Where human factors fits in the design process

Kathleen Potosnak,
Human Factors Editor

Applying human factors engineering to software design is more than just following guidelines on screen layouts and use of attributes such as color and blinking. In addition to guidelines, the bag of tricks carried by a human factors engineer contains
- a research base in journals, proceedings, handbooks, newsletters, textbooks, and special reports;
- models to predict human behavior and describe tasks;
- standards representing consensus on particular design parameters;
- principles for guiding the design process;
- methods of finding out about users, their tasks, and how they use systems;
- techniques for creating and testing system designs; and
- tools for designing user interfaces and for evaluating designs.

To demonstrate the full range of human factors information and techniques, I've outlined how these things can be used in the traditional software-design process. (The generalized description of software development is adapted from the article "Design Education in Computer Science and Engineering" by Taylor Booth et al. on pp. 20-27 in the June 1986 Computer.)

This issue's report is an overview of where human factors fits in the design process. It is a broad look at the forest of human factors engineering rather than a complete description of how to do it. Future reports here will take a closer look at the trees, so don't worry about details for now.

Requirements definition. Once the general concept for a system has been proposed, software developers traditionally begin by defining requirements the system will meet; establishing system functions, inputs, and outputs; and listing economic considerations and constraints on system performance.

During requirements definition, human factors engineers:
- Provide input on strategies for successfully applying human factors techniques to the design process.
- Identify users' work habits, skills, capabilities, expectations, goals, and experience levels.
- Evaluate existing systems and perform task analyses. (Task analysis is an observational method for collecting quantitative and qualitative information about how people work with a system. It involves systematically recording people's actions while they work.)
- Determine system functionality and the general level of usability needed.
- List environmental constraints on the design such as lighting and noise levels. This information is helpful in choosing hardware, designing color screen layouts, and selecting auditory signals.

Specifications development. In the traditional development process, software designers then determine the specifications for exactly what the system must do, and they specify acceptance tests the system must pass.

At this stage, human factors engineers:
- Translate information about users into system specifications by creating a user profile. The profile contains lists of user characteristics such as perceptual abilities, educational levels, memory demands, attitudes, expectations, computer proficiency levels, typing skills, and reading ability. Each user characteristic is translated into one or more system specifications. For instance, people who are not touch-typists may prefer a system that lets them select options from a menu rather than one that makes them type their requests.
- Develop a model of the task and begin organizing the functions to support the task procedures.
- Define criteria for usability, learnability, and user acceptance against which the system will be evaluated.

Criteria are stated in measurable terms. For example, "Users will learn the 10 fundamental functions of the system in 20 minutes. After this learning period they will be able to use these functions to complete a benchmark task in five minutes with no more than one error."

Preliminary design. During this stage, designers develop models of possible solutions for satisfying the requirements and specifications; determine advantages and disadvantages of each solution; and choose one or two promising solutions for further evaluation and investigation.

Human factors engineers:
- Determine which parts of the task will be performed by the user and which will be performed by the computer system. For example, calculations are best performed by the machine and subjective decisions are best left up to the user.
- Generate ideas for the design of the user interface. This includes providing input on the nonsoftware aspects of the system, such as hardware configurations, documentation, training, and support.
- Use human factors theories, models, and guidelines to evaluate different interface styles and to select promising interfaces for further development and testing.
- Begin the Wizard of Oz technique if this method is being used. The Wizard of Oz technique combines prototyping, simulation, and iterative testing. A bare-bones system that functions at a minimal level is configured so all inputs to and outputs from the system can be monitored and interrupted by a human intermediary (the wizard). The intermediary observes real users working with the system and intervenes in the dialogue when difficulties are encountered. For instance, when the user enters a valid request in a way the system won't understand, the wizard supplies an understandable message that was prepared before the testing session began. Since users don't know that an intermediary is in the loop, they naturally react as if working with a complete system. The system is iteratively built.
based on users' interactions with it.

**Intermediate design.** The intermediate design phase is the point at which designers start to fill in some of the details. They develop the selected designs to demonstrate system organization and structure, and they use models and analysis methods to evaluate these designs.

