USER CENTERED DESIGN

What Is the Human-Machine Interface?

Nowadays humans interact more with computer-based technology than with hammers and drills. Unlike tools, the visible shape and controls of a computer do not communicate its purpose. The task of an HMI is to make the function of a technology self-evident. Much like a well-designed hammer fits the user's hand and makes a physical task easy, a well-designed HMI must fit the user's mental map of the task he or she wishes to carry out.

Figure 1: What Is the Human-Machine Interface?

In nearly every technological solution, the effectiveness of the HMI can predict the acceptance of the entire solution by the intended users. Often, as far as consumers are concerned, the HMI is the product — the user's experience with the interface is far more important than the architecture of the internal workings.

HMI effectiveness is measured by a number of components, such as learnability and productivity. These components are sometimes brought together under the title of "usability," also known as "quality of use."

ISO Definition of Quality of Use

The ISO 9241 standard defines three components of "quality of use" applicable to the design of HMIs:

- Effectiveness
  Does the product do what the users require? Does it "do the right thing?"

- Efficiency
  Can the users learn the HMI quickly? Can they carry out their tasks with minimum expended effort, including a minimum of errors? Does it improve the productivity/effort ratio? Does it "do things right?"
- Satisfaction
  Do users express satisfaction with the product? Does the new product reduce stress? Do the end users now have a more satisfying job?

Usability Engineering

These components are not intrinsic qualities of a product. The designer cannot take a tape measure to an HMI and measure its effectiveness, efficiency, or satisfaction scores. Effectiveness depends on the users' intentions, goals, or tasks; efficiency depends on users' understanding of the product and on their previous experience; and satisfaction can only be expressed by the users.

As a result, the unifying principle of design techniques that deliver usable products is that each recognizes the need to keep users at the center of the process. The overall design process that brings these techniques together is known as "usability engineering."

Usability: Why Is It Important?

In an increasingly competitive marketplace, those products and services that do not meet customer needs fail. Ease of use is a real user need and most product reviews, whether in consumer magazines or professional journals, usually compare products based on their usability. Research also shows that about 50% of the code in new software applications is devoted to the user interface, making it a significant cost component. Finally, applications have become increasingly complex — and delivering this increased complexity while maintaining ease of use is a challenging endeavor.

The importance of usability will vary from product to product. For safety-critical applications such as nuclear power station management or air traffic control, usability is crucial. Where a high value is placed on productivity, or perhaps a high cost is associated with human error (such as a financial dealing room or telecommunications network management center), usability is key as well.

In marketing terms, where significant additional revenue will be generated from increased usage (for example, telephony services), usability is also a priority.

Usability Engineering Principles

Principle #1: Know Your Users

Three simple design principles which underlie the development of products and services ("Know Your Users," "Involve Users Early And Continuously," and "Rapid And Frequent Iteration Towards Measurable Usability Targets") will be outlined in this and the following two modules. The next module, "The Usability Engineering Lifecycle," shows how these three design principles are kept in focus by specific activities during development.

The best way to meet users' needs— including usability— is to understand the users intimately. A user-centered approach assumes that although people vary widely, they all have particular needs that must be met. For example, where a business process is being automated, instead of automating whatever can be automated and leaving the remaining tasks to the users, a user-centered approach will assign specific tasks to the users and the system, taking into account the users' needs.

Users can often be a source of product improvement and innovation— especially lead users and early adopters. The most difficult and demanding customers often become the best partners for product improvement. Users will often use products in ways that were never intended nor expected—these uses and abuses, and the problems users have as a result, often sources of inspiration for improvement or differentiation.

Users are experts in their requirements—they understand their goals and their tasks, they know the objects and artifacts they produce and use, the work-arounds they invent (not just the official, formal procedures), and the problems they have. However, users are not always good at describing, explaining, or predicting their behavior, and users do not often make good designers, so they have to be involved in effective ways.

In particular, users develop their own conceptual model of their work. This conceptual model is *never* the same as the designer's model. Users *always* behave in surprising ways, which is why they must be involved in the design process. A successful HMI maps the users' conceptual model directly onto the software or hardware so that the user may not even be aware of the HMI components.

Who Are the Users?

The first issue to be resolved is how to choose which users will be involved in the design. For consumer products, the answer will lie in the demographics of the target user population. For business productivity applications, target users are often easier to define, although sometimes more difficult to access or to involve in the design process.

