Ten Ways to Improve Usability Engineering—Designing User Interfaces for Ease of Use

Robert E. Opaluch
Yao-Chung Tsao

This paper discusses 10 methods for improving a product's or service's ease of use. These methods can benefit not only human-factors engineers, but also project managers and system developers. The methods are general user-interface (UI) principles that engineers and designers have applied to many projects with excellent results. These methods demonstrate proven techniques for ensuring effective UI design. Although each one is important, the extent to which designers apply them to individual projects depends on many variables. These variables are also discussed. Not all UI issues can be resolved simply and easily, but the 10 methods presented in this paper will help to improve usability and end-user acceptance.

Introduction

Product engineers and designers can optimize customer satisfaction by early and consistent use of design and testing processes that emphasize rather than ignore ease of use. An easy-to-use product or service provides important end-user benefits, including reduced learning time, enhanced productivity, and fewer operating errors. Ease of use can also positively affect customers' and end-users' bottom line. Products and services having superior usability often sell at a premium price, but save money over the long term because they accelerate and simplify many complex and difficult end-user procedures.

Engineers and designers can achieve ease of use through UI designs that focus on key usability criteria, that meet specific needs, that improve on established procedures, and that maintain UI consistency. Additional information about this topic and other UI issues, principles, and methods is also available.[1–10]

A UI includes all aspects of the relationship between a product or service and an end-user. Traditional UI design considerations include computer displays, dials, controls, keyboards, work station furniture, documentation, on-line help, training, procedures, tasks, instructions, descriptions, and pictorial representations. Recent UI technological advances include voice input, voice synthesis, real-time video, and electronic pen input.

A UI does not include specific hardware and software with which an end-user does not interact. For example, central office equipment, operating system kernels, programming subroutines, transmission equipment, and protocol layers typically are not part of a UI. However, these examples could be part of a product or service that has a UI. Or, they may be the hardware or software that drives a UI. Therefore, a UI is defined as only that portion of a product or service that an end-user sees, hears, feels, interacts with, or thinks about.

The technologies to develop a UI, to write code, to test software, or to build hardware are not included in this discussion.

Ten Methods to Facilitate Usability

The 10 methods for improving ease of use of products and services are:

- Identifying/Knowing End-Users
- Assigning a UI Designer Early On
- Obtaining Team-Member Input
- Enumerating End-User Tasks
- Conducting Competitive Analyses
- Prototyping a UI
- Using UI Standards and Style Guides
- Reviewing UI Design Work Iteratively
- Conducting Usability Testing
- Project Management Support.

Panel 1. Abbreviations, Acronyms, and Terms

alpha release—an early version of a product or system having limited functionality

beta release—an interim, mostly functional, version of a product or system that allows testing in the end-user environment with actual tasks

prototype—working model or early representation of a UI without complete system functionality, allowing simple simulations or demonstrations that lack full functionality

QFD—quality function deployment; a method for analyzing the relation of customer needs to product features

rapid prototyping—repeated cycles of redesigning and evaluating a prototype by end-users

task—a set of human actions, with clear starting and ending points, that contributes to a specific functional objective and ultimately to the output goal of a system

task analysis—a description and application of a task

task scenario—a common sequence of task steps

UI—user interface; includes all aspects of the relationship between a product or service and an end-user

usability—degree to which a system is easy to use, easy to learn, and optimized from the end-user's perspective

usability engineering—methods to assure that UIs are easy to learn and optimized from the end-user's perspective

usability testing—measuring end-users while they complete tasks with the tested UI, with the intent of improving the UI to optimize end-user performance

user-centered design—designing for ease of use by including end-users on a project team

Identifying/Knowing End-Users

To successfully design a product or service, UI designers and engineers must ensure that they get to know end-users. This may involve a change in thinking for many engineers, who often assume everyone has the same knowledge as they do, or that they automatically know what end-users want.

Engineers sometimes spend little if any time determining end-user needs. Instead, they may focus their attention only on designing a product or service for a general type of end-user. Many times, the only individuals who provide design feedback are other engineers or designers, who are usually far more technically adept than most end-users.

Before beginning to design a UI, usability experts suggest that management assemble a project team that includes engineering, marketing, and human-factors professionals. Although atypical, the inclusion of end-users on the project team is strongly recommended.

The team identifies who the target end-users are. This identification process includes developing a detailed, written description of end-users' background and characteristics. Designing and testing the UI, using the written description, is then begun.

The project team should use the end-user description for selecting subjects for testing, for design reviews, and for participatory design work with member end-users. To double-check the description's accuracy, the project team reviews its contents with team-members, as well as outside end-users.

User-Centered Design. Often, the research and development community develops products or services based on the availability of a technology, then hopes that a market for the product or service will emerge. This technology-driven approach can result in a product that centers only on the technical community's perceptions and preferences, and not on those of customers.

A better approach is to identify the end-users; determine the features and usability criteria important to the end-users; and iteratively design and test for usability to ensure that the product or service meets end-user needs. Such a user-centered design approach helps to ensure maximum user interest, and results in practical, easy-to-use products and services.

Without user-centered design, developers risk producing technologically advanced products or services that no one will buy, use, or even understand. Products and services designed by a technical staff may appear to be easy to learn and use. But nontechnical end-users may find such products and services intimidating and difficult to grasp. There are numerous examples of engineer-designed software UIs that are so complicated and inconsistent that average end-users make frequent errors and never learn to use the programs effectively. Many people do not take advantage of potentially useful technology, such as television program recording using a video cassette recorder, because the UI is poorly designed, not because of an inherent lack of interest in the technology.

Usability is the degree to which a system is easy to use, easy to learn, and optimized from the end-user's perspective. Usability is measured not by feedback from designers or developers, but by feedback from end-users. No matter how elaborate a product or service may be, if an end-user cannot use it

easily and effectively, it rates poorly in usability. Engineers determine a product's ease-of-use rating through usability testing.

Usability can incorporate several components, depending on end-users' needs. Usability typically includes ease of learning, ease of use, and end-user satisfaction. Usability factors can also include:

- Efficiency and productivity enhancements
- Minimal error rates
- Clear notification of errors
- Easy recovery from errors
- Easy maintenance
- Ease of learning
- Support for incremental, spontaneous learning
- Support for end-users under stress and in non-optimal conditions
- End-user safety and health
- End-user delight and enjoyment
- End-user acceptance and preferences.

