Лекция 12. Технические и психологические аспекты использования графических обозначений в современных интерфейсах

Цель лекции: *в данной лекции рассмотрены вопросы технических и психологических аспектов использования обозначений в пользовательских интерфейсах. Раскрыты понятия метафоры с точки зрения психологии.*

Вопросы для рассмотрения: *Понятие метафоры интерфейса. Факторы, воздействующие на комфорт пользователя. Вопросы оценки проекта*

Основные термины: *метафора, интерфейс, Макинтош, PARC, оценка проекта*

Введение

Интерфейс - система правил и средств, регламентирующая и обеспечивающая взаимодействие нескольких процессов или объектов.

Пользовательский интерфейс (ПИ) - система правил и средств, регламентирующая и обеспечивающая взаимодействие программы с пользователем. В понятие пользовательского интерфейса (ПИ) входит не только, и даже не столько, картинка на экране - трехмерная, анимированная или просто выполненная в модном дизайне, а способы взаимодействия пользователя с системой.

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

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

Итак, есть метафора для интерфейса. Теперь нужно сделать концептуальный дизайн интерфейса. То есть в рамках метафоры необходимо разработать систему интерфейсных элементов, своего рода алфавит взаимодействия, изучив который, пользователь сможет легко делать то, что ему нужно. Еще необходимо найти изящный способ изображения, как отдельных элементов,
так и их групп. И, наконец, надо выбрать общий изобразительный стиль, который был бы легко узнаваем и приятен для глаз.

Концептуальный дизайн интерфейса должен базироваться на идее интерфейсной среды. На время работы с системой пользователь погружается в среду интерфейса. Слово среда применяется как обозначение типичной для поведения человека в различных средах

связки «сигнал - действие».

Эта идея принадлежит психологу Гибсону, который утверждает, что наше восприятие основано на мотивации в том смысле, что если мы хотим есть, то видим только съедобные вещи, а если устали - то только предметы мебели, предназначенные для отдыха. То есть, человек не просто видит, а опрашивает среду, руководствуясь различными мотивами. В свою очередь, среда подает человеку разные сигналы. Наряду с ответами на его запросы, есть сигналы первоочередные (или всегда запрашиваемые), связанные с физической опасностью. Опираясь на полученные сигналы, человек осуществляет различные действия.

Для искусственных сред такая модель верна. Гибсон считает, что она верна и для естественных сред. Во всяком случае, как отправная точка для дизайна интерфейса, она очень продуктивна. Так, кнопки различных диалогов в стандартном оконном интерфейсе можно трактовать как сигналы к их нажатию. Но эти сигналы крайне слабы, поскольку все кнопки выглядят одинаково, отличаясь только текстами в них, а функции у них различные. То есть из всего разнообразия изобразительных средств - формы, размера, цвета, текста - в кнопках диалогов используется только текст. Считается хорошим тоном иметь кнопки одного размера и аккуратно расположенные, чтобы вынудить пользователя каждый раз прочитывать текст. Исключением, подтверждающим правило, является кнопка OK, которая смотрится не как текст, а как изображение (иероглиф).

Понятия среды и понятие метафоры близко связаны. Если среда по виду и некоторым опорным элементам будет напоминать пользователю что-то уже знакомое, он сможет быстрее приспособиться к ней. Вместе с тем выбранная метафора может продиктовать все изобразительные решения дизайна интерфейса. Однако следует остерегаться фотографической схожести среды в компьютере с выбранной метафорой. Все-таки компьютерная среда искусственная, и полностью повторить все элементы взаимодействия из физического мира не удастся.

А фотографическая схожесть может спровоцировать пользователя на то, чтобы пользоваться этой искусственной средой в точности как той, которую она напоминает. В первый же раз, когда пользователь натолкнется на различие, он испытает тяжелый психологический шок, который может привести к полному отторжению системы.
Мы подходим к важному принципу построения дизайна интерфейса - балансу между интерактивными возможностями программы и сложностью ее изобразительного ряда. В интерфейсе должен обеспечиваться баланс между функциональными возможностями программы, возможностями манипуляции ею и ее изобразительным рядом.

