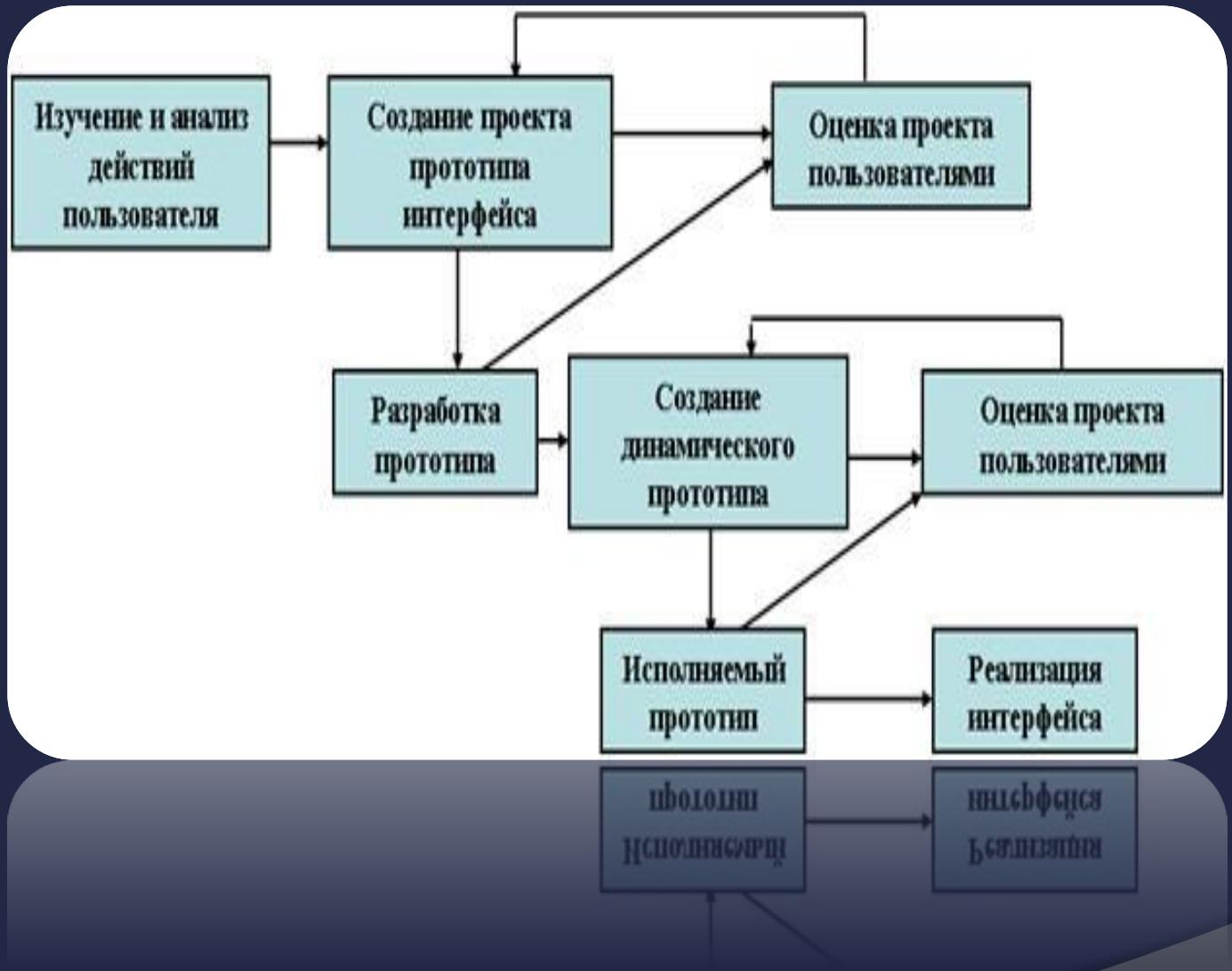
ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

Федоришин Н.Е.


