WINNING THE BUSINESS SOFTWAR

Bold Tactics to Help You Join the Winning Side of the Software Revolution and Get the Productivity You've Been Expecting

Winnning the Business Softwar

Published by Simax
8904 Carlisle Road
Wyndmoor, PA 19038-7412
(215) 836-9147
webmaster@UsabilityInstitute.com

Printed in the Unites States of America

The Productivity Gap 6

Big Problems Call for Small Solutions 7

And We Mean Business 8

A Computer User's Bill of Rights 9

What's Going on Here, Anyway? 10

- The Undesigned Design 10
- The Mother of All Problems 11
- Most Programs Are Only Half Finished 12
- Form Follows Dysfunction 13
- The Big Disconnect 14
- There Are Very Few User Interface Designers 15
- The Shifting Role of the User Interface 16

Rules For User Friendliness 17

- Part 1: Simple Wording 17
  - Use Verbose Phrasing 17
  - Use Perfectly Accurate Words 18
  - Be Explicit, Not Implicit 19
  - Build Error Details Right into the Program 20
  - Show the Level at Which Options Are Invoked 21
- Part 2: Easy Programming Changes 22
  - Put All Functions on Menus 22
  - Display from the General to the Specific 23
  - Put Functions Where Users Will Look for Them 24
  - Don't Change Menus and Dialogs On-the-Fly 24
  - Always Provide Visual Cues 25
  - When the User Does Not Have Control, Announce it Prominently 25
- Part 3: Big System Buyers' List 27
  - Wizards: Step-Through Dialogs 27
  - Design for the Learning Period, Not Just "Production Use" 27
  - Let Users Work with Descriptive Values, Not Internal Codes 28
  - Provide Layer-Testing Diagnostics 29

Help! 30

- Provide Real Information 30
- Provide Help in the Program 31
- User-Sensitive Help 32
- User-Contributed Help 33

Documentation 34

- Don't Proofread, Prove 35
- Timing is Everything 36

Print a Short Read-Through Guide 36
What's in a Read-Through? 37

Training 38

Shift to Productivity Consulting 38
Distribute Periodic Literature 39
Organize Peer-to-Peer Productivity Sessions 40

PC Hardware 41

Write It Down 41

Support and Troubleshooting 43

A Beginner's Guide to Backup 43
Directory Assistance 44
Your Own Knowledgebase 45
Technical Troubleshooting Primer 46
Questions to Ask About Support 48

Bellis's Law:

For every computer problem,
even hardware problems, there is a corresponding remedy that could be accomplished
through the design of the software, particularly the user interface.

It is not your fault...
the problem is not user error!

The Productivity Gap

I called a professional printer to get an estimate on producing the booklet you're reading now. His estimating software crashed as he was entering the job information—yes, really. He said he'd call me back.

Productivity. That was the promise, right? And what a payoff we got. You can now operate the equivalent of a publishing house from a home office. With programs like Macromedia Director or Intuit's Quicken, you can learn in weeks what used to take a lifetime apprenticeship, and get results with a fraction of the time and money that used to be required. That's the good news. But the average computer user does not go half a day without some sort of unnecessary frustration, problem, or time loss. Reversing that situation is what this booklet is all about.

The printer whose program crashed was using an estimating program that he had created himself, and he deserves a lot of credit for the accomplishment. In fact, he gave me the first part of the pricing information in a matter of seconds, just before the program had a problem. This put him head-and-shoulders above the other printers I called, who all seemed to act as if obtaining a price estimate required conjuring spirits. So his experience was very typical: computers both elevate his business and make it more frustrating.

The "Little Things" that make the difference between good business and bad
are only little when you do them.
If you dont do them, they're very big things.

Big Problems Call for Small Solutions

It's been over 15 years since the beginning of the computer revolution, and there's widespread recognition that we have some big problems. But there's not widespread agreement about how to effect positive change. Some feel we need to overthrow the powers that be.

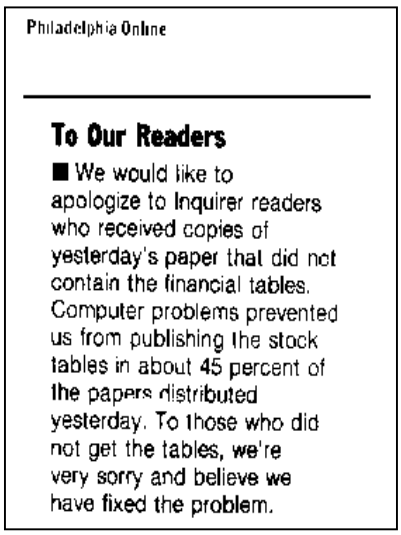
The premise of this booklet, however, is that complicated or expensive solutions are not needed to cure most of what's wrong with the computer world. The lion's share of the problems can be fixed by little things, simply improving communication, mostly in the form of the user interface—the design of the software components with which users interact. Often these changes are as simple as more and better words on the screen, and these are usually the easiest, least expensive changes to make.

> *By some estimates, businesses spend more on computer technology than on the natural gas, automotive, petrochemical, steel, and mining industries combined. Multiplied over such a base, imagine the true enormity of what we routinely shrug off as minor annoyances.*

More technology—in and of itself—is not the answer. The current flavor-of-the-month is called Java, the latest in an endless parade of programming languages, which developers must learn again from a standing start. According to the industry it will free us from the Microsoft dictatorship and deliver us to the Promised Land. To me, it's another distraction from the real goal: following through on established practices to fully capitalize on the great technology that we already have.

And We Mean Business

The focus of this booklet is the PC world, not mass-market software... business systems like the one referred to in the little blurb shown here, published recently in the Philadelphia Inquirer.



Philadelphia Online

To Our Readers

■ We would like to apologize to Inquirer readers who received copies of yesterday's paper that did not contain the financial tables. Computer problems prevented us from publishing the stock tables in about 45 percent of the papers distributed yesterday. To those who did not get the tables, we're very sorry and believe we have fixed the problem.

Mass market software, is improving much more rapidly than business software. Fierce competition and other factors—those that represent what's good about the PC world—are improving this category at a fairly successful pace. But the business software world is getting little or none of the warm breezes of progress that the mass market is experiencing. It often seems to be moving backwards. Too many projects repeat past mistakes, and very few provide the level of usability necessary to hit the ground running. Consider the following, somewhat dated but still troubling figures that categorize software project success rates:

- Delivered but never used: 47%
- Paid for but never delivered: 29%
- Extensively reworked before use: 19 %
- Used after changes: 3%
- Used as delivered: 2%

Software projects often languish because their champions move on, or for other political reasons. But that can't explain such dismal statistics. Let's see what can. But first, a simple list of expectations, a destination that, like a mirage, seems to move away as quickly as we approach it…

A Computer User's Bill of Rights

"We hold these truths to be self-evident…" that all computer users recognize when they're being asked to do a task that the computer should be doing for them… that computer users are not dumb, but rather, unable to read the minds of the designers… that all we users need is a fighting chance. So here's our list of our basic rights. Demand these rights at every opportunity.

We shouldn't have to read a manual,
certainly not a huge one.

We should be able to accomplish every task and entry with the fewest possible keystrokes.

We should be able to do things out of order
without being penalized.

We should be able to make mistakes without being terminated, executed, canceled, re-booted, or erased.

We should be able to understand why the program
does what it does.

We expect that all of what we type into the computer
will be saved, by default.

We expect to be forewarned when any work is
over-written, undone, or erased.

We expect to have most of our work retained
after the power is interrupted.

What's Going on Here, Anyway?

We won't spend much time dwelling on the problems themselves, but it's valuabld to understand the lay of the land and the major obstacles in our quest. In the next few topics, we'll discuss exactly how we've gotten into such a big mess. This knowledge and perspective will be important in helping us judge which battles we should even consider fighting.

Throughout this booklet, we'll use Microsoft Word for many examples. And through the miracle of computers, this includes both good and bad examples.

> *In his book, The Underground Guide to Microsoft Word, Woody Leonard aptly refers to Word as "The worst word processor in the world, except for all the other ones."*

The Undesigned Design

> *A man purchased a new car, and, after signing the paperwork was mortified when the salesman said "Now go down the street to the Joe's Engine Store and buy an engine."*
>
> *"You mean there's no engine in this car?" he asked.*
>
> *"No," the salesman explained, "this way you have the freedom to buy whatever engine you like, and the competition of the free market system encourages engine manufacturers to develop the best engines in the world."*

As silly as the notion sounds for cars, this is basically how the PC world has evolved, having the software and hardware components designed separately from one another. And it's true, the competition *has* made it the platform with the world's most powerful engines at the lowest prices. But the Windows-Intel architecture was essentially conceived out of wedlock, without a fully thought out scheme or commitment, and its offspring have forever been affected.