Designing for usability encompasses refining and optimizing whatever is important to end-users, including processes and tools that focus product or service features on end-users' needs and demands. Primarily, end-users drive and constrain UI design. Marketing, systems engineering, and systems development assume secondary UI design roles. Methods differ in the specific way that end-user perspective is brought into the design process. Some of these methods may be combined:

- End-user interviews before design—interviewing future end-users to understand their needs and to solicit their opinions on a proposed design
- End-user focus groups before design—meeting of 8-12 end-users, conducted by a specially trained moderator
- End-user modeling—developing a model and description of the end-user, documenting end-user assumptions, or developing an end-user simulation system
- Quality function deployment (QFD)—often helps focus team efforts on customer needs. It relates what the team learns about customer needs to the product or service being designed. QFD, when implemented at the front end of a product or service life cycle, typically includes: identifying and prioritizing customer needs; identifying features that address those needs; mapping features to needs; and identifying feature emphasis.
- End-users as design team members—one method of improving the design team's effectiveness in designing for ease of use is to include end-users on the team. End-users can participate in design and can provide a wealth of information about the product or service environment. They can also explain and demonstrate the importance and significance of proposed features.

A user-centered design approach ensures that the project team recognizes and develops only those features end-users want and need. This approach gets end-users directly involved with the design process, minimizing the chance of producing a technically complex, difficult-to-learn product or service. User-centered design eliminates the need to change a UI in future releases because the UI was designed correctly from the beginning. User-centered design creates a technically powerful product or service that most people can understand and use effectively, and it also increases chances for success in a competitive marketplace.

Easy Ways to Start. Identification is one of the most important aspects of getting to know the principal end-users. Quality function deployment (QFD) is a method for analyzing the relation of customer needs to product features.[11] QFD can be used for focusing on end-user needs. In addition, project and marketing managers can identify end-users by developing a written description, which includes a page or two about background, experience, education, computer skills, motivation level, normal workday activities, and systems familiarity. Managers distribute the description to end-users and product marketing to verify the information. Afterward, the write-up goes to the project team.

Design-team members consider the background and skill-set summary in the description to tailor a UI to end-users' needs. Then, a list of usability criteria is developed. These criteria can give specific minimums and expected levels for parameters such as: how long it takes 80% of end-users to learn an important task; maximum number of errors that 90% of typical end-users would make doing some specific task; maximum time it would take 75% of end-users to perform an important sub-task. The design team tests for these usability criteria to determine whether or not a UI design meets end-user needs.

Assigning a UI Designer Early On

Too often, project managers assign usability engineers or human-factors professionals to work on a project after important decisions are made and milestones are passed. New team members then can only evaluate work already done and point out any UI and

engineering-process defects. It is too late for them to fix the defects without major cost overruns and schedule delays. So, either the usability defects remain and the project proceeds with a design having reduced end-user satisfaction, or the project team takes time to rework the UI and fix the defects at substantial additional cost.

From the beginning, project managers should include usability engineers as part of a project team to minimize such difficulties and maximize end-user satisfaction. Any project that has developers working on UI design should receive formal support from usability engineering. For that support to be effective, it should start at the project's inception and continue through to the end, with particular emphasis during the early design stages. Usability engineers should have the opportunity to completely redesign the UI if they or the end-users believe redesign is needed.

Timing of UI Design and Testing. UI design and testing methods are most effective when discussed and decided at the product-planning stage, before UI development begins. Usability engineers should have the opportunity to completely redesign the UI if they or end-users believe rework is needed. Specifically, it is very important to have consensus on UI design before beginning detailed development. Otherwise, either end-user satisfaction will suffer, or the cost and time for UI redevelopment will skyrocket.

UI Design and Usability Testing Plan. Once involved, designers prepare a UI design and usability testing plan for the project team's review and concurrence. Designers use the plan to execute and track usability engineering procedures.

Likely Penalties for Late Involvement. Two significant difficulties will have to be faced if a UI designer, or any other critical team member, is assigned to a project late. Either the project team will have to proceed with a non-optimal design, or costly rework of the UI will be necessary.

Easy Ways to Start. Management should hire a human-factors professional from a support organization to join the project team when it is formed. This individual should be empowered to prototype the initial UI design, as well as to locate about 10 subjects who match the end-user description. The subjects should be tested with the prototype, or at least interviewed and stepped through typical task scenarios with the prototype.

Obtaining Team-Member Input

Gathering information is critically important. It is often time consuming—but can also be time saving. Project managers must continually strive to improve information flow and accessibility between information suppliers and those who need it, especially UI designers.

Reviewing Supplier Input. If possible—and before the design stage—project management needs to get the customer, supplier, and project team to verify and agree on supplier input and design assumptions. Starting design without critical information or without agreement on important assumptions might appear to save time and accelerate development. However, premature design work typically results in faulty decision making and unnecessary rework. Rework increases when a design is based on erroneous information or invalid assumptions. Rework also delays development and increases costs.

Before designing a UI, project teams need to obtain specific information from several key sources:

- Marketing—This organization furnishes some insights about end-user characteristics, including background, experience, motivational and cultural elements, task familiarity, and end-user working environment. It provides information on end-user and customer needs, requests, and priorities. Marketing also supplies strategic direction, competitor information, and informal observations about previous product releases.
- Systems Engineering—This group typically provides a list of planned features and functional details. It may supply a project plan, priorities list, migration plan, and an evolution plan which specifies, for example, future end-users and upgrades to future releases. Systems engineering also provides a draft of technical requirements and architecture constraints such as network configuration, remote systems, data base size, and number of users.
- Systems Development—To ensure design consistency, development engineers provide information or code from similar UIs. They also furnish hardware constraints, as well as software constraints, such as UI platform capabilities.
- End-users and Operations Experts—Information about tasks to be computerized—as well as their priority—are provided by these groups. They can furnish task analyses and data about job-change trends.

They can reinforce marketing data with information about competitors' products and end-user performance on previous releases. They can also provide usability goals and objectives.

