование проводится один на один, удаленно. Респондент и модератор общаются удаленно (по телефону, скайпу и т. п.). Респондент выполняет задания со своего компьютера, подключившись к совместному удаленному рабочему столу.

Модератор дает задания, наблюдает за ходом их выполнения, задает уточняющие вопросы. Задача модератора – держать респондента сфокусированным на задачах (но при этом не «помогать» ему решать эти задачи) и выяснить:

– какие лишние или ненужные действия совершает пользователь на веб-сайте;

- что мешает ему быстрее найти искомый объект (информацию);
- что отвлекает от главной цели (например, слишком яркий дизайн может быть откорректирован после проведения тестирования юзабилити сайта).

Немодерируемое удаленное тестирование. Респонденты выполняют тесты, созданные в одной из специальных систем, самостоятельно. Задания для тестирования формируются в одной из систем для проведения удаленного тестирования. Специализированный сервис дает задания, собирает метрики и обратную связь в автоматическом режиме, без участия модератора. Ссылка на тест рассыпается респондентам, которые самостоятельно проходят тест.

В системе можно задавать любые вопросы (как в анкете). Выполнение заданий происходит на сайте.

Используемые фиксируемые метрики:

- 1) выполнение заданий (при настроенных условиях);
- 2) пути пользователя по сайту;
- 3) время на выполнение;
- 4) ответы на вопросы;
- 5) тепловые карты движения мыши по сайту.

Инструменты, с помощью которых можно осуществлять данное тестирование:

- Loop'11 (<https://www.loop11.com>);
- Usabilla (<https://usabilla.com/>);
- UserZoom (<http://www.userzoom.com/>);
- Webnographer (<http://www.webnographer.com/>);
- другие (<http://remoteresear.ch/tools/>).

A/B-тестирование (англ. *A/B testing, Split testing*) – автоматизированная проверка двух или более версий одного контента на одинаковых аудиториях.

Несколько версий контента (страницы, изображения, письма) с незначительными различиями показываются большим группам пользователей. По результатам статистической выборки измеряются показатели каждой версии и выясняется, какие из изменений улучшают целевой показатель

Разновидностью А/В-тестирования является **многовариантное тестирование**. В этом случае тестируются не два целостных варианта, а сразу несколько элементов продукта или составных частей исследуемого объекта в различных сочетаниях, при которых каждый тестируемый элемент может быть двух видов (А или В).

Метод часто используется в веб-дизайне, типичные применения – исследование влияния цветовой схемы, расположения и размера элементов интерфейса на конверсию сайта. В веб-дизайне часто тестируются две очень похожие веб-страницы (страница А и страница В), которые различаются лишь одним элементом или несколькими элементами (тогда метод называют *A/B/n-тестированием*).

Страницы А и В показываются различным пользователям в равных пропорциях, при этом посетители, как правило, не знают об этом. По прошествии определенного времени или при достижении определенного статистически значимого числа показов, сравниваются числовые показатели цели и определяется наиболее подходящий вариант страницы.

На успех теста влияют:

- понятные различия (можно точно сказать, что повлияло на поведение пользователей);
- наличие большой репрезентативной выборки респондентов (лучше больше 1000 для каждой версии);
- однородность выборок для каждой версии.

Для А/В-тестирования веб-дизайна часто используются инструменты от сервисов веб-статистики; в этом случае также часто важно применение механизма для разбиения пользователей, которым будет показан тот или иной вид дизайна (одному и тому же пользователю нужно показывать тот же самый вариант дизайна).

Для поставщика контента успешная конверсия может быть регистрацией посетителей на сайте, подпиской на почтовую рассылку, скачиванием программного обеспечения или какие-либо другие действия, ожидаемые от посетителей.

Для А/В тестирования можно использовать следующие инструменты:

- Content Experiments GA;
- Optimilezy.com;
- ABTest.ru.

Преимуществом метода является использование при проектировании объективных данных.

Экспертная оценка

Крупный недостаток Ю-тестирования – высокая стоимость. Экспертная оценка является более быстрым и дешевым способом проверки качества интерфейса. Она позволяет обнаружить порядка 80% проблемных мест.

Эксперт (или несколько) проводят аудит системы.

Один или несколько экспертов, профессиональных дизайнеров интерфейса или Ю-специалист по заданным сценариям или метрикам оценивают удобство системы, фиксируют найденные проблемы и (опционально) дают свои рекомендации.

Виды экспертной оценки. Чтобы сделать поиск проблем результивным, нужен хоть сколько-нибудь формальный подход. Наиболее распространенные формальные подходы: *проверка по контрольному списку, эвристическая оценка, мысленная прогонка по интерфейсу*.

Проверка по контрольному списку. Проверка по контрольному списку ближе всего к формальному тестированию качества. Составляется список произвольных требований (чек-листы), после чего интерфейс проверяется на соответствие этим требованиям. В контрольном списке может быть все что угодно, как правило, туда прямым ходом отправляются требования из стандартов и руководств по проектированию интерфейсов. Метод, вообще говоря, чрезвычайно надежен при поиске проблем, вроде грамматических ошибок в интерфейсных тестах.

К сожалению, он имеет следующие недостатки.

1. Чем более полон и детален контрольный список, тем дольше производится проверка. В хорошем списке может быть больше сотни пунктов. Проверки должны выполняться последовательно, больше чем один пункт за раз проверять нельзя. Предположив, что одна проверка занимает пятнадцать минут, можно посчитать длительность полного исследования.

2. Контрольный список, чтобы быть совершенно надежным, не имеет права содержать сколько-нибудь размытых требований.

3. К сожалению, достаточно полный контрольный список, содержащий только строго определенные требования, есть явление недостижимое.

4. Качество проверки по контрольному списку напрямую зависит от качества самого списка. Чтобы создать качественный список требований, нужно затратить годы труда – и самое обидное, что этот список никогда не будет завершен, поскольку все время проявляются новые требования.

С другой стороны, четкий контрольный список может использоваться кем угодно, что дает возможность вынести проверку интерфейса из деятельности Ю-специалиста, передав ее отделу контроля качества.

Эвристическая оценка. Эвристическая оценка была разработана Якобом Нильсеном и Рольфом Моличем, которые надеялись с ее помощью сократить продолжительность проведения проверки по контрольному списку. При эвристической оценке вместо десятков и сотен конкретных требований интерфейс проверяется на соответствие всего нескольким общим принципам. Сам Нильсен (Молич впоследствии самоустранился) рекомендовал следующие принципы.

1. *В любой момент времени система показывает, что с ней происходит.* Этот принцип означает, что пользователь ВСЕГДА должен знать, что происходит и на каком участке пути он остановился. Если это сложная регистрация, указывайте, что это второй шаг из трех. Если что-то закачивается или работает какой-то скрипт – выводите процент обработки.

2. *Система использует термины, понятия и метафоры, присущие в реальном мире, а не обусловленные компьютером.*

3. *В любой момент пользователь контролирует систему, а не наоборот. Любую команду можно отменить или повторить.* Пользователь всегда должен контролировать ситуацию. Например, при заполнении формы обязательно должна быть кнопка «очистить форму». Если форма предусматривает несколько шагов – пользователь должен вернуться на предыдущий шаг, или наоборот, по возможности – пропустить какой-то, чтобы попозже вернуться к нему.

4. *В любой момент времени система выглядит и функционирует единообразным и стандартным способом.* Кампания Microsoft в своих продуктах придерживается этого принципа (хотя часто и нарушает многие другие). У них во всех продуктах одни и те же иконки находятся в одинаковых местах и выглядят одинаково.

5. *Интерфейс системы препятствует появлению человеческих ошибок.* Ошибку проще предупредить, чем исправить. Там, где можно упростить выбор и убрать ненужные действия, следует убрать. Если вам что-то известно о вводимой информации, подсказывайте ее пользователю (формат телефона или код региона) и т. д. В данном случае пользователь не может пройти дальше, пока не разберется с этими настройками и не выберет хотя бы один вариант анализа сайта. Это намного удобней, чем потом сообщать пользователю что-то типа «выберите хотя бы один вариант» или «неверно заданы параметры».)

6. В любой момент времени интерфейс показывает объекты и команды сам, не требуя от пользователя вспоминать их. Максимально упрощайте жизнь пользователю, делайте ему подсказки, запоминайте вводимую ранее им информацию. Например, если у вас опять же многошаговая форма регистрации, показывайте ему уже заполненные поля, если они могут понадобиться в дальнейшем. Кроме того, постарайтесь минимизировать количество текстовых полей, где пользователь должен что-то вводить сам. Давайте ему варианты (подсказки), чтобы он мог выбрать из уже имеющихся вариантов. Подсказка региона выскакивает при вводе хотя бы одной буквы. И если пользователь не знает, как правильно пишется город, он просто кликает по одному из предложенных вариантов.