Изучение и анализ действий пользователя
Создание проекта прототипа интерфейса
Оценка проекта пользователями
Разработка прототипа
Создание динамического прототипа
Оценка проекта пользователями
Исполняемый прототип
Реализация интерфейса

НОВЫ ПОСТРОЕНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Когда говорят о научных основах проектирования пользовательских интерфейсов, в первую очередь упоминают термин HCI. HCI — это аббревиатура английского Human-Computer Interaction, что переводится как "взаимодействие человека и компьютера". На Западе HCI — это целая профессия, ей обучают в университетах, выдавая дипломы "Специалист по HCI". Издается много журналов по этой теме, существует большое количество Web-сайтов. В России, к сожалению, эта наука не пользуется особой популярностью например, у нас настоящих специалистов по HCI можно буквально пересчитать по пальцам одной руки.

Как легко догадаться по названию, составными частями HCI являются:

- человек (пользователь)
- компьютер
- их взаимодействие.

Пользовательский интерфейс (англ. user interface, UI) является своеобразным коммуникационным каналом, по которому осуществляется взаимодействие пользователя и компьютера.

Лучший пользовательский интерфейс — это такой интерфейс, которому пользователь не должен уделять много внимания, почти не замечать его. Пользователь просто работает, вместо того, чтобы размышлять, какую кнопку нажать или где щелкнуть мышью. Такой интерфейс называют *прозрачным* — пользователь как бы смотрит сквозь него на свою работу.

Если говорить о самых **общих принципах проектирования пользовательских интерфейсов**, то можно назвать три основных положения:

- 1. Программа должна помогать выполнить задачу, а не становиться этой задачей.
- 2. При работе с программой пользователь не должен ощущать себя дураком.
- 3. Программа должна работать так, чтобы пользователь не считал компьютер дураком.

Довольно эмоциональные формулировки, но, тем не менее, поразительно верные.

РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕСА

Интерфейс пользователя - эта та часть программы, которая находится у всех на виду. Некоторые программисты склонны оставлять дизайн интерфейса пользователя на потом, считая, что реальное достоинство приложения - его программный ко. который и требует большего внимания. Однако часто возникает недовольство пользователей из-за неудачно подобранных шрифтов, непонятного содержимого экрана и скорости его прорисовывания, поэтому работу над интерфейсом также нужно воспринимать серьезно. Пользователь не вши, программного кода, зато интерфейс (хороший или плохой) всегда перед ним.
Разработка эффективных форм

Формы - это строительные блоки интерфейса пользователя. Хороший дизайн форм включает нечто большее, чем просто добавление элементов управления и программирование процедур обработки событии. Чтобы создать хорошо спроектированную форму, вы должны уяснить ее назначение, способ и время использования, а также ее связи с другими элементами программы. Кроме того в приложении может находиться несколько форм, каждая из которых будет отображаться по мере необходимости. Одни пользователи широко используют многозадачность Windows, другие предпочитают работать только с одним приложением. Необходимо помнить об этом во время разработки интерфейса пользователя (UI) Вы должны максимально реализовать все возможности Windows, чтобы пользователи с любыми навыками работы могли эффективно применять созданное вами приложение.

Проектирование форм ввода данных

Особый вид форм - формы, предназначенные для ввода данных. Они позволяют пользователь в нужном ему темпе, не оглядываясь на программиста. Общий смысл и основное правы: если пользователь собирается ввести в базу данных 10000 записей, вероятно, он не подтверждать ввод каждой записи. В форме ввода данных необходимо максимально использовать свободное пространство, поскольку открытие и закрытие дополнительных форм существенно замедляет работу. При разработке форм ввода данных основное внимание следует уделить скорости их работы. Чтобы максимально ускорить процесс ввода данных, следуйте приведенным ниже основным правилам.

· Всегда назначайте клавиатурные эквиваленты команд; не требуйте обязательного использования мыши. (Кстати, этот совет хорош для всех форм программы, а не только для форм ввода данных.)

· Расположение элементов должно быть согласовано с задачами пользователя. Другими ело вами, не заставляйте пользователя перепрыгивать из раздела в раздел; при вводе информации это совсем не обязательно.

· Не заставляйте пользователя выполнять лишнюю работу. Другими словами, если информация, содержащаяся в полях со 2-го по 10-е, необходима только, когда первое поле имеет определенное значение, не нужно заставлять пользователя заполнять все поля подряд. же время, не ставьте работу формы в зависимость от содержимого отдельных полей. В противном случае это может существенно замедлить работу пользователя.

· Используйте заметную, но ненавязчивую обратную связь с пользователем. Хороший пример - работа редактора программного кода Visual Basic, который проверяет правильной написания переменных и констант.

