Human Factors

“Human factors discovers and applies information about human behavior, abilities, limitations, and other characteristics to the design of tools, machines, systems, tasks, jobs, and environments for productive, safe, comfortable and effective human use.” (Chapanis, 1985)


Part 1 – Introduction
Ch.1 – Human Factors and Systems
Ch.2 – Human Factors Research Methodologies

Part 2 – Information Input
Ch.3 – Information Input and Processing
Ch.4 – Visual Displays of Static Information
Ch.5 – Visual Displays of Dynamic Information
Ch.6 – Auditory, Tactual, and Olfactory Displays
Ch.7 – Speech Communications

Part 3 – Human Output and Control
Ch.8 – Human Physical Activities
Ch.9 – Human Control of Systems
Ch.10 – Controls
Ch.11 – Hand Tools and Devices

Part 4 – Work Space and Arrangement
Ch.12 – Applied Anthropometry and Work Space
Ch.13 – Physical Space and Arrangement

Part 5 – Environment
Ch.14 – Illumination
Ch.15 – Atmospheric Conditions
Ch.16 – Noise
Ch.17 – Motion

Part 6 – Human Factors: Selected Topics
Ch.18 – Human Factors Applications in System Design
Ch.19 – The Built Environment
Ch.20 – Highway Transportation and Related Facilities
Ch.21 – Human Error and Work-Related Topics

What is the User Interface?

The part of the system the user directly interacts with.

- E.g. what makes a Windows machine different from a Macintosh from the user’s point of view.

Tendency to identify interface design problems with superficial properties of the interface:

- "Come look at our screens."
- Screen colors
- Graphic quality
- Widget appearance
- "Look and feel"

User Interface involves:

- Specific hardware features of machine
  - e.g. type of display and input devices
- General environment supplied by the machine
  - E.g. dedicated word-processors vs general purpose computers
- Content of displays
  - What information is present?
  - When and where is the information available?
  - How is the information represented?
- Behavior of software - procedures user must follow:
  - Conventions or user interface standards for a machine or system
  - E.g. the ESC key often has a consistent effect in PC software
  - Specific behavior of a particular application program.
    - What does the user have to do?
    - What does the program do for the user?
  - Specific behavior of the operating system
    - How does it support user tasks?
- Supporting documentation, training, technical support, and on-line help
  - Well designed interfaces require less support.
  - Well designed documentation requires less training and technical support.
Usability is a Scientific and Technical Issue

Not just a matter of opinion.
Not just a matter of "what you are used to."
Certain terms will not be used in this course because they have no technical meaning and no reliable characterizations:
• "User-friendly 
• "Look and feel"
• "Intuitive"
• "Natural"
Rather, in this course, "usability" means:
Ease of learning
- Can the system be learned quickly and easily, either for the long term, or for immediate, "walk up and use," purposes?
Ease of use
- Can the system be used to accomplish work rapidly, accurately, and without undue effort?
These are the two major aspects of usability.
• Often do not have to be traded off against each other.
• Can be measured, quantified, and even predicted.
Some other aspects of interface design are unrelated to usability.
These are not addressed in this course.
• Ease of implementation
• Attractiveness
• Marketability
• Motivating value
• Entertainment value
If a system is difficult to learn or to use, customers are likely to be dissatisfied eventually, even if it is a market success.

How to Design a Good Interface:
The Course Approach

Understand the tasks that the users need to do.
Choose system functions that will help the user do the tasks.
Propose a preliminary system and interface design.
• Driven by what will help the user do the tasks.
• Following platform conventions, as appropriate.
Evaluate the design, and identify how it can be improved.
• Determine whether design is good enough, or is better than an alternative design.
  - User testing - have representative users perform tasks using a prototype of the system, look for problems.
  - Analytic evaluation - characterize how the system is supposed to be used in the tasks, look for unnecessarily slow and complex procedures.
Revise the design, and repeat as long as possible.
• A median 50% improvement per iteration (Landauer, 1995).
The focus: What do users need to do, and how can the system help them do it?

This Course Does Not Provide a Cookbook

Everybody would like an interface cookbook, containing prepackaged solutions to interface design problems like:
• A GUI is always the best choice.
• Design your screens this way.
• Use this color scheme.
• Use pop-ups here, list boxes there, modal dialogs here, on-screen buttons where possible, etc.
• Do it this way, and you'll be successful.
The course will provide considerable practical advice:
• Principles for what is important in human-computer interaction.
  - A conceptual framework for how to deal with important issues.
  - Some guidelines for what design possibilities to consider.
• Techniques to apply during interface and system design.
But not a "cookbook" - solutions must be worked out for each design.
• No short-cut cookbook for usable interfaces is possible.
The characteristics of a good interface depend almost completely on the details of what the user needs to do.
• An interface is good only to the extent that it helps users get their work done.
• Before designing an interface, you have to know what the user needs to do.
Role of platform style guides:
• Platform style guides provide conventions, some consistency, but most were not developed on the basis of usability.
  - Don't mistake following platform conventions for designing for usability.
• Use style guides to make low-level decisions so that at least the user can depend on platform-consistent behavior.
  - E.g. in Motif, users expect a file selector widget to look a certain way.

Aims and Objectives of This Course

• Identify and describe specific usability evaluation techniques and methodologies that can be used to evaluate an interface.
• Explain the strengths and weaknesses of each of the analysis techniques discussed in the course.
• Use task analysis to develop an understanding and model of a user's work activities.
• Build Keystroke Level Models and GOMS (Goals, Operators, Methods, and Selection Rules) engineering models that accurately predict the time required to accomplish a task using a specified interface.
• Design and build a user interface using Java Swing based on a careful analysis of task requirements.
• Analyze that interface using well-established methodologies, and identify improvements to the design based on these analyses.
• Evaluate an interface by conducting a cognitive walkthrough.
• Implement your design improvements, and validate the revised interface.
• Evaluate your interface by conducting a user observation study.