What does this troubling ancestry mean to you? Simply that you must be prepared for the products and solutions you buy to be engineering compromises… designed to make the best of a bad situation. Don't wait for an overthrow of the ruling class—put your effort into countermeasures.

The prevailing logic of the Wintel world, as many people see it, is "more is better." I'm not against more, faster, and better features—I just want to see an equal priority placed on achieving 100% productivity. Now that we have the world's most powerful, inexpensive, cars-with-separately-sold-engines, the question becomes how do we get the best mileage from them.

The Mother of All Problems

Electronic processes are invisible to the naked eye.

More to the point, the inner workings of the computer are so micro-miniaturized as to be completely beyond our ability to cope when things are not entirely as we expect. Almost all other problems are really results of this one deadly killer. (And you thought the problem was Bill Gates!)

How can I blame so much on so singular a factor? The limiting factor with today's computer systems is not their performance when things are working right, but their ability to frustrate us when things are malfunctioning or simply incomprehensible. Whether in software or hardware, true root-cause diagnosis is almost *never* performed. Instead, whole electronic modules and even whole *products* are replaced when there are problems. This is often done without genuine diagnosis, but with a process that requires less expertise: swapping parts in a crude process of elimination.

Unfortunately, computers are as fragile and vulnerable as they are powerful. The story of computers in the workplace is really the story of our complete and utter dependence on electronics. That dependence has become more thorough as more organizations adopt what are called 'mission critical' systems.

Most Programs Are Only Half Finished

The pressure to be first to market in the computer business is a plain case of do-or-die. It's why most programs don't do everything they should. The following quote, attributed to Peter Norton, a prominent computer expert, summarizes this unfortunate tradeoff for users:

> There is always a tradeoff between the convenience of the programmer and the convenience of the user.

Another way of saying it is that good programming takes more work. A lot of the work that makes a program easier to use is in the user interface design, the things you see, hear, and click on. By my rough estimate good user interface design can take about three times as much work.

I don't know what the question is, but the answer, as usual, is money!

Who Does What?

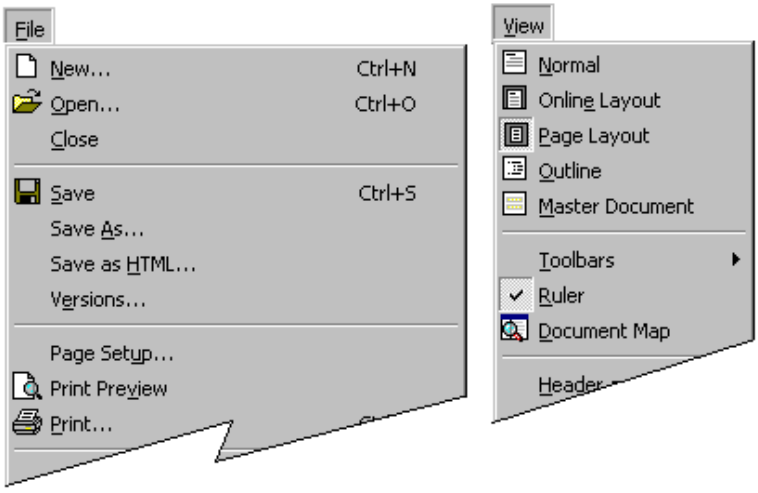
Development Managers: keep a master list of all of the best features (as opposed to program code) of all programs, for use in all new programs. If possible, make it a working prototype, or or at least a printed functional specification.

Vice Presidents: read the rest of this booklet to learn what makes a program complete. Then create an environment where these principles are common knowledge, the rule rather than the exception, by inculcating these ideas throughout the organization.

Form Follows Dysfunction

A common saying about design in general—not just programming—is that "form follows function," meaning that a thing becomes what it is because of what it does or how it is used. But in my experience, this is decidedly not true in many manufacturing instances, and computer programming is no exception.

Bad design is often the result of designing things for the convenience of the manufacturing process, rather than the end user. A classic example of this problem is the location of the Print Preview feature in many programs. It's on the File menu, near Print, instead of on the View menu where you might expect to find a preview feature. These functions are shown on the following menus:



Which Menu Should Print Preview Really Be On?

In this case form has followed the convenience of the programmer, instead of the function expected by the user. Previewing a document as it would be printed should be on the View menu, since it is simply another viewing option. Instead, because the *programmer* must perform the same tasks when previewing as when preparing the document for printing, it is placed on most File menus, adjacent to Print. As you become a more educated observer of usability issues, you will see many places where users struggle to find or apply a feature because it has been designed from the point of view of the manufacturer, not the consumer.

Stubbornness does have its helpful features.
You always know what you are going to be thinking tomorrow.
— Glen Beaman.

The Big Disconnect

A man walked into a computer store to buy a custom-tailored suit. (Bear with me, this is an analogy. If you must, imagine the year is 2010 and you have to go to a computer store no matter what you want to buy.) The salesman took a few measurements, did some clipping with his scissors and sewing with his machine, and immediately wrapped up the new suit. The customer remarked, "Don't you want me to try it on?" The salesman replied, "Oh, no sir, that's not how we do things in the computer industry."

Computer people are starting to acknowledge the importance of usability testing, which is a fancy name for—imagine this—actually trying to see if people can do what they bought your program for. Wow, what a concept! Most of today's business software, is designed, developed, and sold in a "we-know-what's-best-for-you," one-way fashion. Generally, systems are not adjusted based on observing actual use. If users are involved in design decisions, it's usually for verbal input, not an actual examination of the system in use.

What can you do?

Whether you are a developer, writer, purchaser, or manager, work with end users constanly. Add frequent feedback sessions to your written and spoken expectations, particularly your timelines.

Watch untrained users try to cope with your program. Don't tell them what to. Just watch and take notes. Then get a designer to recommend changes.

Plan on your first release of a system as simply a starting point.

There Are Very Few User Interface Designers

> *In his "Software Design Manifesto," Mitchell Kapor, creator of Lotus 1-2-3, observes, "The most important social evolution within the computing professions would be to create a role for the software designer as a champion of the user experience…"*

Every day, thousands of programming jobs are advertised in newspapers, but there is virtually no mention ever made of program interface *design* as a skill, let alone a job classification. Only the largest companies have someone whose sole job it is to design the human engineering of the programs.

It's important that interface design is an independent task because it needs to have someone constantly fight for its cause, against the pressures imposed by time and money. When it is added to the tasks of an existing role, the struggle is too arduous, and the erstwhile designer will move on to less contentious tasks.

Who Does What?

Everyone: lobby for an interface designer in your company.

IS Managers: Make user interface design a job category, even if it's part-time, contracted out, or a portion of a current job. The mandate of this job should be to make your systems comply with my Computer User's Bill of Rights.

The Shifting Role of the User Interface

This one's not so much a problem as a constant challenge. Products and services that do not answer the challenge are not giving you enough for your money.

> The goal of user interface design is to reduce the need for training and documentation to the lowest possible level.

The user interface must help you cope with the fragile, invisible electronics, the hiding places called disk drives, the unfriendly operating systems, and the incomplete programs described earlier. The relative success of an interface is determined by the skill and desire of the programmers and the usual business limitations: time and money. Designing user interfaces involves a special area of expertise that has come to be know by the lofty name of "human factors engineering."

As programs have evolved from the command prompt systems of the 70's to the menu systems of the 80's and the full graphical user interface (GUI) of the 90's, the significance of the words on the screen has evolved too.

Command prompt systems, with only a C:> staring at you, required you to get all of your instructions from documentation or training. *You* communicated with the *computer*, but it did almost no communicating with you. Menu systems provided more power and more words on the screen to guide you. Today's most advanced systems do everything for you except those tasks that absolutely require a human operator.

During this evolution, the mission of the user interface has changed to fully encompass communication. Today's wizard techniques epitomize this: the screen is full of pictures and text, guiding you through a relatively small number of critical decisions. In other words, the screen is mostly documentation.

This is the logical evolution. After all, users shouldn't need books to use programs. The information should be embedded in the system! The goal of a computer program is to perform actions, but the goal of the user interface is to communicate with the user. This enlarged communication role now leads us to what will be our first area of focus, the words being used. Sparse words, wrong words, and indirect words are a blight on the user interface landscape. Fortunately, the cost of correcting these problems is low, and it's where your greatest return on investment can be found.

Rules For User Friendliness

And now... specific recommendations to improve user interface design and make your software self-training, easily discoverable, and as fun to use as it should be. This section is arranged with the easiest recommendations first. The first two sections require no changes to the logic of a system, so the changes are childishly simple for any serious developer. If you can implement even the first group of recommendations, you can go a long way toward turning a difficult system into a productive one.