Sometimes, information that should be provided by other suppliers may not be available. If some material cannot be obtained, a written summary about the missing information should be developed and distributed by the UI designers to ensure common understanding and agreement among all project team members. Otherwise, too much rework of UI prototypes could take place based on erroneous assumptions. Marketing, systems engineering, systems development, UI designers, and other team members must review the written summary and reach consensus.

Likely Penalties for No Design Input. Designing a UI without needed input will lead to much rework. Other individuals with relevant design information will realize that the project team is not aware of important constraints and will insist that the UI be changed to reflect those realities. Although a designer may feel too busy to obtain some needed information and is making progress without important input, the project may actually be jeopardized. In this case, the designer should be ready to change direction and backtrack when other key individuals review the UI design work completed up to this point.

Easy Ways to Start. Rather than beginning UI designs on computer displays or hardware before having needed input, designers should first obtain the information discussed earlier in this section. If the information cannot be obtained, designers should develop a written summary about their assumptions on the missing information and distribute the write-up to all other project team members for feedback.

Enumerating End-User Tasks

The task analysis approach developed in the early 1950s is widely used by engineers today to analyze human performance requirements in a work process. Task analysis can yield detailed information about the type and sequence of information that a system must provide an end-user, and the type of information or choices that an end-user must be able to enter into a system. Engineers can use task-analysis information from existing systems or environments to construct more realistic scenarios for testing UI designs. Human-factors engineers highly recommend learning how to complete task analysis and how to build test scenarios.

A comprehensive review of the history, definitions, and methodologies of task analysis is available.[12] A description of task analysis, in the context of computer-to-human interface design, is also available.[13]

Defining a Task. A task is a set of human actions that contributes to a specific functional objective and ultimately to the output goal of a system.[12] The size and scope of tasks may be a rather subjective decision. However, engineers generally recommend that size and scope be comparable among different tasks within a system. A task should have clear starting and ending points. Engineers can define and observe input to and output from the task. Input and output should have a significant impact on attainment of overall objectives. An example of a task is reading an electronic mail message and responding to it. This task is a clearly defined action, it can be observed, and its execution enables the system to accept the data entered.

Phases of Task Analysis. There are four phases of task analysis for a typical system development project: *phase 1*, system description and analysis; *phase 2*, task description list; *phase 3*, descriptive data collection; *phase 4*, applications analysis.[12]

The principal purpose of phase 1 is to identify a system's tasks and functional flow. This phase helps engineers understand how the system functions to achieve specific goals.

In phase 2, tasks identified at phase 1 are sequenced for the entire system. Engineers show tasks to end-users to validate each one. Together with end-users, they also construct scenarios to ensure that a UI will support specific tasks in the best possible way. Engineers use scenarios as a design aid, as a way to check their work, and as a method to suggest design improvements. Scenarios describe each task in detail concerning, for example, its criticality, activation frequency, error potential, prerequisite end-user knowledge and training, and its association with the relevant tasks.

In phases 3 and 4, engineers use task-analysis information to test a UI prototype by a walk-through using task scenarios. They can step through task scenarios, doing the task with the aid of a specific UI to analyze the design.

Following a systematic task-analysis procedure, as discussed, will provide important benefits for a system being designed: reasonable task allocation among end-users and system; elimination of irrelevant tasks or steps; and logical sequencing of tasks that do not exceed end-users' capabilities.

Likely Penalties for Skipping Task Analysis. Without systematic attention, it is possible to overlook important tasks, meaning that some key capabilities could be left out. In addition, project teams could fail to make common or critical tasks easy to do, or perhaps even make them nearly impossible to do. Tasks may be too cumbersome and convoluted for efficient completion, they may be error prone, or they may be too difficult to learn and remember. Tasks may overwhelm end-users with so many options and possibilities that the capabilities that end-users really need are difficult to access. Ultimately, without task analysis, design teams may waste time and money developing capabilities that end-users do not actually need.

Easy Ways to Start. A design team first develops a list of all the tasks end-users will perform. Tasks are broken down into sub-tasks until the team can identify all the information, choices, and actions a system must provide to an end-user. The list is reviewed by marketing, as well as design team members, including end-users, to ensure it is complete. Tasks not needed are deleted. If some tasks lack important information, the team creates a proposed task and adds it to the list. The design team then uses the task list to ensure the UI can provide the necessary functionality.

The design team also develops a second list of task scenarios, containing only a few vital tasks that account for a large portion of the task domain. A rule of thumb states that a task list containing 20 percent of tasks is done 80 percent of the time. Task scenarios are the critical, core set of tasks arranged in the sequence end-users complete them. This task set does not include each and every required task, but it does provide vital sequence information for the most important task set.

The design team confers with end-users, marketing, and operations experts to verify the validity of task scenarios. Such cross checks help the team determine how well a UI design supports the task sequence in each scenario. The team then modifies the UI design, if required, so that end-users can complete each task scenario quickly, efficiently, without errors, and with as little training as possible. If all end-user tasks are not straightforward and easy to complete, the UI must be redesigned to ensure that they are.

Conducting Competitive Analyses

Before designing a UI, a design team investigates and evaluates similar competitor offerings. Competitive analysis allows designers to identify UI difficulties in existing designs in order to solve or avoid such problems in their own future offerings. Designers also identify weaknesses in competitors' UIs that may allow new products to "leapfrog" the competition.

QFD offers a framework that a project team can use to relate competitive information to both customer needs and to product characteristics and features. The team can conduct QFD to understand how competitors are meeting customer needs. By using QFD, a project team may identify their own product's weaknesses, or the team may discover opportunities to improve where the competition is weak.

Likely Penalties for Ignoring Competition. Without conducting competitive analysis, designers could overlook identifying usability improvements the industry is implementing. Usability objectives that must be met to remain competitive may not be understood. Market share could be lost due to not recognizing product weaknesses, important market segments, and competitor vulnerabilities. Much money and time could be spent developing UI solutions that are commonplace or even outdated. And without competitive analysis, there is a risk of continuing in a line of business that should be abandoned.

Easy Ways to Start. In conducting competitive analysis, it is usually best to begin by having the UI designer or project team learn to use all the top competitors' systems. Learning about the most innovative or best systems in their class could greatly help in designing and developing a new, competing product. It is also highly desirable to hire a QFD professional to help the project team conduct quality-function deployment cost effectively.

