Human Factors

Building Systems that Work for People

- Interaction of people with computers.

**Goal:** to make the system easy and comfortable to use so the user can be more productive on the job.

- Human factors focus the design team on serving the needs of the user.

- Once an elegant interface to the system is built it cannot be easily changed.

WHY!
Human Factors

The design team MUST

- plan for user interactions from “day one”
- understand and care for the customer's perspective

The customer is key!

Quality: the Customer’s View

1. Easy to install
2. Easy to learn and obtain help
3. Works the first time (quality)
4. Works the next time (reliability)
5. Intuitive command interface where the user is always in control

Quality

6. "Expert" mode for time saving
7. Quick and predictable response
8. "Undo" capability
9. Safe data storage (security and easy data backup/ recoverability)
Quality

10. Easy to maintain and upgrade the software
11. Compatibility with earlier versions and other systems
12. Easy to modify, customize and program
13. Low first cost with incremental add-on features
14. Good value (cost/benefit)

Human Factors

- Traditionally -- focused on safety and productivity (physical workstation design)
- Increasingly, focus is on the user interface

Human Factors

- Hardware Interface for ease of use
  - keyboard/screen make way for mouse/table/audio
  - Response to repetitive stress injuries
- Sensors
  - keyboard clicks assist the user -- HOW?
- Menu Designs
- Screen formatting
Menu Designs

If number of commands are small, place on a single menu

otherwise

Hierarchy of Menus

Studies show that:

- novices make fastest selections from alphabetical listings rather than from functional or random listings
- expert users perform at about the same speed using alphabetical or functional order
- random ordering is a poor third

Screen formatting

- key = impact on readability and acceptability
  - color matters
- use high contrast for the important menu items
  - makes them easier to read
- graphics MAY enhance a computer interface if the graphics are well done
What makes the screen work?

- Don’t clutter the screen
- Get important information on the screen first.
- Adequate spacing and blank lines matter
- Use the whole width of the screen
- If information on the screen is not needed for the next transaction, or is not needed, it should be removed from the screen

User Interface

The user interface is the first contact the user has with any interactive program

User Interface

The user interface can determine whether a program is considered
- wonderful
- acceptable
- intolerable
User Interface

If the system does what it is supposed to do, is a model of efficiency with regard to CPU time and memory. It is not worth the paper it was debugged with -- if user doesn't like it!

Conversely, a system that is pleasant to use can mask some deficiencies in the system. Remember, from the user's viewpoint, the major part of the system is in the "look and feel" of the interface.

"Look and Feel"?? -- Legal Issues
User Interface

- End users are typically unable (or unwilling) to understand a formally defined specification of the required interface.
- Graphical or forms-based user interfaces are increasingly common.
- A significant portion of development costs goes into the user interface.

User Interface

- Designers cannot IMPOSE their view of what a system should look like on users.
- To the degree possible, the user should be involved in the development of the user interface.

User Interface

- Rapid prototyping can be a helpful communicate tool - to help users understand how a system works and ensuring it will meet their needs.
- In building a GUI prototype
  - provide prompts to the user and handle input
  - call appropriate functions for each response, but need not process any data.
  - tests output, by demo-ing displays.
The user interface is the most unpredictable part of any interactive program.

**Murphy's Law:**
"Anything that can go wrong will GO WRONG!"

User Interface : Murphy's Law
These include:
- invalid input by the user
- invalid output by the system
- errors that crash the system

User Interface : Murphy's Law
Error Handling must be used to check for:
- **input information**
  - whether from the keyboard, files or anything else
- **output information**
  - make sure that it is reasonable from the user's view
- **internal states that could cause termination**
  - such as end-of-file conditions
User Interface: Murphy's Law

- When errors are detected, the user MUST be given information on what action to take. This is usually in the form of a text message.
- **Antibugging**: The idea of constructing traps to detect potential errors

User Interface: In practice

- Recognize that *program design is fallible*
- Build error traps --
-- to contain and identify *any conceivable faults that might cause loss of execution control.*

User Interface: In practice

- In an interactive system, the first goal of error handling is to *retain control* no matter what *stupidity is inflicted on the system by the user.*
- If the user can get to the operating system through any sort of bad data entry, your system has failed!
  (this does not include OS overrides such as CTL-C or Reset)
User Interface: In practice

Regardless of the time and effort put into designing the prompts and other information messages, sooner or later, the users will enter *incorrect information*!

User Interface: In practice

While this *MAY* be due to incompetence on the user's part or a lack of clarity on the programmers part many other possibilities exist:

User Interface: Problems?

1) the cat may stroll across the keyboard, producing strange error messages we might not otherwise see
2) an incorrect key may be pressed due to inattention to the screen
   - they baby is crying
   - the boss walks in
   - a message is heard on the radio
3) someone may rest an arm on the keyboard while reaching for a cup of coffee
User Interface

- The programmer must guard against all these possibilities, which means trying to think of anything and everything that can go wrong short of turning the system off.
- Users have the "right" to expect that "obviously wrong" data will be rejected or at least flagged by the system.

User Interface: Actions to Recover

- Error messages should indicate not only what happened but also what to do about it.
- Try to give the user some idea of what went wrong, but in the briefest possible terms.
- A good set of error messages should allow an average user to correct errors and recover without having to read the manual every time.

User Interface: Actions to Recover

- The manuals should have more detailed descriptions of how to help the novice user.
- A message such as "ERROR 43" is not acceptable in a delivered product since it requires the user to refer to the manual.
- "Cute" messages, such as: "YOU DID IT AGAIN, JERK" should NEVER be used!
User Interface: Error Messages should never:

- insult the user
- be ambiguous
- be unhelpful
- be just a number

User Interface: Error messages should always:

- clearly indicate the nature of the fault
- suggest actions to recover
- avoid jargon as much as possible
- classify the error, e.g. data entry, system software, hardware

On the opposite end of the interface:

- Users become particularly irritated when a computer system behaves in a way that is unexpected.
- Interfaces need to be designed so that similar actions always produce similar results.
User Interface Design:
Remember

- Must take into account the needs, experience and capabilities of the system user.
- Physical and mental limitations of the individuals who use computer systems
- Short term memory is very limited and can be easily overloaded.

User Interface Design:
Remember

Users should not be forced to adapt to an interface because it is convenient to implement.

or
Because the designers “like” it!