Разработка интерфейса «человек-компьютер»

Общие рекомендации:

- Интерфейс необходимо проектировать отдельно (как, например, отдельно можно разрабатывать структуру файлов). Состав и форма представления входных и выходных данных должны стать предметом тщательного анализа разработчиков интерфейса.

- При проектировании интерфейса необходимо максимально полно учитывать особенности используемых аппаратных и программных средств.

- Крайне желательно соблюдение стандартов (либо общих, если таковые есть, либо принятых в некотором коллективе разработчиков).

- Необходимо соблюдение общепринятых в эргономике рекомендаций (разумеется, с учетом конкретных особенностей разработки).

- Необходим учет особенностей, целей, проблем потенциальных пользователей. Необходимо привлечение пользователей к оценке (и необходимой модернизации) разрабатываемого интерфейса.

- Интерфейс должен быть адаптируем к изменению круга пользователей, к изменению потребностей пользователей.

Критерии оценки качества интерфейса:

- простота освоения и запоминания операций системы (*сколько времени требуется определенному пользователю для достижения заданного уровня знаний; какова должна быть его подготовка; насколько легко возобновить деятельность после некоторого перерыва в работе и т.п.*);

- быстрота достижения целей задачи, решаемой с помощью системы (*учитывать не быстродействие системы, а время, необходимое для достижения некоторой цели, например, «обработка за час не менее 20 банковских счетов с ошибкой менее 1 %»*);

- субъективная удовлетворенность при эксплуатации системы (*явное мнение пользователя или же количество невыходов на работу*).

Общие требования к диалогу:

- ***Естествен*** (для человека, решающего привычные задачи), т.е. не вынуждает пользователя существенно изменять свои традиционные способы решения задачи; стиль ведения диалога должен быть разговорным, а не письменным; следует избегать как чрезмерной напыщенности, так и «фамильярности»; диалог должен вестись на родном языке пользователя (или на другом понятном и привычном ему языке).
- ***Последователен*** (логически); последовательность (единообразие) в организации диалога с различными модулями системы, в построении сообщений, в использовании форматов данных, в размещении данных (однотипных) на экране.
- ***Неизбыточен***; ввод минимума информации, необходимой для работы системы; выходные сообщения должны содержать именно ту информацию, которая требуется пользователю, причем в приемлемой для восприятия форме; следует широко использовать принцип «по умолчанию».
- ***Обеспечивает поддержку пользователя***; поддержка пользователя – мера помощи, которую система оказывает пользователю при его работе; основные аспекты: количество и качество имеющихся инструкций, характер выдаваемых сообщений об ошибках, подтверждение тех или иных действий системы.
- ***Гибок***; гибкость диалога – мера того, насколько хорошо он соответствует различным уровням подготовки и производительности труда пользователя.

Рассмотренные нами критерии эффективного взаимодействия человека с компьютером касаются не только структур диалога и содержания отображаемых сообщений системы, но и того, как эти сообщения выглядят на экране.

Процесс размещения данных на экране – Форматирование экрана.

Прежде всего разработчику интерфейса следует определить:

- **какая** информация должна появляться на экране (с учетом неизбыточности диалога);
- каков основной **формат** этой информации;
- **где** эта информация должна появляться (область вывода для каждого поля; пустые места);
- какие средства используются для **выделения** (цвет изображения и фона, инверсное изображение, мерцание текста или фона, яркость и др.).

После этого можно:

- **разработать проект** размещения данных на экране;
- **оценить эффективность** этого размещения.

Рекомендуется оценка эффективности предложений во взаимодействии с пользователями / потенциальными пользователями.
Этот процесс часто носит итеративный характер.

Данные должны располагаться на экране так, чтобы пользователь мог просматривать экран в логической последовательности и мог легко:

- выводить нужную информацию;
- идентифицировать связанные группы данных;
- легко определять исключительные ситуации (сообщения об ошибках, предупреждения);
- определять, какое действие он должен выполнить (и должен ли вообще что-то делать) для продолжения выполнения задания.

Какая информация должна выводиться на экран?

Желательно – на экране находится та и только та информация, которая действительно необходима пользователю на данном этапе работы. Например, в меню не следует включать пункты, соответствующие недоступным в данный момент действиям, или же пункты, с которыми работает узкий круг пользователей.
Однако есть и общие правила/рекомендации: оставлять примерно половину площади экрана пустой; оставлять пустую строку после 4-5 строк таблицы (или выделять строки, например, поочередно, фоном); оставлять несколько пробелов между столбцами таблицы.
Логически связанные данные должны быть представлены отдельной группой.
Фрагменты текста должны располагаться на экране так, чтобы взгляд пользователя сам перемещался по экрану в нужном направлении.
Общепринятую схему просмотра (начать из левого верхнего угла и перемещаться слева направо и сверху вниз) желательно менять только в крайних случаях (расположение материала на экране задает другую схему).
"Нельзя допустить, чтобы пользователь при заполнении экранной формы мучился так же, как разработчик при ее составлении." [2, с. 246].