Prototyping a UI

The term *prototyping* can have several different meanings, and engineers can prototype a UI in a number of different ways. This section focuses on prototyping tools and methods that do not require system development. For this discussion, a UI prototype is assumed to be one without full system functionality.

Prototypes can be working models or early representations of a system. For example, when prototyping

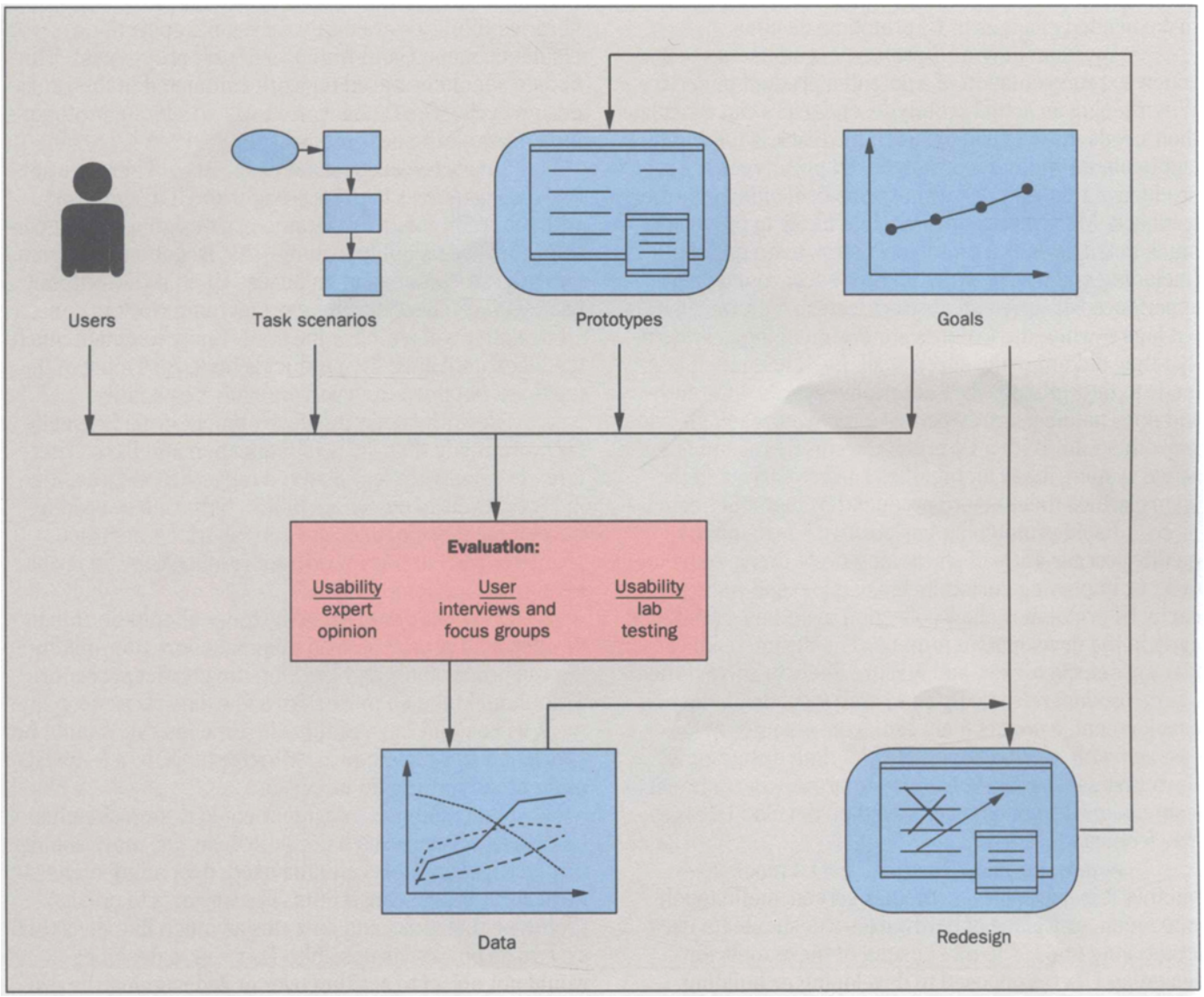

Figure 1. This illustration is an overview of rapid-prototyping methodology. It shows how end-users, task scenarios, prototypes, and project-team usability goals are used to evaluate and redesign a prototype. A decision to retain or redesign a UI prototype is made depending on end-users' needs and final evaluation results.

UI screens for software systems, screen mock-up prints or transparencies of key UI displays can serve as the earliest system prototypes. Pencil and paper sketches can also serve as preliminary prototypes. Prototypes can represent many different types of products and services. Some examples include computer screens, mock-ups of newly designed telephone sets, and audio touch-tone menus for voice services.

Rapid Prototyping. Repeated cycles of redesigning and evaluating a prototype by end-users is known as *rapid prototyping*. Rapid prototyping is iterative, with each successive iteration incorporating more end-users' needs and suggestions (see Figure 1). To incorporate end-user feedback, a design team needs tools that allow easy prototype changes. Otherwise, the cost and cycle time of iterative prototyping become prohibitive, and designers are less likely to incorporate feedback and

make needed changes in UI prototype designs.

By examining a UI prototype, end-users see a concrete representation of a potential product or service. When seeing an actual prototype, end-users can articulate their needs more explicitly. Such feedback is more difficult to obtain without a prototype. UI prototypes are more productive than abstract discussions in eliciting end-user feedback. Mere discussions are less likely to provide as much detail as does a prototype. Often, team members (including end-users) with different backgrounds and experience will agree on abstract feature lists, only to find out later—when the features are designed into a system—that they did not really agree at all. Each team member tends to interpret abstract descriptive words differently and does not necessarily perceive the features in the same way others do. With a UI prototype closely resembling the actual system, many ambiguities surface early, and the team clarifies them before product development begins.

