ВВЕДЕНИЕ

Взаимодействие оператора с вычислительной машиной является важным звеном вычислительного процесса при решении различных прикладных задач как научного, так и производственного плана. Создание программ в области организации рыночных отношений при создании информационных сайтов различных организаций и предприятий, при создании программ управления производственными процессами, учета выпускаемой продукции и ее реализации, управления качеством и даже при такой задаче, как сортировка электронной почты секретарем, требуется разработка удобного для пользователя взаимодействия с ЭВМ.

Определение интерфейса.

В общем, **интерфейс (interface) –** это совокупность логических и физических принципов взаимодействия компонентов технических средств вычислительной системы (ВС), т. е. совокупность правил алгоритмов и временных соглашений по обмену данными между компонентами ВС (логический интерфейс), а также совокупность физических, механических и функциональных характеристик средств подключения, реализующих такое взаимодействие (физический интерфейс).

Интерфейс нередко называют также технические и программные средства, реализующие сопряжение между устройствами и узлами ВС.

Интерфейс распространяется на все логические и физические средства взаимодействия вычислительной системы с внешней средой, например с операционной системой, с оператором и т.п.

Виды интерфейсов

Интерфейсы различают по таким характеристикам, как структура связей, способ подключения и передачи данных, принципы управления и синхронизации.

1. ***Внутримашинный интерфейс*** – система связи и средств сопряжения узлов и блоков ЭВМ между собой. Внутримашинный интерфейс представляет собой совокупность электрических линий связи (проводов), схем сопряжения с компонентами компьютера, протоколов (алгоритмов) передачи и преобразования сигналов.

Различают два варианта организации внутри машинного интерфейса:

- многосвязный интерфейс, при котором каждый блок ПК связан с другими блоками своими локальными проводами;

- односвязный интерфейс, в результате которого все блоки ПК связаны друг с другом через общую или системную шину.

2. Внешний интерфейс – система связи системного блока с периферийными устройствами ЭВМ или с другими ЭВМ

Здесь можно выделить также несколько типов внешнего интерфейса:

- интерфейс периферийных устройств, подключаемых с помощью шин ввода-вывода (ISA, EISA, VLB, PCI, AGP, USB IEEE 1384 SCSI и др.);

- сетевой интерфейс, типа одноранговой сети или сети клиент-сервер с топологиями типа звезда, кольцевая или шинная.

3. ***Интерфейс «человек-машина» или интерфейс «человек-компьютер» или пользовательский интерфейс*** – это способ, которым вы выполняете какую-либо задачу с помощью каких-либо средств (какой-либо программы), а именно совершаемые вами действия и то, что вы получаете в ответ.

Интерфейс является ориентированным на человека, если он отвечает нуждам человека и учитывает его слабости.

Машинная часть интерфейса – часть интерфейса, реализованная в машине (аппаратно-программной ее части) с использованием возможностей вычислительной техники.

Человеческая часть интерфейса – это часть интерфейса, реализуемая человеком с учетом его возможностей, слабостей, привычек, способности к обучению и других факторов.

Наиболее распространенные интерфейсы определены государственными и международными стандартами.

В дальнейшем изложении будет рассматриваться только интерфейс пользователя.

Классификация интерфейсов пользователя

Как было указано выше интерфейс – это, прежде всего набор правил, которые можно объединить по схожести способов взаимодействия человека с компьютером.

Различают три вида интерфейсов пользователя: командный, WIMP и SILK – интерфейсы.

Взаимодействие перечисленных интерфейсов с операционными системами и технологиями показано на рис.1:

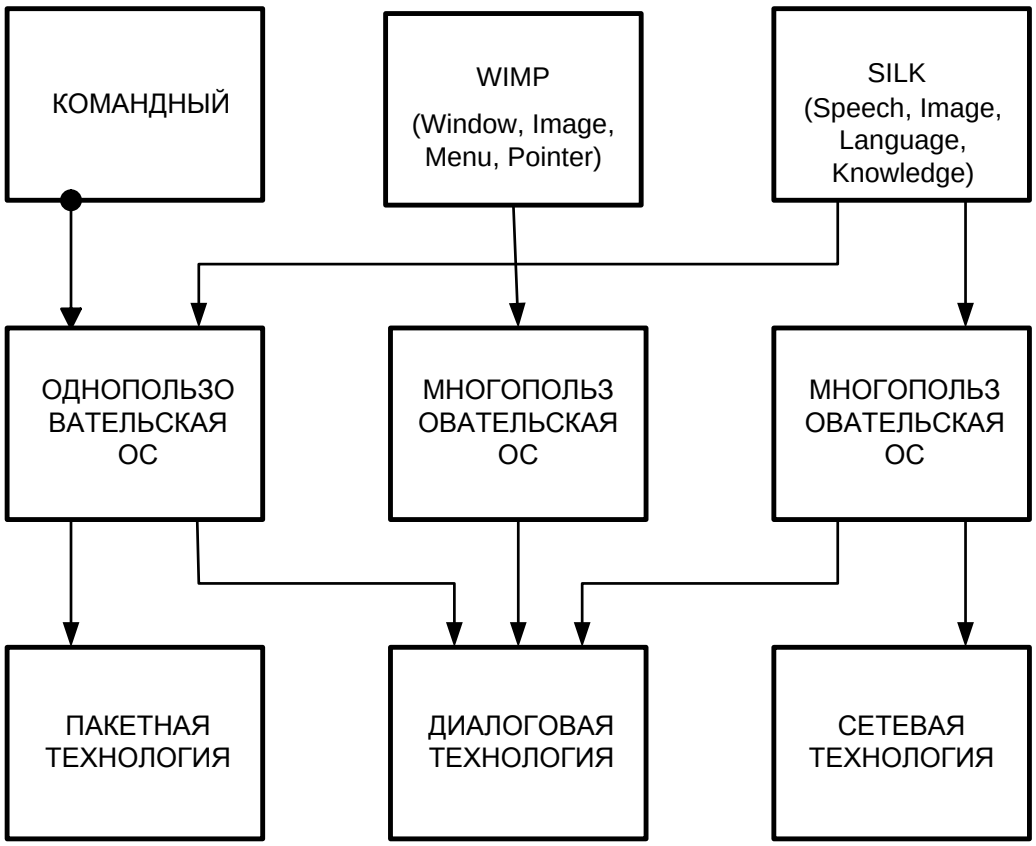

Рис. 1. Взаимодействие интерфейсов пользователя их технологий и операционных систем.