· Если возможно, выполняйте добавление и редактирование записей в одной и той же форме, тогда пользователю не придется осваивать несколько методов доступа к одним и тем желанным.

- **Работа с несколькими формами**

Если интерфейс пользователя должен содержать несколько форм, вам предстоит принять самое важное решение: какой использовать вид интерфейса- однодокументный (SDI) или многодокументный (MDI).
В SDl-приложениях окна форм появляются совершенно независимо друг от друга. Однако, не имеет значения какой тип интерфейса SDI или MD1 выбран; взаимодействие пользователя с формами происходит одинаково -посредством обработки событий, поступающих от элементов управления формы. Поэтому, если в вашем приложении предусмотрено несколько форм программу необходимо написать так, чтобы у пользователей не было возможности нарушить предписанные ход ее выполнения (например, у пользователя не должно быть средств вывести форму, для которой еще не готова информация).

- **Эффективные меню**

Еще одна важная часть разработки форм - создание содержательных и эффективных меню. Приведем некоторые важные рекомендации:
· Следуйте стандартным соглашениям о расположении пунктов меню принятым в Windows File, Edit, View, и т.д.
· Группируйте пункты меню в логическом порядке и по содержанию.
· Для группировки пунктов в раскрывающихся меню используйте разделительные линии
· Избегайте избыточных меню.
· Избегайте пунктов меню верхнего уровня, не содержащих раскрывающихся меню
· Не забывайте использовать символ троеточия для обозначения пунктов меню, активизирующих диалоговые окна.
· Обязательно используйте клавиатурные эквиваленты команд и "горячие" клавиши.
· Помещайте на панель инструментов часто используемые команды меню.

- **Ощущение скорости**

Ощущение - это реальность. Здесь мы попытаемся объяснить, как ощущения пользователя могут повлиять на то, понравится ему ваша программа или нет. Простой пример - скорость работы приложения. У вас может быть самый быстродействующий программный код, но это ничего не значит, если с точки зрения пользователя он работает медленно. Когда пользователи жалуется на скорость, программисты защищаются, утверждая, что "пользователь не знает, что делает программа". Однако, если воспользоваться некоторыми уловками, то можно сделать та что будет казаться, будто программа работает быстрее.

Пользователь гораздо более расположен к ожиданию, если считает, что компьютер работает с максимальной скоростью. Хороший пример- загрузка Windows, которая обычно требует достаточно много времени. Однако вывод графики, сопровождающие звуки, шум жесткого диска отвлекают настолько, что пользователь не ощущает ожидания. Описанная ниже техника поможет в создании "более быстрых" приложений.

- **Информируйте пользователя о ходе процесса**

Когда есть видимость работы приложения, пользователи более легко переносят длительное ожидание в работе программы. Один из способов информирования пользователя о ходе выполнения работы - использовать в форме индикатор процесса. Если вы обновляете записи базы данных, можно использовать такой индикатор для отображения числа записей, над которыми операция уже произведена. Для этого добавьте пару строк кода обновляющих показания индикатора по мере перехода к следующим записям.

Выводы по проектированию пользовательского интерфейса

Хотя ни одно ухищрение не гарантирует создания удачного интерфейса пользователя, плохой интерфейс гарантирует отсутствие пользователей вашей программы. Однако при стремительном появлении новшеств, в сфере пользовательских интерфейсов, понятие "хорошего" интерфейса очень быстро изменяется. Возьмем, например, процесс настройки часов видеомагнитофона. Раньше часы программировали "вслепую" кнопками и переключателями, потом стали применяться дисплеи, для которых использовался экран телевизора, а теперь в некоторых моделях этот процесс выполняется автоматически по радиосигналу. Так и интерфейс пользователя программ будет "эволюционировать" по мере того как индустрия будет устанавливать новые стандарты, и вы, в свою очередь, должны быть всегда в курсе, как в наибольшей степени удовлетворить ожидания пользователя.

ПРИНЦИПЫ ПОСТРОЕНИЯ ИНТЕРФЕЙСА

- **Золотое сечение**

Золотое сечение — это самая комфортная для глаза пропорция, форма, в основе построения которой лежит сочетание симметрии и золотого сечения, способствует наилучшему зрительному восприятию и появлению ощущения красоты и гармонии.