В каком виде информация должна выводиться на экран?

Естественность диалога предполагает, что информация представлена таким образом, что ее можно срезу же использовать. Недопустимо требовать, чтобы пользователь обращался к справочникам или выполнял какие-то промежуточные вычисления.
Общепринятая система больших (прописных) и малых (строчных) букв в тексте облегчает его восприятие. "Программу для вывода формы, представленной на рис. 7.2, наверняка составлял программист с устаревшими взглядами, который даже мыслит заглавными буквами!" [2, с. 243].
В таблицах поля и данные (и их наименования)должны четко различаться.

В каком месте экрана должна располагаться информация?

Плотность расположения данных (это понятие субъективное) зависит и от задачи, и от категории пользователя.
Данные на экране для опытного оператора/пользователя могут располагаться плотнее, чем для начинающего.

Выделение информации на экране.

Выделение информации – использование средств, позволяющих привлечь внимание пользователя к некоторой области экрана (и, следовательно, к представленной в этой области информации).
Неудачное выделение информации может помешать работе пользователя. Поэтому при выборе способов/средств выделения информации следует учитывать психофизиологические особенности человеческого восприятия (а при проектировании интерфейса разработчику рекомендуется сначала расположить информацию без выделения полей и только после этого решить какие поля следует выделить и как).
Рекомендации по использованию цвета:
– минимальное количество цветов (не более 3-4 на одном экране) замечание о психолингвистике;
– для больших прямоугольников следует выбирать цвет фона;
– яркие цвета для выделения данных, а более спокойные тона – для фона;
– для выделения двух областей уместны контрастные цвета (из разных концов спектра);
– цвет должен соответствовать представлениям (ассоциациям) пользователя;
– целесообразны эксперименты с различными композициями цветов на реальном экране.

Время ответа (интервал между событием и реакцией системы на него):

Время ответа системы – время от момента ввода последнего символа команды/запроса до момента вывода первого символа ответа системы.
Быстрый ответ системы: не задерживает пользователя,
впечатляет, создает благоприятное представление о системе.

Психологические аспекты скорости ответа системы на запрос:
- преемственный и предсказуемостный отклик системы;
- ответ системы не должен быть слишком быстрым, иногда необходимы паузы;
- гипотеза Ингве (7 ± 2) об объеме оперативной памяти человека;
- ограниченность времени хранения информации в оперативной памяти (около 2 секунд).

Рекомендации по допустимому времени ответа:

Подтверждение факта	- 0.1 – 0.2 с
Выполнение простой команды	- 0.5 – 1.0 с
Выполнение команды связного диалога	- 1.0 – 2.0 с
Обработка сложного запроса	- 2.0 – 4.0 с
Пакетная обработка	- 10 – 20 с

Иногда считается, что если время ответа более 20 секунд, то система не является интерактивной.

Если все же процесс обработки длителен по сути (например, при поиске вирусов на жестком диске большого объема), необходимо информировать пользователя (постоянно или время от времени) о нормальном протекании процесса.

Для этого используются, в частности:

- подвижная шкала, наглядно показывающая объем проделанной и объем оставшейся работы;
- оценки времени объема оставшейся работы;
- информация о проценте объема проделанной работы;
- комбинации этих приемов (например, шкала и оценка времени оставшейся работы).

Почему трудно добиться быстрой реакции системы?

Время ответа зависит от загрузки ресурсов системы:
- времени доступа к файлам,
- времени работы ЦП,
- времени передачи информации по сетям.

При возрастании нагрузки на систему:
- возрастает загрузка ресурсов,
- образуются очереди,
- возрастает время ответа (резко возрастает при использовании системы на 80% и выше).

Реакция на загрузку зависит от:
- среднего числа задач,
- среднего «размера» задач,
- разброса размера задач относительно среднего.

Необходимость гарантии того, что система работает (сообщения о состоянии).

Адаптация

Фиксированная адаптация – пользователь явно выбирает уровень диалоговой поддержки. Первое решение – выделение двух уровней:

- ***подробный диалог***, обеспечивающий всемерную поддержку начинающих;

- ***краткий диалог***, предназначенный для экспертов и обеспечивающий небольшую поддержку либо совсем лишенный ее.