Основной проблемой в интерфейсе с пользователем является синхронизация точки внимания пользователя и точки активности системы. Эта проблема должна решаться совместно. С одной стороны, пользователь должен уметь сказать системе, где и что он хочет изменить (обычно это делается щелчком мыши в нужном месте). С другой стороны, система должна уметь привлечь внимание пользователя к месту наиболее актуальных изменений.

При переходе от алфавитно-цифровых дисплеев к графическим полям дисплея проблема синхронизации точки взаимодействия была самой сложной. Ее решение было выполнено по принципу «разделяй и властвуй». Поле экрана разбивалось на прямоугольники - окна, и вся работа велась только в одном из них, так называемом активном окне. Одновременно сменилась форма текстового курсора, и, что очень важно, он начал подмигивать. Это требовалось для облегчения проблемы поиска текстового курсора в окне. Поиск же курсора мыши при его потере из поля внимания пользователь (до сих пор) выполняет подергиванием мыши.

На самом деле используется очевидный факт, что движущийся предмет легче привлекает

внимание. Во многих приложениях используются разные формы динамики изображения, которые называются мультимедиа.

Две анимированные среды интерфейса разработаны в фирме XEROX PARC, в которой появились идеи оконного интерфейса (в группе Стюарда Карда, которому принадлежит авторство этой идеи). Одна - «Конические деревья» - является визуализацией файловой системы компьютера и похожа на систему детских пирамидок, каждый уровень которой соответствует уровню файлового каталога. Сами файлы из каталога отображаются в виде трехмерной карусели под своим каталогом. Суть модели в том, что нужный файл можно приблизить поворотом карусели (может быть, не одной), идущим в режиме анимации.

Вторая модель - «Стена в перспективе» - также отображает файловую систему, но вне ее иерархии, согласно двум каким-то параметрам, например частоте обращения к файлу и его размеру. Это нормальная стена, только очень длинная, разбитая на три отрезка. Средний из них отображается на экране плоско, а два крайних уходят в перспективу. Пользователь может сделать средним любой отрезок стены, причем это тоже происходит в режиме анимации. Для Карда анимация - принципиальный момент, так как анимация сохраняет в восприятии пользователя идентичность объекта, то есть пользователь легко соотносит объекты в конечной точке движения с объектами в начальной.
В графическом интерфейсе пользователь имеет дело с последовательностью картинок. Программисты, хвастаясь скоростью своих программ, замеряют время, теряемое между картинками. Однако психологи, занимающиеся интерфейсом, говорят о совсем другом времени, - времени, когда пользователь может начать взаимодействие с новой картинкой на экране. В этот интервал входит не только время вывода новой картинки на экран, но и время осознания ее пользователем. Ведь определенное время и усилия тратятся пользователем на то, чтобы понять, как каждая следующая картинка соотносится с предыдущей. Анимация за счет увеличения времени перехода от одной картинки к другой (а именно времени анимированного преобразования картинок) существенно сокращает время осознания новой картинки. В психологическом смысле новой картинки и не существует, существует преобразованная старая, а так как все преобразования шли на глазах у изумленных зрителей, то пользователь практически немедленно готов к взаимодействию.

Существует еще одно свойство анимационного пользовательского интерфейса, которое существенно улучшает его полезность по сравнению с графическим интерфейсом, а именно динамически визуальные сигналы.

Динамические визуальные сигналы - это изменение изображения на экране с целью дать пользователю дополнительную информацию. Уже в стандартном оконном интерфейсе мы можем видеть примеры таких сигналов. При выполнении программой длительных действий курсор мыши приобретает форму песочных часов. Это сигнал о том, что на действия пользователя система временно реагировать не будет. Второй пример - изменение изображения кнопки при нажатии на нее мышью. Это сигнал о том, что система считает, что пользователь взаимодействует именно с этой кнопкой.