7. В интерфейсе есть методы ускорения работы, предназначенные для опытных пользователей и не мешающие пользователям неопытным; благодаря таким ускорителям опытные пользователи получают резерв для повышения собственной производительности. Одна из самых больших проблем – это как соединить простоту интерфейса и его функционал. Как сделать так, чтобы один и тот же интерфейс был понятен и удобен как профессиональному пользователю, так и новичку? В Google есть простой поиск для большинства пользователей, а также расширенный, для опытных. При этом функции расширенного поиска спрятаны довольно далеко, и только опытный пользователь может их найти, т. е. по сути тот, кому он нужен.

8. Интерфейс эстетичен и в любой момент времени не содержит ненужной сейчас информации. К сожалению, этим принципом практически никто не пользуется. В 90% случаев при регистрации вас спрашивают и домашний телефон, и мобильный, и адрес, и e-mail, и даже дату рождения. А все для того, чтобы поздравить вас с Новым годом и Днем рождения. Все дополнительные данные вы сможете уточнить позже, при необходимости. А изначально, например, при регистрации или при оформлении заказа спрашивайте только ту информацию, которая вам действительно необходима.

9. Интерфейс помогает пользователям обнаруживать и исправлять проблемы, включая человеческие ошибки. Скажите, что означает ошибка базы данных в строке 433? Или длинные жуткие ошибки в строке памяти LXR-XXX-5438645? Все эти системные

сообщения должны сохраняться в логах для администратора сайта/системы. Пользователю же пишите нормальным языком, в чем конкретно ошибка. Более того, пользователя по сути интересует только одно: виноват он или система? И можно ли что-то сделать?

10. Справка доступна в любой момент времени. Она достаточна, но не избыточна; к ней легко обращаться; она не абстрактна, а нацелена на решение конкретных задач пользователя; в ней описываются конкретные шаги по решению проблем. В идеале, конечно, система должна быть настолько понятной, что никакая помощь или документация не должна быть востребована. Но увы, этого не всегда можно достичь. Поэтому у вас всегда должна быть разработана система помощи. Помощь или документация должна быть простой, понятной, быть легко доступной, соответствовать задачам пользователя. Если все-таки документ получается достаточно объемным, сделайте по нему краткую навигацию для быстрых переходов по разным разделам, а также поиск по помощи.

Эти эвристики достаточно результативны. В то же время существуют две проблемы этого метода.

Во-первых, оценку должны проводить от трех до пяти экспертов, иначе она не будет сколько-нибудь результативной. Требование к числу экспертов постоянно игнорируется. В результате для многих заказчиков, испробовавших этот метод, вся дисциплина юзабилити оказывается полностью дискредитированной.

Во-вторых, эвристическая оценка никак не учитывает деятельности пользователей, в лучшем случае она покрывает выполняемые пользователями операции. В результате эвристическая оценка не выявляет структурные проблемы интерфейса, которые, как правило, заметно важнее локальных проблем.

Мысленная прогонка по интерфейсу. Если предыдущие методы формализовали правила, которым должен следовать интерфейс, то мысленная прогонка формализует метод, согласно которому интерфейс оценивается.

Если исходить из того, что интерфейс предназначен для использования функций, можно проверить, как эти функции вызываются и используются.

Если просто проговорить словами, как работают интерфейсы всех функций, становится понятно, какие из них неоправданно подавлены, а какие работают недостаточно хорошо. Разумеется, для этого тоже необходим опыт эксперта.

Проблема есть и здесь. Хотя метод неплохо работает при простом проговаривании работы функций, *наилучшие результаты* он дает, если *функции* не просто проговариваются, а *записываются* («мысль изреченная есть ложь»). Сам факт записи позволяет увидеть множество противоречий и недостатков в интерфейсе. Увы, запись требует очень много времени, что не позволяет использовать этот метод всегда и всюду.

Можно сделать следующие выводы.

Экспертная оценка – метод неполноценный, но достаточно эффективный и замечательно дополняющий юзабилити-тестирование. Использовать ее без тестирования разумно только в ситуациях, когда подобрать адекватных респондентов для тестирования по каким-либо причинам невозможно.

«Чистое» юзабилити-тестирование, лишенное параллельной экспертной оценки, неэффективно.

Если проводить и тестирование, и оценку, результаты становятся наиболее полными. Тем более, что составление списка пользовательских задач, формализация контекста использования и целевой аудитории продукта делаются и при тестировании, и при оценке.

Этапы Ю-тестирования

Юзабилити-тестирование включает следующие этапы: 1) определение проблемы; 2) формирование гипотез; 3) определение метрик для тестирования; 4) определение персонажей и сценариев; 5) подбор респондентов; 6) заполнение анкеты; 7) вводный инструктаж; 8) проведение ю-тестирования; 9) опрос респондентов; 10) анализ результатов; 11) определение требований для проектирования сайта.

1. Определение проблемы. Вначале необходимо сформулировать проблему. Например, «Пользователи заходят на страницу (интерес есть), но не выполняют целевого действия. Почему?».

2. Формирование гипотез. Необходимо выдвинуть свои предположения, почему на данной странице возникают проблемы. Например, предполагаем, что пользователи могут: не видеть кнопки/ссылки/заголовки/разделы, двойственno понимать назначение элементов управления, иметь недостаточно информации, неправильно воспринимать информацию.

3. Определение метрик. Мы должны определить, что будем измерять. В качестве метрик можно взять следующие:

- справился/не справился;
- ошибки;
- отклонение от идеального сценария;
- понятность сообщений;
- последовательность действий;
- время на выполнение задания.

4. Определение персонажа и сценария теста. Персонаж – это собранный образ вашего целевого посетителя. Определить персонажа необходимо для понимания, кого необходимо приглашать на тестирование. Нет смысла тестировать бухгалтерскую программу на медработнике, а интерфейс кассового аппарата на художнике.

Определение сценария теста. Сценарий теста – это задание, которое необходимо выполнить пользователю на сайте с предысторией. Следует составить не больше 10 ситуативных заданий, приближенных к реальным сценариям взаимодействия, имитирующих пути пользователя. При этом учитываем следующее:

- сценарий теста должен имитировать реальный путь пользователя, заинтересованного в товаре/услуге/информации;
- сценарий теста должен быть понятным, логичным и простым для выполнения;
- сценарий теста должен состоять из целей, а не предлагать инструкции, как этой цели достичь.

Тестовое задание – это то, что получает от вас респондент, задание, позволяющее провести респондента через фрагмент интерфейса системы и определить характеристики этого фрагмента. Тестовые задания, помимо того, что должны соответствовать пользовательским задачам, должны обладать еще и следующими свойствами.

Однозначность. Задания должны быть сформулированы так, чтобы исключить их неправильное толкование респондентом. Если респондент поймет задание неправильно, вам почти наверняка не удастся по ходу теста направить его на правильный путь, не подсказав ему одновременно последовательности выполнения задания.

Полнота. В тексте задания должна присутствовать вся информация, необходимая для выполнения этого задания.

Краткость. Если вы замеряете скорость выполнения заданий, задания должны быть достаточно краткими, чтобы длительность чте-

ния заданий респондентами не влияла на продолжительность выполнения самих заданий (люди читают с разной скоростью). Если текст задания будет велик по объему, вам придется вручную отсекать длительность чтения для каждого задания, что очень трудоемко.

Отсутствие подсказок. По тексту задания не должно быть понятно, как это задание нужно выполнять. Например, недопустимо использовать терминологию системы – вместо каждого термина нужно расписывать его значение, иначе респонденты просто нажмут кнопки с теми же словами, и вы не выявите никаких проблем.

В задании должна присутствовать *точка начала выполнения задания*, т. е. должно быть прописано то окно или экран, на котором респондент должен находиться вначале. Если такой информации представлено не будет, респонденты неизбежно будутходить к другим фрагментам интерфейса, а значит, задание разными респондентами будет выполняться по-разному, что делает бессмысленным все статистические расчеты.

Тестовые задания, предъявляемые респондентам, должны быть распечатаны, каждое задание на отдельном листе, чтобы респондент не мог забежать вперед и прочесть задания, которые он еще не выполнил.

5. Подбор респондентов. Сначала необходимо определить общие требования к респондентам.

Опыт работы с системой. Общее правило: если оптимизируется интерфейс существующей системы, половина респондентов должна иметь опыт работы (на них можно определить проблемы переучивания при внедрении), а половина нет (на них определяется скорость обучения). Если существуют конкурирующие системы, лучше другая пропорция: треть с опытом работы с предыдущей версией, другая треть – с опытом использования конкурирующих систем, оставшаяся треть – без опыта работы с системой.

Уровень компьютерной грамотности. При прочих равных предпочтительным выбором является реальный, т. е. совпадающий с опытом целевой аудитории, уровень у трех четвертей респондентов и низкий уровень – у оставшейся четверти (на ней можно определить большее количество проблем).

Уровень компьютерной грамотности удобно определять по следующей шкале: высокий, выше среднего, средний, низкий, очень низкий.

Возраст. Оптимальная пропорция: три четверти респондентов имеет возраст целевой аудитории системы, оставшаяся четверть старше (на ней можно определить большее количество проблем).