- During intermediate design, human factors engineers:
  - Select the dialogue types (for example, commands, menus, and graphics) and the input and output devices that will be used.
  - Determine the order in which task steps will be performed to define the structure and organization of the user interface.
  - Create a prototype or develop a simulation of the system if iterative design and testing will be used.
  - Begin collecting data with users if they are using the Wizard of Oz technique.

**Detailed design.** During this phase, system designers create a detailed architectural design of the proposed system. They evaluate it to determine if it meets the specifications and requirements. Prototypes of subsystems are sometimes used for evaluation. If the detailed design (or the prototype) is unacceptable, designers go back to previous steps in the process and modify the design.

- During detailed design, human factors engineers:
  - Apply human factors guidelines and standards to fill in the details of the user interface, translating guidelines from human factors research into design rules for this particular system.
  - Theories of human behavior and models of the task help resolve trade-offs required by inconsistent guidelines.
  - Test specific aspects of the proposed user interface in controlled experiments with real users to determine the effect on learnability, usability, and user acceptance. A simulation of the system is tested if a working prototype is not available. User opinions are collected only after people have tried the prototype or simulated system.
  - Use iterative design methods to monitor the usability of a prototype as it is developed. Iterative testing continues until all parts of the system meet the measurable criteria for usability, learnability, and user satisfaction that were established.
  - Continue testing and development if the Wizard of Oz technique is being used.

**Implementation.** Software designers and the programming staff use the architectural design to implement the software. They test each subsystem to determine if it operates properly.

- There is little direct human factors involvement during implementation if it is performed as a separate step in the process. However, implementation frequently is integrated with the detailed design stage. In this case, testing and revisions are done to improve the design as the system is built.

- Human factors engineers may become directly involved in implementation when a user-interface management system is used to create the user interface. The tools provided by many user-interface management systems let nonprogrammers create interfaces. Because many decisions are made or changed during implementation, direct involvement of human factors engineers is beneficial.

**There is no single way to apply human factors engineering.**

**Verification and acceptance.** Once the system has been built, developers test the entire system to verify that it meets the requirements and specifications.

- At this stage, human factors engineers:
  - Evaluate the entire system (including documentation, on-line help, and support services) to verify the levels of usability, learnability, and user acceptance. The system is measured against validated models of human behavior or is evaluated with task analysis, experiments, field studies, or questionnaires.
  - Make any necessary changes before product release. Test the system again after such changes to determine if the usability, learnability, and user acceptance criteria have been met.
  - After product release, follow up with field tests to determine the system's success and to gather suggestions for future product designs.

**Deciding which techniques to use.** Just as no single design strategy will work for all products, there is no single way to apply human factors engineering. However, there are several factors that can guide the selection process.

**The application.** Certain methods have been used more often with certain types of software. A design strategy that has been found to be successful for similar applications is likely to be successful again.

Most human factors models have been created around text-editing tasks and can be used with some confidence in this domain. The Wizard of Oz technique has been used primarily for finding out how people naturally "talk" to computers. This method works well for developing programs to respond to natural language or for selecting command names that people use spontaneously.

Simulations have been successful for testing speech input and output, as well as for some experiments on the syntax and naming of commands. Prototypes, experiments, and questionnaires have been used with all sorts of systems. Field studies have focused on large systems already in use and have also been helpful in determining the organizational implications of particular systems.

**Knowledge availability.** Theories, models, and guidelines are most useful in areas where the knowledge base is well-documented and well-tested. The database of original research is the only place to look for information on newer technologies, although very recent innovations are not likely to have been tested extensively (if at all). When previous research is not available, it may be necessary to use human factors methods to collect data for designing new systems.

**Usability criteria.** If the usability criteria can be met without resorting to detailed analysis, less robust methods (such as field studies and questionnaires) are sufficient to solve large usability problems. However, if the criteria are more restrictive, it will be necessary to perform more controlled analyses to resolve smaller difficulties and to meet the usability goals.