Создавая анимационный интерфейс, надо закладывать систему динамических визуальных сигналов с самого начала, поскольку они являются столь же естественной, сколь и необходимой частью анимационного интерфейса.

Кроме того, информационная емкость (т.е. количество разных различимых вариаций) динамических сигналов огромна. Современные дисплеи отображают миллионы цветов, но даже если человеческий глаз и в состоянии отличить столько оттенков, человеческий мозг не в состоянии придавать им смысл. С другой стороны, такой простой сигнал, как мигание, имеет действительно миллионы хорошо осознаваемых оттенков, связанных с изменением яркости объекта во времени.

Однако, решая многие проблемы для пользователя, анимационный интерфейс, как это часто бывает, ставит сложные проблемы перед программистом и дизайнером.

Для использования анимационного интерфейса придется переходить к программам, управляемым временем. Вне зависимости от активности пользователя программе, построенной на анимационном интерфейсе, всегда есть что делать (например, менять фазу мигания). При этом, естественно, она должна постоянно быть доступной для взаимодействия, но, в отличие от многих сегодняшних мультимедиа-программ, не прерывать отображаемый поток, а плавно изменять его в соответствии с воздействием пользователя.

Такие требования легче всего реализуются в специфической архитектуре программ, управляемых временем. На каждом такте работы такой программы заново строится изображение на экране, а события, инициированные пользователем, например ввод с клавиатуры, отрабатываются всего лишь изменением состояния программы. Соответствующее изменение на экране происходит (быть может, не сразу) на очередном временном такте. Таким образом, к двум привычным уровням программы - функциональному и интерфейсному - добавляется визуальный.

Для дизайна конкретной программы требуется разработка собственной среды взаимодействия (направленной на реализацию конкретной функциональности) на базе общепринятой системы динамических визуальных сигналов. Предпочтительно иметь сквозное визуальное решение.

После выработки сквозного визуального решения необходимо прорисовать картинки, называемые у аниматоров фонами. Точнее назвать их неподвижной составляющей подвижного изображения. На каждом фоне надо расположить анимированные элементы взаимодействия. И, наконец, самое трудное - надо спроектировать визуальные переходы между существенно отличающимися состояниями. Распространенные ошибки при разработке интерфейсов программ

Перегруженность элементами управления. Перегруженность окон программы элементами управления встречается часто. Взять интернет-браузер. Кроме того, что он сам содержит несколько панелей инструментов, более десятка кнопок, меню, скроллеры, и т.д., содержимое большинства сайтов включает множество элементов управления - списки, кнопки. За всем этим пользователю очень сложно найти действительно нужную ему информацию. Часто разработчики программ не знают, как сделать по-другому, и располагают всю необходимую, по их мнению, информацию в одном окне, которое при этом превращается в нагромождение надписей и элементов управления. Даже опытный пользователь будет работать с такой программой с трудом, не говоря уже о новичке.

Программы предназначены для того, чтобы помогать пользователю делать свою работу. Пользователь не должен тратить свое время на манипуляции с программой. Перегруженность элементами управления приводит к тому, что человек отвлекается от основной задачи и принимается рассматривать окно программы, пытаясь понять, что здесь, где, и как с этим работать. Большое количество посторонних элементов заслоняет необходимую информацию. В результате производительность работы с программой снижается. Кроме того, пользователю в этом случае практически невозможно создать у себя в голове модель поведения этого окна, и позднее ее использовать. Каждый раз при работе с этим окном процесс познавания начинается сначала. Необходимо всегда стремиться к минимизации числа элементов управления на экране. Чем меньше их будет, тем лучше. Может показаться, что это абсурд - ведь получается, что идеальный интерфейс не должен быть виден совсем. Так оно и есть. Существует понятие программ-агентов, которые выполняются в фоновом режиме, собирая и накапливая информацию, на основе которой выполняют определенные действия. Срабатывающий сразу пункт главного меню. Пользователи привыкли, что пункт главного меню в программах содержит подменю.