1. Командный интерфейс, при котором взаимодействие человека с компьютером осуществляется путем подачи компьютеру команд, которые он выполняет и выдает результат пользователю. Командный интерфейс может быть реализован в виде пакетной технологии и технологии командной строки. В настоящее время пакетная технология практически не используется, а технология командной строки можно встретить в виде резервного способа общения человека с компьютером.

Пакетная технология.

Исторически этот вид технологии появился первым на электромеханических вычислительных машинах К. Цюзе, Г. Айкина, а затем на электронных вычислительных машинах Эккерта и Моучли, на отечественных ЭВМ Лебедева, Брусенцова, на ЭВМ IBM-360, на ЕС ЭВМ и так далее. Идея его проста и состоит в том, что на вход компьютера подается последовательность программ, набитых, например, на перфокартах и последовательность символов, определяющих порядок выполнения этих программ. Человек здесь имеет малое влияние на работу машины. Он может лишь приостановить работу машины, сменить программу и снова запустить ЭВМ.

Технология командной строки.

При этой технологии в качестве способа ввода информации оператором в ЭВМ служит клавиатура, а компьютер выводит информацию человеку с помощью алфавитно-цифрового дисплея (монитора). Комбинацию монитор-клавиатура стали называть терминалом или консолью. Команды набираются в командной строке, представляющей собой символ приглашения и мигающий курсор, при этом набранные символы можно стирать и редактировать. По нажатию клавиши «Enter» («Ввод») ЭВМ принимает команду и начинает ее выполнять. После перехода в начало следующей строки компьютер выдает на монитор результаты своей работы. Наиболее распространенным командный интерфейс был в операционной системе MS DOS.

2. ООМУ (окно, образ, меню, указатель) WIMP (window, image, menu, pointer) - интерфейс. Характерной чертой этого интерфейса является то, что диалог пользователя с компьютером ведется не с помощью командной строки, а с помощью окон, графических образов меню, курсора и других элементов. Хотя в этом интерфейсе подаются команды машине, но это делается через графические образы.

Идея графического интерфейса зародилась в средине 70-х годов в исследовательском центре фирмы Xerox Palo Alto Research Center (PARC). Предпосылкой графического интерфейса явилось уменьшение времени реакции компьютера на команду, увеличение объема оперативной памяти, а также развитие элементной базы, технических характеристик ЭВМ и в частности мониторов. После появления графических дисплеев с возможностью вывода любых графических изображений различного цвета графический интерфейс стал неотъемлемой частью всех компьютеров. Постепенно проходил процесс унификации в использовании клавиатуры и мыши прикладными программами. Слияние этих двух тенденций привело к созданию такого пользовательского интерфейса, с помощью которого при минимальных затратах времени и средств на переучивание персонала можно работать с любыми программными приложениями

Этот вид интерфейса реализован в виде двух уровней:

- простой графический интерфейс;
- полный WINP – интерфейс.

Простой графический интерфейс, который на первом этапе очень походил на технологию командной строки со следующими отличиями:

- при отображении символов с целью повышения выразительности изображения допускалось выделение части символов цветом, инверсным изображением, подчеркиванием и мерцанием;

- курсор мог быть представлен некоторой областью, выделенной цветом и охватывающей несколько символов и даже часть экрана;

- реакция на нажатие любой клавиши во многом стало зависеть от того, в какой части находится курсор.

- кроме часто используемых клавиш управлением курсором стали использоваться манипуляторы типа мыши, трекбола и т.п., которые позволяли быстро выделять нужную область экрана и перемещать курсор;

- широкое использование цветных мониторов.

Появление простого графического интерфейса совпадает с широким распространением операционной системы MS DOS. Типичным примером его использования является файловая оболочка Norton Commander и текстовые редакторы MaltiEdit, ChiWriter, Microsoft Word для DOS, Лексикон и др.

Полный WIMP-интерфейс, явился вторым этапом развития графического интерфейса, который характеризуется следующими особенностями:

- вся работа с программами, файлами и документами происходит в окнах;

- программы, файлы, документы, устройства и другие объекты представляются в виде значков (иконок), которые при открытии превращаются в окна;

- все действия с объектами осуществляются с помощью меню, которое становится основным элементом управления;

- манипулятор выступает в качестве главного средства управления.

Следует отметить, что WIMP-интерфейс требует для своей реализации повышенного требования к производительности компьютера, объему его памяти качественного растрового цветного дисплея программного обеспечения, ориентированного на этот вид интерфейса. В настоящее время WIMP-интерфейс стал стандартом де-факто, а операционная система Microsoft Windows стала ярким его представителем.

3. РОЯЗ (речь, образ, язык, знания) SILK (speech, image, language, knowledge) - интерфейс. Этот интерфейс 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. Результаты выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса требует больших аппаратурных затрат, поэтому находится в стадии разработки и совершенствования и используется пока только в военных целях.

SILK- интерфейс для общения человека с машиной использует:

- речевую технологию;
- биометрическую технологию (мимический интерфейс);
- семантический (общественный) интерфейс.

Речевая технология появилась в середине 90-х годов после появления недорогих звуковых карт и широкого распространения технологий распознавания речи. При этой технологии команды подаются голосом путем произнесения специальных стандартных слов (команд), которые должны выговариваться четко, в одном темпе с обязательными паузами между словами. Учитывая, что алгоритмы распознавания речи недостаточно развиты, требуется индивидуальная предварительная настройка компьютерной системы на конкретного пользователя. Это простейшая реализация SILK- интерфейса.

Биометрическая технология («Мимический интерфейс») возникла в конце 90-х годов и в настоящее время находится в стадии разработки. Для управления компьютером используется выражение лица, направление взгляда, размер зрачка и другие признаки человека. Для идентификации пользователя используется рисунок радужной оболочки его глаз, отпечатки пальцев и другая уникальная информация, которая считывается с цифровой камеры, а затем с помощью программы распознавания образов из этого изображения выделяются команды.