Недостатки: навыки меняются со временем; пользователь может хорошо знать одну часть системы и совсем не знать другие; выбор уровня (пользователем) может быть неверен; автоматическое определение уровня затруднительно.

Косметическая адаптация – приспособление компьютерной системы к развитию у пользователя навыков работы с ней без учета поведения пользователя и без однозначного выбора им конкретного стиля диалога:

- умолчания,
- сокращения,
- синонимы,
- "горячие клавиши",
- опережающий ввод,
- многоуровневая помощь.

Косметическая адаптация, т.к. вносит лишь поверхностные изменения в базовую структуру, но тем не менее весьма полезна, поскольку снижает утомляемость и делает интерфейс более универсальным.

Гибкость при сравнении (входного сообщения и "цели").
1. Полное совпадение.
2. Совпадение сокращений:

Слово	Правильные сокращения
мотор	мо мот мото
содержимое	с со сод содерж содержим
собственность	с со соб собств собствен
конец	к ко кон

Ситуации: нет совпадений, однозначное совпадение с целью, неоднозначное совпадение.
3. Частичное совпадение.
Простейший вариант. Удаление по одному символу с конца "цели" до совпадения с входными данными (так, можно работать с опечаткой типа – **prpo** вместо **prop**).
Более серьезные методы предполагают использование метрик, статистики ошибок, учета их причин (в том числе, расположения клавиш на клавиатуре, регистра и др.).

Синонимы
Пример: в MS-DOS синонимичны **DEL**, **ERASE** и **del**.
Преимущество синонимов – пользователь может выбирать наиболее естественный/привычный для него вариант написания идентификатора/команды.

Умолчания
Система заранее настраивается на стандартный (наиболее вероятный) ответ/сообщение пользователя (например, дата – обычно текущая дата; номер следующего документа – обычно следующее натуральное число, принтер – обычно основной принтер, подключенный к компьютеру). Для принятия соответствующего значения: "нулевой ввод" (**Enter**), нажатие "активной" экранной кнопки.
Проблемы: неэффективно, если вводятся разные данные; стандартный ответ может меняться при смене пользователя и с течением времени.

Опережающий ввод
Опережающий ввод символов.
Если при вводе первых символов сообщения система обнаруживает, что сообщение целиком хранится в некотором буфере (файле, адресной книге и т.п.), система дополняет введенную цепочку и выдает один или несколько вариантов полного сообщения. Это сообщение хранится в системе и было когда-то введено пользователем (или другими пользователями) ранее.
Можно использовать и механизмы, обеспечивающие опережающий ввод цепочки ответов/команд.

Многоуровневая помощь
Система циклически выводит сообщения и запрос о необходимости более детального объяснения (до тех пор, пока пользователь не будет удовлетворен либо не будут исчерпаны все уровни помощи.

Полная адаптация – формирование в компьютерной системе модели пользователя (или группы пользователей) и модификация этой модели в процессе работы.

Изящная адаптация и модели пользователя
Любые интерфейсы предполагают наличие (обычно неявной, ***"вшитой"*** в них) модели мира, лежащей в основе их взаимодействий с пользователем. Их мир – некоторый класс прикладных задач. Каждый пользователь имеет свою собственную модель этого мира приложений, используемую им при общении с компьютерной системой (формирование задания и входных данных, интерпретация сообщений системы).
Требование ***естественности диалога*** – требование (как минимум) сопоставимости этих моделей.
Вспомним об индивидуальных языковых моделях и проблемах человеческого общения на естественном языке – Лекция № 7.
Отличия в моделях приводят к ошибкам "взаимопонимания"и затрудняют общение пользователя с системой.
Любой интерфейс содержит неявную модель мира приложений, "внесенную" в него разработчиками и отражающую их взгляд на эту модель.
"Большинство из более или менее удачных интерфейсов человек-компьютер потерпели провал из-за того, что модель на которой проектировщик основывал систему, существенно отличается от модели, сложившейся у пользователя" [2, с. 401].
Примеры с "эксцентричным" цветным кодированием [2] и выходом из системы с помощью ***F1***.
Проблемы:
пользователей может быть несколько;
даже в однородной группе пользователей могут существовать индивидуальные различия в моделях;
даже единственный пользователь со временем меняется.
Вопрос в духе "Диполя Тыугу": "Почему пользователь должен подстраиваться под системную модель, а не системная адаптируется либо сама, либо кем-то, подстраиваясь к текущей модели пользователя" [2, с. 402].
Путь – отделение содержания и представления системных сообщений и правильных ответов от структуры диалога; в дальнейшем, персонификация диалога.