Пол респондентов (меньшее влияние на результат). Стоит увеличить количество женщин среди респондентов по сравнению с пропорцией в целевой аудитории, поскольку на женщинах легче определять проблемы при внедрении (женщины, в целом, медленнее обучаются, но, обучившись, лучше работают).

Уровень эмоциональной открытости респондентов. Чем более скован респондент, тем меньше он способен сказать вам ценного. Даже определив наличие проблемы, вы не сможете добиться от него никакой информации о том, что эту проблему вызвало. Существует прекрасный способ решить проблему недостаточной эмоциональной открытости – стоит иметь базу респондентов и использовать их повторно. Респондент, уже знающий на опыте, что в юзабилити-тестировании нет ничего страшного, значительно охотнее идет на контакт и вообще более разговорчив.

В 1992 г. Роберт Вирзи (Robert Virzi) предположил, что для теста достаточно пяти респондентов. Через год эстафету приняли Якоб Нильсен и Томас Ландауэр (Jakob Nielsen и Thomas K. Landauer), которые утверждали, что пяти респондентов достаточно для того, чтобы выявить 70% проблем, и еще три респондента нужны для того, чтобы довести результативность до 85%. Все три автора писали только о тестировании маленьких систем.

Для более сложных систем придется выполнять несколько разных тестов на разных респондентах; без этого охватить весь интерфейс системы окажется попросту невозможно.

6. Заполнение анкеты. Разработка и тестирование надежных анкет является очень долгим и трудоемким процессом, так что рассчитывать на скорое появление качественных отечественных анкет не приходится.

Помимо анкетирования можно подсчитывать эмоциональные реакции респондента. К примеру, респондент улыбнулся – ставим плюс, ругнулся или поморщился – ставим минус. Количество и знак реакций и составляет искомую величину показателя.

7. Вводный инструктаж. На первом листе нужно выводить *вводную форму*.

8. Проведение ю-тестирования. При проведении тестирования рекомендуется фиксировать следующие наблюдения.

Скорость работы. Между заданиями переводите секундомер на новый круг. Если респондент по какой-либо причине отвлекается, ставьте секундомер на паузу.

Ошибки. На листе бумаги ставьте черточку при каждой человеческой ошибке. Удобно ставить мелкие черточки при мелких ошибках и длинные – при ошибках крупных. После теста достаточно посчитать количество черточек. Если вы раздельно считаете ошибки разных типов (например, просто ошибки и отдельно неправильно выбранные элементы меню), лучше пользоваться разными кодами, например, теми же черточками при простых ошибках и буквами «М» при ошибках, связанных с меню.

Проблемы, которые вы видите сразу же. Кратко записывайте на бумажке сущность проблемы и текущее время (время первым!). Если вы будете точно знать, когда произошла проблема, вам будет легче найти соответствующий фрагмент видеозаписи.

Эмоциональные реакции респондента. Ставьте знак плюса при положительных реакциях и знак минуса – при отрицательных. Реакции, происходящие в момент завершения тестовых заданий, не считаются.

9. Опрос респондентов. Перед проведением теста у респондента нужно выспросить, что он ожидает от системы. После теста – насколько показанный интерфейс соответствует его ожиданиям. Здесь респонденту уже вполне можно верить: во-первых, он подготовлен своими предыдущими ответами, а во-вторых, показанный ему интерфейс может побудить его сформулировать требования, которых он раньше не осознавал.

10. Анализ результатов. Анализировать результаты можно как во время, так и после теста. Оптимальной стратегией является начало анализа во время теста.

Почти любое юзабилити-тестирование направлено на поиск и выявление проблем. Результаты можно подсчитывать по разработанному заранее алгоритму.

Сами по себе собранные количественные данные практически не способны показать *сущность* проблемы, они показывают только количество проблем.

С другой стороны, они могут быть использованы для сравнения старого и нового интерфейсов. Кроме того, сведенные вместе, они способны показать, где именно выявлены самые значительные

проблемы – что полезно, т. к. начинать оптимизацию приятнее всего с областей, обещающих максимальную отдачу.

11. Определение требований для проектирования сайта или формирование рекомендаций по дальнейшей модификации интерфейса. Все результаты тестирования обобщаются с тем, чтобы сформулировать рекомендации относительно модификаций прототипа интерфейса. Модификации могут быть связаны с изменениями содержимого экранных форм, элементов навигационной системы, терминологии и даже функциональности тех или иных элементов интерфейса.

Глава 9

ОСОБЕННОСТИ РАЗРАБОТКИ ИНТЕРФЕЙСОВ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

Понятие и особенности мобильной среды

Формирование самой мобильной среды тесно взаимосвязано с развитием мобильных технологий, которые позволили создать мобильные устройства такими, какими мы видим их сейчас.

Мобильная среда – сфера существования переносных и носимых вычислительных устройств.

Сам термин «мобильная среда» включает в себя все переносные устройства, однако специалисты по разработке, чаще всего, сужают его исключительно до следующего списка мобильных устройств: телефоны, смартфоны, фаблеты (смартфон, имеющий сенсорный экран, размер которого находится между размером типичного смартфона и планшетного компьютера: с диагональю от 5,5 (ранее в качестве фаблетов рассматривали модели с диагональю экрана от 5,0) до 6,9 дюймов и шириной не более 135 мм), КПК (карманый персональный компьютер), планшетные компьютеры, носимые устройства (умные часы, браслеты) и (реже) электронные книги. Подобное сужение обусловлено популярностью и первоочередностью разработки именно для этих устройств.

Основные характеристики мобильных устройств следующие: с пользователем находится постоянно; может использоваться немедленно; является персональным; постоянное перемещение пользователя; может использоваться для подключения к сети; площадь экрана меньше, чем у обычного компьютера; ограниченность ресурсов мобильного устройства; важно наличие, скорость и надежность сетевого соединения.

Среди особенностей мобильной среды можно выделить:

- оперативность (самая высокая степень в общедоступном плане);
- более низкий показатель единицы трафика на пользователя в сравнении с десктопными устройствами;
- самая высокая активность пользователей как в потреблении, так и создании контента;

- низкая конкуренция на рынке;
- пользовательская аудитория наиболее выгодна для рекламы, поскольку является более платежеспособной и активной в сравнении с аудиторией десктопных устройств.

Мобильное приложение – компьютерная программа, разработанная для использования на смартфоне или планшетном компьютере и нацеленная на решение конкретной задачи. Мобильные приложения создаются под конкретную операционную платформу (наиболее популярные – iOS, Android, Windows Phone). Приложения, написанные четко под определенную операционную систему, называют **нативными** (от англ. *native* –aborиген).

На сегодняшний день не существует устоявшейся классификации мобильных приложений ввиду взрывного роста технологий. На рынке постоянно появляются качественно новые продукты и видоизменяются старые. Можно разделить мобильные приложения на несколько основных групп с точки зрения выполняемых ими функций:

- *приложения развлекательного характера* (их доля на рынке наиболее существенна, а мировой оборот составляет десятки миллиардов долларов): сюда входят небольшие аркадные игры, менеджеры рингтонов, калькуляторы, календари, проигрыватели аудио- и видеофайлов, просмотрщики изображений и электронных документов и т. д.;

- *приложения для коммуникации*, в частности, для общения посредством электронной почты и социальных медиа, мобильные приложения для средств массовой информации;

- *справочные приложения* (программы для перевода, туристические гиды, электронные энциклопедии, галереи, различные каталоги и базы данных; технологии GPS и позволяющие определить местоположение мобильного устройства);

Рынок приложений не очерчивается упомянутыми группами программных продуктов – они лишь наиболее заметны в общем потоке. Существует множество смежных, пограничных видов приложений, совмещающих в себе две или более функции. Таким образом, наблюдается тенденция к гибридизации приложений. Разработчики создают программные продукты на стыке описанных выше групп, чтобы повысить функциональность и привлечь как можно более широкую аудиторию.

Процесс создания мобильного приложения подразумевает его разработку, т. е. процесс, при котором приложения разрабатываются для небольших портативных устройств, таких как КПК, смартфоны или сотовые телефоны. Эти приложения могут быть предустановлены на устройство в процессе производства, загружены пользователем с помощью различных платформ для распространения ПО или являться веб-приложениями, которые обрабатываются на стороне клиента или сервера.

Разработчики приложений могут предлагать и публиковать свои программы в магазинах приложений с возможностью зарабатывать от распределения доходов по продажам. Самыми известными являются AppStore, где только одобренные приложения могут распространяться и запускаться на iOS устройствах, и Google Play, приложения в котором работают на устройствах с Android OS.

Кроме приложений, для мобильных устройств возможно и другое техническое решение: мобильная версия сайта.

Мобильная версия – это специализированный сайт, адаптированный для просмотра и работы на мобильных устройствах: смартфонах и планшетах. При проектировании таких программных продуктов учитываются особенности мобильных устройств и мобильного интернета.

От основного сайта мобильная версия отличается тем, что ее дизайн значительно упрощен и жестче структурирован. Ввиду скромных размеров экрана становится чрезвычайно актуальной проблема выбора информации, которую стоит донести до пользователя. Задача эта ложится на плечи дизайнеров и разработчиков, и от их решения напрямую зависит качество конечного продукта и удовлетворенность им пользователей.