Семантический (общественный) интерфейс возник еще в конце 70-х годов XX века, с развитием искусственного интеллекта. Его трудно назвать самостоятельным видом интерфейса, так как он включает в себя и интерфейс командной строки, и графический, и речевой, и мимический интерфейсы. Основной его особенностью является отсутствие команд при общении с компьютером. Запрос формируется на естественном языке, в виде связанного текста и образов. По сути - это моделирование общения человека с компьютером. В настоящее время используется для военных целей. Такой интерфейс крайне необходим в обстановке ведения воздушного боя.

В настоящее время наиболее актуально изучение технологии WIMP-интерфейса, так как он наиболее часто используемый для всех типов компьютеров. Основное внимание будет уделяться программной части интерфейса пользователя, в которой важным звеном является человеческая часть.

Наши компьютеры и сотовые телефоны, оснащенные самыми последними моделями чипов и другой электронной начинкой. Современные операционные системы способны радовать глаз великолепными цветными заставками и стремительными трехмерными эффектами, но когда Вы начинаете пользоваться этой системой, выясняется, что в некоторых случаях она ограничивает вас своим непредсказуемым поведением. Из тысячи команд, предусмотренных в системе, вам не удается найти ту, которая нужна в данный момент, а простые стандартные процедуры выполняются бесконечно долго. Разрешение этого противоречия лежит в разработке удачного интерфейса пользователя. В основе разработки хороших интерфейсов лежат принципы, которые на сегодня не являются общеизвестными.

Несмотря на то, что интерфейсы непрерывно совершенствовались в течение двух десятилетий, опубликованы руководства по созданию интерфейсов и созданы средства их разработки проблема совершенствования интерфейсов пользователя с учетом более глубоких познаний менталитета и психологии пользователя является актуальной. Тем более что проблема разработки однопользовательских интерфейсов еще не решена, а если индивидуальное взаимодействие с некоторой системой не проходит для пользователя легко и комфортно, то в результате этот недостаток будет негативно отражаться на качестве работы всей системы, независимо от того насколько она хороша в других своих проявлениях.

Интерфейс пользователя связан с такими науками, как психология и эргономика.

Эргономика – это наука о принципах взаимодействия системы «человек-машина».

Авиационная эргономика – это наука о компоновке шкал, подвижных и неподвижных указателей, сигнализаторов, рукояток управлении отдельных приборов и пультов управления, а также о компоновке индикаторов, пультов, сигнализаторов, органов управления и рабочих мест в кабине экипажа с учетом человеческих факторов для оптимального взаимодействия интерфейса «человек-машина».

Психология – наука о поведении человека в различных ситуациях, его менталитете, интеллектуальных способностях, привычках, реакции на внешние и внутренние раздражители.

Когнетика и локус внимания

Когнетика так же как и эргономика, учитывает статистическую природу различий между людьми.

Изучение прикладной сферы наших ментальных способностей называется когнитивным проектированием или ***когнетикой.*** От слова Cognate – родственный, сходный.

Сознательное и бессознательное – это термины из области психологии, философии и истории применяются для описания аспектов функционирования нашего мышления.

В плане проектирования человеческой части интерфейса имеет смысл пользоваться более ограниченными понятиями ***когнитивного сознательного*** и ***когнитивного бессознательного*** или более точно ***эмпирическое сознательное*** и ***эмпирическое бессознательное.***

Когнитивное бессознательное – это те ментальные процессы, которые вы не осознаете в тот момент, когда они происходят.

Когнитивное сознательное включается в тот момент, когда вы сталкиваетесь с ситуацией, которая кажется новой или представляет угрозу или когда требуется принять нешаблонное решение.

Свойства когнитивного сознательного и когнитивного бессознательного сведены в таблицу 1.

Таблица 1.

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

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 храниться в течение 10 секунд.

Память

Возможности человеческой памяти существенно влияют на качество взаимодействия пользователя с системой. Для нас актуально знать про две подсистемы памяти, а именно про *кратковременную* и *долговременную* подсистемы.

Кратковременная память

Свойства, а точнее ограничения кратковременной памяти (КВП) являются очень важными факторами при разработке интерфейса. Дело в том, что вся обработка поступающей информации производится в КВП, в этом кратковременная память сходна с ОЗУ в компьютерах. Сходство, однако, не является полным, так что думать о КВП, как об ОЗУ не стоит.

Что попадает в КВП. Удобно, хотя и неправильно, считать, что в КВП попадает всё, что кажется нужным и имеющим какой либо смысл. Соответственно, чтобы что-либо попало в КВП пользователя, пользователь должен это заметить (для чего, собственно говоря, и полезно проектировать интерфейс с учетом возможностей человеческого восприятия) и счесть полезным лично для себя. Как правило, для опытного пользователя оценка полезности не представляет проблем, неопытные же почти всегда посылают в КВП наиболее заметные детали. Таким образом, самое важное в интерфейсе должно быть наиболее заметным.

Изменение содержимого. Другая интересная особенность КВП заключается в том, что смена содержимого в ней происходит скорее из-за появления новых ситуаций, нежели чем просто от времени. С одной стороны, это значит, что без новых ситуаций КВП остается неизменной. С другой стороны, поскольку отсутствие новых ситуаций есть труднодостижимый идеал, содержимое КВП постоянно меняет-

ся. Практический смысл этого наблюдения состоит в том, что нельзя допускать, чтобы пользователь отвлекался, поскольку новые ситуации при отвлечении стирают содержимое КВП. Но и это есть только мечта. Приходится удовлетворяться тем, чтобы максимально облегчать пользователю возвращение к работе.

Объем КВП. Чуть ли не единственным правилом интерфейсной науки, известным широкой публике, является утверждение, гласящее, что в группе чего угодно не должно быть более 7±2 элемента. Проблема в том, что это правило, как показывает опыт, не является абсолютным для создания интерфейса пользователя и практически оно выглядит по-иному.

Во-первых, потому что, 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 элементов, но только в тех случаях, когда они сгруппированы. Соответственно, всегда можно сгруппировать элементы и поместить в КВП пользователя больше информации.

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

И, наконец, самое важное. Информация не только хранится в КВП, она в нем же и обрабатывается. Это значит, что один этап обработки занимает место как минимум одного элемента КВП. Более того: контекст предыдущих действий тоже хранится в КВП, снижая доступный объем.