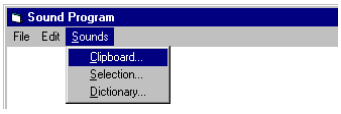
Part 1: Simple Wording

In the following topics we'll suggest some techniques you can use to make software easier to use with the simplest of all changes, the words you read on the screen.

Use Verbose Phrasing

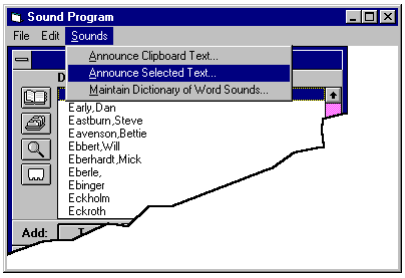
Many of today's programmers got into the field when there was as little as 64K of RAM and fixed-disk storage was 10 MB (stop laughing). This might explain why, when you drop down a menu, the captions are very terse. *You might think that each letter was expensive*, or that brevity was a virtue for some other reason. Indeed, brevity can be a virtue when a function's purpose is perfectly apparent, in which case scanning a list of single words is quicker than scanning one of full sentences.

But as programs have become more sophisticated, more detailed functions are prevalent. The tendency has remained, however, to try to identify these functions with one- or two-word names. Here's an example of a menu as it is typically worded with terse labels:



Example of a Terse Menu

Can you tell what they do? Recently, you can see that programmers are placing almost full sentences on menu—a change for the better. Let's look at the same menu reworded with verbose labels:



Example of a Verbose Menu

Notice, above, that there's usually both a noun and a verb in a complete phrase. Let's look at another excellent example that drives home the point. A very good graphic conversion program has the following three functions on a menu:

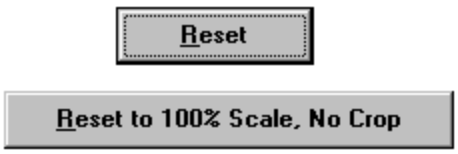
- Resize
- Redimension
- Crop

It takes several minutes of experimenting to decode these choices into these verbose descriptions:

- Rescale (Stretch/Shrink) Image or Selection
- Change Size of Selection Frame
- Crop Window Size to Current Selection

This is an admittedly difficult set of functions to describe well in only a few words. But the awkwardness of the terse names is highlighted by the fact that it was hard for me to remember which was which, among the first two items, even *after* I figured out what they do. Notice in the verbose suggestions that nouns have been added, identifying what the object of the action is.

Even on buttons, there's no longer any reason to be stingy with words. Spell out exactly what the button does. The following figure shows before and after buttons, one terse, one verbose, when formatting pictures in Word:



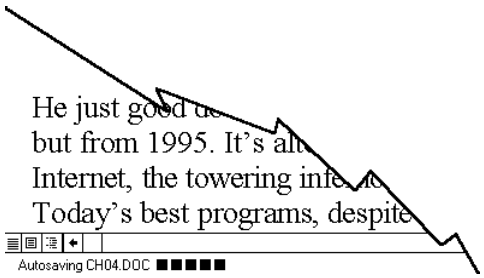
Terse and Verbose Button Labels

The first button requires you to guess whether the picture will be reset to the immediately prior state or its original state. The second leaves no doubt.

Use Perfectly Accurate Words

> *I went up in a friend's small plane one clear, sunny day for a little jaunt around the neighborhood. We flew into a nearby, uncontrolled airport, which means there is no tower operator. My friend checked the gauges, the situation on the ground and in the air, and everything looked A-OK for landing.*
>
> *When we descended and got about 200 feet from the ground he acted a little flustered. We landed without a hitch, but it turns out that the ground came up on him about 500 feet too soon, because he momentarily mistook elevation for altitude—a harmless mistake unless you're piloting an airplane. Fortunately he wasn't flying by instruments. Splat.*

Some wording choices are not as easily made as in the previous story. Often there are two points of view, but a little study usually reveals the preferable choice. Consider the following situation. When you turn on Microsoft Word's automatic, timed backup feature, it periodically saves a backup copy of the current file. In the status bar at the bottom of the screen, it says something like "Autosaving up CH04.DOC":



However, it is not saving a copy named CH04.DOC... it is saving a specially named, temporary file, perhaps named ~MY999.ASD (the ASD stands for Auto Save Document). It also doesn't list the directory in which it is saved, but that's a matter of explicitness, which we'll talk about next, not accuracy. This inaccurate labeling is not a big deal until you lose some work and are bound-and-determined to unravel the mysterious workings of the backup method, as I was.

Be Explicit, Not Implicit

Explicitness is different from accuracy. The more explicit your wording is, the less interpretation it requires.

For instance, a graphics program has an option entitled "Use stretchy lines." This feature, when activated, enables the size of the selection box to be changed, instead of having a fixed size. (A selection box lets you drag a rectangle around a portion of artwork.) A more explicit name would have been "Changeable Selection Box Size."

> *I worked at one of the world's foremost science museums. The exhibit coordinator one time told me and my boss, in reference to a minor prototype that just wasn't developing into a usable exhibit, "Can it."*
>
> *So we threw it out.*
>
> *Some time later, much to our surprise, he asked us to resurrect it. Our jaws dropped open and we looked at him, dumbfounded, as we explained that there was nothing to resurrect. He meant 'can it' in the theatrical sense, where movie films are put in tin cans for safe-keeping. Oops.*

Often explicitness means expressing things with the same terms used elsewhere in the user interface. For instance, instead of presenting a message that says: "If you insert a new index entry you will have to update all fields..." it should say "If you insert a new index entry you will have to use Edit/Update All Fields."

20

When computer programs are made, many new functions are created and must be named. In naming these functions, I've noticed that there seems to be a choice to be made between a descriptive phrase that spells out the purpose of the function, and a sort of shorthand nickname. I think that programmers have felt an obligation to name their features, rather than just describe them. The nicknames are almost always a disservice. Here are some examples:

- An imaging system has two different functions for attaching editors' comments to a scanned image. They are Notes (one per document) and Annotations (one or more per page). The user has to investigate the difference, then try to remember which is which. Instead, they could simply have been named Document Annotations, and Page Annotations.
- At the bottom of the File menu on many programs, the most recently saved files are listed, for convenient re-opening. In an INI file, this list was identified as the 'Pick List.' A more explicit name would be the Recently-Saved File List, eliminating the need for documentation explaining what a Pick List is.

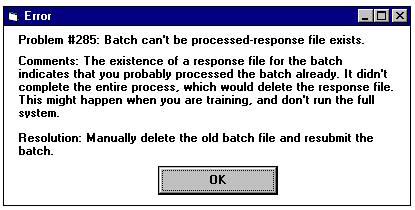
Who Does What?

Development managers: Have someone specialize in user interface design and weed out (excuse me, find and remove) implicit, metaphorical, indirect wording. Always use words that directly describe things in common terms or terms used elsewhere in the program.

Build Error Details Right into the Program

One system that I worked on had about a thousand error messages listed in the appendices of its 21 books, along with the causes and resolutions. But when users encounter an error, here's what they see on the screen:

You would then read a more detailed explanation in the back of the user guide, or just call support. Instead, the information could go right on the screen, like this:



The additional half-megabyte of text is nothing compared to today's software size, and it is infinitesimal compared to the time lost.

Who Does What?

Programmers: put error information right into the code. Work with the tech writers to add problem resolution info to your messages.

Big system purchasers: preview the documentation. If you see lots of error messages, find out if the information is displayed directly by the user interface.

21

Show the Level at Which Options Are Invoked

Most programs have many options that can be set to the liking of the user, sometimes numbering well into the hundreds. And each option has a sphere of influence. In other words, each usually affects only a specific range of actions or data, such as the file you are currently working on, or all files. The problem is that today's programs can have five to ten layers of technology and data, and users have a difficult time working with some features because it is unclear at which level the options are put into effect.

Consider Microsoft Word; here are just some of the levels at which options might work:

- The current window
- The current session
- All templates
- The current template
- All documents
- The current document
- All Microsoft Office programs (yes, some Word options such as spelling even affect other programs)
- Windows restart
- Power on/off

A great deal of wasted time could be prevented by simply identifying, on the screen, which level every option is associated with, and when the option is actually enforced if you select a new setting. Indicating the levels at which options are invoked is not sophisticated work. They can easily be indicated by providing words directly on the screen, grouping the options according to level, or explicitly describing the levels in context-sensitive help.

Part 2: Easy Programming Changes

This next group of techniques is not as simple as wording… but almost. To use them, none of the logic of a system must be altered, just some the arrangement of various items. Rarely do such changes impact the operation of the software, so they can be done with little expense.