For almost all products, there will not be a single user or user role. Although the end user may be the primary person affected by your design, there will also be secondary users—people who have requirements which need to be taken into account in the design and who are affected by the design even if they do not actually press the keys. The task of identifying the users and their different requirements is known as "stakeholder analysis."

For example, in a call-center for customer assistance and queries, the call-center operators are the primary users, but the call-center managers and the customers who call are other stakeholders. Similarly, the people who decide to buy a videoconferencing service for a major corporation are often not the people who have to use it, but both of these stakeholders—the users and the choosers— have important needs that need to be met.

During the early gathering of information, designers will start to understand the users' range of concerns, goals, and priorities. It is often helpful to develop a series of stereotypes— imaginary individuals whose life details and images are representative of the main user population. Developers and users alike can often relate to these portraits more easily than to dry statistics. These imaginary individuals can also star in the storyboards and scenarios that are used to gather users' requirements and explore solutions, as will be described later.

Principle #2: Involve Users Early and Continuously

Early design decisions are usually those concerning the product concept, its architecture, and its priorities. As a result, these are the decisions which are the most costly to change later. If these decisions are not user-centered, the end product will not be usable. For example, adding well-designed icons to a flawed menu structure will not rescue a poor product.

For this reason, users should be involved as early as possible in the design process. Users usually contribute to early efforts to gather information, through observation, by the use of questionnaires, focus groups and interviews, or through more detailed task analysis. At this stage, designers will be building models of the users' domain and establishing task priorities and relationships. As was noted, users may not be good at articulating their requirements or describing or predicting their own behavior. For this reason, field observation of user behavior is often most effective.

Users are also better at critiquing an existing HMI than designing one from scratch. But user involvement must be cost-effective. Simply placing users in front of a new application and asking for improvements will lead to an unprioritized wish list. This is why prototyping is crucial. Users should be given a number of alternative designs — whether high-level or detailed — to compare and critique. The alternatives will help them generate more ideas and also show that their comments are welcome and useful.

Figure 2: Involve Users Early and Continuously

As the design becomes relatively stable, user activities are aimed at refining and validating detailed design. Usability testing becomes most effective when measuring performance and productivity, including error rates and causes, and validating terminology and icons.

Typically, users will be asked to carry out specific tasks designed to test parts of the interface or to address particular design issues which could not be resolved earlier. Well-designed user trials will get maximum results for the time and effort invested by designers and users.

Principle #3: Rapid And Frequent Iteration Toward Measurable Usability Targets

The key to involving users is to take an iterative approach. Each iteration is an opportunity to bring in real users and evaluate different aspects of the evolving product. Early iteration prevents major architectural decisions from leading the development down erroneous and costly paths. Iteration should start as early in the development cycle as possible, with "lo-fi" prototypes, often pencil and paper designs that can be changed quickly.

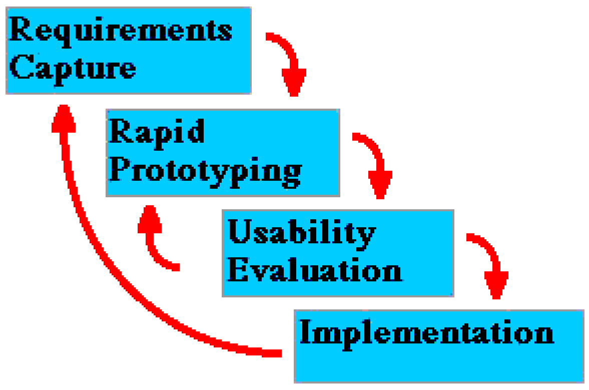
If a physical product is being designed, then card, foam, or plastic models should be built to test acceptability of size, weight, and shape and to ascertain the location of main components. One potential downfall is that with iteration it is difficult to know when to stop. Each iteration needs to be focussed on a desirable target and should improve the design. Usability metrics are the key, derived from an understanding of the users' priorities, the main tasks they will carry out, and the desired productivity. Even if a development is "time-boxed," with a pre-specified number of main iterations, there should be some minimum usability acceptance criteria.