Rapid prototyping has positive effects on the quality-process goals of shortening cycle times, reducing defects, improving customer satisfaction, and reducing costs. UI prototypes allow collection of end-user feedback early in the development process. The design team can identify defects earlier, and is more likely to correct them before product release. By collecting feedback before UI development, a project team can make changes at lower cost and with less likelihood of extending design time. Customer satisfaction is likely to improve when a project team makes design changes based on detailed UI prototype feedback.

Prototyping Tools. In creating a UI mock-up—whether it is a graphical computer screen, audio touch-tone menu, or a piece of hardware—a design team uses prototyping tools. The most useful of these tools for designing UIs (as opposed to developing or building UIs) allows high-resolution, quick prototyping. A UI designer can assemble and update a prototype with minimal time and effort. The UI appears very similar to the final product. Sometimes, such a prototype appears so functionally complete that end-users do not realize it is not the finished product. Depending on the prototyping tools a design team uses, software they have created may be reused for later system development, becoming part of the system being developed.

Updating Tools and Platforms. A UI designer should invest some time near the end of a usability engineering effort to update platforms, tools, and prototype libraries (libraries are easily accessible collections of elements, menus, and windows of past prototypes). This update should be based on work completed in the latest design cycle. The time invested will accelerate prototyping efforts on the next project.

Likely Penalties for Not Prototyping. There are several disadvantages to proceeding with UI design work without prototyping. For example, a designer cannot conduct effective usability testing—that is, gather data from potential customers and end-users about a UI—without prototyping. Therefore, if no prototyping work is done, a UI designer will not have the opportunity to obtain much feedback until after the product is built. And most of the feedback obtained then will probably be negative.

Rework late in the development cycle is usually far more costly than early identification and fixes. Therefore, designers are less likely to repair defects, resulting in lower levels of customer satisfaction. Unless product-development cycle times are very short, UI interface problems resulting from skipped prototyping will probably linger for a long time.

Easy Ways to Start. A designer should be able to develop a UI prototype with minimal effort after obtaining the proper tools and learning the correct procedure. This should take no more than a few days. If prototyping work will extend beyond this, the time interval should be shortened by obtaining easier-to-use tools, or a less elaborate prototype should be created.

For example, a designer could draw a sketch of UI display screens as a basic prototype. Or, more sophisticated software tools could be used, depending on hardware and UI style constraints. The object is to create a prototype that looks and operates as much like the actual system as quickly as possible. This way, a designer would not object to starting over or redesigning the prototype if end-users or other team members suggest radical changes from the initial concept.

After prototyping is completed, a designer must realize that the tools and techniques will need continuous improvement. This helps to ensure that the same mistakes will not be made, and that the prototyping can be completed faster and more smoothly on succeeding projects.

Using UI Standards and Style Guides

The easiest way to design and evaluate a UI prototype—with a minimum investment of time and money—is to emulate successful UI designs. Groups of

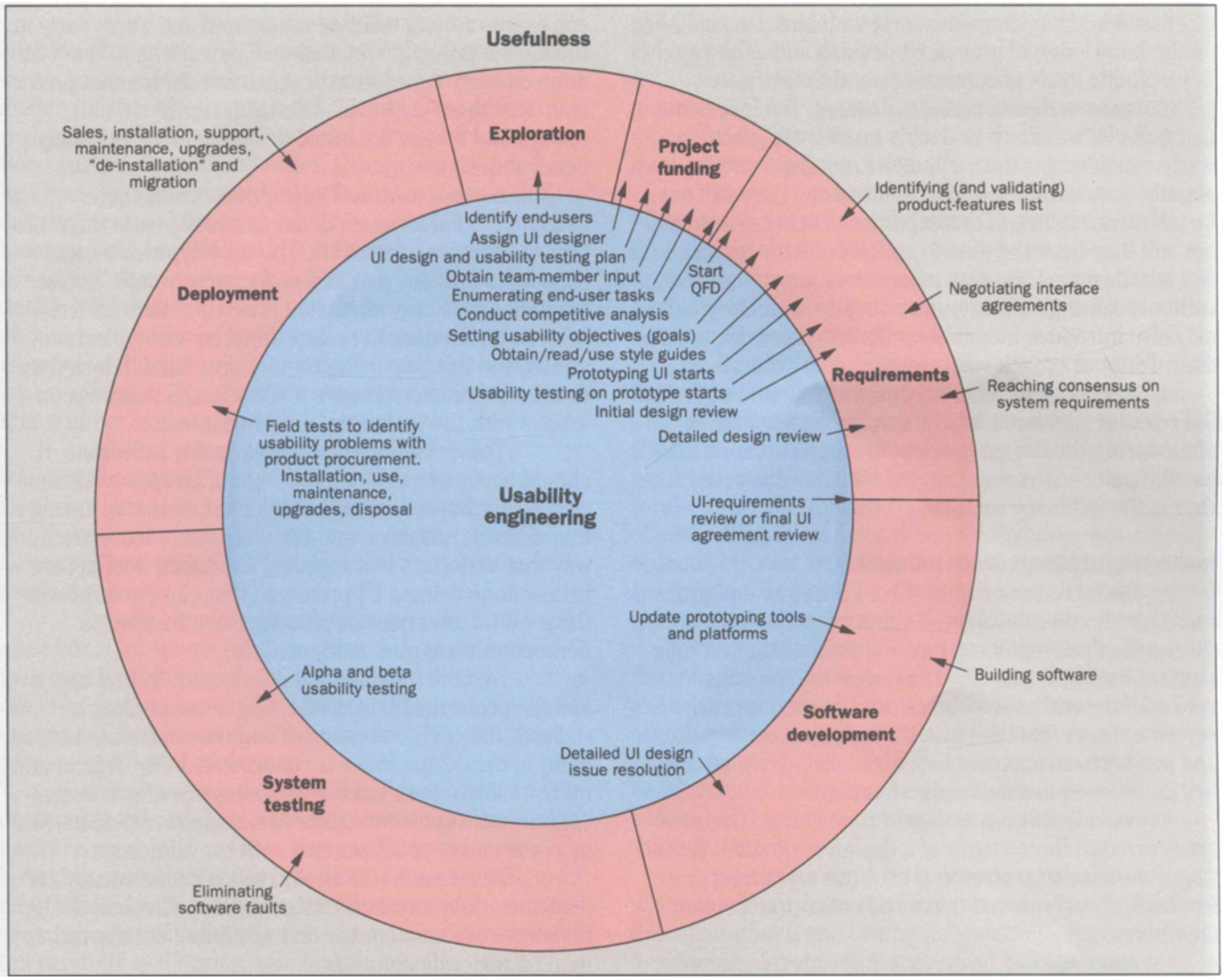

Figure 2. The key techniques that usability engineers can implement to ensure efficient UI design are shown in this illustration of a UI development cycle. The milestones of UI design—exploration, project funding, requirements, software development, system testing, and deployment—appear clockwise in the outer circle. Significant factors affecting usefulness comprise the entire circle, and important usability-engineering criteria complete the inner circle.