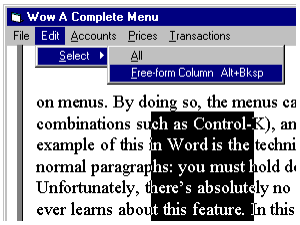
Put All Functions on Menus

This is probably the single greatest piece of advice I have for the entire computer community.

> The purpose of menus is to teach you how to use a system…

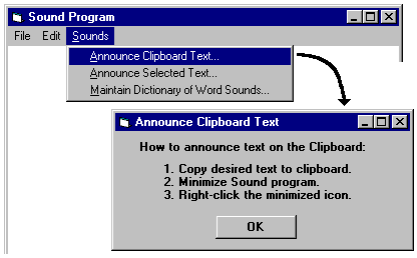
…not simply to enable you to invoke functions. Therefore, even functions that might rarely be invoked from a menu should be accessible and thus learnable from a menu.

In fact, even functions which *cannot* be invoked from menus should be placed on menus. By doing so, the menus can show you hot-keys (fancy key combinations such as Control-K), and mouse techniques, if appropriate. An example of this in Word is the technique for selecting a column of text within normal paragraphs: you must hold down the Alt key while dragging. Unfortunately, there's absolutely no indication of this on the menus, so no one ever learns about this feature. In this case the menu item might be setup as shown below, Edit/Select/Free-form Column.....Alt-Drag:



Selecting Columns in Word with Alt-Drag

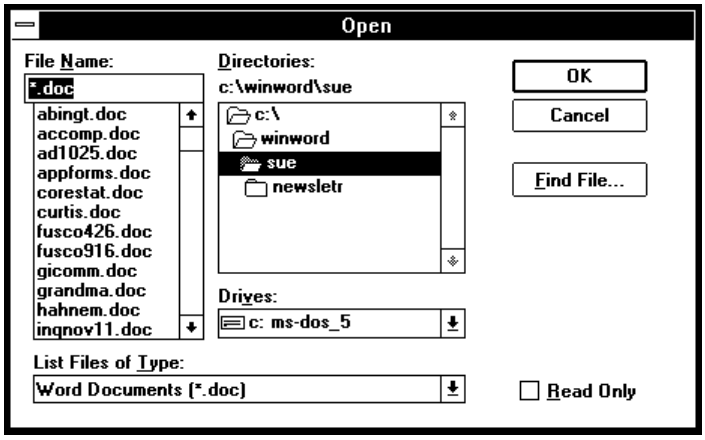
Even when *no key combination is relevant*, a dialog box can display brief instructions explaining how to invoke the function with the mouse. For instance, a sound program that used a contrived keystroke (right mouse clicking a minimized icon) to invoke a function. Even a peculiar technique like that could be made somewhat user-friendly by adding a menu function, perhaps as shown below: "Announce Clipboard Text." By simply displaying an instructive message, telling how to click the minimized icon, it will do the job:



Menu Function Even When There's No Key Equivalent

Display from the General to the Specific

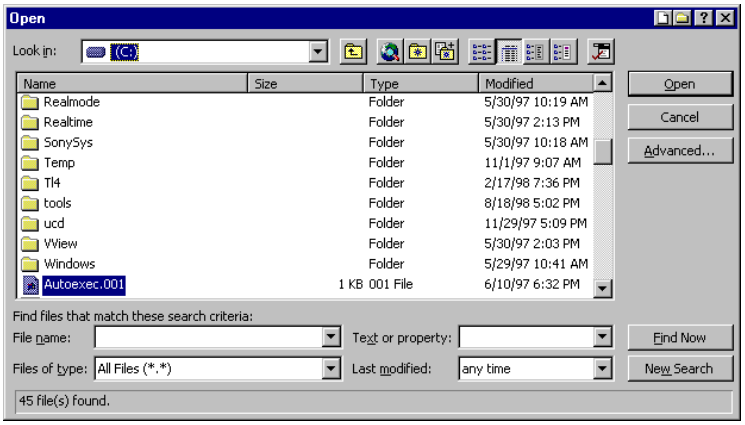
When several functions are provided in a dialog, present the most general ones toward the top and left. A classic example of violating this rule is the standard dialog box that Windows 3.x provides for opening files, shown below.



Incorrect Placement of Most General Item

In this dialog box, the most general selection, the one for choosing the disk drive, has been placed inappropriately below the other three. In this example, the problem might not seem too important, but as you find yourself using less familiar functions the design error becomes more apparent.

Here's the same dialog, redesigned in Windows 95:



Corrected Placement of Most General Item

Notice that another important, very general field, Files of Type, is still in the wrong place, at the bottom of the dialog. It should be above the file list because it determines what appears in the list.

I recall seeing a very experienced user misplace all of his template macros in Word. We ultimately determined that the accident was encouraged because of the same problem with the Macros dialog: the selection for the range of visible macros was below the list of macros, not above it. So he didn't realize he was saving the macros to a different template.

Put Functions Where Users Will Look for Them

Sounds easy, huh? But it's not.

It's a difficult to be sure that you're putting functions where users will look for them, not where they are most relevant in the eyes of the programmer. For instance, as we previously pointed out, in Word, the Print Preview function should be on the View menu with other viewing options, not on the File menu. And the Page Setup function should be on the Format menu with other format-setting functions.

You might even find it necessary to put functions in two different places on your menus. There's nothing wrong with this. A reasonable analogy could be drawn to rooms in a building: would it necessarily be the case that all rooms should have only one entrance? Of course not.

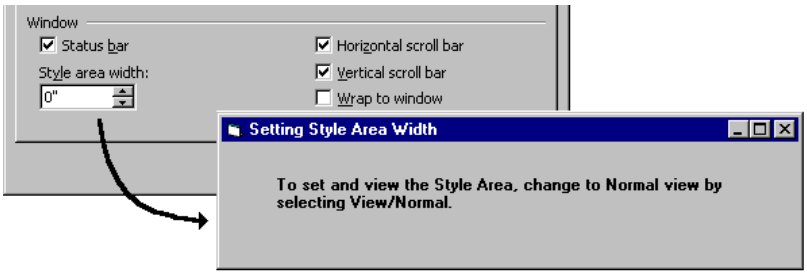
> Who Does What?
> Developers: when in doubt, ask a few people where to put functions; put them in more than one place; and make your menus customizable. Have a disinterested party make a recommendation.

Don't Change Menus and Dialogs On-the-Fly

Dynamic menus are those that change as different features become applicable, making the program something of a moving target, or guessing game. Some programs also change the appearance of items on dialog boxes, an equally offensive practice.

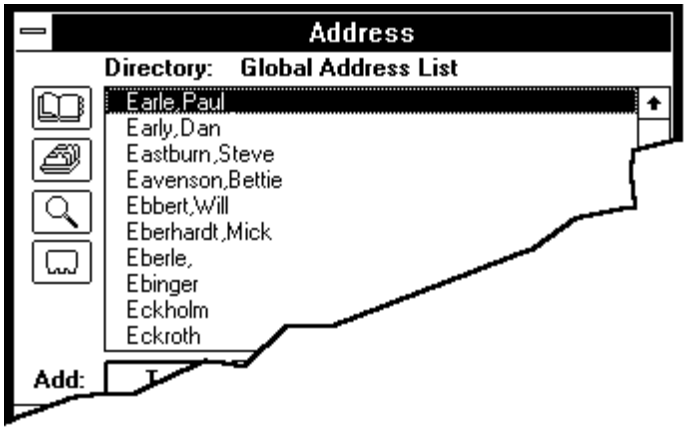
For instance, in Word, the Tools/Option/View dialog displays an item called Style Area Width, but in Page Layout view this item disappears from the dialog. This leaves the user disoriented, until you start poking around to find out why it has changed. The motivation for this technique is good but the effect is counterproductive because the system does not communicate with you.

In all cases, it would be better if the appearance of the menus and items did not change, but rather, behave differently if they must, perhaps displaying a message, as shown below. That's what usable design is all about—communication.



Always Provide Visual Cues

Whenever a program does something, it should tell the user or give some sort of visual feedback. A particularly subtle example could be found in an older version of Microsoft Mail, which used a clever but secretive method on its main addressee look-up list, shown below.