The Usability Engineering Lifecycle: Requirements Gathering and Rapid Prototyping

Many techniques have been employed in usability engineering, and user involvement can range from simple consultation to participatory design in which end users become part of the design team. As this tutorial is aimed at newcomers to usability engineering, only the core activities of a usability engineering lifecycle will be described.

The basic usability engineering lifecycle contains four phases, as shown in Figure 3. An initial intensive requirements gathering phase is followed by a series of rapid iterations between prototyping and user testing before final implementation. Field implementation then becomes a source of requirements for future products, and the cycle repeats.

Figure 3: Usability Engineering Lifestyle



Requirements Gathering

There are three primary activities in the requirements capture phase: user observation, interviewing, and task analysis, often carried out in that order. Other techniques, such as focus groups, questionnaires, and surveys, may be employed. Additional information on potential product features may be obtained from user groups, from sales and marketing forces, from maintenance engineers and help lines, and from analysis of competitors' products.

A stakeholder analysis should first be carried out to ascertain which parties' requirements should have a bearing on the design. Other stakeholders then need to be involved in the design process. Stakeholder analysis can radically change design acceptance criteria. For example, with increasingly powerful and reliable telecommunications technology, maintenance costs are becoming a more significant cost factor. A new feature's value may need to be traded off against its potential to demand increased maintenance.

User observation will be the designer's primary source for a model of the users' domain, main tasks, and priorities. Observation is especially helpful in studying work habits of which the users themselves may not have been aware such as "work-arounds," failures, and exceptions. When looking for innovation, these breakdowns can be a rich source of new product ideas.

Observation gives the designers access to informal work practices — for example, observing the decisions made and the work conducted in the corridors, rather than by the formal corporate process. Observation will also bring to light the work context around use of a particular product or application and the relationship of the use of the product to other products. Interviews with users about their work will typically be semi-structured and focused around the designer's agenda but will allow the focus of the investigation to change.

There are many different task analysis techniques, usually involving a more formal and detailed interaction with users. The designers may ask users to carry out tasks while describing what they are doing and why, or the designer may even run through some user tasks, with users commenting and guiding. Video records of such sessions are often kept, so that the designers can refer back to them. One successful task analysis technique involves asking pairs of users to carry out a task, forcing the users to communicate with each other to complete the task. Formal task documentation, such as existing training material or company process descriptions, will be valuable at this stage, but the designer must still take into account the informal activities of users.

The results of task analysis typically describe:

- Roles and related tasks
- Sequences of events and relationships between them
- Objects involved in tasks and their attributes, including flow of information via paper and electronic documents, etc.
- Users' actions and resulting behavior
- Breakdowns and problems
- Users' terminology

This phase can produce large amounts of information, making it easy to lose sight of the main user priorities. One of the best techniques for ensuring continued focus on key user requirements is to capture main priorities in "scenarios." Scenarios are vignettes or stories, often produced as storyboards or short videos, which are taken from the task analysis and observation. They characterize the target user population (the stereotypes mentioned earlier), distill the key user tasks, incorporate critical design qualities, and communicate main product features.

Rapid Prototyping

Requirements capture obviously demands users' involvement. However, the design activities that follow requirements capture have traditionally failed to continue this user focus. Iterative development, by employing prototypes, is the key to ensuring that users remain at the center of the process while the product is being refined and cost trade-offs are being made. Users also often find it difficult to understand written specifications and prefer to critique rough prototypes.

Early and late prototypes

There are two kinds of prototypes, each with a different purpose. Early prototypes are used to validate the designers' understanding of the users' domain and to explore design alternatives. These prototypes should be partial, cheap to produce, and discarded frequently. Later prototypes are evolutionary and are gradually refined to become the finished product.

Early prototypes should be "lo-fi," such as paper-and-pencil prototypes of user interfaces or inexpensive mock-ups of hardware. These lo-fi prototypes are important because they allow users to be involved early in the process and because users will feel better able to critique artifacts which are obviously unfinished. Producing high quality prototypes at this stage is inappropriate and can divert effort from the main design issues onto small details.

When the main design issues are relatively stable, or when a pre-defined project deadline has been reached, a prototype will be produced which is to be refined into the final product. This prototype will need to be robust, reliable, and meet software design quality requirements. Users still need to test and critique this prototype, as there will still be design trade-offs and probably some HMI compromises to be made which will affect end users.