UI professionals have identified and solved many design problems, summarizing and publishing their findings in UI style guides. Project teams just beginning UI prototype design can refer to the guides to evaluate their own work and to avoid rethinking issues that have already been successfully addressed. Consulting the style guides helps avoid hidden problems in prototypes, which often escape notice until after a UI is produced.

Nothing hinders ease of use more than inconsistency. After becoming accustomed to doing things one way, end-users will be frustrated with a UI that works in unfamiliar ways. By following the UI design principles

discussed in the style guides, project teams can capitalize on the knowledge of proven UI designs and achieve the key usability goals of consistency and ease of use.

Penalties for Not Using the Guides. Designers not using the UI standards and style guides will probably waste considerable time and effort redoing work that has already been completed and reported on. They will not be taking advantage of the expertise of other designers, nor will they have the time to consider all the implications of a new design. They may even create new UI problems without realizing it. Designers not consulting the guides may also introduce inconsistencies into a new UI, even if their design approach was superior to any other.

Easy Ways to Start. UI designers should obtain all the relevant standards, read them, and consult them often during the design process. Designers should also use the guides as review criteria, which is discussed further in the following section.

Reviewing UI Design Work Iteratively

To minimize rework, the UI designer and project manager should establish a number of quality milestones during the design process (see Figure 2). The first milestone should be a new-designs review or conceptual review of the earliest UI ideas and concepts. An early review ensures that the basic UI design meets end-user and project team approval before detailed design work has progressed to detailed development.

Starting a Design Review. Project teams can progress through three stages of a design review. All three stages emphasize prototyping to obtain and incorporate feedback, from both end-users and other project-team members.

Initial Review. In the first stage, designers produce an initial or conceptual prototype. The first UI quality milestone should occur when designers show end-users the key tasks, a rough outline of the general UI strategy, or general UI characteristics. Designers then get general agreement on the initial prototype among all project stakeholders.

A UI designer, with unique perspectives on end-user needs, may be able to suggest more targeted design elements to replace the abstract, conceptual ideas of project team members. This is particularly likely if the designer practices rapid prototyping with minimal user-model and task-model input. It is important to minimize rework and defects in the final design by achieving consensus among team members and end-users early in the design process. Detailed design work should not be done on early conceptual designs that fail to meet project team members' approval. The UI designer should rework and review the initial design before beginning detailed design work.

Detailed Review. During the second stage, designers build as much detail as possible into the prototype to model the actual UI. The prototype looks, and appears to operate, just as a real system would, but its functionality is only simulated. The UI details are present, so end-users can review them, try out typical scenarios, and imagine using the system. The UI designer then incorporates end-user feedback, and reaches consensus with the end-users and project team.

This review is the second quality milestone. It should occur when detailed design of an approved concept is nearing completion. A detailed review will ensure that controls, displays, and tasks are designed in such a way that end-users will approve. A detailed design review is best done using a UI prototype that closely simulates the product or service. A prototype can be used to demonstrate a typical task scenario.

Despite the time and effort spent by end-users and the project team in conducting detailed design reviews, the early collection of end-user and stakeholder input in the design process reduces usability defects and more costly rework later. By making this effort, designers minimize cycle-time intervals, costs, and defects, and increase end-user satisfaction with the UI design.

Team members should reach consensus on UI design work before final design review. The final-design reviewers can concentrate on last-minute details, rather than on major design issues.

As in the initial review, the detailed design-review process should include all stakeholders: paying customers who make purchasing decisions, graphic artists, and other UI design specialists.

Final Agreement. In the third stage, designers develop and write detailed specifications from the prototype designed earlier, incorporating information from reviews. Designers provide these specifications to their developers as a blueprint for building the actual system. Another way to document the agreement can also be used. The prototype itself, on which everyone has signed off, can serve as the actual agreement. This is possible if the prototype tool can generate useful software code.

A third quality milestone is the final review. It should occur when a UI design is completed and requirements or specifications are written, or another final design deliverable is completed. The purpose of final review is to reach consensus among project team members, customers, and end-users. Otherwise, rework may be necessary, or serious design flaws may remain during development and deployment. End-users and other team members can provide final UI details and approval far in advance of development completion. Modification of a UI design during development should be minimized by obtaining end-user input throughout the design phase. However, a small number of unanticipated development issues will arise. These issues should have a relatively minor effect on the approved UI design.

Listening and Acting on Reviews. Listening to sources of constructive criticism is key to improving the UI design. Implementing input from end-users and other project-team members will improve usability and will help avoid end-user dissatisfaction and complaints after product release.

Incorporating feedback from reviewers' critique of an initial prototype can be a great time saver in the long run. By thinking about all aspects of the UI, reviewers help designers avoid problems and rework. Designers can use reviewers' feedback to improve usability on present and future projects.

Likely Penalties for Skipping Reviews. Unless a UI designer is the only project-team member, and the only user of a product or service, there will be others who have differences of opinion about a UI design. Designers need to build consensus for effective teamwork, and consensus begins with listening. It is most important to listen carefully to differing opinions about an initial UI design, otherwise:

- There will be a continuous stream of complaints about the UI.
- A designer will not learn how to improve procedures on the next project.
- Complaints might be serious enough to cause customers to seek out competitive offerings.
- A designer may find that no one wants to cooperate to build the system or pay for it.

Listening to others review an initial prototype is a must. Reviewers are doing much thinking for a designer, helping to avoid costly problems and rework later on. Designers should be grateful to receive such good support, and they should use reviewer feedback to improve the UI design.