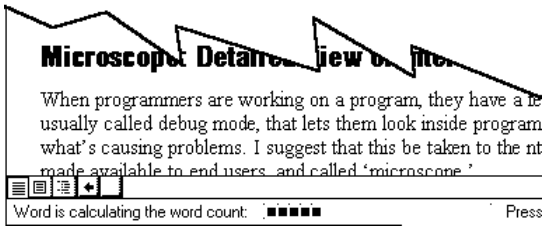


Here's what it does when you don't know a recipient's name, and you try to get it from the list: if you type the first few letters of the recipient's name, it goes deeper into the list, matching all of the letters you type (not just the first one). That's fairly conventional, and apparent as you watch the screen. But if you delay a few seconds before typing more letters, it starts over again as if you were typing the first letter of their name again. In other words, it "times out."

The concept is fine, but until you decipher what is going on you might go crazy, because there is no indication on the screen when it times out and discards the letters you typed. To turn it into a great feature they simply need to provide a text box that highlights the letters which currently comprise the search criteria; and when it times out, clear the box, indicating that the next letter you type will start things over again.

When the User Does Not Have Control, Announce it Prominently

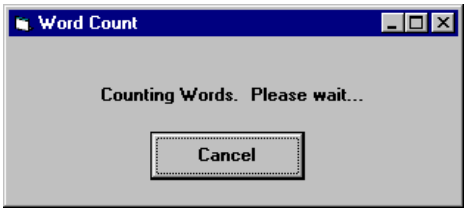
Often when using a program, you will encounter lengthy automated processes that keep the computer busy, while you must wait. For instance, if you tell Microsoft Word to count the words in your document, you will not be able to use any other Word functions (other than cancel the process) while it is busy counting. The problem is that most programs use the Status Bar at the bottom of the screen to tell you that the system is not paying attention to you:



Status Bar Message, Too Subtle When User Has No Control

Experienced users are learning to look in the status bar when the machine seems non-responsive, but the Status Bar is so subtle as to be unnoticeable... it has a passive presence, not an active one. The Status Bar is entirely appropriate

for reporting the mode you are in, file information, screen coordinates, and other continually changing data. But if the 'status' is so severe that the program is ignoring you, the message should be right in the center of the screen:



A Better Message When the User Must Wait

Part 3: Big System Buyers' List

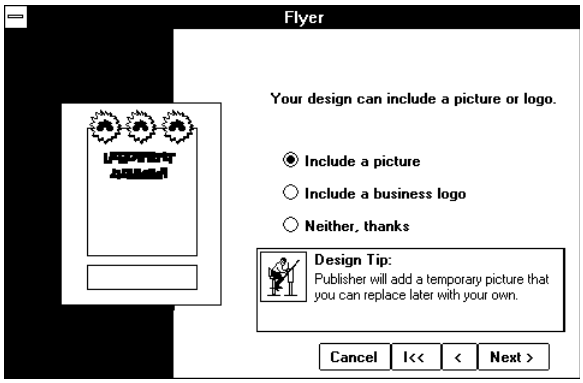
The following techniques take things to another level, and require accordingly more effort, but they're justified if you are staking your reputation on a major system.

Wizards: Step-Through Dialogs

Whenever a fundamental or important procedure requires more than two steps, provide a step-through dialog, also called a wizard.

Despite the cutesy name, wizards are not a trend or incidental development. They represent the culmination of the last 15 years of interface design. Don't underestimate them just because they are not applicable to all design tasks. Where they are applicable, they raise a software product to its highest, most profitable rung for both user and developer.

With a wizard, the computer does the maximum amount of work possible for a given task, and requires you to do the minimum. At each step, the screen presents either information or choices and asks you to make limited decisions. Here's a good example from Microsoft Publisher:



Who Does What?

Programmers: learn to always ask yourselves, "What is the most the computer can do to automate this task?" The wizard approach will be the answer for a surprising range of functions.

I'd like to suggest that we start referring to this technique by the explicit name of 'step-through dialog' rather than the metaphorical, implicit name, wizard.

Design for the Learning Period, Not Just "Production Use"

This rule concerns a problem that occurs during installation and training. Invariably there are features that only make sense after the system has real data in it, when the user is faced with actual circumstances. For instance on a transportation system with which I worked, there was no feature to browse for a railroad car number—you had to know the number of a car already on the system.

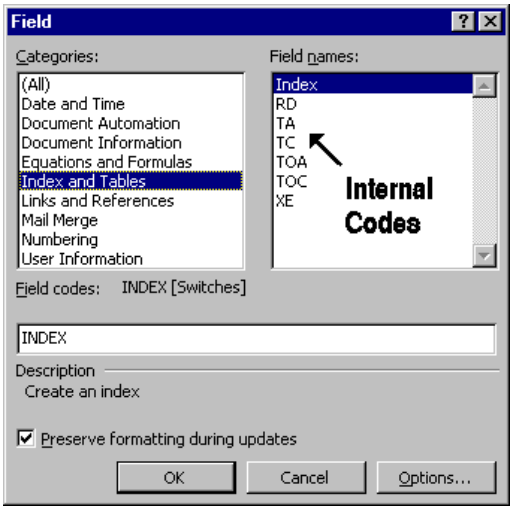
The absence of a browse capability was rationalized by the fact that in actual use, you would always be looking for a specific car whose number would be supplied to you by an outside party. But when first using the system, the inability to browse through the available data makes the learning process very difficult .

What can you do?
Stop rationalizing software design decisions based solely on 'actual use.' Allow in your design for those who don't know what they're doing, as well as those who do. Never presume what people will or won't know to do.

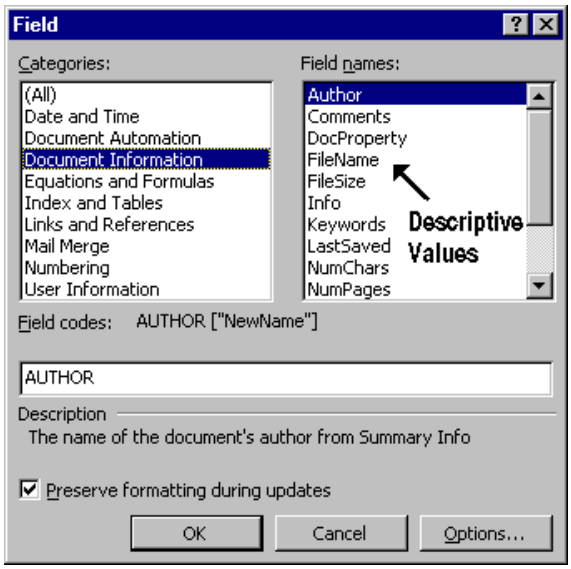
Let Users Work with Descriptive Values, Not Internal Codes

At their deepest core, computers deal with nothing but ones and zeros. Each successive layer of programming insulates us from this internal gibberish, but at the expense of programming effort. There's a constant tension between the need of users to work with friendly, descriptive terms and the starting point of all those ones and zeros. So there are still many places where internal codes sneak through into the user interface.

A nice example is in Word, where, right on the same dialog are two features, one using internal codes and the other using more descriptive identifiers. First the one with codes. Notice the Field Names list, with mostly meaningless codes such as RD, TA, TC, and so on.



Then look at the Field Names that appear when you select Document Information:



All of a sudden, the Field Names are meaningful, a good side-by-side demonstration of the difference between internal codes and descriptive values.

Provide Layer-Testing Diagnostics

Today's client-server systems have seven or more layers of technology. When a breakdown occurs, you will learn how few people can truly diagnose problems. But you might notice that they do various hit-or-miss tests to prove the viability of the layers.

For example, when toady's Internet connections go kerphlooey, some network expert may eventually type something like "ping 123.663.9.03 on your system to prove that you have a healthy TCP/IP layer. Another example: I had trouble with my modem, and the vendor told me to type in "c:\echo atdt1234>com3" to see if the modem was working, irrespective of Windows and the other software I was using. It failed the test.

The best you can do in this regard is to make sure diagnostic tools like this are available for every layer in your system, to quickly isolate the problem.

Who Does What?

Software Purchasers: make a written requirement that all major components must have independent, menu-driven, diagnostic tools that clearly establish the health of their layers. Include the delivery of these tools as milestones in your implementation and payment plan(!)

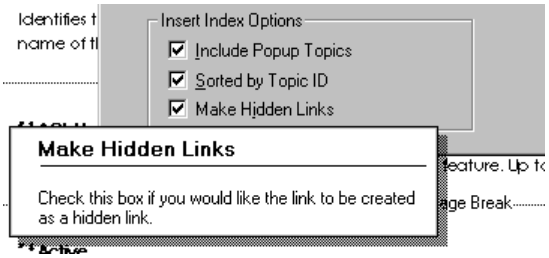
Help!