With this understanding, software tools such as Visual Basic, Supercard, or Macromedia Director may be used for prototyping. The World Wide Web is a useful prototyping medium, as HTML and JavaScript are easily changed and immediately accessible from a range of platforms.

Horizontal and vertical prototypes

Prototypes also come in horizontal and vertical varieties. Horizontal prototypes are broad but shallow, showing the range of facilities or features available without implementing any of them in depth. Vertical prototypes implement features in depth without necessarily illustrating all of those available. Most prototypes should be a hybrid, showing a broad range of intended features and implementing the key ones in depth.

Prototyping the whole product

If some "moments of truth" — important stakeholder interactions which are critical to the success of the product — have been identified, then it is quite appropriate to prototype these interactions as closely as possible, even if these are not directly concerned with the HMI. These situations should already have made their appearance in the scenarios used to focus design priorities.

For example, for a product sold mainly through retail outlets, observing buying behavior with a mock-up of the new product packaging on a shelf alongside competitive products will more accurately reflect potential buying behavior than questioning buyers about their likely behavior. Similarly, installation can be a moment of truth, as it brings together all the elements of the product in an important interaction that can set the tone for the customer's future perception of the product and manufacturer/supplier.

The user manuals, helplines, and other support also make up a vital part of the whole product. The user guide should be prototyped and tested in parallel with the product, not written as an add-on to try to cure usability problems. Similarly, the user guide need not be "idiot-proofed" but should adopt diverse strategies to installing and learning a new product.

Managing the prototyping process

The two main challenges with a prototyping approach are, first, ensuring that each prototype is an improvement over the previous version and, second, knowing when to stop. In order to manage these issues, it is important to set minimum and desirable usability performance targets as well as cut-off dates for iteration.

Figure 4: Usability Evaluation

Usability targets should correspond to the key scenarios that have been articulated. The actual values will be based on comparison with competitors' products, with previous versions of the product, or perhaps with the manual performance of a task which is being automated.

Usability targets may be at different levels of detail (see examples).
Each usability target should include:

- the target users (stereotypes)
- the task or activity being carried out
- the relevant product features
- relevant aspects of the environment

Two examples of usability targets:

- ♦ "Target market segments must be able to make their first telephone call within 5 minutes of opening the product packaging in the home."
- ♦ "A network manager with at least 2 years experience, and with no measurable red-green visual deficiency, must be able to detect 99% of critical errors represented by color change from [rgb values to be defined] of a one-pixel wide line connecting nodes on a visual display [specify makes], in lighting conditions representative of operations center X, over a 1-hour period of typical peak hour traffic."

The Usability Trial

A prototyping approach allows the usability metrics to be monitored constantly, from the moment that the relevant product feature has been included until the desired target has been reached. In a typical usability trial, which is usually recorded on video, users will be asked to carry out specified critical tasks. The session may conclude with a questionnaire or debriefing interview. Typical usability targets will refer to time of completion of a task, number of errors made, or quality of completed task. The usability trial, however, is also an opportunity to obtain users' subjective reactions to the prototype and to follow up on marketing issues where appropriate.

Usability trials often take place in laboratory conditions when the measures need to be precise and the environmental conditions carefully controlled. However, field tests will more accurately reflect the end use situation.

Implementation

The heart of the user-centered usability engineering process is the cycle between prototyping and evaluation. However, sooner or later — ideally triggered by achieving the usability targets — the product is implemented and there is a greater investment in packaging, marketing, production, selling, and maintenance.

Implementation can be seen as an evolution of the prototype-evaluation cycle, moving out of the development laboratory and into the marketplace. Every product can be seen as a set of hypotheses with the ultimate test of success being profitability.

Figure 5: Implementation

In theory, all key issues should have surfaced during the previous design activities. There are always last-minute trade-offs to be made in implementation, however, and surprises which only emerge when a product finally goes to market. At least with a user-centered design approach, these further trade-offs can be made while keeping in mind the user priorities outlined earlier in the process.

Continuing the product evaluation activities post-launch will also help capture any surprise issues as soon as possible and give quick feedback for the next version of the product. Post-launch there are also rich new sources of data to be exploited from the sales force, retail, and distribution channels, from trade reviews and user groups, from maintenance engineers, and from customer help-lines.