В математике пропорцией называют равенство двух отношений: a : b = c: d.

Отрезок прямой АВ можно разделить точкой С на две части следующими способами:

- на две равные части АВ : АС = АВ : ВС,
- на две неравные части в любом отношении (такие части пропорции не образуют);
- таким образом, когда АВ: ВС = ВС: АС.

Последнее и есть золотое деление или деление отрезка в крайнем и среднем отношении.

Золотое сечение — это такое пропорциональное деление отрезка на неравные части, при котором весь отрезок так относится к большей части, как сама большая часть относится к меньшей; или другими словами, меньший отрезок так относится к большему, как больший ко всему.

a : b = b : c или c : b = b : a.

Отрезки золотой пропорции выражаются бесконечной иррациональной дробью 0,618..., если с принять за единицу, а = 0,382. Отношение же отрезков а и b составляет 1,618.

Прямоугольник с таким отношением сторон стали называть *золотым прямоугольником*. Он также обладает интересными свойствами. Если от него отрезать квадрат, то останется вновь золотой прямоугольник. Этот процесс можно продолжать до бесконечности. А если провести диагональ первого и второго прямоугольника, то точка их пересечения будет принадлежать всем получаемым золотым прямоугольникам.

Есть и золотой треугольник (равнобедренный треугольник, у которого отношение длины боковой стороны к длине основания равняется 1,618), и золотой кубоид (прямоугольный параллелепипед с ребрами, имеющими длины 1,618, 1 и 0,618).

Золотое сечение не является искусственным явлением. Оно очень широко распространено в природе: золотое сечение можно найти в пропорциях тел многих растений и животных, а также морских раковин и птичьих яиц. Но наиболее впечатляющий пример "применения" природой принципа золотого сечения — человеческое тело. Оно целиком и его части (лицо, руки, кисти рук и т. п.) насквозь пронизаны пропорцией 1,618.

Принцип золотого сечения был открыт людьми еще в глубокой древности. Знаменитые египетские пирамиды в Гизе, например, основаны на пропорциях золотого сечения. Более молодые мексиканские пирамиды и античный храм Парфенон также содержат в себе пропорцию 1,618.

С развитием дизайна и технической эстетики действие закона золотою сечения распространилось на конструирование машин, мебели и т. д. Проектирование компьютерных интерфейсов — не исключение. Формы диалоговых окон и элементов управления, стороны которых относятся как 1,618, очень привлекательны для пользователей. Например, очень много восторгов у пользователей программыChameleon Clock (http://www.softshape.com) вызывает такая, казалось бы, обыденная вещь, как вид диалогового окна Свойства. А все потому, что при его проектировании использовался именно принцип золотого сечения.

- **Кошелек Миллера**

Этот принцип назван так в честь ученого-психолога Г. А. Миллера, который исследовал кратковременную память, проверяя выводы, сделанные ранее его коллегой, Г. Эббингаузом. Эббингауз пытался выяснить, сколько информации может запомнить человек без каких-либо специальных мнемонических приемов. Оказалось, что емкость памяти ограничена семью цифрами, семью буквами или названиями семи предметов. Это "магическое число" семь, служащее своего рода меркой памяти, и было проверено Миллером, который показал, что память действительно в среднем не может хранить более семи элементов; в зависимости от сложности элементов это число может колебаться в пределах от пяти до девяти.

Если необходимо в течение короткого времени сохранить информацию, включающую больше семи элементов, мозг почти бессознательно группирует эту информацию таким образом, чтобы число запоминаемых элементов не превышало предельно допустимого. Например, номер банковского счета 30 637 402 710, состоящий из одиннадцати элементов, будет, скорее всего, запоминаться как 30 63 740 27 10, т. е. как пять числовых элементов, или восемь слов (тридцать, шестьдесят, три, семьсот, сорок, двадцать, семь, десять).

Применяя принцип кошелька Миллера в дизайне интерфейсов, следует группировать элементы в программе (кнопки на панелях инструментов, пункты меню, закладки, опции на этих закладках и т. п.) с учетом этого правила— т. е. не более семи в группе, в крайнем случае — девяти. Взгляните, например, на главное окно программы-словаря ABBYY Lingvo 6.0: четырнадцать кнопок на верхней панели, между которыми нет ни одного разделителя, воспринимаются гораздо хуже, чем кнопки на панели внизу, которые разделены на группы.