Лаконичность верстки преследует цель уменьшить коммуникативные шумы, возникающие из-за конструктивных особенностей передатчика и контекста использования мобильных устройств: контент часто просматривается на ходу, внимание пользователя рассеяно, а время, которое отведено на ознакомление, ограничено. Мобильные версии, как правило, более схожи с официальными сайтами, чем приложения. Дизайн последних на сегодняшний день стоит особняком.

Пользовательский интерфейс мобильных приложений может строиться на различных платформах. При выборе платформы

разработки приложения необходимо сначала определиться с ее типом. Существуют три варианта: родная (нативная), web и гибридная.

Родные (нативные) платформы позволяют создавать интерфейс приложения, который замечательно выглядит и воспринимается. Обратная сторона разработки родных приложений – для каждой платформы нужно писать свой код. Загружаются они через магазины приложений (AppStore, Google Play, магазин приложений Windows и т. д.) и устанавливаются в ПО смартфона. *Важным отличием является то, что нативные приложения разрабатываются специально под конкретную платформу (например, под iOS для iPhone, под Android для устройств под управлением ОС Android или под Windows для Windows Phone и т. д.).*

Таким образом, для пользователя нативными являются приложения, которые требуют установки. В целом, это верно, как и то, что такие приложения разрабатываются специально под мобильные платформы (iOS, Android, Windows Phone). Поэтому от разработчика требуется навыки программирования в конкретной среде разработки (xCode для iOS, eclipse для Android). На выходе это дает приятный внешний вид и беспроблемное взаимодействие приложения с мобильной ОС.

Нативные мобильные приложения позволяют использовать оборудование мобильных устройств так, как не может сделать ни один из мобильный браузеров. В частности, речь идет о доступе к адресной книге, работе с SMS, встроенными сенсорами, камерой и записи аудио.

Такие приложения с наименьшим поглощением ресурсов используют камеру, микрофон, акселерометр, плеер и прочие функции.

Нативные приложения разрабатываются под конкретную платформу и потому имеют такой уровень доступа к системным ресурсам устройства, каким веб-приложения похвастаться не могут. Это означает, что при взаимодействии с пользователем интерфейс нативных приложений работает, как правило, более быстро и гладко. Попытки воспроизвести такой интерфейс в браузере часто приводят к сбоям и задержкам в работе.

Нативное приложение также намного опережает и гибридное, и веб-приложение в вопросах безопасности.

Условно нативные приложения можно поделить на две группы: приложения, которым необходимо интернет-соединение, и офлайн приложения.

Данный вариант обладает как большим количеством положительных сторон, так и большим количеством недостатков.

Web-приложения обычно создают один раз и запускают на разных платформах, но такие приложения могут уступать родным по внешнему виду, восприятию и производительности. Их можно написать на HTML5, CSS и JavaScript.

Web-приложения не случайно называют html5-приложениями. Это, по сути, сайт, оптимизированный под смартфон. Пользовательский интерфейс создается при помощи стандартных веб-технологий. Их не нужно загружать из магазина приложений, но они могут находиться в специальных магазинах веб-приложений, которые есть в некоторых современных браузерах, например у Chrome. Веб-приложения используют для работы браузер телефона. Главной особенностью таких приложений является их кроссплатформенность – возможность работать на всех устройствах, без дополнительной адаптации.

Независимо от установленной ОС такие приложения не могут использовать ПО смартфона. Для обновления информации в приложении необходимо подключение к интернету, скорость работы ограничена возможностями интернет-соединения провайдера услуг.

На самом деле, грань между веб-сайтом, оптимизированным под мобильное устройство, или с адаптивной версткой, которая способствует адекватному его отображению на любом устройстве, и веб-приложением очень тонкая.

Веб-технологии развиваются так стремительно, что разница начинает размываться, и сайты все более становятся похожими на веб-приложения. Разницу, хотя и спорно, можно по-простому описать так:

- сайт представляет собой в большей степени статическую информацию (по сути, цифровая брошюра или листовка);
- веб-приложение, если пользователь может взаимодействовать (менять тексты местами, менять оформление, создавать собственные страницы и т. д.), с информацией.

Веб-приложения для того и создаются, чтобы пользоваться сайтом с телефона. Так что, по сути, это тот же сайт, оптимизированный под мобильные устройства. В отличие от нативного приложения, веб-приложения устанавливать не нужно – они работают в браузере телефона. Поэтому от модели телефона (от мобильной

платформы, если быть точнее) ровным счетом ничего не зависит. Также, вне зависимости от платформы, веб-приложения не могут работать с нативными функциями телефона.

Гибридные инструменты и среды ликвидируют этот разрыв, сочетая в себе простоту программирования и универсальность web-приложений с блеском интерфейсов родных приложений.

Гибридные приложения сочетают в себе некоторые функции нативных и веб-приложений: кроссплатформенность и возможность использования ПО телефона. Такие приложения могут быть загружены через магазины приложений, и при этом имеют возможность независимого обновления информации. Гибридные приложения требуют подключения к интернету, поскольку веб-часть обновляется через интернет. Это, наверное, самый популярный способ построения мобильных приложений, так как у него органическая среда распространения. Пользователь же скорее всего не заметит разницу между нативным приложением и гибридным.

Гибридное приложение потому и называется гибридным, что сочетает функции, которые имеют нативное приложение и веб-приложение. Это кроссплатформенное приложение, которое имеет возможность работать с ПО телефона. Эти приложения так же, как и нативные, загружаются из магазина приложений, но данные обновляют автономно. Поэтому им всегда нужно подключение к интернету – без него веб-функции не работают. Разработка гибридного приложения дешевле и быстрее, чем создать нативное приложение. А пользователи разницы все равно не заметят. Поэтому гибридные технологии наиболее популярны.

Значение при выборе между приложением и мобильной версией играют характеристики целевой аудитории, плановый бюджет проекта, цели проекта (объективная необходимость и/или имидж). Если ваше приложение никак не может работать без нативных функций мобильных устройств, если очень важна высокая скорость обработки данных (игры, соцсети, геолокация), то лучше, чем нативное приложение, ничего не найти. Когда скорость работы можно пренебречь, подойдет гибридное приложение. Веб-приложение стоит делать, когда пользователю от вас не нужно ничего, кроме информации, которую он мог бы получить с телефона при наличии интернета.

Преимущества и недостатки разных типов приложений

Тип приложения	Преимущества	Недостатки
Нативное	Максимальная функциональность и скорость работы. Не требуется интернет-соединение для использования. Имеет доступ к ПО смартфона (GPS, плеер, камера). Распространение через магазины приложений	Выше стоимость и длиннее сроки разработки. Требует от разработчика знаний определенной среды программирования. Работает только с одной платформой. При косметических изменениях необходимо выпускать обновление
Веб (HTML5)	Кроссплатформенность. Не требует загрузки из магазина мобильных приложений. Можно легко адаптировать обычный сайт. Легче найти веб-разработчика нежели разработчика под определенную платформу. Простота создания и поддержки	Требует подключения к интернету. Не имеет доступа к ПО смартфона. Не может отправлять push-уведомления. Должен быть запущен интернет-браузер. При продаже требуется использование своей платежной системы
Гибридное	Функциональность нативного приложения на независимой платформе. Запускается не из браузера в отличие от веб приложения. Возможность независимого обновления. Распространение через магазины приложений	Загружается из магазина мобильных приложений (необходимо соответствовать требованиям). Разработчик должен быть знаком с разными API

Мы рассмотрели три типа мобильных приложений – нативные, web-приложения и гибридные – и их особенности. От того, какие цели вы планируете достичь, создавая мобильное приложение, зависит, какой функциональностью его следует наделить.

Проектирование мобильных версий сайтов без построения родного приложения базируется на принципах адаптивного дизайна.

Адаптивный веб-дизайн (англ. adaptive web design) – дизайн веб-страниц, позволяющий корректно воспроизводить их на разных компьютерах. Философия адаптивного дизайна – предоставить всем пользователям, независимо от их технических средств, равные возможности в доступе к контенту.

Адаптивный подход позволяет не менять ссылку и не переадресовывать пользователя на отдельный мобильный сайт. Подход позволяет не создавать различные версии под каждое отдельное устройство, не зависеть от разрешения и формата экрана.

С точки зрения разработки, адаптивный дизайн строится на эластичных шаблонах и CSS3 MediaQueries. Адаптация дизайна производится через медиа-запросы в CSS файле, где для заданных параметров экрана устройства пользователя применяются определенные правила.

Адаптивный дизайн представляет собой не что иное, как совокупность нескольких шаблонов страниц, каждый из которых «заточен» под определенное разрешение экрана. Как правило, разработчики предлагают три основных шаблона: для полноразмерной версии сайта (например, от 1280 пикселей по ширине), для планшетов (1024) и для смартфонов (640).

При первоначальной загрузке страницы будут загружаться все три шаблона, со всеми соответствующими им скриптами, но на экран будет выводиться только тот, который максимально удовлетворяет текущему разрешению экрана или ширине окна браузера.