Easy Ways to Start. Prototyping work should be planned in three stages. Each stage emphasizes prototyping for the purpose of *obtaining and incorporating feedback* from end-users and other team members.

During the first stage, a rough initial or conceptual prototype is produced. General agreement on the prototype is obtained before detailed design work begins. It does not make sense to work on project details when the initial prototype may not be acceptable to end-users or other key stakeholders.

After reaching consensus on the general UI concept, the design team injects as much detail as possible into the prototype to closely model the actual system. This should be completed within a few day's time. The prototype should appear and operate much as the actual system would. However, the prototype's functionality is only simulated; it is nothing more than a UI that does not actually operate, but that appears to be genuine. All of the details are there for end-users to inspect. They can try out typical user scenarios, and imagine using the system.

After reaching consensus on the details, the design team documents their agreement by developing explicit written requirements for developers to build the actual system. Designers should solicit final-review feedback from the project team to ensure consensus has been reached. Designers hope that developers will be able to build the system with little or no deviation from this agreement.

Designers must be sure to focus on listening when showing the prototype to others, rather than only demonstrating it and asking for questions. A good UI designer is also a good listener who builds consensus, not a good debater whose designs win out over the objections of others.

It is typically not possible to hold good reviews without following structured reviews. This implies that designers should enlist help from a trained moderator to conduct the review meetings, record all suggestions and comments noted in the review, and focus on understanding the comments rather than responding to them. Feedback should be incorporated as much as possible.

Conducting Usability Testing

Usability testing is the most effective means of assuring that a UI is designed to meet end-users' needs.

Table I. UI Evaluation Alternatives

Evaluation Criteria	Impact on UI Design	Cost to Change UI Design	Similarity to Actual Product (External Validity)
Review Early UI Sketches	High	Very Low	Limited
Review Early UI Prototype	High	Low	Fair
Beta Test in Laboratory	Low	Very High	Very Good
Beta Test in Field	Low	Very High	Excellent

Although usability can be enhanced by a number of other types of analyses, usability can be measured only by testing. Testing can be performed in a number of ways, which differ in reliability, cost, and time intervals (see Table I).

Usability-testing methods also differ in their points of occurrence. Some testing occurs too late to affect the design of a particular release. Only later releases could be modified, based on the data obtained through usability testing. Alpha and beta pre-releases can provide feedback before a final release, thereby allowing some changes. An *alpha release* is an early version of a product or service and has limited functionality. A *beta release* may lack the full functionality of a final release, but has sufficient functionality for it to be used by customers in their own environment.

UI designers can construct prototypes to allow usability testing much earlier in the design process than would otherwise be possible. Feedback from end-users can be collected early in the design process by using UI prototypes, to allow end-users to influence redesign. UI prototypes elicit more feedback from end-users early in the design phase. This encourages incorporating end-user feedback into the design, resulting in increased usability. This process also helps to minimize defects, reduce costs, and improve cycle-time intervals. Avoiding later redesign can offset the cost of prototyping and usability testing. Even if prototypes are not constructed, typical end-users could participate in reviews early in the design process.

Usability-testing methods differ in cost, and in the methods of fixing defects identified through testing. The earlier that usability testing feedback is collected, the less costly it is to modify the product or service based on the feedback.

Costs include those of redesign and development, increases in cycle times, and lost revenues when the releases are delayed due to redesign and other schedule delays. Feedback collected during design reviews and from early system prototypes leads to changes that are less costly to implement than changes based on later alpha or beta feedback. Likewise, changes based on alpha and beta feedback will be less costly to implement than waiting to complete last-minute testing of a final release before making changes.

It takes under a week for an experienced UI designer with the right tools to construct a UI prototype, to include end-users in design reviews, or to conduct informal prototype reviews. If it takes much longer than this, rapid prototyping is not being completed, and feedback is not being incorporated into a design often enough. Designers should be willing to start over whenever feedback suggests trying a different approach. In general, formal usability testing in a laboratory costs only a month's worth of time, as compared with development costs or the cost of fixing defects in a final release, which can easily amount to the cost of multiple head count.

When changes to a product or service are costly to make, they are often deferred or never implemented. A design team might identify a usability defect, but the defect may be left alone because it was discovered too late and is now too expensive to fix. If the defect was identified earlier in the design cycle, the cost to fix it would have been much lower and it most likely would have been removed. Therefore, early usability testing is more effective than later testing because defects are more likely to be eliminated through redesign.

Usability testing methods differ in the reliability and scope of the feedback they provide. Participation in early design reviews sometimes gives end-users so little information about a final release that their participation cannot be called usability testing. Later testing gives end-users something much closer to an actual product or service. Not only is later testing more reliable; it also allows earlier investigation of some questions. A beta or final release can be used in an end-user environment to

perform actual tasks. Early, nonfunctional UI prototypes typically cannot be used because they are only simulations, which lack the functionality of a final release.

Alternate types of usability testing are not mutually exclusive. As stated previously, early usability testing has many advantages and is the most cost effective. Later usability testing is more realistic, permitting a wider variety of questions. Different usability testing methods can be used to complement one another, rather than relying on a single method to deliver all the feedback.

Alpha and Beta Release Testing. UI prototypes are less costly to construct—and can be developed much earlier in the design process—when compared with alpha and beta releases. Therefore, prototype usability testing can be done earlier and at less cost than testing alpha and beta releases. However, alpha and beta releases have more than just UI functionality. They have some system functionality that, in several ways, more closely approximates the finished product.

Alpha releases facilitate intermediate testing, which includes some (but not all) end-user tasks. Beta releases allow testing in the end-user environment with actual tasks. Early prototype testing only allows simple simulations or demonstrations that lack full realism. For this reason, testing with an alpha version—and especially with a beta release—is very valuable in providing user feedback before final release. Unfortunately, only minor product or service modifications are possible during beta testing. Many defects can only be fixed in the next release, if there is one. However, documentation and training can be modified to help resolve some of the issues raised during beta testing that could not be fixed by the project team in the final release.

Early Usability Testing. As previously mentioned, fixing a UI problem early in the design process is substantially less expensive than doing so late in the product development cycle. Human-factors professionals universally recommend early usability testing with prototypes—and prompt attention to suggested design improvements—as the best investments a UI design team can make to ensure ease of use.