С практической точки зрения важно еще и следующее. Если на интерфейс смотрит опытный пользователь, почти вся необходимая ему информация содержится не в кратковременной, а в долговременной памяти, что обозначает, что объем КВП уходит на второй план. Более того, зачастую и для неопытных пользователей объем КВП не важен. Если предположим, такой пользователь смотрит на раскрытое меню, зная контекст предыдущих действий он сканирует меню и,

найдя наиболее необходимый элемент, выбирает его, при этом ни один из ненужных ему элементов в КВП не попадает. Проблемы с КВП начинаются только тогда, когда ни один из элементов не покажется ему более необходимым, чем другие. Только в этом случае пользователю придется поместить все элементы меню в КВП и сделать выбор. Соответственно, большинство таких проблем может быть решено визуальной организацией элементов и правильным наименованием отдельных элементов (чтобы по тексту элемента сразу была понятна его применимость). Необходимо всегда помнить, что количество элементов, предоставляемых пользователю для одновременного обозрения необходимо минимизировать.

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

Нагрузка на КВП. В целом, использовать КВП пользователям неприятно. В этом заключается самая большая проблема КВП применительно к интерфейсу, большая даже, чем человеческие ошибки, вызванные выпадением элементов из памяти. Объясняется это неприятие КВП просто – и запоминание, и извлечение информации из памяти требует усилий. Более того. Поскольку содержимое КВП теряется при поступлении новых ситуаций, 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 её в другом. Лучшим способом достижения этой цели является непосредственное манипулирование, у которого, кстати, есть ещё множество других достоинств.

Вообще говоря, любой ввод параметров не значениями, а воздействием на управляющие элементы (т.е. верньеры) сильно снижает нагрузку на память. Другой разговор, что верньеры занимают много места на экране, плохо подходят для точного ввода значений и всегда оказываются хуже непосредственного манипулирования, поскольку при непосредственном манипулировании пользователям не нужно помещать в КВП алгоритм действия.

Долговременная память

Объем ДВП очень велик, информация, попавшая в ДВП, хранится довольно долго, а может даже и вечно, но с точки зрения дизайна всё это не представляет особого интереса. Интерес представляют два вопроса:

- как и при каких условиях, информация попадает в ДВП?
- каким образом происходит вспоминание?

Оба вопроса очень имеют большое значение с точки зрения обучения пользователей, второй вопрос, важен с точки зрения улучшения способности пользователей для сохранения навыки работы с системой в течение длительного времени, так как это одна из основных характеристик хорошего интерфейса.

Внутрь ДВП. Считается, что информация попадает в ДВП в трех случаях. Во-первых, при повторении, т.е. при зубрежке. Во-вторых, при глубокой семантической обработке. В-третьих, при наличии сильного эмоционального шока (стресса). Последний вариант можно не рассматривать, так как для стрессового состояния проектировать интерфейс нерационально (тем более что после шока запоминание прерывается).

Что касается повторений можно отметить, что чем их больше и чем меньше времени проходит между повторами, тем больше шансов, что информация будет запомнена, следовательно, пользователи надежно обучаться работать с системой.

С семантической обработкой несколько сложнее. Дело в том, что информация хранится в ДВП в сильно структурированном виде (например, зрительные образы на самом деле хранятся не в виде картинки, а как список объектов, находящихся в изображении, изображения же отдельных объектов хранятся отдельно). Чем больше человек думает о какой-либо информации, чем больше он соотносит её с другой информацией, уже находящейся в памяти, тем лучше он запомнит то, о чем думает (т.е. текущий стимул). Если пользователь долго мучается, стараясь понять, как работает система, он запомнит её надолго, если не навсегда.

Несколько помогает понять устройство механизма запоминания его антипод забывание. Современная наука утверждает, что забывание обусловлено тремя факторами, а именно: затуханием, интерференцией и различием ситуаций. Самое простое объяснение имеет затухание: когда информация не используется долгое время, она забы-

вается. Несколько сложнее с двумя оставшимися факторами. Предполагается, что если сходной семантической обработке подверглись несколько фрагментов сходной информации при запоминании, эти фрагменты перемешиваются в памяти и интерферируют друг с другом, делая практически невозможным воспроизведение поврежденного фрагмента. Иначе обстоит дело с различием ситуаций. Предполагается, что для успешного воспоминания требуется соответствие признаков во время кодирования в процессе запоминания с признаками во время воспроизведения.

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 обработка есть способ мощный, но дорогой: без необходимости пользователи не будут задействовать свой разум на семантическую обработку запоминаемой информации, предоставить же им повод для этого сложно. Лучше всего в качестве средства запоминания работает аналогия, неважно, как она представлена, как метафора интерфейса, или как эпитет в документации.

Цена вспоминания. Обращение к ДВП стоит довольно дорого. Разные понятия вспоминаются с разной скоростью, слова, например, вспоминаются быстрее цифр, а визуальные образы – быстрее слов. Очень сильно влияет объем выборки, т.е. вспомнить одно значение из десяти возможных получается быстрее, нежели из ста возможных. Наконец, частота вспоминания влияет на скорость вспоминания (т.е. тренировка существенно ускоряет вспоминания). Так что для обращения к воспоминаниям мозг выполняет работу, сходную с поиском книги в библиотеке (только более сложную; попробуйте методом самонаблюдения вспомнить, например, всех своих одноклассников). Соответственно, когда человек вспоминает, он углубляется в свою память и находит всё больше признаков искомой информации.

При проектировании интерфейса удобно пользоваться следующим правилом. Для обычных пользователей, у которых нет навыков извлечения из ДВП информации, присущей проектируемой системе, следует снижать нагрузку на ДВП; для опытных пользователей, у которых эти навыки сформировались, обращение к ДВП может быть более быстрым, нежели любой другой способ поиска информации.

Важно, однако, сознавать, что для опытных пользователей ДВП, будучи быстрым, не обязательно является предпочтительным. На-

пример, если стоит задача снизить количество ошибок, меню будет более эффективно, чем, скажем, командная строка, поскольку оно не позволит отдать заведомо неправильную команду.

Формирование привычек

Любая привычка означает отказ от внимания к деталям.