Даже если вы заходите на сайт с адаптивной версткой с мобильного устройства и видите «усеченную» версию основного сайта, велика вероятность того, что вы загружаете его полностью, со всеми компонентами. Даже если часть элементов сайта скрыта, вы все равно их загружаете. Это, во-первых, неrationально с точки зрения логики, а, во-вторых, серьезно подвергает сомнению сам принцип удобства мобильного пользователя, на которого перекладываются лишние расходы на неэффективный трафик.

Мир адаптивного дизайна серьезно изменил метод Люка Вроблевски в своей книге «Сначала мобильные!», вышедшей в 2009 г. Автор предлагает начинать дизайн сайтов не с десктопных версий, а с максимально маленьких экранов, с которых будет

просматриваться контент. Вроблевски предлагает создавать дизайн, исходя из особенностей и слабых мест мобильных устройств: скромные размеры дисплеев, низкая производительность и пропускная способность систем и прочее. Такой подход дает качественный дизайн, однако долг, дорог, организационно сложен и достаточно кардинален. Преимущества данного подхода, как показывает практика, тяжело объяснить заказчикам. Неизвестный интерфейс, к тому же, может вызвать негативную реакцию аудитории.

Несмотря на все проблемы, данный подход нашел множество сторонников и продолжает развиваться, что неудивительно при настоящих темпах развития мобильных технологий. К тому же, он учитывает преимущества, которые дает HTML5 и, в принципе, мобильная природа гаджетов: GPS, камеры, акселерометр, геолокация, сенсорный экран.

Принципы адаптивного дизайна. Проектирование начинается с адаптивной версии веб-сайта для мобильных устройств. На этом этапе дизайнеры стремятся правильно передать смысл и основные идеи с использованием небольшого экрана и всего одной колонки. Содержимое при необходимости сокращают, удаляя второстепенные информационные блоки и оставляя самое важное. Основные принципы адаптивного дизайна:

- проектирование для мобильных устройств с самых ранних этапов («mobile first»);
- применение гибкого макета на основе сетки (flexible, grid-basedlayout);
- использование гибких изображений (flexibleimages);
- работа с медиазапросами (mediaqueries);
- применение постепенного улучшения.

Типы адаптивных макетов. Основные типы адаптивных макетов, существующие в настоящее время, представлены ниже.

«Резиновый». Простой в реализации и очевидный для пользователя тип представления сайта. Основные блоки сжимаются до ширины экрана мобильного устройства, где такое невозможно – перестраиваются в одну длинную ленту.

Перенос блоков. Очевидный способ для многоколоночного сайта: при уменьшении ширины экрана дополнительные блоки (айдбары) переносятся в нижнюю часть макета (рис. 9.1).

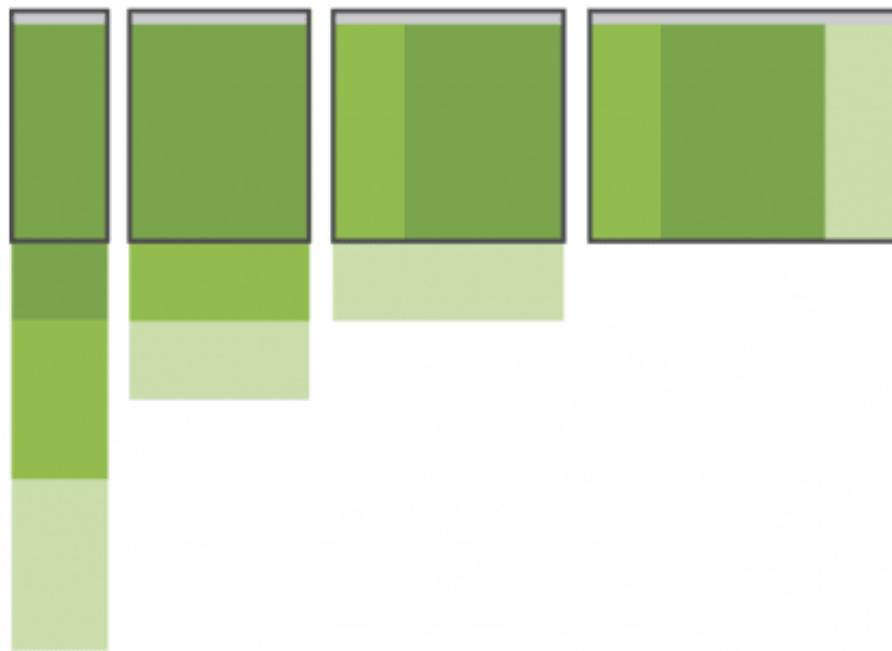


Рис. 9.1. Перенос блоков. Фрагмент сайта Lukew Ideation + Design

Переключение макетов. Этот способ наиболее удобен при чтении сайта с различных устройств: под каждое разрешение экрана разрабатывается отдельный макет. Способ трудоемок, поэтому менее популярен, чем предыдущие два (рис. 9.2).

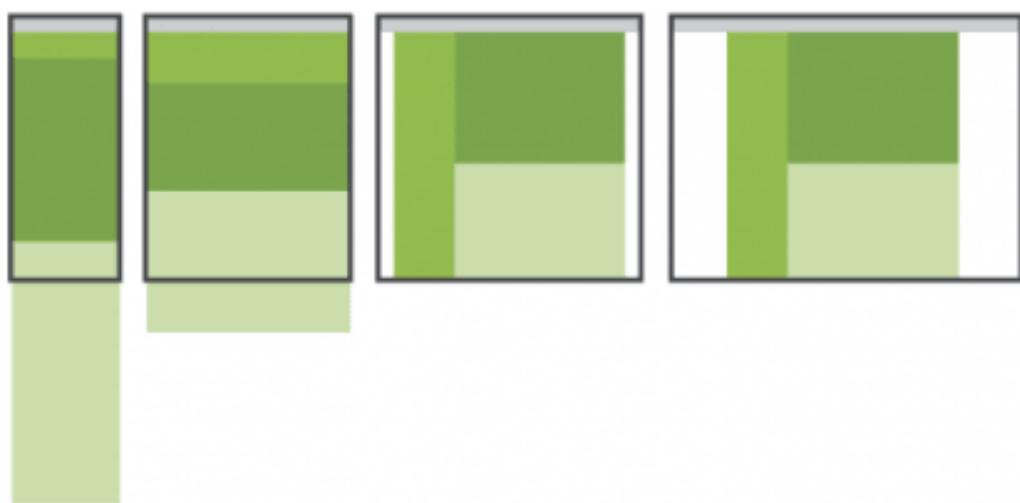


Рис. 9.2. Переключение макетов. Фрагмент сайта Lukew Ideation + Design

Адаптивность «малой кровью». Очень простой способ, который подходит для несложных сайтов. Достигается элементарным масштабированием изображений и типографики. Не очень популярен, т. к. не обладает гибкостью (рис. 9.3).

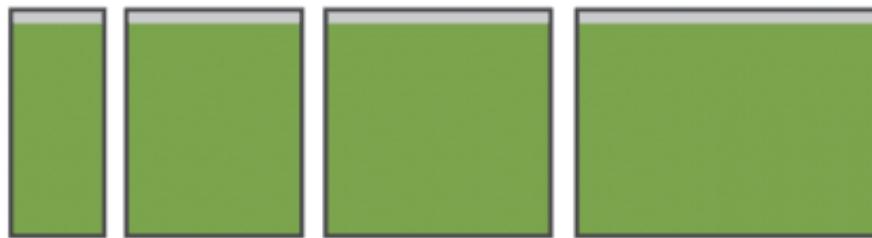


Рис. 9.3. Адаптивность «малой кровью». Фрагмент сайта Lukew Ideation + Design

Панели. Способ, пришедший из мобильных приложений, где дополнительное меню может появляться при горизонтальном или вертикальном тапе. Главный недостаток – неочевидность действий для пользователя: очень непривычно видеть мобильную навигацию на веб-сайтах. Но со временем способ может стать достаточно популярным (рис. 9.4).

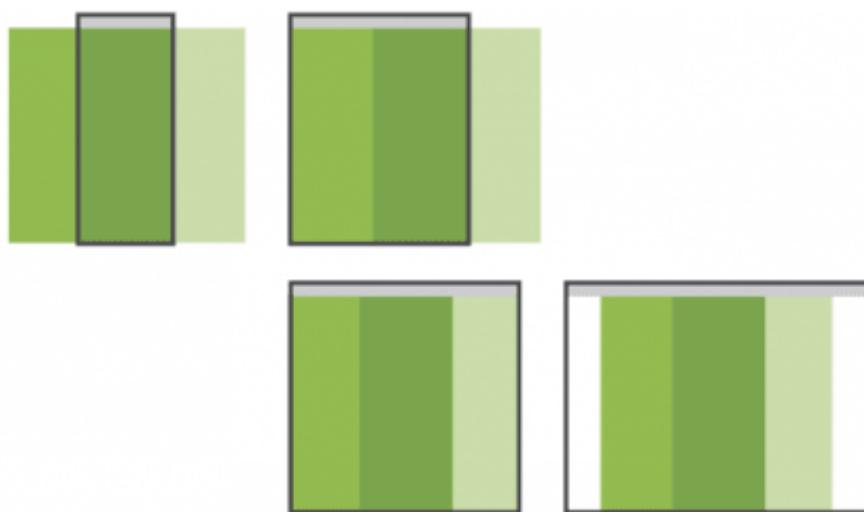


Рис. 9.4. Панели. Фрагмент сайта Lukew Ideation + Design

Представленные выше макеты не являются универсальными решениями – для каждого проекта необходимо выбирать наиболее подходящий под нужды и возможности способ.