Large usability problems are those such as what functions to implement, how tasks are ordered, which interface style is best, and which input devices to use. Small problems include such things as specific message wording and syntax, screen layout, and default choices. The sooner a big difficulty can be corrected, the better because these aspects of a design are more difficult to change as the design progresses. It is usually easier to improve minor aspects later in the development process.

For example, even a quick test will inform designers that a program is totally unusable, but it may take many
Software Engineering Opportunities

RADE*, a Micrognosis product, is an integrated workstation that is designed to put the active traders' need for fast breaking information at their fingertips. Leading banks, brokerage firms, insurance companies use Micrognosis systems as their link to the world of financial information.

We have exciting opportunities for qualified professionals in the following areas:

Software Manager
Product Engineering
You will be responsible for the supervising of project teams during all phases of software enhancements.

An ability to set and meet full project lifecycle goals and objectives in a highly technical environment is required. We expect you to have a solid knowledge of "C" and PLM operating under UNIX® in an on-line environment.

Software Manager
Research & Development
You will direct project teams involved in the development of state-of-the-art, microprocessor-driven software for the Financial Services Industry.

You must possess strong analytical, conceptual and technical management skills and a knowledge of "C," PLM and data communications under UNIX and VMS.

Group Leader
Software Systems Test
You will be responsible for the verification, validation and evaluation of product and system releases.

You must have at least 6-8 years experience with a previous supervisory background, and experience in microprocessor-based systems with knowledge of VAX and microVAX, UNIX-based with "C" experience a plus.

Senior Systems Test Engineers
Preferably, you should have 5-7 years experience in the design, implementation and test of microprocessor-based or VAX-based software systems.

A knowledge of various software systems test methodologies is required along with a knowledge of "C" and data communications under UNIX would be a plus.

Group Leader
Software Test Tools & Simulation
To design, develop and customize test tools and simulation for microprocessor-based and VAX-based real-time distribution systems.

For this senior level position, with group leader responsibilities, you must possess 7-10 years experience and a knowledge of VAX, UNIX and "C."

For all of the above positions a BSCS or BSEE or its equivalent is required or preferred.

IT'S ABOUT TIME!

Micrognosis can offer you an outstanding salary, comprehensive benefits and exceptional opportunities for professional growth.

Join a company that can provide real career momentum. It's time to send your resume, indicating position of interest, salary history and requirements in confidence to: Vito Santoro, Manager of Staffing & Development, MICROGNOSIS, Inc., Dept. IEEES 1001, 100 Saw Mill Road, Danbury, CT 06810. An equal opportunity employer.

Applying technology to trading

*UNIX is a trademark of AT&T Bell Laboratories

QUALITY TIME
Continued from p. 89

prehensibility, modifiability, defect-proneness, and the like.

The future. Improving the use of software measurements requires that the relationship between the research and industrial communities continues to grow. The research process too often suffers from little resources. Industry can enhance its future benefits from better software measurement by considering itself as a critical resource that the research community must be able to access to improve both training and the external validity of empirical research. To avoid an empirical literature dominated by studies of student programming, industry must provide researchers with access to professional programming.

Companies that have supported a serious measurements program and have understood the importance of a research component will be able to reap the benefits of greater understanding, predictability, and control of their software life-cycle process.

HUMAN FACTORS
Continued from previous page

trials in a controlled experiment to determine that one way of presenting information to users results in faster performance than another arrangement.

Experiments and task analysis usually require strict controls. Questionnaires and field studies are, by nature, less robust. Iterative testing can be more or less robust depending on the degree of experimental control exercised during the test. A major advantage of iterative design is that the large problems are revealed early in the development process and smaller refinements are discovered after those have been corrected.

Working together. Human factors issues crop up at all stages of development. Therefore, the best results are obtained when system designers and human factors professionals collaborate throughout a design project. With human factors research, models, standards, guidelines, principles, methods, techniques, and tools, designers can make better, more objective decisions about how the system should work. Looking at the full range of human factors inputs makes it clear that the fit between human factors engineering and software development certainly is a tight one.