Project Post-Mortems

Project post-mortems are not frequently carried out, but should be viewed as an opportunity to answer questions such as:

- Did the product really meet the targets?
- Were these the right targets?
- What trends will change the targets for the next generation?
- What are the unexpected field issues?
- What are the key manufacturing and distribution issues?

Project post-mortems and the continuation of the prototype-usability evaluation cycle into the marketplace improve the design process on its next cycle.

Where to Find Data on Usability

Books

Bias, R. G. and Mayhew, D. J. *Cost-Justifying Usability*. Boston: Academic Press, 1994.
Kirwan, B. and Ainsworth, L. K., Eds. *A Guide To Task Analysis*. London: Taylor & Francis, 1992.
Nielsen, J. *Usability Engineering*. London: Academic Press, 1993.
Norman D. *The Design Of Everyday Things*. New York: Doubleday, 1990.
Rubin, J. *Handbook Of Usability Testing: How To Plan, Design, And Conduct Effective Tests*. New York: John Wiley & Sons, Inc, 1994.
Winograd, T. *Bringing Design To Software*. New York: ACM Press and Addison, 1996.

International Standards Organization (ISO)

- ISO 9241 "Ergonomic Requirements for Office Work with Visual Display Terminals," produced by ISO Technical Committee 159, Signals and Controls Group 4, Working Group 5 (TG159/SC4/WG5). Covers many areas from task, keyboard, and workplace requirements, to providing guidance on dialogue design, menu design, and the use of color. Part has been implemented in European legislation (Directive 87/391/EEC).
- ISO 14915 standard on multimedia, also produced by TC159/5C4/WG5. Covers design of controls and navigation, media combination/individual media requirements, and domain specific multimedia aspects.

ISO-lEG Joint Technical Committee 1 is a combined activity of ISO and the International Electrotechnical Commission. Working Group 9 of Sub-Committee 18 is developing standards in keyboard layout, dialogue interaction, and symbols. Its work includes:

- ISO-lEG 11581 "Graphical Symbols on Screens."
- ISO-lEG 13714 "User Interface to Telephone-Based Services — Voice Messaging Applications."
- ISO-lEG 11580 "Names and Descriptions of Objects and Actions Commonly Used in the Office Environment."

More information may be found at the ISO web site http://www.iso.ch.

American National Standards Institute (ANSI)

- ANSI/HFES 200 "Ergonomic Requirements for Software User Interfaces." Extends ISO 9241, including a chapter on guidelines for making user interfaces usable for people with disabilities of different kinds, written in response to the Americans with Disabilities Act.
- The Information Infrastructure Standards Panel (IISP) was set up under ANSI to accelerate the development of standards which will be crucial to the building of the United States' national information infrastructure and the taking of a global perspective.

- ANSI T1M1.5 provides technical contributions on "Telecommunications Architecture, Interfaces, and Protocols" to Study Group 10 of the International Telecommunications Union

More information may be found at the ANSI web site http://www.ansi.org.

Organizations And Conferences

The main international organization is the SIOCHI (Special Interest Group in Computer-Human Interaction; home page: http://www.acm.org:82/sigs/sigchi) within the ACM (Association for Computing Machinery). There are many national and local groups within SIGCHI and affiliated to it, for example, the Usability Professionals' Association (http://www.UPAssoc.org). There are also many national and international human factors groups; examples are the North American Human Factors and Ergonomics Society (LIFES) and the United Kingdom's Ergonomics Society (http://www.ergonomics.org.uk).
Most of these societies have special interest groups focusing on telecommunications, for example, the communications technology group within the LIFES. Apart from local conferences, the two main international conferences are the annual CLII conference and Human Factors in Telecommunications, held every 2 or 3 years.

Web Sites

The World Wide Web is becoming an increasingly valuable source of research information. In addition to the sites listed above, the following sites are great places to start an exploration of the latest global research and practice in usability engineering and good HMI design.

- AGM SIGCHI home page: http://www.acm.org:82/sigs/sigchi
- The human-computer interaction virtual library at: http://web.cs.bgsu.edu/hcivl
- The University of Delft's UI Design pages: http://www.io.tudelft.nl/uidesign

World Wide Web Consortium (W3C)

W3G is responsible for defining Internet-specific standards, including the HTML hypertext formatting language.
More information may be found at the W3C web site http://www.w3.org.