Итак, принцип кошелька Миллера говорит о семи плюс-минус двух элементах. Но если взглянуть на программы, интерфейс которых совершенствовался годами (тот же Microsoft Word), то можно заметить, что число объектов (пунктов меню, кнопок на панелях инструментов) в группах доходит до шести-семи довольно редко, а в основном элементы сгруппированы по три-четыре объекта. Такие небольшие группы объектов наиболее хорошо воспринимаются взглядом пользователя, уже слегка утомленного сложными интерфейсами современных программ. Я думаю, при проектировании интерфейсов программ верхнюю границу кошелька Миллера — семь-девять элементов — нужно применять очень осторожно, стараясь обходиться группами, содержащими максимум пять объектов.

- **Принцип группировки**

Согласно этому правилу, экран программы должен быть разбит на ясно очерченные блоки элементов, может быть, даже с заголовком для каждого блока. При этом группировка, естественно, должна быть осмысленной: как расположение элементов в группах, так и расположение самих групп друг от друга должны быть продуманы.

Примеров реализации этого принципа очень много: это уже упоминавшиеся при разговоре о кошельке Миллера пункты меню, кнопочные панели инструментов, а также сгруппированные по назначению флажки и переключатели, с помощью которых настраиваются параметры работы программы в диалоговых окнах Свойства, Настройка и т. п..

- **Бритва Оккама или KISS**

Философский принцип, носящий название "Бритва Оккама", гласит: "Не множить сущности без надобности". Или, как говорят американцы, KISS ("Keep It Simple, Stupid" — "Не усложняй, болван").

На языке интерфейсов это означает, что:

- любая задача должна решаться минимальным числом действий;
- логика этих действий должна быть очевидной для пользователя;
- движения курсора и даже глаз пользователя должны быть оптимизированы.

Простым на первый взгляд требованиям из этого списка на самом деле не так уж легко следовать. Для проектирования сложного по своим функциям и простого для понимания интерфейса требуется немалые опыт, знания и особое чутье. Как пишет Лу Гринзоу: "Если и есть в мире что-то такое, что почти все программисты постоянно повторяют наизусть, как мантры, но при этом откровенно игнорируют, так это — принцип KISS"'.

Принцип KISS перекликается с несколькими из эвристических правил Якоба Нильсена — "Эстетичный и минималистический дизайн", "Равенство между системой и реальным миром", "Понимание лучше, чем запоминание". KISS более универсален и применяется практически во всех сферах человеческой деятельности, в том числе и в области, очень близкой к теме книги—в программировании.

- **Видимость отражает полезность**

Смысл этого принципа состоит в том, чтобы вынести самую важную информацию и элементы управления на первый план и сделать их легкодоступными пользователю, а менее важную — переместить, например, в меню

Этот вопрос уже немного затрагивался при разговоре о принципе Якоба Нильсена "Эстетичный и минималистический дизайн", правда, в привязке к отражению полезности.

Отличие принципа "Видимость отражает полезность" как раз и состоит в том, что интерфейс программы должен быть построен вокруг объектов, с которыми манипулирует пользователь, и отражать состояние текущего объекта. Реализацию этого принципа вы видите каждый раз, когда пользуетесь компьютером: контекстные панели инструментов в программах пакета Microsoft Office, которые меняются в зависимости от того, с какой частью программы (редактором, предварительным просмотром, рисованием и т. п.) и данный момент работает пользователь. Еще один пример — уже упоминавшееся меню Пуск в Windows ME и Windows 2000: по умолчанию в них видимы наиболее часто используемые, т. е. полезные для пользователя, пункты.

- **Умное заимствование**

Заимствование широко распространенных приемов дизайна интерфейсов и удачных находок авторов конкурирующих программ позволяет резко сократить время обучения и повысить комфорт пользователя. При работе он будет использовать уже приобретенные навыки — этот вопрос затрагивает и принцип равенства между системой и реальным миром.