"Everyone spoke of an information overload, but what there was in fact was a non-information overload."
—Richard Saul Wurman, What-If, Could Be Philadelphia, 1976

It's no longer easy to tell where a program ends and its online help begins. This is a good trend and well underway, demonstrated by intro panels that can be disabled after you read them, and wizards that guide you through major procedures. As more and more procedural information is converted to these sort of user interface techniques, I predict that the only substantial need for printed documentation will be introductory concepts. The following ideas will help you understand world-class help, so you will know what you have a right to expect, and how to ask for it.

Provide Real Information

The starting point for good help is something that is surprisingly rare: good information. The following example shows non-information, also called a "circular reference." Notice the help text for "Make Hidden Links."'



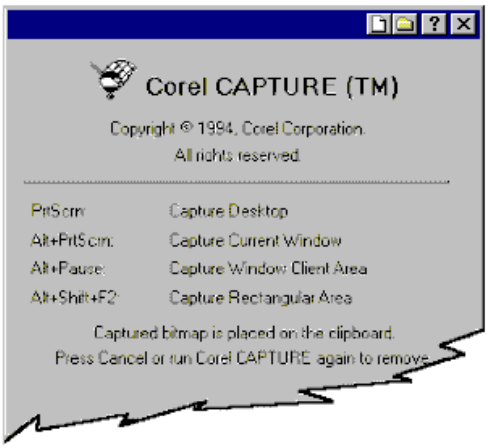
What, you may wonder, is a hidden link? A link is text that you click on to jump to another spot in a help file; a hidden one is not underlined, as links generally would be.

Your first goal in purchasing or creating help or other documentation is to insist on real information. Obtaining real information is hard work, so expect it to be more expensive.

Provide Help in the Program

Good programs actively communicate, taking the lead. No amount of passive, add-on information, whether printed or online, is a substitute for embedding the instructions right in the interface.

The following figure shows a good example of introductory help, from Corel Capture, a screen capture utility. This panel appears each time you launch the program.



Another similar technique is a message that appears the first time you use a feature, and offers you the option to disable the message thereafter.

User-Sensitive Help

Context-sensitive help is now commonplace. User-sensitive help is another evolutionary step waiting in the wings, especially suited to multi-user business systems. It adjusts the help based on username, and can be controlled or disabled entirely, at the option of the user.

Here's how it works. When you start using a program, it prompts you for your user name, suggesting the most recent username, if available. For new users, it defaults to display a lot of introductory help screens, as some programs do now, and offers you the option to disable those screens. Unlike the way that most programs implement introductory help, however, user-sensitive help would track what the user has been helped with and adjust the level of help accordingly.

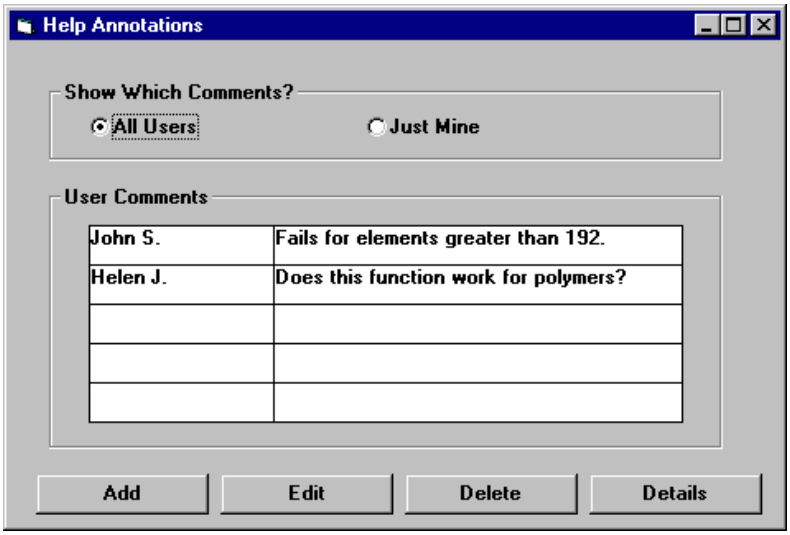
User-sensitive help tracks which functions users have accessed, and how many times. At appropriate intervals, based on this information, it alerts you to under-used features. For instance, after using a word-processor for months, you might get a message that you've never used the Styles feature, a feature that can help you establish consistency and productivity.

User-Contributed Help

This level of help is most useful in large, specialized business systems used by many technical experts (for instance, chemists) over a corporate network.

In such systems, it is unrealistic to expect that the software creators will be able to put genuinely valuable information in the documentation, online or otherwise. The literature will usually explain how to use the interface, not how to make the system work with your actual circumstances and problems. This is because the systems are developed faster and the application areas are becoming more esoteric. Accordingly, it is time for help systems to 'grow up.'

It's time to let users add to the help system, and enter questions and issues directly from the application and/or help screens. Today's routine Windows help already lets you add your own comments, called annotations, but these are stored on the user's workstation, so they are not shared; and there is no capability to send messages to the help desk, when information is lacking. The system I envision would have a button in the button bar of the help system, labeled 'Notes.' This button would bring up a dialog like this:



Users could add comments based on their experience, and declare the comments as public or private. They could also enter questions and issues that would automatically result in an e-mail to the help desk. This would reduce the vicious cycle of frustration that occurs when users can't find the information they need and can't get assistance quickly enough.

The most valuable of all talents is that of never using two words when one will do.
Thomas Jefferson

Documentation

Users don't want good technical writing,
they want answers!!!

Printed documentation and online help have their differences, but at the core, they share an important issue. The main thing that counts is not the appearance, accessibility, spelling, or even the technical correctness. The important thing is the value of the information provided. Does it answer the user's question?

This is the fundamental issue facing documentation—the value of information. Sometimes, value is established by providing an intense level of detail. At other times, it is established by better choosing which information to zero in on. And, as programs have become more friendly, value can even be increased by making *shorter* documentation. More on that later.

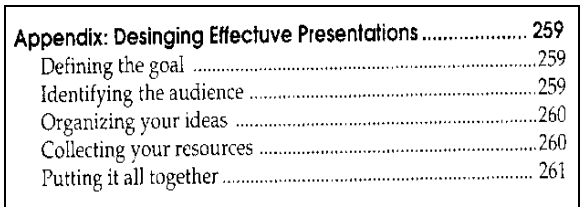
Who Does What?

Writers: spend your time writing about the things that you can't discover directly from the user interface: startup concepts, business tasks, non-intuitive and incomplete things, and problems.

Documentation department leaders: revolutionize your direction. Stop documenting everything. Isolate and target the genuinely valuable issues.

Don't Proofread, Prove

The following is an actual portion of a computer book:



If you're "desinging" documentation, don't fool yourself into thinking that proofreading makes it "effectuve." It only makes you feel better about documentation whose value you haven't proven. This is an issue of resource allocation. The more time you are devoting to profreading, the less you are spending on uncovering valuable, desperately needed information.

Working at a big company, I learned a lot about some style and grammar issues. I specifically remember learning to use 'might' instead of the more common 'may' when the issue was likelihood, not permission. ("You may perform function A or B. You might find function B more useful.") It turns out that there are very few instances in software documentation where 'may' is appropriate. I finally got it drilled into my head when I was corrected a thousand times.

Then I moved to a small company where the owner proofread my work. He 'corrected' every might to may.

What can you do?
Stop spending time proofreading your user documentation. Give the books to uninitiated users and ask them to try to follow the instructions.

Timing is Everything

Whether you like hearing this or not, the best solution to the problem is to write the books after the system is in use. In this way you can write about the real experiences of users… and include real world information and solutions.

Most businesses cling to the notion that the 'docs' must be ready as soon as the system is ready; this encourages gratuitous non-information—nothing more. In most cases, your initial user base is in too much of a guinea pig situation to bother with the books anyway, so don't ruin the books by enforcing what is really an artificial deadline. And besides, much of the writing effort done before the system is complete is very inefficient because of development glitches and changes.

Who Does What?
Documentation department leaders: start your serious writing after deployment. Spend 25% of your effort during development and 75% after deployment.

Print a Short Read-Through Guide

If you struggle with the need to print huge user guides for custom software, consider this strategy. Don't include everything in your printed matter. Instead, for each major system or module, print an introductory book of up to 75 pages. I suggest 75 pages as the maximum because that's the most that any reader is ever likely to read in a continuous effort, even over the course of a few days. This is based on my own surveying of computer users, and my own experience; I've never found anyone who has read the greater part of any user guide that is much longer.

I suggest that we start referring to this up-to-75-page introductory printed matter as a 'read-through.' The closest word we currently have is 'guide,' but there are guides of 1000 pages out there. No one reads them through! Delivering them on paper adds incremental value but considerable cost. Worse, they are a *disservice* because they hide vital concepts in a cloud of minutia that can be learned later.