При постоянном использовании какого-либо интерфейса у вас формируются определенные привычки, которые впоследствии трудно преодолеть. В этом смысле задача дизайнеров заключается в том, чтобы создавать интерфейсы, которые не позволяют привычкам вызывать проблемы у пользователей. Мы должны создавать интерфейсы, которые, во-первых, целенаправленно опираются на человеческую способность формировать привычки и, во-вторых, развивают у пользователей такие привычки, которые позволяют упростить ход работы. В случае идеального ориентированного на человека интерфейса доля участия самого интерфейса в работе пользователя должна сводиться к формированию полезных привычек. Многие проблемы, которые делают разрабатываемые программы сложными и неудобными в использовании происходят из-за того, что в используемом интерфейсе «человек-машина» не учитываются полезные и вредные свойства человеческой способности формировать привычки.

Хорошим примером в этом случае является тенденция предусматривать сразу несколько путей решения одной и той же задачи. В этом случае множество вариантов приводит к смещению фокуса внимания пользователя с самой задачи на выбор наилучшего пути решения задачи.

Одновременное выполнение задач

На языке когнитивной психологии любая задача, которую вы научились выполнять без участия сознания, так же как и любая последовательность действий, которую вы регулярно выполняете, становится **автоматичной.**

Автоматизм позволяет выполнять несколько действий одновременно. Все одновременно выполняемые задачи, за исключением не более чем одной являются автоматичными. Та задача, которая не является автоматичной, естественно находится в фокусе вашего внимания.

При интерференции одновременно выполняемых задач, чем более предсказуемой, автоматичной и бессознательной становится задача, тем больше становится эффективность ее выполнения одновременно с другими задачами, и, тем менее, она конкурирует с ними.

Единственный способ предотвратить неправильные действия в результате привычек – это удерживать выполняемую полезную задачу в фокусе вашего внимания, что не так просто, так как наше внимание все время «гуляет». В интерфейсах в этом случае ставиться вопрос, например в задаче удаления файлов «Вы уверены …» Правда, и это не совсем спасает от неправильных действий, так как любой запрос подтверждения, требующий установленного ответа, вскоре становится бесполезным. Более эффективный подход заключается в том, чтобы дать пользователю возможность отменить ошибочную команду.

Все люди обладают только одним фокусом внимания. После приобретения нового фокуса внимания прежний фокус теряется. Человеку необходимо около 10 секунд для того, чтобы переключиться с одного контекста на другой. При обычных ординарных обстоятельствах ваше внимание не отвлекается, и вы можете не заметить событий, происходящих в окружении. Интерфейсы следует разрабатывать с расчетом на то, что пользователь, поглощенный своей задачей не станет даже реагировать на ваши попытки пообщаться с ним. Интерфейс должен хорошо работать независимо от степени поглощенности пользователя.

Если во время использования какого-либо интерфейса компьютер начинает работать неожиданным образом, то по мере того, как ваше волнение, вызванное возникшей проблемой, возрастает, вы с меньшей вероятностью сможете замечать подсказки, сообщения и другие средства помощи пользователю.

Интерфейс должен быть таким, чтобы пользователь не мог совершать ошибок связанных с работой интерфейса, или, другими словами, чтобы пользователь имел возможность сразу отменить результаты любого действия, а не просто получал предупреждения о потенциальных его последствиях.

Возобновление прерванной работы

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

Очевидно, что когда вы открываете документ в каком-либо приложении, например в текстовом процессоре, вы должны вернуться к

тому месту, где вы с ним работали в тот момент, когда закрывали или сохраняли его в последний раз.

Значения, режимы, монотонность и мифы

Содержание (content) – это информация, которая находится в компьютере или другом устройстве, предназначенном для ее обработки, которая является для Вас осмысленной и полезной и которую вы предполагаете создавать или изменять с помощью упомянутого устройства.

Графическое устройство ввода (ГУВ) – это механизм для передачи системе информации об определенном местоположении или выборе места на экране монитора (мышь, световое перо, трекбол, планшетный карандаш, джойстик или тачпад).

Режимы (modes) – это состояние интерфейса, при котором интерпретация данного конкретного жеста остается неизменной. Режимы являются важным источником ошибок, путаницы, ненужных ограничений и сложности в интерфейсе. При создании интерфейсов их необходимо по возможности исключать

Разделение интерфейса на ограниченные области является неизбежным следствием наличия режимов.

Жест (gesture) – это последовательность действий человека, которая выполняется автоматически (после старта).

Набор состояний, в котором жест имеет конкретную интерпретацию, называется *диапазоном (range) жеста.* Группировка команд по разным диапазонам (по приложениям) позволяет понять и научиться использовать сложные интерфейсы.

Для фиксации и определения режимов применяются кнопки, флажки и переключатели. Рекомендуется использовать переключатели вместо выключателей. На выключатели стоит полагаться только в том случае, когда можно видеть значение контролируемого состояния и оно находится в локусе внимания пользователя или в кратковременной памяти

Интерфейс, полностью ориентированный на человека, должен состоять только из одного диапазона.

Объединение последовательности действий в жест, связанный с определенным психологическим (процессом) состоянием определяется как формирование *модуля (chunking)* т.е. соединение отдельных элементов когнитивного процесса в единый ментальный модуль, что позволяет воспринимать множество элементов как целое.

Интерфейс «человек-машина» является модальным по отношению к данному жесту, если во-первых, текущее состояние интерфейса не находится в локусе внимания пользователя и во-вторых если в ответ на некоторый жест интерфейс может выполнять одно из нескольких возможных действий в зависимости от текущего состояния системы.

Немодальный интерфейс не должен быть модальным для любого жеста.

Степень модальности интерфейса определяется формулой

Q=Σp(Ni),

где **Q** – степень модальности каждого жеста интерфейса может быть определена через классификацию того или иного жеста, изменяется от 0 – полностью модальный, до 1 – полностью немодальный.

p(Ni) - вероятность применения немодального жеста.

В общем случае, чем меньше режимов, тем интерфейс в большей степени считается ориентированным на человека.

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

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

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

Наиболее наглядно изобилие режимов можно показать на существующем в настоящее время интерфейсе пилота с вычислительной системой самолетовождения (ВСС), в состав которой входит собственно компьютер ВСС и комплексный пульт управления и индикации (КПУИ). К этому же интерфейсу относится и экранная система индикации (ЭСИ) или комплексная система электронной индикации и сигнализации (КСЭИС).