Добавление ускорителей (опережающий ответ, гибкость при сопоставлении) обеспечивает бо̀льшую адаптивность. Желательно, чтобы и пользователь мог сам модифицировать содержание диалога (например, перефразировать подсказки).
Мы говорили ранее о различных уровнях подготовки пользователей (групп пользователей) в связи с выбором схемы диалога.
Но как определить автоматически уровень подготовленности пользователя?
Интерфейсу/его разработчику необходимо иметь не только модель задач приложений, но и модели пользователей (***профили пользователей***).
Сложности создания таких профилей.
Учет психофизиологических факторов (характер, способность к обучению, знание задачи, знание интерфейса). Часть из них (личностные) относительно устойчивы, другие – динамичны.
Что можно учесть? Время отклика пользователя; характер ошибок ввода (вспомнить Лекцию № 6), степень, в которой используется опережающий ответ; историю работы с системой.
Изящная адаптация (вероятно, ***smart***) – термин, введенный для описания такого типа интерфейса, который обеспечивает разумную меру самоадаптации:
гибкость с точки зрения возможных входных сообщений и выходной информации;
способность к персоналификации;
не прерывает пользователя, но информирует о неоднозначных и нераспознанных входных данных;
интерфейс должен "следить" за вниманием пользователя;
в идеале желательны "совместные" действия интерфейса и пользователя, аналогичные тем, что возникают при общении людей.
Изящное взаимодействие – лежит в основе **Интеллектуального интерфейса** (как мы его определяли на Лекции № 2).
Интеллектуальный интерфейс – совокупность программных и аппаратных средств, позволяющая конечному пользователю решать на компьютере характерные для его повседневной деятельности задачи без помощи посредников-программистов.
Расширение взаимодействия между человеком и компьютером с помощью:
- увеличения диапазона способов ввода и вывода;
- обогащения грамматики ввода и вывода;
- попытки кооперации с пользователем в достижении целей.

Эти расширения отражают тенденцию приближения человеко-машинного интерфейса к общению между людьми. Они требуют наличия в системе модели проблемной среды, в которой работают человек и компьютер и которая близка к модели, существующей в представлениях человека.

Еще несколько замечаний об интерфейсе.
(**Факультативный материал**; в программу второго коллоквиума и экзамена не входит! Но прочитать рекомендуется!)
Джефф Раскин в своей книге " Интерфейс: новые направления в проектировании компьютерных систем" [4, с. 7-13] отмечает (в цитируемые фрагменты лектором внесены отдельные мелкие изменения):
"1) Разработчики интерфейса должны не только включать в него четкие указания для пользователя, но и предусматривать возможность нелогичных действий с его стороны (в программистском сленге для обозначения этого есть термин "защита от дурака"). На самом деле, нежелательные последствия – это, конечно, в первую очередь результат непредусмотрительности программиста. Необходимо продумывать возможные стратегии поведения пользователя, чтобы не позволять ему "попадать в ловушку" (например, неожиданный выход из системы) или идти единственным путем: у него должен быть хотя бы небольшой выбор (например, чтобы не повторять каждый раз уже отработанную цепочку).
2) Это позволит как новичку, так и пользователю, выполняющему работу, быстро пройти дальше. Одновременно это обеспечит возможность человеку, который захочет исследовать интерфейс, поиграть во "что – если".
Раскин подробно рассматривает требования, которым должны соответствовать исследуемые интерфейсы.

- **Не следует помещать в главном меню пункты, срабатывающие сразу после нажатия**. Пользователи привыкли, что любой пункт главного меню в программах всегда содержит «подменю». Помещение в главное меню пункта, который срабатывает сразу при нажатии на него, делает программу непредсказуемой. Чаще всего это бывает пункт "Выход".
- **Делайте действия обратимыми.** Люди исследуют интерфейс не только с помощью навигации. Иногда они хотят знать, что случится, если они сделают что-то потенциально опасное. Иногда они не хотят этого, но такое может произойти случайно. Если действия обратимы, пользователь может исследовать интерфейс и не бояться за последствия.