Помещение в главное меню пункта, который срабатывает сразу при нажатии на него, делает программу непредсказуемой. Еще хуже, если это пункт
«Выход», который закрывает программу без всякого предупреждения. Представьте, пользователь исследует меню программы и неожиданно программа исчезает. Чаще всего таким пунктом действительно является пункт
«Выход». Например, пункт «Выход» из программы лучше располагать самым нижним пунктом самого левого подменю, независимо от того, подходит он туда по смыслу, или нет. Пользователи начинают искать его в первую очередь именно здесь. Несоответствующие рисунки на кнопках. Практически ни одна программа не обходится без панелей инструментов с кнопками. Кнопки имеют рисунки. Нажатие на кнопку вызывает определенное действие. Кажется логичным, что рисунок на кнопке должен соответствовать смыслу действия, однако многие про это забывают. Некоторые разработчики просто не умеют рисовать картинки, некоторые просто не хотят, и берут готовые (из других программ). В результате трудно без подсказки определить, что означает та или иная кнопка. Рисунки иногда получаются абсолютно не связанными с содержанием. Четкий и понятный рисунок способствует повышению эффективности работы. Если рисунок соответствует действию, эта ассоциация легко запоминается пользователем, и в дальнейшем один только взгляд на кнопку позволяет мгновенно вспомнить ее назначение. Если же нет, то назначение приходится вспоминать, либо узнавать заново каждый раз. Красный цвет. Правильно примененный цвет может, например, передавать тонкие различия между однородными элементами. Неправильно примененный цвет может мешать работать с программой.

Особенно это относится к красному цвету. Известно, что для людей красный цвет ассоциируется с некой опасностью. Большое количество красного цвета в каком-либо месте на экране привлекает внимание, заставляет пользователя настораживаться, думать, что здесь что-то не так.

Дорожные знаки красного цвета либо запрещают, либо предупреждают об опасности. Поэтому, если кнопка на экране окрашена в красный цвет, независимо оттого, что на ней написано, пользователь будет стараться избегать нажатия на нее. В малых количествах красный цвет может служить в качестве ненавязчивого указания наличия каких-либо проблем. Например, если получившееся в результате расчета число превышает норму. Красный цвет может использоваться в паре с другим. Существуют две метафоры - «термометр», когда красному противостоит синий, и «светофор» - зеленый. Обе они должны использоваться в случае, если это уместно. Терминология. Так как программы пишут программисты, они часто забывают, что пользоваться ими будут обычные люди, которые не знакомы с их терминологией. Большинство людей в действительности толком не представляют себе что такое, например, база данных или понятие записи. Файлы и манипуляции с ними тоже сложны для пользователей. Такая терминология пугает пользователей, в результате чего снижается эффективность их работы. Практически в подобных случаях можно называть вещи более понятными именами. Программа должна говорить с пользователями на их языке. Миф о метафоре. Разработчики программ часто говорят о нахождении правильной метафоры в качестве основы для интерфейса. Они думают, что если наполнить интерфейс картинками хорошо узнаваемых объектов из реального мира, то пользователи очень быстро научатся работать с программой. Поэтому они создают интерфейсы, которые выглядят как офисы со столами, папками документов, телефонами и адресными книгами, в надежде создать программу, которая легка в обучении. Некоторые из лучших дизайнеров интерфейсов считают выбор метафоры одной из первых и самых важных задач. Три парадигмы интерфейсов. Для пользовательских интерфейсов программ существует три парадигмы: технологическая, метафорическая и идиоматическая. Технологическая парадигма основана на понимании механизма работы программы - сложный подход. Метафорическая основана на интуитивном понимании - проблематичный подход. Идиоматическая парадигма основана на знании о том, как решать ту или иную задачу - естественный для человека процесс. Технологическая парадигма. Технологическая парадигма пользовательского интерфейса проста и широко распространена в компьютерной индустрии. Она означает, что интерфейс выражается в понятиях его конструкции, как он был построен. Чтобы успешно им пользоваться, пользователь должен понимать, как работает программа. Метафорическая парадигма. Современный графический интерфейс пользователя был изобретен в Исследовательском Центре Пало Альто фирмы Xerox (PARC) и был востребован промышленностью. Графический интерфейс

пользователя, разработанный в PARC, состоял из различных объектов: окна, кнопки, мыши, иконки, метафоры, меню.