На этом пульте для выполнения различных операций имеется 9 клавиш режимов и 6 клавиш подрежимов в каждом режиме, кроме того, в каждом подрежиме необходимо пользоваться наборным полем кнопок алфавита и цифровой информации.

Пользовательские настройки и временные режимы.

Пользовательские настройки – это такие изменения в программном обеспечении, которые не отражаются в документации.

Снабжая программу настройками, мы обременяем пользователя задачей, которая не относится к его рабочим функциям. Время, которое тратится на изучение и выполнение пользовательских настроек, большей частью является потерянным с точки зрения текущей задачи. Существует идея, по которой интерфейс должен адаптироваться в соответствии с эмоциональным состоянием пользователя.

Режимы, которые исчезают после однократного применения, создают меньше ошибок, чем те, которые работают постоянно.

Если вы разрабатываете модальный интерфейс, учитывайте, что пользователи будут всегда совершать модальные ошибки, за исключением тех случаев, когда состояние, контролируемое данным режимом, находится в локусе внимания пользователя (и он может его видеть) либо в его кратковременной памяти. Задача разработчиков состоит в том, чтобы показать, что данный режим используется правильно или что преимущества данного режима перевешивают его неизбежные недостатки.

Режимы и квазирежимы

Использование клавиши “Caps Lock” для набора заглавных букв существенно отличается от удержания клавиши “Shift” для этой же цели. Первый случай устанавливает режим, второй – нет. Режимы, которые физически удерживаются пользователем, называется ***квазирежимом.***

Исследования показали, что удержание кнопок в нажатом состоянии, нажатие на педаль или любая другая форма физического удержания интерфейса в определенном состоянии не приводит к возникновению модальных ошибок. Это явление объясняется тем, что наша нервная система функционирует таким образом, что постоянный стимул порождает сигналы, которые, которые со временем снижают свою способность привлекать наше внимание до полного их прекращения. Однако физические усилия не позволяют снижать способность привлечения нашего внимания к выполняемому действию.

Квазирежимы являются весьма эффективным средством с точки зрения устранения режимов, однако требуют запоминания десятков команд: “Control”, “Alt”, “Esc” и т.д. Для сохранения эффективности число квазирежимов должно быть от 4 до 7.

Примером использования квазирежимов является выбор вариантов в выпадающем меню, при котором в компьютере «Макинтош» пользователь нажимает на кнопку графического устройства ввода и удерживает ее до выбора нужного элемента из списка выпадающего меню, а затем отпускает ее.

Элементы интерфейса можно называть привычными, если они могут легко использоваться с закрытыми глазами («слепым» пользователем).

В основном существует только два типа команд, которые вы вводите в компьютер или в другое устройство для обработки информации:

- команды, которые служат для создания содержания;
- команды, которые используются для управления системой.

В этом отношении предлагается следующий практический подход: квазирежимы должны использоваться для управленческих функций, тогда как для создания содержания должны применяться операции без задействования квазирежимов.

В командах действия к некоторому объекту рекомендуется сначала указывать и, следовательно, выбирать предмет, а затем действие. Например, в текстовом процессоре, когда необходимо в некоторой фразе или абзаце изменить шрифт текста рекомендуется в первую очередь выделить текст, а затем выбрать команду «изменить шрифт». Это, так называемая модель «существительное-глагол». Анализ, проведенный с точки зрения локуса внимания пользователя, показывает преимущества такой модели:

- уменьшение количества ошибок;
- увеличение скорости выполнения работы;
- простота и обратимость при использовании отмены или отката команды.

Таким образом, подход «предмет-действие» является более предпочтительным. Применение метода «глагол-существительное» («действие-предмет») должно ограничиваться только выбором из палитр, если они предназначены для непосредственного использования.

Видимость элементов интерфейса.

При работе с компьютером не всегда ясно, какие функции сейчас доступны, что они выполняют и как получить к ним доступ.

Элемент интерфейса можно считать видимым, если он либо в данный момент доступен для органов восприятия человека, либо он был настолько недавно воспринят, что еще не успел выйти из кратко-

временной памяти пользователя. В противном случае мы говорим, что элемент интерфейса невидимый. Для нормальной работы интерфейса должны быть видимы только необходимые вещи – те, что индицируют части работающих систем, и те, что отражают способ, которым пользователь может взаимодействовать с устройством. Видимость отображает связь между предпринимаемыми действиями и реальной отдачей.

Перед разработчиком интерфейса стоит задача сделать каждый элемент своего продукта видимым.

Поиск и визуализация информации

Большинство задач, выполняемых с помощью компьютера, сводятся к созданию, хранению, поиску, просмотру и редактирования текстовой и численной информации, причем поиск и просмотр лидируют по затратам времени и усилий. Это делает задачу всемерного облегчения этой работы чрезвычайно важной.

Четыре вида поиска и еще два

Существует четыре основных вида поиска информации:

- Поиск конкретных данных (сколько сделок было совершено за последние два месяца?, когда родился Пушкин?)
- Поиск расширенных данных (кто еще участвовал в этой сделке, которая принесла нам столько проблем?, какие еще произведения, помимо «Мертвые души», написал Гоголь?)
- Свободный поиск (есть ли связь между этой сделкой и какими-нибудь другими?, есть ли любовные сцены в «Кому на Руси жить хорошо»?)
- Проверка доступности (у нас есть вообще какие-нибудь данные о том, почему этот контракт был подписан? А у меня есть какие-нибудь книги Толстого?)

В этой классификации есть еще один вид поиска, который в список не попал, частично потому, что он плохо формализуется, а частично потому, что его значение до сих пор слабо осознано. Этим видом является «совсем уж свободный поиск», при котором основной вопрос, который искатель ставит перед системой, звучит так: «а есть ли тут что-нибудь интересное?».

Эти четыре вида поиска (или пять) существуют очень давно, с появления первых библиотек. За сотни лет библиотекари научились очень многому, чтобы поиск нужной информации был эффективен и прост. Потом появился компьютер, вскоре придумали многомерные базы данных и королями поиска стали программисты. Проблема в

том, что до сих пор информацию ищут примерно так же, как искали её сто и двести лет назад.