- **Всегда предоставляйте отмену (undo)**. Если Ваша программа не предоставляет возможности отмены, Вам неизбежно придется создавать массу диалогов с вопросами типа "Вы действительно уверены?". Это, безусловно, замедляет работу, но отсутствие таких диалогов (если Вы опустили "отмену") будет замедлять работу еще больше. Специальные исследования показали, что люди в опасной среде делают не больше ошибок, чем в отзывчивой и более ясной среде, но работают гораздо медленнее и осторожнее, чтобы избежать необратимых ошибок.
- **Всегда предоставляйте запасной выход.** Пользователи не должны чувствовать себя пойманными в ловушку. Путь назад должен всегда быть четко обозначен. Однако делайте так, чтобы остаться в программе было легче. Более ранние программы создавались так, чтобы их трудно было покинуть. С приходом Internet чаще появляются программы, в которых трудно остаться. Браузеры напоминают гирлянды объектов и опций, которые не имеют ничего общего с самой программой. Если пользователь имеет на экране набор элементов, в котором 49 элементов ведут напрямую к разрушению работы и только один-два способны реально помочь – такой интерфейс не является исследуемым.
- **Предлагайте пользователю постоянные наглядные подсказки, чтобы он чувствовал себя "как дома" (очень эффективны "всплывающие" подсказки).** Стабильные визуальные элементы не только позволяют людям быстро перемещаться, но и работают как указатели, давая пользователю ощущение психологического комфорта. Если в ставший привычным пользовательский интерфейс начать вносить некоторые изменения, то его, как и любой язык в такой ситуации, будет труднее понимать. Люди в большинстве своем консервативны по природе, не любят отказываться от привычек и привыкать к чему-то новому, особенно когда этого нового много. Этот фактор необходимо учитывать. Поэтому, если разрабатывается программа, аналог которой уже существует и которым пользуются многие, то имеет смысл посмотреть, как организовано взаимодействие с пользователем в уже существующей программе, и позаимствовать ее основные принципы, тогда новая программа не вызовет отторжения, так как сделана в привычном для пользователя стиле. Она будет более привычной для пользователя, а, главное, ему будет легче перейти к новой программе, если она лучше аналогичной.
- **Стремитесь к единообразию интерфейса в разных подсистемах (частях программы).** Необходимо тщательно продумать и осознать схему программы, приведя ее к системе (структуре, выстроенной по определенным правилам), и реализовать пользовательский интерфейс в соответствии с этой системой. Пользователю достаточно будет «схватить» суть, чтобы по аналогии разобраться в остальном. Следует избегать лишних элементов: чем меньше их будет, тем проще пользователю будет разобраться.
- **Обеспечивайте защиту работы пользователя.** Удостоверьтесь в том, что пользователь никогда не потеряет свою работу в результате либо своей ошибки, либо неустойчивости соединения, или по какой-то другой причине, кроме неизбежных, таких, как неожиданная потеря питания".

Еще одна проблема, на которую обращает внимание Д. Раскин [там же], – **Сообщения, подтверждающие действие системы.**

"Сообщения, подтверждающие какое-либо действие системы, требуются для того, чтобы пользователь мог еще раз убедиться в том, что система выполнила или будет выполнять требуемое действие. Подтверждать ввод данных следует тогда, когда эти данные приведут к необратимым действиям системы (например, удаление записи из файла) или когда вводится код и пользователю необходимо проверить по соответствующему описанию, нужная ли запись выбрана (например, о конкретном энергоресурсе или административной области). Также может оказаться желательным подтверждение, что было выполнено некоторое действие (например, добавлена запись в файл).

Может оказаться полезным не только подтверждение выполнения какого-то действия, но и вывод сообщения о состоянии системы (чтобы пользователь знал, где он находится). Это можно реализовать, отведя на экране специальную область, в которой будет отображаться путь, "пройденный" пользователем до текущего состояния.

Механизм ***статуса*** используется, чтобы держать пользователя в курсе того, что происходит. Механизмы статуса жизненно необходимы для предоставления информации человеку, чтобы он мог соответственно реагировать на изменение условий. Простой пример: человек, не получая информации о статусе, в течение периодов ожидания, пока процесс действительно не завершится, будет напрягаться и уставать без необходимости, так что в следующий раз у него может не хватить физических и умственных ресурсов на это.

Информация о статусе должна быть актуальной и легкодоступной. Пользователи не должны искать информацию о статусе. Они должны иметь возможность, одним взглядом окинув свою рабочую среду, получить хотя бы приближенное понятие о ходе работы и состоянии системы. Информация о статусе может быть, например, такой: иконка папки "Входящие" может иметь три состояния - пустая, наполненная и переполненная. Однако здесь нельзя переусердствовать. Иконка "мусорной корзины" на Макинтоше долгие годы изображала неминуемую опасность взрыва, когда хотя бы один документ находился внутри, так что люди старались очищать корзину сразу же после каждого документа. Это не только превратило одношаговую операцию в двухшаговую (перетащить в корзину, очистить корзину), но и лишило смысла саму идею мусорной корзины - возможность отмены удаления".