Адаптивный подход ставит перед дизайнером две проблемы: различные разрешения экранов и выбор контента.

Проблема разрешения решается при помощи технологии медиа-запросов, позволяющей менять ширину макета, количество колонок, формат изображений и текста. На данный момент их поддерживают все мобильные браузеры. Резиновые макеты позволяют не ломать колонки с другими элементами интерфейса.

Модульные сетки существенно облегчают создание сайта, одинаково корректно работающего в различных системах. Особенностью модульных сеток в адаптивном дизайне являются блоки различных размеров и постоянная подгрузка содержимого. Подгрузка как прием изначально пришла из приложений для социальных сетей. Оказалось, что наиболее интуитивно понятное движение – протягивание края страницы вверх («свайп»), часто сопровождающееся характерным звуком. Этот шаблон моторики настолько общепринят, что менять его не имеет смысла.

Особое значение приобретает иерархия контента, так как при адаптивном подходе нет отдельной десктопной версии, куда можно было бы обратиться за полной информацией. Для того, чтобы уместить на экране необходимое и достаточное количество данных, дизайнеры прибегают к замене части текстового контента на аудио и видео, упрощают навигацию и убирают менее значимые элементы под выпадающие списки. Однако не всегда эти методы приводят к положительному пользовательскому опыту.

Скорость работы на мобильном устройстве – величина, которую нужно учитывать не только программистам, но и дизайнерам для того, чтобы оптимально ранжировать контент, необходимо изучить аудиторию издания.

Таким образом, адаптивный подход к дизайну перестает быть просто способом решить проблему различных форматов и разрешений. Адаптивный метод – это набор гибких сценариев.

Юзабилити мобильных приложений

При работе с мобильным устройством пользователь, как правило, находится в движении: едет в транспорте, стоит в очереди, идет пешком, сидит в кафе, беседует, слушает лекцию и т. п.

Следует также отметить, что изначальная функция мобильного телефона – голосовая связь, и сеанс такой связи может также прервать работу пользователя с продуктом.

Такая деятельность (по сравнению с идеальными офисно-лабораторными условиями) отнимает существенное количество когнитивных ресурсов: пользователь часто отвлекается, прерывается для выполнения других (внешних по отношению к используемому приложению) задач.

Соответственно, хотя бы для частичной компенсации и учета этих факторов следует выполнять следующие требования:

– должен быть четко обозначен контекст выполняемой в приложении операции – в первую очередь за счет коротких, но информативных заголовков, корректного именования ссылок и командных кнопок, других объектов;

– максимальное количество действий должно носить обратимый характер (программа должна быть терпима к случайным ошибкам пользователя – например, случайный, неправильный ввод возможен за счет непроизвольных и случайных толчков при движении в толпе или транспорте);

– в интерфейсе не должно быть ничего лишнего, что отвлекало бы внимание пользователя или запутывало бы его в принятии решений. Структура возможных действий и их последствий должна быть максимально явной и очевидной. Каждый символ, каждый пиксель должен быть обоснован, служить цели и выполняемой задаче пользователя;

– не следует экономить на памяти пользователя – критическая информация, важная для выполнения задачи, должна быть доступной.

Цели пользователей. Пользователи мобильных устройств (МУ) часто имеют интересы, отличающиеся от пользователей стационарных устройств: пользователи МУ обычно имеют более прямые, непосредственные и целенаправленные намерения.

Например, показательной типичной целью мобильного пользователя может быть нахождение информации, непосредственно связанной с контекстом среды, в которой находится пользователь: найти информацию о расписании движения транспорта во время совершающего путешествия. Пользователи МУ обычно мало заинтересованы в длинных документах и их тщательном прочтении, рассматривании, изучении.

Требования к интерфейсу мобильного устройства:

1. Перед началом проектирования интерфейса продукта следует четко специфицировать цели, задачи и информационные потребности пользователя продукта, описать среду и условия применения.

2. Дизайн приложения должен быть максимально простым и понятным.

3. Элементы управления на экране должны бросаться в глаза. Смартфоны используются людьми, которые стоят, ходят, находятся в разнообразном оживленном окружении.

4. Информация должна быть представлена в сжатом, но информативном виде.

5. Лишних и второстепенных деталей не должно присутствовать в интерфейсе.

6. Следует использовать принцип «вложенности», т. е. давать информацию порциями, когда каждый информационный элемент может быть более подробно расшифрован на отдельной странице.

7. Объемную информацию (приемлемую для отображения на больших дисплеях) следует специально форматировать в виде, пригодном для отображения на МУ. Так выводы и итоговые данные следует помещать в начало страницы, ключевые слова и критические данные – в начало абзаца, в таблицах отображать только самые важные столбцы, итоговые строки выносить в начало таблицы и т. д.

8. Следует особое внимание уделять соблюдению наивысшего контраста текста: располагать темный текст на светлом фоне, или наоборот – светлый текст на темном фоне (наивысший контраст дает отношение черный/белый). Экран мобильного устройства редко находится в условиях идеального освещения. ЖК-дисплеи, широко применяемые в мобильных устройствах, чувствительны к яркому (солнечному) освещению – при таком освещении дисплеи теряют контрастность, и разглядеть на них текст бывает затруднительно.

9. Желательно избегать ярких фоновых заливок, использования графических обоев, затрудняющих восприятие текстовой информации. Если все-таки таковые используются, то следует применять двухцветные шрифты (черные символы с белой окантовкой).

10. По возможности следует применять крупный шрифт или обеспечить пользователя возможностью изменять размер шрифта.

Требования юзабилити, налагаемые особенностями оборудования мобильного устройства

Малоразмерный экран. Большинство мобильных устройств имеет ограниченный размер экрана (большинство устройств имеет экран размером 128×128 пикселей). На таком экране невозможно разместить много информации.

Создавайте визуальные якоря. Для решения задачи пользователю смартфона часто приходится пройти через несколько экранов

нов. Нужно применять визуальные якоря, помогающие пользователям ориентироваться

Элементы управления следует делать достаточно крупными, чтобы их можно было активировать пальцами. Минимальный размер – 7–10 мм.

Используйте крупные шрифты без засечек. Четко указывайте наличие дополнительной информации за пределами экрана. Многие люди не привыкли к идее маленького экрана, требующего прокрутки информации. Если данных больше, чем помещается на экране, не забудьте подчеркнуть это обстоятельство. В идеале дайте пользователю понять, как получить доступ к дополнительным данным.

Если предполагается перевод интерфейса на другие языки, необходимо учитывать, что в некоторых языках слова длиннее, чем в других. Необходимо выделять элементам, содержащим текст, достаточное количество драгоценного места на экране.

Учитывайте мобильный контекст использования. Люди имеют обыкновение пользоваться смартфонами на ходу, стоя, в общественном транспорте и т. д. Интерфейс должен быть таким, чтобы приложением было удобно пользоваться в разных ситуациях.

Необходимо использовать технические возможности смартфонов при проектировании интерфейсов, так как это влияет на пользовательский интерфейс. Практически все смартфоны на Android снабжены сенсорными экранами, реагирующими на нажатие пальца. Большинство современных смартфонов поддерживает технологию мультитач (мультикасание) – функцию сенсорных систем ввода, осуществляющую одновременное определение координат двух и более точек касания. Мультикасание используется в жестовых интерфейсах, например, для изменения масштаба изображения. Кроме того, экраны мультикасания позволяют работать с устройством одновременно нескольким пользователям.

Следует уделить особое внимание расположению элементов управления. Области, прикосновение к которым вызывает серьезные последствия (уход с экрана, удаление данных и т. п.), не должны быть доступны для случайного нажатия. Рекомендуется их располагать вверху. В то же время часто используемые элементы лучше всего размещать внизу. При этом не следует назначать одному из нижних углов наиболее используемое действие. От четверти до трети пользователей часто или даже всегда держат

телефон в левой руке. Считая, что телефон будут держать исключительно в правой, вы рискуете получить негативные отзывы и потерять часть клиентов.

Стивен Хубер в своем исследовании использования мобильных устройств определил, что 49% людей для решения задач полагаются только на большой палец (рис. 9.5). На рисунке ниже, диаграмма, показанная на экране мобильного телефона, является ориентировочной схемой охвата, в которой цвета указывают, каких областей пользователь может достичь большим пальцем, чтобы взаимодействовать с экраном.