USABILITY ENGINEERING IN PRACTICE

What Is Usability Engineering in Practice?

Four assertions conspire to make usability a key pursuit in today's software development:

- Human beings are remarkably similar
- Computer programmers are human
- Human beings are remarkably different
- Product development resources are finite

Human beings are seductively similar — most of us have two functioning eyes, arms and legs; fingers roughly the same size; memory capabilities ranging only from fair to "I wish it were better." And so, because programmers are human, they have the perfectly understandable tendency to think that if they can figure out how to use some computer application, other humans will likely be so able.

However, these human similarities are illusory. In fact we are remarkably different in our cognitive capabilities and our experiences. Thus we differ wildly in what we find "usable." Because of this, the world of computer software has a long and inglorious history of unusable applications produced by programmers who depended on their intuitions for what would be seen as usable. The solution to this problem is a strong focus on "user-centered design" (Norman, 1990), whereby user data are collected to inform product designs.

Relatedly, since product development resources are finite, the field of Usability Engineering, complete with "discount usability methods" (Nielsen, 1993) for collecting these data, has emerged as a key approach to user-centered design.

What Is Usability?

Usability is easy, intuitive (safe, fulfilling) access to a system's underlying functionality. It is a measurable system/product/process characteristic, best designed in from the beginning and best achieved by understanding the users and their environment.

Usability is not all art and no science. It is not tacked on at the end, not best fixed via documentation, nor stumbled onto by accident. It is not free, but given the now popular discount usability engineering methods, it need not be expensive. Employed properly, usability engineering saves development time and costs, plus yields greater sales and customer satisfaction.

Convergent industry trends toward open systems, downsizing, lower training costs and flexible (exchangeable) work assignments have combined to elevate the importance of usability to users of computer software and, therefore, to software vendors. Indeed, Norman (1990) refers to usability as "the next competitive frontier" (p. vi).

Usability Isn't Just a "Nice to Have"

No longer considered just aesthetics or a final adjustment of the graphics, good usability translates to reduced customer training costs, increased customer productivity, reduced customer support costs, higher customer satisfaction, better press (internally and externally), and concomitant higher sales (or, for internal development organizations, higher usage and throughput). There is particular attention paid to, and expectation of, usability in the open systems, client/server world. However, even in legacy, character-based systems, level of usability can mean the presence or absence of high throughput and high user satisfaction.

Usability Engineering

Good usability engineering begins with user-centered design — an early and ongoing attention paid to who the product's users will be and what tasks they'll be trying to carry out with the product. Unless all your users are just like the product developers, it's a bad idea for those developers to depend solely on their intuitions about what is and is not usable. Thus, it is best to assume an "empirical" design approach — design informed by user data.

Following are methods and tools you can employ to help ensure your customers find your system, product or process usable:

- Task analysis — understanding users, tasks and environments
- Design guidelines — for how to do a good job with the first draft
- Prototype testing — long before underlying function is coded
- Usability walkthroughs — like code walk throughs, but with a focus on usability
- Heuristic evaluations — a usability professional's "professional judgment"
- Usability lab testing — in-house, or vended out
- Field testing — but don't make this your only investment in usability
- User surveys — cheap, but effective

Discount Usability Engineering

The aforementioned discount usability engineering methods, such as usability walkthroughs, are methods that usability professionals (and sometimes the software developers themselves) can employ that have a lower cost (when compared to traditional lab testing of end users) and yet yield nearly the same amount and quality of usability data to help steer product re-designs. Thus, in any formal or informal cost-benefit analysis conducted to cost-justify the use of usability engineering techniques, these discount methods usually prove to be well worth their time and cost.

Why Has Usability Been a Problem?

There are various reasons why software user interfaces are not wondrously usable. Software developers' previously mentioned tendency to depend on their own intuitions is one. Another is an organizational inhibitor to good user interface design: "the limitation of contact between designers/developers and users" (Mayhew and Bias, 1994, p. 296). Whether it has been for reasons of corporate propriety, or because of the expense, or from a fear of allowing customers to interact with "bit heads," software developers have tended not to have a direct line of communication with users.