Starting Usability Testing. The simplest form of usability testing is to interview end-users as they are reviewing a prototype. The process involves designers listening to end-user comments and suggestions and writing them all down, even if some comments seem irrelevant. Designers can grade themselves as usability engineers by what percentage of comments they incorporate into a design, with over 90% being excellent, over 80% good, over 70% fair, and less than 70% needing improvement.

Designers may find it helpful to take a camcorder with them when interviewing end-users, recording end-users' comments and reactions about a prototype. Designers can videotape end-users manipulating the prototype, and ask them to think out loud while trying to perform task scenarios. Designers can note when errors are made, indicating the need to redesign the UI. Videotape of the interview provides a visual reminder of any difficulties that the project team needs to address.

Designers must constantly remind themselves of their primary mission during an interview: to concentrate on collecting end-user feedback, ideas, frustrations, comments, and points of view. End-users are a designer's customers. The more attentive designers are to end-users' needs, the more successful designers will be in developing an effective UI.

Project Management Support

Project managers must provide the appropriate support to ensure effective usability engineering. They must verify that team members are following UI standards, and assign UI design responsibilities to individuals who are appropriately qualified and not occupied with other tasks.

Project managers conduct audits to obtain outside opinions about UI design and testing activities. They may commission an outside organization, experienced in usability engineering audits, to review design and testing processes, as well as the UI design itself. Project managers typically ask usability experts to review compliance with UI standards, design guidelines, checklists, good design practices, and internal/external UI-standards consistency.

Project managers must ensure that usability engineers work closely with software developers in meeting end-users' needs. Managers need to provide—and assure the availability of—the tools their project teams need to support effective UI design and testing. These tools include video equipment, laboratory space, customizable questionnaires, sample usability testing plans, detailed end-user scenarios for usability-testing scripts, and appropriate support documentation.

Benefits of Project-Management Support. Project-management support of usability engineering affords several important benefits. It re-focuses design teams away from debating usability issues among themselves

and toward obtaining end-user feedback about UI needs and desires.

Project-management support helps to avoid scheduling and budget difficulties by encouraging design teams to complete UI rework early in the development process. It ensures that design teams focus their attention on UI design and usability engineering, and that they identify product features they can eliminate without reducing end-user satisfaction.

Project-management support encourages design teams to follow established UI standards and good design practices, and also reinforces the need to address end-user complaints about the suitability of product performance.

Support from project management can reduce the extent of pre-sale and post-sale product support and the necessity for customer help lines. Through effective design-team management, project managers can also minimize product returns and order cancellations due to end-user frustration with a UI.

Conclusion

The 10 ways to improve usability engineering discussed in this paper are guidelines; not so specific that they can be applied in a step-by-step fashion. Designers must think creatively about how to apply them.

UI design professionals agree on the fundamental importance of these 10 techniques, but they may disagree on the extent and exact method of applying them. The nature of a project normally determines the application method. For example, task analysis on a well-defined, mature application, such as referring trouble tickets to specific work centers, can be far more defined and complete than identifying task scenarios for a future-services software platform.

In both cases, UI professionals would agree that using specific task scenarios to evaluate UI designs is important. But designers could complete the well-defined task analysis for the trouble-ticket application far more easily and comprehensively than a task analysis for the future-services platform. How well, and to what extent, a designer can apply each of the 10 techniques often depends on experience, creativity, and effort level.

For any UI designer, engineer, or project team, investing in a proven UI design and testing process will pay dividends in shorter cycle times, fewer user-defined defects, lower development costs and—most important—significantly improved customer satisfaction.

References

1. R. M. Baecker and W. A. Buxton (eds.), *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Morgan Kaufmann Publishing Co., San Mateo, California, 1987.
2. S. Card, T. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*, Erlbaum Publishing Co., Hillsdale, New Jersey, 1983.
3. P. Heckel, *The Elements of Friendly Software Design*, Sybex Publishing, San Francisco, California, 1991.
4. M. Helander (ed.), *Handbook of Human-Computer Interaction*, North Holland Publishing, New York, 1988.
5. B. Laurel (ed.), *The Art of Human-Computer Interface Design*, Addison-Wesley Publishing, New York, 1990.
6. J. Nielsen, *Usability Engineering*, Academic Press, Boston, Massachusetts, 1993.
7. D. A. Norman, *The Psychology of Everyday Things*, Basic Books Publishing, New York, 1988.
8. G. Salvendy (ed.), *Handbook of Human Factors*, John Wiley & Sons, New York, 1987.
9. B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interactions*, Addison-Wesley Publishing, Boston, Massachusetts, 1987.
10. B. Tognazzini, *TOG on Interface*, Addison-Wesley Publishing, New York, 1992.
11. Y. Akao, *Quality Function Deployment—Integrating Customer Requirements Into Product Design*, Productivity Press, Cambridge, Massachusetts, 1990.
12. C. G. Drury et al.,"Task Analysis," in G. Salvendy (ed.), *Handbook of Human Factors*, John Wiley & Sons, New York, 1987.
13. M. D. Phillips et al., "A Task Analytic Approach to Dialog Design," in M. Helander (ed.), *Handbook of Human-Computer Interaction*, North Holland Publishing, New York, 1988.

(Manuscript approved July 1993)

Robert E. Opaluch *is a member of the technical staff in the UI Architecture organization at AT&T Bell Laboratories in Holmdel, New Jersey. He is involved in usability engineering (consulting), defining multimedia UI standards. Mr. Opaluch has a B.A. in philosophy and a B.S. in applied mathematics from Brown University in Providence, Rhode Island, and a Ph.D. in psychology from U.C.L.A. in Los Angeles, California. He joined AT&T in 1988.*

Yao-Chung Tsao *is a member of the technical staff in the New Business Concepts Directorate at AT&T Bell Laboratories in Holmdel, New Jersey. He researches and manages the development of inbound service concepts. Mr. Tsao has a B.S. and M.S. in psychology from National Taiwan University in Taipei. He also holds an M.S. in industrial engineering and a Ph.D. in psychology from the State University of New York at Buffalo. He joined AT&T in 1984.*