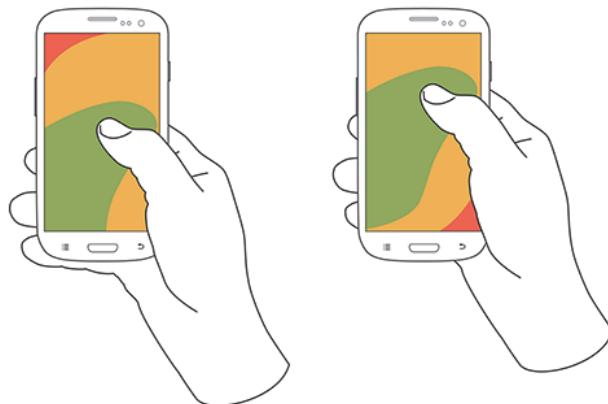


Рис. 9.5. Принцип большого пальца

Зеленый цвет указывает на область, которой пользователь может легко достичь; желтый – область, которая требует усилия; и красный – область, требующую от пользователя изменения способа удерживания телефона. Положение рук и сцепление влияют на размещение элементов управления в мобильном дизайне. Очень важно поместить меню верхнего уровня, часто используемые элементы управления и общие действия в область, которой пользователь может легко достичь, чтобы до них было проще дотянуться большим пальцем. Поместите негативные действия, такие, как удаление, в труднодоступную область, чтобы пользователь случайно на них не нажал.

Элементы интерфейса должны быть ясно видны. Поэтому необходимо обеспечить достаточный цветовой контраст между элементами, чтобы пользователи со слабым зрением могли видеть и использовать ваше приложение без проблем.

Цвет шрифта и фона должны достаточно контрастировать – это упущение может привести к недостаточной разборчивости.

Рекомендуются следующие показатели контрастности для основного текста и текста изображения:

- небольшой текст должен иметь коэффициент контрастности не менее 4,5 : 1 к его фону;
- большой текст (14 пт жирным шрифтом / 18 пт обычным, и выше) должен иметь коэффициент контрастности не менее 3 : 1 к его фону;
- иконки и другие важные элементы также должны использовать вышеуказанные рекомендуемые показатели контрастности (рис. 9.6).

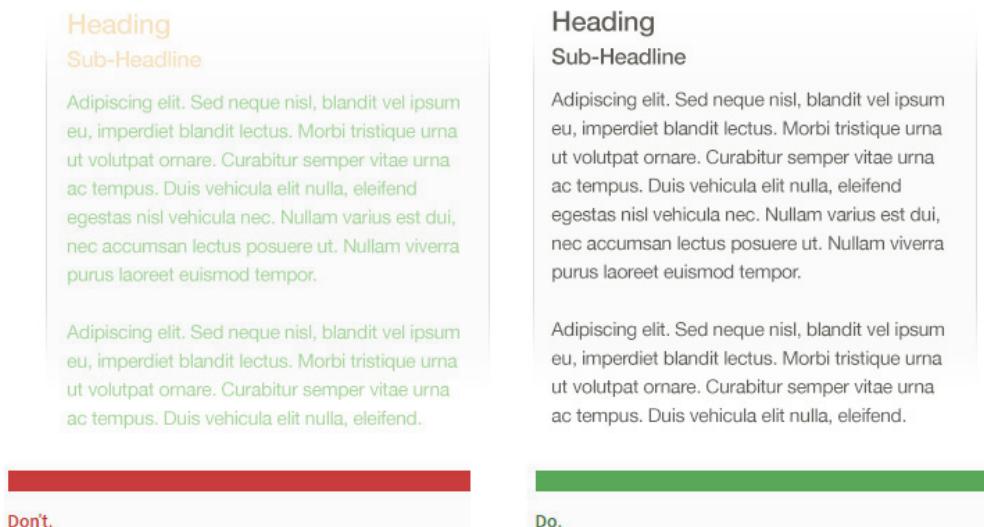


Рис. 9.6. Пример контрастности

Необходимо учитывать размер пальцев пользователя. Да, пальцы многих пользователей гораздо больше, чем могли бы представить любители утонченного дизайна. Поэтому обязательно нужно адаптировать свою программу к пальцам разного размера.

Кнопкам просто нужно оставлять достаточно места. Если кнопки слишком малы или расположены слишком близко друг к другу, некоторые люди просто не смогут попадать по ним. Как следствие, пользователи будут раздражаться и, может быть, прекращать работу с такой программой. Наши пальцы действительно большие. Их ширина составляет около 45–57 пикселей, что больше, чем рекомендует большинство руководств для тестовых устройств. Apple, например, рекомендует цель квадратной формы

с размером стороны в 44 пикселя. А этого далеко не всегда достаточно.

Свободное пространство является важным элементом интерфейса МУ. Чем больше свободного места, тем меньше элементов и, соответственно, тем проще для пользователя бегло их просмотреть. Искусство применения пустого пространства состоит в том, чтобы предоставить пользователям столько информации (в блоках), сколько они смогут обработать, а затем убрать все лишние детали (рис. 9.7).

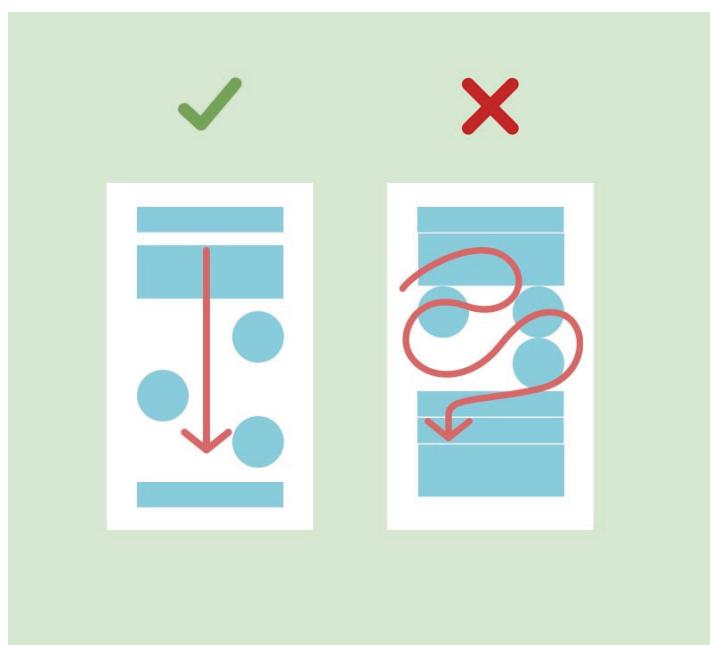


Рис. 9.7. Применение пустого пространства

Влияние размера экрана на разработку приложения. Устройства Android различаются не только физическими размерами, но и разрешением экрана. В руководстве для разработчиков Android приводятся следующие определения.

Разрешение – общее число физических пикселей на экране.

Экранная плотность – количество пикселей в физической области экрана, как правило, называют DPI (точек на дюйм).

Плотностно-независимая точка (DP) – это виртуальная единица-пиксель, которая используется при определении интерфейса макета, чтобы сделать размеры макета или его положения независимым от плотности образа. Плотностно-независимая точка эквивалентна одному физическому пикслю на 160 DPI, которая является базовой плотностью и берется в системе как «средняя»

плотность экрана. Во время работы система в фоновом режиме обрабатывает любое масштабирование единиц DP по мере необходимости, исходя из фактической плотности использованного экрана. Преобразование единиц DP в пиксели выполняется следующим образом: пиксель = DP · (DPI / 160).

Для упрощения ситуации устройства Android по физическим размерам подразделяются на два типа: телефонный (разрешение меньше 600 DP) и планшетный (разрешение больше или равно 600 DP). Android делит экран плотности на пять основных плотностей:

- 1) LDPI (низкий);
- 2) MDPI (средний);
- 3) HDPI (высокий);
- 4) XHDPI (очень высокий);
- 5) XXHDPI (очень-очень высокий).

Это важно, поскольку разработчику необходимо предоставить все графические ресурсы (растровые изображения) в наборах различной плотности. Для любого смартфонного приложения необходимо иметь хотя бы MDPI и HDPI наборы.

Это связано с минимальным размером элементов управления, допустимым для тач-интерфейсов. При этом получается компромиссное решение, учитывающее и общую плотность доносимой до пользователя информации, и возможности элементов управления (рис. 9.8).