Первой, успешной в коммерческом плане реализацией интерфейса PARC, стал Макинтош, с его метафорами рабочего стола, мусорной корзины и папок с файлами. Метафоры плохо масштабируются. Метафора, хорошо работающая для простого случая в простой программе часто перестает работать, как только задача усложняется и увеличивается в размере. Пиктограммы для обозначения файлов были хорошей идеей, когда компьютеры работали с дискетами или с 10 мегабайтным жесткими дисками. В наши дни гигабайтных дисков и тысяч файлов пиктограммы уже довольно неуклюжи. Метафоры мы понимаем интуитивно. Мы схватываем смысл метафорического элемента управления в интерфейсе, мысленно отождествляя его с каким-либо другим процессом или предметом, на познавание которого мы уже затратили время и силы. Эффективность этого метода велика, потому что она использует уникальное оружие человеческого ума - способность делать логические выводы. Процессор этого делать не умеет. Слабая сторона этого метода в том, что он зависит от человеческого ума, который может не иметь знаний или логических способностей, необходимых для совершения отождествления. Метафоры не ответственны за то, как их понимают. Идиоматическая парадигма. Третий метод разработки пользовательских интерфейсов решает проблемы двух предыдущих. Называют его идиоматическим, потому что он основан на том, как мы узнаем и используем идиомы, или фигуры речи. Мы понимаем идиомы потому, что уже знаем их. При запоминании идиом человеческий разум проявляет большие способности. Ему не приходится сравнивать их с уже известными ситуациями или понимать, как они работают. Он и не должен делать этого, потому что большинство идиом вообще не имеют метафорического смысла. Большинство элементов управления в графическом интерфейсе пользователя - идиомы. Кнопки, выпадающие списки и полосы прокрутки - это то, что мы узнаем автоматически, а не догадываемся метафорически.

Хорошо знакомая мышь не является метафорой чего-либо. Люди обучаются работе с ней идиоматически. Можно не знать, как устроена мышь, но можно легко ею пользоваться. Это и есть идиоматическое обучение.

Заметим, что большинство знакомых нам элементов GUI (Graphical User Interface - графический интерфейс пользователя), которые считаются метафорическими, на самом деле являются идиоматическими. Такие артефакты, как кнопки закрытия окна, окна с изменяемыми размерами, бесконечно вложенные папки с файлами, щелчки мышью и перетаскивание пиктограмм - не метафорические операции, потому что их нет в реальном мире. Их сила лишь в простой идиоматической узнаваемости. Проблемы. Основных проблем, которые возникают, если зависеть от метафор при

создании интерфейса, две: метафоры сложно найти, и они ограничивают наше мышление.

Для физических объектов, как принтеры или документы, легко найти визуальную метафору. Но для таких понятий как процессы, связи, службы и преобразования это сделать трудно, или даже невозможно. Сложно найти хорошую визуальную метафору для покупки билета, смены каналов, приобретения товара, нахождения ссылки, установки формата, вращения инструмента или смены разрешения экрана, хотя именно такие операции чаще всего встречаются в программах.

Проблема метафор возникает, если интерфейс привязывается к артефактам механической эры. Например, интуитивно легко понять, как работать с буфером обмена, потому что это метафора. Но, придерживаясь метафоры, можно обнаружить, что его возможности очень слабы. Он не может хранить больше, чем один объект, не помнит, что хранил ранее, не может определить, откуда взялось изображение, он не может показать уменьшенные картинки того, что содержит, и не хранит своего содержимого от запуска до запуска системы. Все эти действия неметафоричны. Возможно, что будущие интерфейсы будут идиоматическими, основанными на естественной способности человека легко и быстро узнавать новое.