Основной проблемой поиска всегда было обилие информации. Нетрудно найти нужные сведения, когда у тебя всего один листок бумаги. Когда же нужные сведения нужно найти в библиотеке, состоящей из десятков и сотен тысяч (если не миллионов) листов, жизнь становится значительно более насыщенной. Для решения этой проблемы были придуманы (еще библиотекарями) картотеки, содержащие основные сведения о каждом объекте. Человек формулировал поисковый запрос, а потом тем или иным способом отбирал подходящие карточки.

Этот метод жив и поныне, хотя, конечно, в несколько других формах. Теперь это делается на компьютере (что действительно облегчило жизнь), а поиск производится языком SQL и иже с ним. Метод карточек хорошо справляется с поиском конкретных данных и проверкой доступности. Со всеми остальными видами поиска он справляется из рук вон плохо. Возьмем, например, свободный поиск. Его цель состоит в том, чтобы найти некий образец, закономерность, нечто, что в начале поиска вообще неизвестно («найди то, не знаю что»). Пользуясь методом карточек, приходится совершать огромное количество поисковых запросов, держа при этом в голове полученные ранее данные. Вероятность того, что при этом будет найдена *информация, а не данные*, невелика.

Но выход есть. Чтобы его найти, нужно углубиться в историю и задать себе три вопроса. Вопрос первый, зачем появились карточки. Ответ: потому, что мы не в состоянии охватить взглядом всю информацию. Вопрос второй – почему в результате поиска мы вытаскиваем часть карточек из картотеки? Опять-таки потому, что мы не можем охватить их взглядом, равно как мы не можем охватить взглядом место этих карточек в общей куче. Вопрос третий – как нам оценить отобранные карточки в целом, не отвлекаясь на частности? Ответ: разложить их на столе, молясь, что стола хватит.

Всё это издержки и ограничения самой концепции поиска карточек и вытаскивания их из общей кучи. Эти ограничения имелись, пока мы не имели ничего, кроме карточек. Теперь, когда у нас есть компьютеры, от этих ограничений необходимо избавляться. Компьютер позволяет так визуализировать данные, что появляется возможность увидеть все данные (пускай издали), видя при этом в этих данных информацию.

Т.е. при таком поиске искатель не формулирует запрос, получая на выходе ряд записей базы данных, но задает правила визуализации *всех* данных и видит, какие данные либо выбиваются из общего ряда, либо наоборот слишком уж обычны. Это позволяет, как найти нужные сведения, так и сразу увидеть взаимосвязи и необходимые образцы.

При этом стандартный поиск с последовательностью запросов имеет еще один важный недостаток: он слишком абстрактен. Большинство же людей, хоть и способно создавать сложные алгоритмы, плохо управляется с абстракциями. Не имея ощутимых, не побоюсь слова «видимых», промежуточных результатов, многие люди неспособны сформировать сложный, многоступенчатый вопрос. Визуализация, напротив, позволяет это ограничение обойти.

Но не поиском единым сильна визуализация. Она позволяет также многократно сократить время, затрачиваемое на восприятие найденной информации, за счет того, что визуально выраженные закономерности воспринимаются гораздо быстрее и легче, нежели численные или цифровые (но об этом позже).

Теперь перейдем к практике.

Как визуализировать

Предположим, есть два числа: 83 и 332. Эти числа содержат в себе всю информацию, которую из них можно тем или иным способом вычислить (включая устройство Вселенной). Проблема состоит в том, что они никак не помогают найти нужную информацию, например, мало кто может сразу сказать, что второе число больше первого в четыре раза.

Рис. 1. Если показывать не только цифры, но и их визуальные изображения величины (слева), скорость восприятия взаимосвязи между ними многократно увеличится. Если отказаться от показа цифр (в центре), оставив только изображения из величины, скорость возрастет еще больше, но потеряется возможность точно определить искомые значения (что зачастую не страшно). А если явно показать различия между значениями (справа), либо относительные, как на иллюстрации, либо абсолютные, скорость возрастет еще больше, более того, появится возможность передавать дробные значения.

Это делает числа негодными средствами для быстрой передачи их реальных значений и взаимосвязи между ними. Даже тот факт, что в десятичной системе счисления ширина числа кое-как показывает размер числа (так, понятно, что 20 меньше 200, поскольку число 200 шире), не делает их лучше.

Всегда выводите цифры, предназначенные для сравнения, шрифтом одной ширины.

Таким образом, отдельные числа зачастую необходимо показывать не как последовательность цифр, но как визуальные объекты, свойства которых тем или иным образом связаны с самим числом.

Несколько иная ситуация с текстовыми данными. Понятно, что более-менее сложный связный текст *автоматически* визуализировать невозможно («Мороз и солнце…»). К счастью, обычно такой необходимости и не возникает. Почти всегда нужно визуализировать сравнительные и/или описательные атрибуты, т.е. нет особой разницы между текстовыми и численными данными.

При визуализации массивов отдельных параметров важно добиться не просто красоты и понятности отдельных блоков, но легкости прочтения многих блоков за малое время. Популярно говоря, каждый вариант проверяйте на большом количестве данных, не ограничивайтесь проверкой на одном значении.

Монотонность исполнения элементов интерфейса.

Когда требуется сделать выбор между несколькими методами, ваш локус внимания смещается с текущей задачи на принятие решения о выборе. Это является главным обоснованием монотонности системы. Если условия для принятия решения остаются достаточно простыми и ясными, то в каждом случае вы можете поступить неким привычным способом, тем самым, делая ситуацию монотонной. Таким образом, перед разработчиком интерфейсов стоит задача по поиску монотонного решения для того, чтобы обеспечить такие преимущества как легкость изучения, простоту внедрения, минимум документации и небольшой размер расходов на обслуживание. Монотонность заключается в том, что для вызова одной и той же команды не должно использоваться множество жестов.

Интерфейс, который не имеет режимов и является – насколько это возможно - монотонным, был бы чрезвычайно полезным в ис-

пользовании при условии, что все другие характеристики имеют, по крайней мере, нормальное качество, принятое для современных интерфейсов.

Хорошо разработанный, человекоориентированный интерфейс совсем не требует разделения на подсистемы для новичков и экспертов.