Заимствование чужих интерфейсных находок не является чем-то зазорным. Программы, лидирующие па рынке, являются неистощимым источником вдохновения для разработчиков более мелких программ, поразительно напоминающих легендарный Norton Commander: FAR, Volcov Commander, DOS Navigator,DISCo Commander

СОЗДАНИЕ ПРОТОТИПА ИНТЕРФЕЙСА

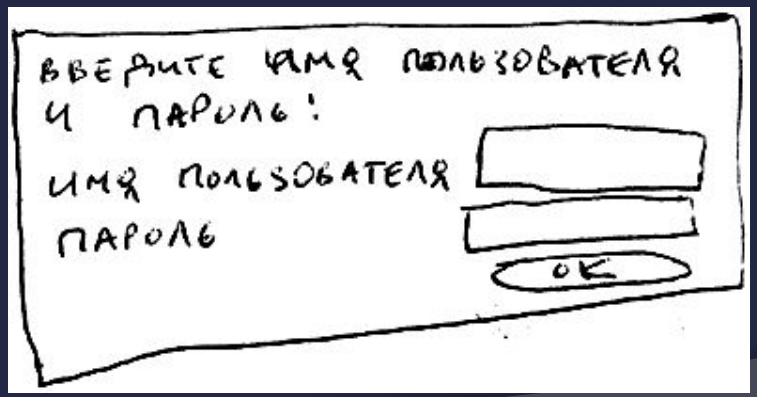
Создание максимально эффективного прототипа интерфейса является чрезвычайно важной задачей. Прототип должен хорошо выглядеть, чтобы понравиться заказчику и не вызвать вопросов у субъектов тестирования, он должен быть максимально дешев, максимально полон и, что немаловажно, должен с легкостью обновляться.

- **Первый тип – примитивный**

Самым эффективным способом создания ранних прототипов является рисование интерфейса на бумаге. Достоинствами бумаги являются исключительная простота и скорость рисования. Кроме того, бумага помогает не думать о том, как интерфейс выглядит, позволяя сосредоточиться на том, как интерфейс работает. При рисовании прототипа на бумажке очень важно не стараться нарисовать его так, чтобы он был красив, например, не нужно стараться рисовать линии прямыми. На ваше понимание работы интерфейса это не повлияет, зато здорово замедлит работу. Красоту же всё равно придется выбрасывать, когда вы нарисуете новую версию. Так что основным критерием живописности должна стать скорость работы.

Элементы интерфейса, которые нельзя нарисовать однозначно (например, раскрывающиеся списки, у которых значения до поры скрыты) эффективнее всего рисовать неоднозначными, важную же информацию из них лучше всего словами писать на полях.

Если вам хочется идти в ногу с прогрессом, вы можете воспользоваться системой DENIM. Эта программа «эмулирует» листок бумаги и ручку с ластиком, при этом позволяет снабжать получившийся псевдо-бумажный прототип зачаточной интерактивностью. Так, например, вы можете без труда сделать так, чтобы кнопка, которую вы нарисуете, автоматически открывала другой экран. К сожалению, DENIM обладает определенными недостатками. Во-первых, он очень функционально беден (как-никак проект некоммерческий). Во-вторых, DENIM сам является полигоном интерфейсных решений (в результате обычной панелью инструментов оказывается очень неудобно пользоваться). Впрочем, это имеет и свои достоинства – где ещё, например, можно увидеть в действии круглые меню?



- **Второй тип – полуреальный**
- Самым распространенным инструментом для создание прототипов второго типа является MS Visio. Некоторую конкуренцию ему могут составить MS PowerPoint и Macromedia FreeHand (и вообще любой иллюстративный пакет, обладающий возможностью работать с несколькими страницами), но возможности PowerPoint для такой работы слишком малы, а возможности FreeHand, напротив, слишком велики. Ни то ни другое скорость работы не увеличивает. При работе с Visio можно выбрать одну из двух возможностей: можно либо рисовать все экраны на одном листе, соединяя взаимосвязанные элементы управления и экраны линиями, либо рисовать каждый экран на отдельном листе, связывая экраны ссылками. Первый вариант хорош для вас, поскольку позволяет оценить интерфейс в целом, второй – для заказчика и субъектов тестирования, поскольку его легче понять. Как правило, превратить второй вариант в первый оказывается гораздо легче.
- Среди многочисленных наборов заготовок в Visio есть и набор с элементами интерфейса Windows, однако эти заготовки выполнены довольно топорно, с огромным количеством лишних деталей. Пользоваться можно только заготовками для радиокнопок и чекбоксов (которые реализованы неплохо), а также заготовкой для полоски прокрутки. Поскольку создание собственных интерактивных заготовок непроблематично, я рекомендую создать свой набор и пользоваться им. Сам я, впрочем, так не делаю, а просто рисую нужные мне элементы на ходу. Получается немножко медленнее, нежели пользоваться готовыми, но зато выглядит прототип лучше (времени же для создания собственного набора у меня нет).
- Большим достоинством Visio является возможность записывать результат в HTML-файл, что позволяет без проблем тестировать интерфейс на территории субъектов (вытянуть субъекта тестирования к себе довольно проблематично). Раньше это мог только PowerPoint, чем, во многом, и объяснялась его популярность в качестве инструмента для создания прототипов. Сейчас это умеет и Visio (обратите внимание, что сохранение в HTML начало нормально работать только в Visio 2001, с другой стороны, более ранние версии лучше работают с русским языком).