Что влияет на удобство работы с системой?

Чтобы работа с компьютером была удобной, пользователь должен при взаимодействии с системой ощущать комфорт. Таким образом, на удобство работы влияют факторы, которые вызывают чувство комфорта. Их можно разделить на три большие группы (табл. 16).

Таблица 16

Факторы, воздействующие на комфорт пользователя

Факторы	Вызываются	Влияют на
Социальные факторы	Психологическим климатом	Эмоциональный комфорт
Физическая эргономика	Аппаратным обеспечением	Физический комфорт
Психологическая эргономика	Качеством разработки программного обеспечения	Умственный комфорт

Общий психологический климат в организации (например, стиль работы администрации и социальная защищенность работника), а также форма преподнесения сведений о вычислительной системе могут вызвать предубеждение против этой системы задолго до того, как пользователь познакомится с ней практически. Функции, которые возлагает система на пользователя, и способ выполнения этих функций могут нарушить сложившиеся рабочие группы, лишить исполнителя привычного общения или испортить его отношения с руководством. Эти социальные факторы могут

усилить или ослабить опасения пользователя относительно системы. Рассмотрим выбор рациональной стратегии проектирования программного обеспечения, которая может способствовать успешной работе пользователя.

Если пользователь относится к системе без предубеждений, эргономические характеристики реальной системы могут существенно улучшить или ухудшить его отношение к ней. Основные эргономические характеристики:

- конструктивные особенности оборудования;
- качество разработки диалога;
- доступность и надежность систем;
- чувствительность систем.

Конструктивные особенности оборудования (как компьютера, так и дополнительных устройств) и размещение его могут повлиять на чувство физического комфорта пользователя при работе с системой.

Конструирование оборудования - это «золушка» эргономики. Разработчики большинства вычислительных систем используют имеющиеся конструкции, а не разрабатывают их заново.

Третий фактор - это психологическая эргономика. В то время как физическая эргономика занимается изучением соответствия функций системы физиологическим процессам человека, психологическая эргономика занимается изучением соответствия функций системы психологическим процессам человека. Чтобы проиллюстрировать разницу, рассмотрим процесс считывания сообщения с экрана. Пользователь, который не может физически различить символы из-за бликов или плохого контраста, ощутит физический дискомфорт. Пользователь, который может прочесть текст, но не может понять его смысл из-за непонятных слов или непривычной формы представления текста, ощутит психологический дискомфорт. Нет смысла прилагать усилия для создания условий, когда у пользователя не болит спина, и не устают глаза, если от сообщений системы его голова пойдет кругом! На этот аспект работы систем, получивший название диалога, разработчики программного обеспечения могут повлиять как в положительную, так и в отрицательную сторону.

С понятием психологической эргономики тесно связаны еще два фактора. Это доступность и чувствительность системы.

Может ли пользователь получить доступ к системе в любое время и в любом месте? Разработчик должен гарантировать, что система будет доступна

именно в то время, когда это нужно пользователю. Кроме того, надежность системы должна обеспечивать доступ в любое удобное время. Имеет значение не только общее время потерь из-за сбоев, но и количество сбоев: несколько сбоев подряд в сети ЭВМ продолжительностью по нескольку секунд каждый могут выбить оператора из колеи сильнее, чем один часовой сбой. Число рабочих станций должно быть достаточным для обслуживания всех потенциальных пользователей.

Такой же отрицательный эффект, как невозможность получить доступ к системе, имеет и длительное ожидание ответа на запрос в течение 20 с и более. Еще хуже, если в какие-то дни ответ выводится через 2 с, а в другие приходится ждать 20 с: разное время реакции системы наводит пользователя на мысль, что система испортилась! Обеспечение приемлемого времени реакции системы - это один из технически важных и дорогостоящих аспектов разработки интерактивных систем.

Почему важен критерий удобства работы?

В 60-х и начале 70-х гг. удобство работы пользователя игнорировали большинство разработчиков систем. Большая ЭВМ была окружена кондиционерами и обслуживалась армией операторов, т.е. условия работы диктовала машина, а не наоборот.