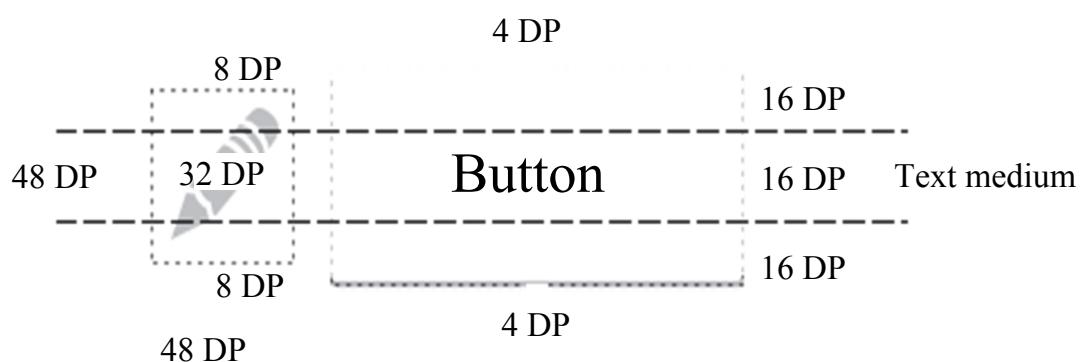


Рис. 9.8. Размеры элементов управления

Элементы управления должны быть размером по крайней мере 48 DP. Расстояние между элементами управления не должно быть меньше 8 DP.

Меню навигации для малых дисплеев. Веб-дизайнерам известно, насколько по-разному ведут себя в сети пользователи мобильных устройств – смартфонов, а также прочих карманных приборов – и пользователи стационарных ПК или ноутбуков. Найти место для схемы навигации в пределах малого экрана непросто, но спроектировать мобильную систему навигации вполне возможно.

Практический метод 1: сокращение количества кнопок в меню. Ограничтесь пятью или менее кнопками в основной схеме навигации, при необходимости дополнив ее вложенными меню.

Практический метод 2: сокращение количества нажатий. Сведите к минимуму количество действий, выполняемых над мобильным устройством. Чем меньше нажатий и кликов приходится совершать пользователю на пути к искомому результату, тем лучше. Всегда включайте в структуру страницы хорошо заметную кнопку «Назад»: не нужно усложнять пользователю жизнь.

Практический метод 3: прокрутка и навигация. Выносите важнейшие элементы навигации вверх. У мобильных устройств дисплеи разной величины, так что тщательно обдумывайте каждое решение, ведь доступ к «нижним» элементам может требовать прокрутки на экранах смартфонов меньших габаритов.

Практический метод 4: «штабелирование» кнопок. Пользователи мобильных устройств видят ваш сайт в портретной ориентации, а не в альбомной, как на экране стационарного ПК. Приспособливайте вашу схему навигации к условиям мобильной среды, вытягивайте ее по вертикали. Пусть страница читается вниз, а не поперек. Лучше всего этот метод срабатывает для сайтов, в чьем меню не более пяти пунктов и расширение его не предвидится.

Практический метод 5: вложенные меню. Иногда такую схему навигации еще называют выдвижной, она удобна для сайтов, страницам которых не обойтись без контента, а не только без меню, да еще в тех случаях, когда со временем ожидается добавление разделов и пунктов меню.

Оптимизация мобильных систем навигации под сенсорные устройства и крохотные кнопки – важная задача. У пользователей мобильных устройств нет «волшебной» мышки для высокоточного попадания одним нажатием по миниатюрным кнопкам или ссылкам меню. Им приходится полагаться только на собственные пальцы, а значит, на участки поверхности экрана они нажимают

гораздо менее прицельно, чем могли бы с помощью мыши. Необходимо очистить пространство вокруг ссылок меню, либо заменить их кнопками доступа к пунктам меню.

Чем крупнее объект, тем он заметнее, тем скорее его обнаружат и кликнут по миниатюрной кнопке телефона или нажмут пальцем на точку поверхности экрана. Для ощутимого контакта с подушечкой пальца средних пропорций нужна площадь поверхности как минимум 44×44 пикселя, кнопки и элементы навигации могут совсем заслонить собой остальное содержимое экрана. Тем не менее, и владелец, и пользователь веб-сайта могут остаться в выигрыше, особенно, если контент домашней страницы менее важен, чем броскость пунктов меню.

В идеале, дизайн страницы должен позволять пользователю сенсорного устройства при необходимости укрупнять кнопки схемы навигации.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Кишкурно, Т. В. Дизайн и юзабилити интерфейсов пользователя: тексты лекций по одноименной дисциплине для студентов специальности 1-40 05 01-03 «Информационные системы и технологии» (издательско-полиграфический комплекс) / Т. В. Кишкурно. – Минск: БГТУ, 2017. – 155 с.
2. Дизайн и юзабилити интерфейсов пользователя: учеб.-метод. пособие по курсовому проектированию по одноименной дисциплине для студентов специальности 1-40 05 01-03 «Информационные системы и технологии» (издательско-полиграфический комплекс) / сост.: Т. В. Кишкурно, Т. П. Брусенцова. – Минск: БГТУ, 2016. – 76 с.
3. Купер, А. Об интерфейсе. Основы проектирования взаимодействия / А. Купер, Р. Рейман, Д. Кронин. – СПб.: Символ-Плюс, 2009. – 649 с.
4. Унгер, Р. UX-дизайн. Практическое руководство по проектированию опыта взаимодействия / Р. Унгер, К. Чендлер; пер. с англ. – СПб.: Символ-Плюс, 2011. – 327 с.
5. Головач, В. В. Дизайн пользовательского интерфейса. Искусство мыть слона / В. В. Головач. – Минск: БГТУ, 2008. – 94 с. – Режим доступа: <http://uibook2.usethics.ru/uibookII.pdf>. – Дата доступа: 01.03.2018.
6. Нильсен, Я. Веб-дизайн. Книга Якоба Нильсена / Я. Нильсен; пер. с англ. – СПб.: Символ-Плюс, 2003. – 512 с.: цв. ил.
7. Круг, С. Веб-дизайн: книга Стива Круга, или Не заставляйте меня думать! / С. Круг. – Пер. с англ. – СПб.: Символ-Плюс, 2008. – 295 с.
8. Калиновский, А. Юзабилити: как сделать сайт удобным / А. И. Калиновский. – Минск: Новое знание, 2005. – 220 с.: ил.
9. Кирсанов, Д. Веб-дизайн: книга Дмитрия Кирсанова / Д. Кирсанов, А. Кирсанова. – СПб.: Символ-Плюс, 1999. – 376 с.: цв. ил.
10. UX-дизайн. Идея – эскиз – воплощение / С. Гринберг [и др.]. – СПб.: Питер, 2014. – 272 с.: ил.
11. Макнейл, П. Веб-дизайн. Идеи, секреты, советы / П. Макнейл. – СПб.: Питер, 2012. – 272 с.: ил.
12. Купер, А. Психбольница в руках пациентов / А. Купер. – СПб.: Символ-Плюс, 2004. – 295 с.

13. Бикнер, К. Экономичный Web-дизайн / К. Бикнер; пер. с англ. Д. С. Ремизова. – М.: НТ Пресс, 2005. – 248 с.: ил.
14. Уодке, К. Информационная архитектура. Чертежи для сайта / К. Уодке. – М.: КУДИЦ-Образ, 2004. – 256 с.: ил.
15. Нейл, Т. Мобильная разработка. Галерея шаблонов / Т. Нейл. – СПб.: Питер, 2013. – 208 с.: ил.
16. Lund, A. M. Measuring usability with the USE questionnaire / A. M. Lund // Usability Interface. – 2001. – № 8(2). – P. 3–6.
17. Контрольный список Веб-интерфейса [Электронный ресурс]. – Режим доступа: http://ddd.exmachina.ru/web/web_cheklist/. – Дата доступа: 01.03.2018.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ	3
<i>Глава 1. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ</i>	4
<i>Глава 2. ВЕБ-ЭРГОНОМИКА И ЮЗАБИЛИТИ</i>	10
<i>Глава 3. КРИТЕРИИ КАЧЕСТВА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ</i>	18
<i>Глава 4. ОСОБЕННОСТИ ВОСПРИЯТИЯ ЧЕЛОВЕКОМ ИНФОРМАЦИИ</i>	52
<i>Глава 5. ПРОЦЕСС ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА</i>	68
<i>Глава 6. РАЗРАБОТКА ПРОТОТИПА</i>	95
<i>Глава 7. ВИЗУАЛЬНАЯ КУЛЬТУРА ДИЗАЙНА ИНТЕРФЕЙСА.....</i>	113
<i>Глава 8. ЮЗАБИЛИТИ-ТЕСТИРОВАНИЕ.....</i>	132
<i>Глава 9. ОСОБЕННОСТИ РАЗРАБОТКИ ИНТЕРФЕЙСОВ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ</i>	149
СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ	170

Учебное издание

**Брусенцова Татьяна Палладьевна
Кишкурно Татьяна Вадимовна**

ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСОВ ПОЛЬЗОВАТЕЛЯ

Пособие

Редактор *Ю. Д. Нежикова*
Компьютерная верстка *Ю. Д. Нежикова*
Дизайн обложки *П. П. Падалец*
Корректор *Ю. Д. Нежикова*

Подписано в печать 22.11.2019. Формат 60×84¹/16.
Бумага офсетная. Гарнитура Таймс. Печать ризографическая.
Усл. печ. л. 10,1. Уч.-изд. л. 10,3.
Тираж 70 экз. Заказ .

Издатель:
УО «Белорусский государственный технологический университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/227 от 20.03.2014.
Ул. Свердлова, 13а, 220006, г. Минск.