Microsoft has figured out how to condense its literature. The following figure shows an edgewise view of the binding of a typical Microsoft technical manual that is approximately 1/4" thick. Can you guess how many pages a Microsoft book this thick might be?

Language Reference	Techy Volume 12

Microsoft's innovative approach—don't laugh—is to make the pages thinner. The book above would be about 250 pages.

For a large system, print a read-through for each key user: business manager, administrator, clerk, and integration programmer.

What's in a Read-Through?

A popular story is told in which Abe Lincoln once told the recipient of a letter, "Sorry this letter is so long, I didn't have time to make it shorter."

The most important information in a read-through is a conceptual introduction to your system, information that is key to understanding the way the system is meant to be used, often called theory of operation or how the business processes map to the system's functions. A read-through also should describe the major features of the system and generalizations about their use, particularly concepts that might be counter-intuitive. Remember, the whole point is that this book can be read in a sitting.

The read-through that I am suggesting is different than the minimal documentation that some packages now ship as a getting started booklet; in my experience, those booklets rarely discuss vital concepts or implementation considerations. Let's look at some examples of the type of information that should be in printed read-throughs:

- Issues of technique are important to put in front of users, rather than wait for them to seek it. For instance, I did a lot of sound editing recently. It took me a long time to realize that constantly readjusting the zoom level (how how many sounds you see on the screen at once) dramatically improved the efficiency of the cut-and-paste process.
- Microsoft Word stores its paragraph formatting information in the marks at the *end* of paragraphs. Every new user must be told this information. A read-through is the ideal vehicle.
- Corel Photopaint has great features called objects and masks—features that I need to use and which are central to its design—but whose use is unlike any of the simple bitmap editors I've used. An overview of the design premise must be accessible without wading through all of the reference information, as I had to do with the online help.

Training

The future belongs to companies
that train.

Why? Because anything that doesn't require training will eventually be done by machines, most of them computers in one form or another. And the machines themselves will not make good money, we think. We hope.

Shift to Productivity Consulting

If you're satisfied that your training is hitting the mark, don't change a thing. Otherwise ask yourself if your training approach has changed as substantially as your training predicament.

My recommendations for your training strategy have one common theme:

> Reallocate some of the time and money that you use for initial training to follow-up training techniques and productivity consultations.

Most training programs attempt to do all of the training at the initial roll-out of a new product. As a consequence, new users are often overwhelmed with details that they can't possibly remember, let alone use. And, typically, a few weeks or months later, many users are struggling with the system. Some cope, some stick to a limited sampling of comfortable techniques even if they're inefficient, and some simply get frustrated.

Today's computer systems require us all to be 'technicians' of a sort, whether we have any technical inclination or not. Even the most technically astute can benefit from an expert's tips and advice to get the most out of the technology. Most of us can even learn a lot about how to do our work from other users who are at an identical level, technically. That's an unavoidable consequence of the intricacy and newness of today's systems.

Another major factor is the constant, rapid bombardment not just of new features and changes, but whole new ways of working. The expectations we've had for traditional training are now unrealistic.

We need a whole new set of training habits…

Distribute Periodic Literature

Who Does What?
Trainers and Technical Writers: produce newsletters and brochures instead of big training guides.

In your training guides, cover only the core skills per role. Then create a one-page introductory brochure to highlight these key topics:

- New features and benefits of the system
- Special business techniques the system supports
- Speed techniques
- Items that are not apparent on the screen

Then follow it up periodically with a one-page, low-tech newsletter entirely devoted to productivity and information sharing—no birthday announcements. You might publish it as frequently as every week or two weeks during a very active deployment, and then scale back to every month or quarter as your system falls into a schedule of routine upgrades. Include topics such as these:

- Bugs
- Creative user's techniques
- Recommendations for enhancements
- Stories about great accomplishments with the system
- Questions from users
- Department profiles: how we use it; what's most valuable to us
- Pointers to shared templates, or reusable boilerplate content

Keep it short and sweet. If you feel you are stretching to create content, you are. Stop and publish what you have, or wait for more. Distribute it on paper if feasible, to encourage reading it in its entirety, over the course of a few days. If you must e-mail it, consider putting it in PDF format and recommending that recipients print it out.

Organize Peer-to-Peer Productivity Sessions

Almost every time I watch someone else use a computer, I seem to notice a time-saving technique they use that I've never thought of. *And I'm a user that constantly tries to use every shortcut I can find!* Imagine if you could take every user's most productive habits and share them… creating a productivity composite, of sorts. That's the point of this recommendation.

> Who Does What?
> Trainers: spend one day a week organizing sessions where individual users visit with other individuals for a half-day at a time, to simply observe them work on the computer system. Ask people to share with you techniques they discover, but don't make that a condition fro participation.

The most effective way to do this is to wait until users have established some competency with their new system and then set up the sessions. It's amazing the things you pick up when you see someone else do what you thought was the same work you're doing.

If you're concerned that one-on-one sessions might be unproductive, consider having small groups with common skill levels get together with a moderator to share thoughts and techniques, without doing any hands-on work. Users should be encouraged to bring their problem situations. When you collect enough valuble information, publicize it.

PC Hardware

Hardware's the fun part. In fact there are no hardware solutions, only funny stories.

I have no idea if this story is true, but fortunately, that's not a criteria for a good story.

> *As the story goes, Unisys had been supplying high-speed line printers—for all you youngsters, this is a printer that prints a line-at-a-time with a mechanical impact wheel—to a state transportation authority who used them to print vehicle registration cards. At one point, they improved the design of the printers, eliminating an annoying waviness to the lines of printed letters. Subsequently, the state police were upset because, prior to the 'improvement,' it had been easy to distinguish counterfeit cards- the print on counterfeits was perfectly straight!*
>
> *This is believed to be the origin of the tiresome computer saying, "That's not a bug, that's a feature."*

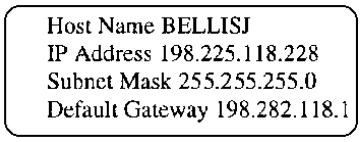
Write It Down

Your best protection from wasting time on hardware problems is labeling everything meticulously. And keep elaborate, organized written records. If communication is important in software it is doubly so with hardware. Every time you realize—too late—that you need a piece of information about your hardware that you didn't write down, you will pay dearly.

> *I got into a peculiar predicament with a laptop computer when I set it up to use a full-size monitor instead of the liquid crystal display (LCD) screen that is built into it. I worked all day with the laptop connected to the full-sized monitor. Then I turned it off.*
>
> *The next morning I disconnected the monitor, re-connected the LCD screen, and went off to work somewhere else. When I turned on the laptop, the screen was blank because it was configured to output only to the external (full-sized) monitor. I had to figure out what keystrokes to enter—blindly—to reconfigure it for the LCD. I believe the NEC tech support talked me through it. From then on I kept the following little blurb in indelible ink right on the frame of the LCD panel: SSALEEY. It means press the first letters of the words: Setup, Screen, Active, LCD, Exit, Exit, Yes.*
>
> *This was on a 4.77 megahertz monochrome, dual-floppy (no hard-drive) laptop in about 1989. I thought it was a bizarre, one-of-a-kind problem, but I eventually found myself in the same situation seven years later on a 120-megahertz, 1-gig, color, multimedia laptop. In the computer world, when you have the same problems with more powerful equipment, that's called upgrading.*

42

You'll notice that some heavily computerized offices have stickers on all of their PC's with critical information that is either not well stored by the system itself, or unrecoverable if the system fails. Most recently, this includes Internet addresses, host computer names, and mumbo jumbo like that:



I installed a system at a drycleaner, in Puerto Rico—the first time that I installed a new style of keyboard that my company had just developed. I couldn't get the new keyboard to work so I called 'the hardware guys' back home, who asked if I tried the switch on the bottom of the keyboard. I immediately reached under the keyboard and flipped the little switch, at which point the screen went blank, the computer power stopped, and the fan whirred uncerimoniously to a stop. Yes, I had fried the system. Sweat started pouring out of my face in buckets—bear in mind the average temperature in a Puerto Rican drycleaner is 150 degrees, even before system meltdown.

Meanwhile, the hardware guys are on the phone saying "You're not supposed to do that with the power on! Didn't you read the label on the bottom of the keyboard?" Duh gee, you don't say. I asked that in the future they consider putting the label physically on top of the switch, so that one must remove the label before ruining the system.

I spent the rest of the day driving frantically around Puerto Rico for a special fuse, which I actually soldered to the mother board to circumvent the existing one.

What can you do?
Label it, write it down, put it in your log book, keep it in your files.

Support and Troubleshooting

All right, so you've followed all of my advice and you've still got problems. Now you need technical support.

How many support technicians does it take to screw in a lightbulb?