Навигация

В любой системе, использующей единое устройство вывода (читай – экран) для любых задач, качество навигации является важной составляющей: поскольку все физические атрибуты системы фактически отсутствуют, их необходимо передавать с помощью изображения на экране. Наличие в системе развитой навигации здорово облегчает процесс обучения работе с системой, поскольку при визуальной навигационной системе не нужно помнить контекст своих действий (как я сюда попал? что я хотел сделать?), поскольку этот контекст сам по себе показывается на экране. Разработка навигационной системы есть сложная и большая работа, требующая большого количества знаний, навыков и талантов.

Цели навигации Любая система навигации обязана выполнять несколько функций, а именно показывать пользователям:

- **Где они находятся сейчас.** Для понятия «не знать своё теперешнее положение» есть два коротких синонима: заблудиться и потеряться. Ни то, ни другое не приносит удовольствия.
- **Куда они вообще могут переместиться.** Невозможность узнать, что можно сделать дальше, приводит к тому, что ничего не делается.
- **Где они уже побывали.** Невозможность определить пройденный маршрут приводит к тому, что пользователи попадают туда, куда попасть они не хотят (они тут уже были), что является человеческой ошибкой.
- **Куда им разумно пойти.** В большинстве случаев пользователям удобнее не думать самим о том, что им нужно сделать, но воспользоваться готовым вариантом действия, а сбереженные ресурсы потратить на что-либо иное.
- **С какого именно экрана (страницы) они пришли.** Точно знать направление своего движения полезно, поскольку это помогает поддерживать контекст действий.

При этом от навигационной системы требуется множество дополнительных свойств, таких как эстетическая привлекательность, малый размер и так далее (эти свойства не вполне интерфейсные, поэтому здесь они не разбираются).

Так вот, очень важно сделать навигационную систему, которая выполняет максимум этих задач, просто потому, что невыполнение части из них делает систему неполноценной. Всякий раз, создавая навигационную систему, нужно вкладывать в неё максимальное количество ответов.

Битва против меню

Почти любая навигационная система является меню. Это имеет множество достоинств, но также и некоторые недостатки. Во-первых, меню ограничивает возможность выбора только определенным набором вариантов. Это хорошо, поскольку позволяет исключить заведомо неправильные действия. Но в навигационных системах это оказывается нехорошо: взамен свободного передвижения по системе пользователям приходится ходить по заранее проделанным для них тропинкам, т.е. степень свободы пользователей сокращается. Иногда это полезно, но чаще – нет.

Во-вторых, любое многоуровневое меню страдает от каскадных ошибок. Способы борьбы с ними существуют, но от этих ошибок лучше бы вообще избавиться. Сделать же это можно, только изъяв либо само меню, либо потребность его использовать. В-третьих, большинство меню не способно показать пользователям, куда им разумно пойти. Это не значит, что меню плохо, но только то, что нужно как-то усиливать навигацию помимо меню.

Единственным универсальным (и работоспособным) алгоритмом решения всех этих проблем является создание системы из двух не связанных между собой навигационных систем. Первая (меню) показывает пользователю, где он находится и куда он может перейти. Вторая, присутствующая, возможно, не на каждом экране, показывает пользователю, куда он, вероятнее всего, хочет перейти в данный момент. Впервые такая система добилась известности на сайте Amazon.com (хотя идея изначально появилась на firefly.com, к сожалению, этот сайт прекратил существование).

Система анализирует читательские предпочтения, проще говоря, регистрирует, кто что купил и на какие товары перед этим смотрел. Собрав эти данные, система предлагает их пользователям, т.е. позво-

ляет просматривать товары не по абстрактным группам, но по покупательским предпочтениям. Никто, однако, не мешает не собирать информацию, но устанавливать связи, пользуясь результатами анализа действий пользователя и здравым смыслом.

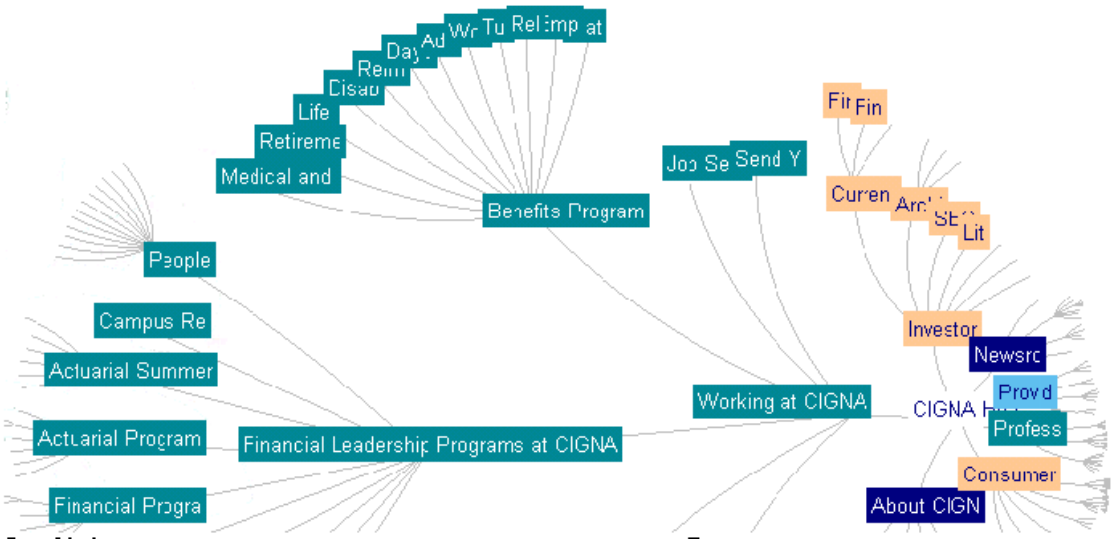

Рис. 29. Это не выглядит как меню, это не воспринимается как меню. Тем не менее, по содержанию это обычное меню, ничем не отличающееся от меню абсолютного большинства сайтов. С другой стороны, такая форма представления меню позволила резко увеличить объем структуры, вмещающейся на экран, и тем самым упростить поиск нужных фрагментов системы (при этом субъективное удовлетворение здорово повышается).

При этом меню становится не основным, а дополнительным навигационным инструментом (точнее, пассивным инструментом: на него можно смотреть, чтобы получить ответы на интересующие вопросы, но с ним не обязательно взаимодействовать).