При переходе к пакетной обработке стало ясно, что на правильное или неправильное функционирование системы влияет, прежде всего, удобство работы с ней. Это понимание расширилось, когда стали разрабатывать не простые счетные программы, а целые системы управления с элементами принятия решений. Для эффективной работы системы не достаточно, чтобы аппаратура и программы обеспечивали правильные результаты для заданных входных данных - не менее важным фактором является деятельность человека-оператора.

Для обеспечения эффективной работы оператора нужно учитывать его эмоциональные, психологические и физиологические особенности. Если пользователь расстроен, раздражен или подавлен, он не сможет работать хорошо. Элементом системы, который может вызвать или наоборот снять стресс, является интерфейс человек-компьютер, т.е. среда, через которую пользователь взаимодействует с системой.

Физические ограничения человека известны. Однако ограничения мозга не так хорошо изучены и часто вообще игнорируются. У нас большая долговременная память и ограниченная кратковременная (оперативная) память. Под оперативной памятью часто понимают последовательность входных и выходных буферов, в которых могут накапливаться промежуточные данные

при любой обработке. Подобно буферам вычислительной системы, эта память имеет ограниченный объем и легко может перегружаться. Долговременная память имеет, вероятно, неограниченный объем и быстрый доступ к данным. Если мы регулярно выполняем какую-нибудь работу, то она легко запоминается без перегрузки оперативной памяти; если мы повторяем ее достаточно часто, то со временем начинаем выполнять ее почти подсознательно. Однако если мы делаем что-то редко, наша оперативная память оказывается полностью занятой на все время работы.

Люди вносят в каждую деятельность свое понимание того, как эта деятельность должна выполняться. Это понимание - модель деятельности - базируется на прошлом опыте человека. Под долговременной памятью часто понимают хранилище различных моделей, на базе которых человек строит свою деятельность по реализации различных умственных стимулов. Вместо того, чтобы хранить мелкие подробности, люди восстанавливают подробности из моделей более высокого уровня. Большинство людей читает экран слева направо и сверху вниз, как книжную страницу. Они хотят, чтобы данные на экране были расположены в структурированном порядке и ищут признаки относительной важности различных элементов данных.

Многие системы, используемые в настоящее время, страдают именно отсутствием интерфейса, который отвечал бы потребностям пользователей. Они не были специально так разработаны. Поэтому нужно разобраться, откуда берутся подобные недостатки систем.

Немного систем проектируется для работы одного пользователя или для работы в однозадачном режиме. Для успешного функционирования интерфейса нужно, чтобы он соответствовал физиологическим и психологическим потребностям пользователя. Представления пользователя о системе, и его терпимость по отношению к ней зависят от характера пользователя и задачи.

Оценка проекта

Как разработчик узнает, что он достиг требуемого результата? Можно предположить несколько критериев, позволяющих оценить интерфейс. Все они охватывают три основных аспекта:

- простота освоения и запоминания операций системы;
- быстрота достижения целей задачи, решаемой с помощью системы;
- субъективная удовлетворенность при эксплуатации системы.

Можно установить контрольное время, необходимое определенному пользователю для достижения заданного уровня знаний. Критерий может также указать тип упражнений, помогающих добиться желаемого результата. Подобный критерий можно сформулировать следующим образом: «После двух дней самостоятельных занятий с системой пользователь, работавший с ней впервые, освоит все команды, необходимые для работы с файлами, хранящимися на диске в иерархически связанных каталогах».

Сохранение полученных рабочих навыков по истечению некоторого времени
- это другой критерий (связанный с первым), который определяет объем знаний, достаточный для возобновления деятельности после некоторого перерыва в работе.

Быстроту решения задачи можно оценить скоростью или точностью. При оценке скорости учитывается не быстродействие системы, а время, необходимое для достижения поставленной цели. Поэтому для системы ввода данных важна не скорость работы с клавиатурой, а контрольная цифра, которую можно указать, например, так: банковский служащий должен за час обработать не менее 20 счетов с ошибкой менее 1 %.