КОНТРОЛЬНЫЙ СПИСОК ИНТЕРФЕЙСА ПО

Использование контрольных списков является эффективным и экономичным средством повышения качества программных продуктов. Их можно использовать и для повышения качества интерфейсов.

Использование контрольных списков является эффективным и экономичным средством повышения уровня качества программных продуктов. Их использование, применительно к интерфейсу, не требует выполнения дорогостоящих процедур юзабилити-тестирования, достаточно большое количество контрольных списков по интерфейсу можно найти в интернете в свободном доступе.

Естественно, что в каждом конкретном случае необходимо разрабатывать свой собственный контрольный список, поскольку он должен учитывать специфику разрабатываемого программного средства и возможности средств разработки. Поэтому настоящий контрольный список является скорее шаблоном, в котором представлены основные разделы (тем более, что в нем не проставлены баллы для каждого пункта). Тем не менее, для компаний, в которых проверка на эргономичность и единообразие не выполняется вовсе, использование даже такого списка может стать серьезным подспорьем для повышения эргономических характеристик интерфейса ПО.

- **Окна**
  - При проектировании было учтено, при каком разрешении, а так же размере монитора и шрифтов будут работать пользователи.
  - **Заголовки**
    - Заголовки короткие и адекватные содержимому окна.
    - Заголовки соответствуют названиям элементов, при помощи которых окна были вызваны.
    - Если окно вызывается элементом, не имеющим явного названия, в заголовке окна отражается название экранной формы.
  - **Дизайн окна**
    - Тип окна (модальное, немодальное, возможность минимизации/максимизации) был выбран осознанно, в соответствии с задачами пользователей.
    - Управляющие и информационные элементы расположены достаточно далеко друг от друга (не менее 7 DLU).
    - Информация в окне адекватно сгруппирована (связанные элементы объединены в группы).
    - Кнопки находятся в секции, на которую они оказывают непосредственное воздействие. Терминационные кнопки (управляющие окном) расположены либо снизу в ряд либо справа в колонку.
    - Переход от элемента к элементу внутри окна, осуществляется сверху вниз слева направо.
  - **Диалоговые окна**
    - В диалоговых окнах отсутствуют меню или инструментальные панели.
    - Диалоговые окна открываются не в центре экрана, а в центре текущего действия пользователя.
- **Меню**
  - **Пункты главного меню**
    - Пункты меню имеют адекватные названия.
    - Первая буква в названии пунктов заглавная.
    - Все пункты первого уровня активизируют выпадающее меню.
    - Каждому пункту меню назначены общепринятые горячие клавиши (выделены подчеркиванием).

- **Раскрывающиеся меню и элементы основного меню второго уровня**
  - Все элементы начинаются с заглавной буквы.
  - Если в меню используются пиктограммы, они расположены слева от названия пункта меню.
  - Все списки содержат более одного элемента.
  - Высота меню не превышает размер экрана (меню не нужно прокручивать).
  - Пункты меню адекватно сгруппированы. Осмысленно использованы разделители в меню.
  - Пункты меню расположены в порядке связанности выполняемых функций, частоте использования, важности.
  - Используются не более двух подуровней меню.
  - Каждый пункт меню имеет соответствующую горячую клавишу.
  - Название пункта меню соответствует названию вызываемого окна.
  - Пункты меню, открывающие диалоговые окна, обозначены в конце многоточием (…).
  - Недоступные пункты меню обозначены серым цветом шрифта.
- **Всплывающие меню**
  - Каждому пункту всплывающего меню соответствует аналогичный пункт в основном меню.