Another Solution: "Develoops"

One rich, yet inexpensive (bordering on free) solution to this problem space is to establish a series of feedback loops between product development and other groups within the corporation: thus "develoops." The objective behind these feedback loops is to capture and attend to user data that are already in-house, the least expensive user data imaginable. The establishment of these feedback loops has met with universal support, but it has required approaching the groups one at a time and a tailored approach to these feedback loops. Thus, now there is a structured vehicle for the collection of user data from:

- Customer Support
- Sales
- Pre-sales support
- Training
- Service consultants

Within Your Company

At your company, you should institute an enthusiastic program of gathering and attending to all available user data. This is just one part of a full program of usability engineering. Next initiatives can include:

- Supporting emerging product integration efforts
- Providing usability support for various products and components
- Establishing (or reinforcing) feedback loops within your company
- Establishing a GUI Guidelines document
- Establishing a presence in the trial and beta programs

An informal education effort can be called the "Usability University”. It will help drive usability awareness and expertise to all corners of the corporation.

Usability *DOs* and *DON'Ts*

The following set of Usability *DOs* and *DON'Ts* should being communicated throughout your company. As you read them, you may want to consider how these could be employed in your environment.

DO:

- **Design from your end users' perspective.**
  — Imagine yourself in their shoes, using your product.
- **Ask: Is your product a member of a family of products?**
  — A product can be usable by itself, but not usable in the context of other user interfaces (UIs).
- **Realize the value of consistency in UIs.**
  — The goal is positive transfer of knowledge; once a user knows how to use one application, he or she is a large fraction of the way toward knowing how to use the next one.
- **Comply with UI guidelines.**
  — If you have some. If you don't, adopt or build some.
- **Test.**
  — Do it yourself, or get help from a usability professional.
- **Know your customers, potential customers and what they do.**
  — Be creative about collecting data from them and about them.

- **Ensure the usability of the whole system — UI, documentation, online help, support, training.**
  — It all goes in to forming the customers' perception.
- **Make your function accessible.**
  — "Underlying function" that your users can't get to is the equivalent of "no function."
- **Imagine the pride you'll feel, when your products are celebrated as having "world-class usability."**
  — Can you?

DON'T:

- **Depend on your intuition.**
  — You may not be representative of your users.
- **Expect to get the design right the first time.**
  — You budget time for debugging. Design isn't any easier than coding.
- **Think that usability is just selection of UI widgets.**
  — Usability is look-and-feel, but it also is how well the objects and their relationships match the users' mental models.
- **Expect to achieve good usability by "fixing it in the documentation."**
  — Good, usable publications are important — as a supplement to a usable UI, not a replacement.
- **Expect usability to be "added on at the end."**
  — Usability is best designed in from the beginning.
- **Design in a vacuum.**
  — We're aware of the competition, because it's helping build user expectations.
- **Externalize the internals.**
  — Your users want to complete a task — they're not interested in how complex the underlying designs are.
- **Let the product be shipped with any known, severe usability problems.**
  — It's your baby.
- **Assume usability testing will slow down development.**
  — Good, timely testing can keep you from working on areas that don't need to be fixed.
- **Be satisfied with "good enough."**
  — You don't feel this way about the function; don't settle for this with the interface either.
- **Assume that usability is someone else's job.**
  — It's yours, too.

And So?

To counteract the tendency to think, "If I can perform this task, anyone can," just observe frustrated users struggling with software applications or read the popular press for stories thereof. User-centered design, driven by usability engineering, is the explicit, systematic approach to building software that users will find usable.

In these times of ever-monitored profit margins, it is important to be able to cost justify any "support" efforts in software development. It is easy to quantify usability costs (e.g., salaries, test subjects' fees, developers' time to produce prototypes). While it may be harder to quantify usability benefits (e.g., more product sold, fewer calls to the help desk, increased customer satisfaction), it is possible nonetheless. And studies on this (e.g., Karat, 1994) have shown that most software usability investment is recovered many times over.

References

Karat, C. M. (1994). A business case approach to usability. In R. G. Bias & D. J. Mayhew (Eds.), *Cost-justifying usability*. Boston: Academic Press.

Mayhew, D. J., & Bias, R. G. (1994). Organizational inhibitors and facilitators. In R. G. Bias & D. J. Mayhew (Eds.), *Cost-justifying usability*. Boston: Academic Press.

Nielsen, J. (1993). *Usability engineering*. Boston: AP Professional.

Norman, D. A. (1990). *The design of everyday things*. New York: Doubleday Currency.