Критерий субъективной удовлетворенности отражает мнение пользователя о системе и удобстве работы с ней. Этот критерий трудно оценить количественно, но его можно выразить, например, с помощью частоты, с которой пользователи обращаются к дополнительным устройствам.

Хотя все три критерия можно отнести к любым областям применения, для конкретных применений важным будет какой-либо один из них.

Для систем, подобных системе управления авиатранспортом, важными являются факторы точности и скорости.

Для систем, рассчитанных на широкое применение, основным требованием является отсутствие предварительного обучения перед началом работы, поскольку часто отсутствует возможность организовать такие занятия.

При работе с системой, подобной электронной почте, пользователи должны чувствовать себя так же удобно, как и при работе с более простыми системами, иначе они просто откажутся от них.

Установить значения для каждого критерия - это часть трудностей. Разработчик должен уметь измерять реальную производительность системы в соответствии с поставленными целями. Для проведения этих измерений используется несколько методик.

Системы могут автоматически создавать и сохранять копию конкретного диалога, заносить в системный журнал время, затрачиваемое на выполнение различных этапов задания, или количество и тип ошибок.

Пользователям предлагается ответить на вопросы или заполнить различные анкеты, чтобы можно было определить, удовлетворены ли они работой системы. За работой системы можно наблюдать визуально или записать на видеокассету для анализа.

При использовании этих методов трудно быть уверенным, что получен действительно правильный результат и что замеченные вами отклонения присущи системе, а не определяются каким-либо внешним фактором.

Статистические методы, часто используемые, требуют более строгого подхода к трактовке природы испытуемого объекта и способа проведения измерений. Известны трудности выбора вопросов анкеты, на которые можно дать точные и четкие ответы. Люди могут изменить свое поведение, если узнают, что за ними наблюдают или их испытывают.

Контрольные вопросы

1. *Дайте определение понятию интерфейс?*
2. *Определите значение слова метафоры.*
3. *Какие существуют метафоры интерфейсов с точки зрения психологии? Опишите их.*
4. *В каком году был описан пользовательский графический интерфейс впервые?*
5. *Какие факторы влияют на комфорт пользователя при работе с пользовательским интерфейсом?*
6. *По каким основным аспектам пользователь может оценить интерфейс и дать оценку?*

Литература:

1. Купер А. т.б. Об интерфейсе: основы проектирования взаимодействия. 4-е изд. – СПб: Питер, 2018. – 720 б.
2. Норман Д. Дизайн привычных вещей. – М: Манн, Иванов и Фербер, 2019. – 384 б.
3. The Encyclopedia of Human-Computer Interaction, 2nd Ed. [электрондық басылым] – URL: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed (соңғы алышған 8/1/2020)
4. Sharp H. Interaction Design: Beyond Human-Computer Interaction. 5th Ed. – Wiley, 2019. – 656 p.
5. Shneiderman B. et.al. Designing the user interface: strategies for effective human-computer interaction. 6th Ed. – Pearson, 2016. – 616 p.
6. Тидвелл Д. т.б. Разработка пользовательских интерфейсов. 2-е изд. – М: Питер, 2011 – 480 б.
7. Круг С. т.б. Веб-дизайн или «не заставляйте меня думать». – СПб.: Символ Плюс, 2008. – 224 .
8. Нильсен Я. т.б. Веб дизайн. СПб: Символ Плюс, 2006. – 512 б.
9. Уильямс Р. т.б. Не дизайнерская книга о дизайне. – СПб: Весь, 2004. – 128 б.
10. http://appcamp.io/ – Онлайн-курс который дает начальное понимание разработки на HTML и мобильных платформах.
11. http://phonegap.com/book/ – Список книг по разработке HTML и мобильных приложений с помощью фреймворкаPhoneGap.
12. http://creator,ionic.io/ – HTML фреймворк пользовательского интерфейса для мобильных приложений.