Please hold and someone will be with you shortly.

Support is necessary evil, best described by the saying, "What doesn't kill you makes you stronger."

I handled some support calls at one job. My favorite call was one where we eventually tracked the problem down to the big red power switch on the side of the computer. It was in the off position. Honest.

A Beginner's Guide to Backup

The best support is the support you never need. And the best way not to need it is backup. The following advice is primarily for users who are relatively new to computers. If you've never lost *any* work on a computer, you are relatively new!

Computer backup has at its core a very simple tenet realized a long time ago when thrill-seekers would perform stunts on the wings of flying airplanes:

The first law of wing-walking: don't let go of one wing until you grab hold of the other wing.

All backup principles are merely incremental applications of this law.

- **Timed Backup** Save your work every 20 minutes. If your program has an automatic, timed backup, set it for 20 minutes. And even though you have an automatic timed save, still save your work manually every 20-30 minutes.
- **Multiple Names** Once a project consumes more than a day of work, save at least one older copy under a different name, still on your hard disk drive or network. Having only a single copy of your data on your hard disk drive is a prescription for misery because today's software is so complex that, on occasion, it might not be able to read its own data.
- **Multiple Media** After adding days of work to a project, copy your data files out of your computer to another medium such as a diskette or tape.
- **Multiple Sites** After working on a project for more than a few days, get your backup diskettes or tape out of the building. I keep copies of this book in my car, to protect against theft, flood, and fire, and randomly rotate new copies back and forth to the car or office.

Directory Assistance

If you've got a network that is not as organized as you think it should be, but don't know what to do about it, you're not alone. Networked directories or folders are more prone to disorganization because they don't physically reside in their respective departments. Since there is no inherent orderliness to a network, you must create and apply some rules. It's not too complicated, but often never happens. Here's the logic:

- Provide a special, network folder called the Transfer folder. Employees can use this to pass files to other users on the network, when they're too large for email. Make it understood that files are deleted from this folder after they are three days old. The existence of a transfer folder will give users the power they need from the network without having to give everyone capability to clutter up the network indefinitely.
- Assign an owner to the Transfer folder, whose responsibility it is to delete files in it older than three days.
- Every folder (or whole shared drive) on the network must have an owner. Only the owner can create folders in that folder. Others must request the owner to make new folders. This is the critical step that promotes organization. The new folders as well, must have only one owner, similarly charged.
- The owner of every folder is responsible for creating and editing a file, named README.TXT, in their folder, explaining the contents of the folder as a whole.

Your Own Knowledgebase

Your only protection against being caught in an infinite loop of troubleshooting problems is to keep a database of problems and solutions that is more powerful than the problems it tracks. The goal is not to solve every problem, but to make sure that no mistake consumes your time twice.

> *Fool me once, shame on you.*
> *Fool me twice shame on me.*

Make your knowledgebase in whatever tool is suitable to your needs. It can be as simple as a table in a word processor or a full-fledged database system, depending on the size of your organization, and more importantly, the likelihood that you can update that format as quickly as problems develop. Don't underestimate a table in Microsoft Word; it has great sort and search capabilites. Another very good option is Microsoft Exchange (mail), because it has folders and full-text search. At the minimum, include the following:

Mftr.	Prog.	Ver.	Problem	Cause	Solution
MS	Word	7 (95)	Getting styles from C drive unexpectedly	As designed, if no networked template found	Turn off Template/ Update From Template
MS	Word	8 (97)	Auto numbers are literal text	Converted from Word 7	Don't do that!

Ultimately, tracking problem information, like any information, is only as good as your administrative follow through:

- Assign an owner of the knowledge base.
- Publicize it to all users, actively and repeatedly.
- Make it everyone's responsibility to add their problems.
- Make it the owner's responsibility to improve and organize the information.

Technical Troubleshooting Primer

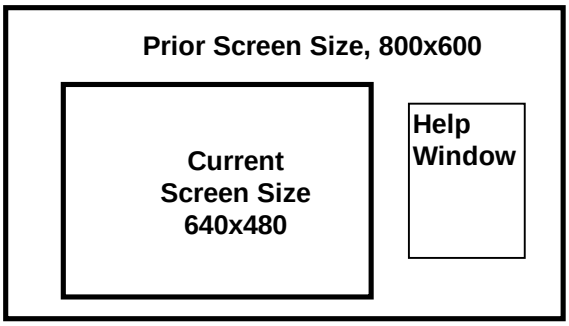
You can't become an instant expert in PC troubleshooting but it's surprising how much you can accomplish with some good technique. Here's a short course on tech support:

The First Law of Software Support: Reboot.

It doesn't matter what the problem was. Turn your computer off and start all over again. Even an exorbitantly priced consultant should probably suggest this first.

Dont overlook the possibility that the computer is actually doing what its supposed to.

Some of the hardest problems to diagnose are not even problems, sort of. For instance, I once had trouble with a Help window that would not appear. It seemed like it wasn't working at all. But it was actually 'displaying'—it was just too far off the right side of the screen to see. The problem was, I had recently changed my screen from 800 x 600 dots to 640 x 480 pixels, and the help window was still set to display at perhaps at the 700th horizontal position. It looked like the Help Window in this diagram:



When you are pounding your head against the wall, try to imagine that the computer is actually doing what it's supposed to. Think of things in a different way to accommodate this possibility, then work from that vantage point.

One of my more interesting support situations ocurred at at a Manhattan drycleaner. A little background, first. Often our customers would report just plain erratic performance, and there was not much we could do to diagnose it. Attempts at serious power monitoring devices were not too successful.

One time, while training the customer at this store, before my very eyes, 'garbage characters' streamed across the terminal monitor! It happened a few times before the store owner realized that it occurred whenever his old-fashioned cash register did its quaint 'kerrrrr-ching' as its cash drawer popped open. The motor in the register was generating interference that got through the power lines to the computer terminals. Finally, we had some reproducible, eyewitness corroboration of corrupted data due to electrical power problems.

Let your fingers do the walking.

47

Before you pull out a screwdriver to start swapping hardware parts, make sure you've exhausted all of the software solutions that don't require such rash measures.

Swap, baby, swap.

OK, now it's time for rash measures. Almost all electronic diagnosis these days is done by swapping out components, until the behavior of the system changes. With computers, this consists of swapping both software and hardware.

For instance, a co-worker once had trouble putting a Lotus ScreenCam display into a Microsoft Word document. We fiddled with Word, we fiddled with ScreenCam. Nothing we did would get it to work. Then we 'swapped' ScreenCam with another program, Windows Paintbrush; lo and behold, *not even a Paintbrush picture* could be pasted into Word, meaning the problem had nothing to do with ScreenCam!

Next we 'swapped' her PC for another, pasting her ScreenCam movie into another computer's Word document, via our office network, and it worked. The problem then was something peculiar about the setup of the laptop computer she was temporarily using for a presentation. It either had an incompatible program called a Dynamic Linked Library (DLL) or had unacceptable settings in a thing called the Registry.

This whole swapping thing is just the computer version of the ol' process of elimination. There's been no name for it, so I hereby name it repladuction, for replacement-deduction. If you must become self-sufficient with your computer, become an expert in this process.

End of troubleshooting course.

Questions to Ask About Support

If you're buying a serious business system, ask as many questions as you can about the vendor's support operation. Here are some suggestions:

- How many support people are there are?
- How many customers are there per support person?

  You'll have to decide whether the ratio is sensible for the type of application you are expecting to buy. The vendor of a point of sale system thought that 100 customers per support person was reasonable. At times it was; often it wasn't.
- What is the average response time to actually work on a problem, not just return the phone call?
- Do you have a call-tracking system to assure that all calls are returned and issues resolved?
- Do you accept e-mail for support questions?
- If they have a call-tracking system, ask to see the printouts of last week's calls.
- What is your procedure for enhancement requests?
- Ask other customers about their support experiences. Get referrals and visit them.
- Visit and sit in with support technicians for a half day.

WINNING THE BUSINESS SOFTWAR

With this simple booklet, learn why the computer world seems destined to keep you in a state of perpetual dependency, and how to counter-attack. Understand the forces at work, learn to recognize the problems before they're yours, and identify the parts of the battle that you can and can't win.

- If you're a project administrator, learn how to guarantee your project's success
- If you're a development leader, learn how to make your system a favorite among its user audience.
- If you're a programmer learn how easy it is to make your work attract attention instead of derision.
- If you're a corporate IT decision maker waiting for the long-promised productivity gains to appear at the doorstep, learn which door the grand prize is really hiding behind.
- And if you're *just a regular ol' computer user*, gain valuable insight and learn specific steps that can help you become part of the solution, and not just a casualty on the battlefield.