Инструментальные панели

- Каждому элементу инструментальной панели соответствует всплывающая подсказка.
- Элементы упорядочены и сгруппированы в соответствии с задачами пользователей.
- Для стандартных действий используются общепринятые графические элементы.

Управляющие элементы

- **Переключатели (Check boxes)**
  - В одном окне используется не более 10 переключателей.
  - Переключатели сгруппированы и каждой группе присвоено название.
  - Внутри группы переключатели расположены строго вертикально.
  - Переключатели не применяются для частого, оперативного использования.
  - В названиях используется только позитивная, утвердительная форма.

- **Командные кнопки**
  - Кнопки имеют краткие и ясные названия.
  - В каждом диалоге используется не более 6 кнопок.
  - Кнопки, выполняющие в разных диалогах идентичные функции, имеют одинаковые названия.
  - Типовые кнопки имеют общепринятые названия и общепринятые горячие клавиши.
  - Кнопки, вызывающие продолжение диалога в вложенных формах, обозначены многоточием (…).
  - Недоступные кнопки имеют соответствующие атрибуты (серый цвет шрифта и т.п.).
  - Опасные для пользователя кнопки не являются кнопками по умолчанию
- **Редактируемые поля со списком (Combo Box)**
  - Имеют функцию авто-выбора.
- **Раскрывающиеся списки**
  - Высота выводимого на экран списка ограничена 3-8 элементами.
  - Если список содержит более 50 элементов, используется фильтр или режим поиска.
  - Если все элементы не умещаются в одном фрагменте списка, автоматически появляется полоса прокрутки.
- **Группы элементов**
  - Каждая группа имеет осмысленное название, помимо рамки отделена от других групп и элементов свободным пространством.
- **Подписи (Labels)**
  - Все элементы имеют подписи.
  - Учтена возможность увеличения (уменьшения) длины подписей при использовании large fonts (small fonts).
  - Подписи выровнены по левому краю поля (если они находятся над полем).
  - Подписи расположены по середине высоты поля (если название находится с боку).
  - Если элемент недоступен, подпись отображается серым шрифтом.

- **Списки**
  - Если список содержит более 50 элементов, используется фильтр или режим поиска.
  - Высота ограничена 3-8 элементами.
  - Если все элементы не умещаются, автоматически появляется полоса прокрутки.
- **Кнопки выбора (Option Buttons или Radio Buttons)**
  - В одной группе используется не более 6 кнопок.
  - В пределах группы кнопки расположены по вертикали.
  - Нет состояния, когда ни одна кнопка не выбрана.
  - Последовательность расположения кнопок в группе учитывает частоту использования.
- **Вкладки (Tabs)**
  - Названия вкладок выровнены по центру.
  - Каждой вкладке присвоено осмысленное название.
  - Количество рядов закладок не превышает двух.
  - Все связанные между собой данные находятся внутри одной закладки.
  - Кнопки, относящиеся ко всему блоку закладок, расположены за пределами блока закладок.
- **Текстовые поля ввода (Text Box or Edit Field)**
  - Для недоступных полей используются серый цвет (название, текст и фон поля).
  - Высота всех текстовых полей в окне одинакова.
  - Содержимое полей выровнено по левому краю, за исключением полей с числовыми значением (напр., для вывода денежных сумм).
  - Длина поля не меньше длины вводимых в него данных.
  - Если в поле вводится численное значение границы диапазона выводятся во всплывающей подсказке.
- **Порядок табуляции фокуса ввода**
  - При открытии окна фокус попадает на элемент внутри окна.
  - Схема табуляции соответствует очередности заполнения полей (слева направо, сверху вниз).
  - Командные кнопки включены в табуляцию.
  - Невидимые и недоступные элементы исключены из схемы табуляции.
- **Пиктограммы**
  - Направление теней во всех пиктограммах одинаково: слева сверху.
- **Взаимодействие с пользователем**
  - Система, завершив какую-либо длительную операцию, пищит через встроенный динамик компьютера.
  - Цифры, предназначенные для сравнения либо для копирования в буфер обмена, выводятся непропорциональным шрифтом.