

Screen Writing: “Brevity is the Soul of Wit”

— published in *The X Journal*, November/December, 1996



Eric Schaffer, Ph.D., CUA, CPE, is Founder and CEO of Human Factors International, Inc. (HFI). He teaches, consults, and speaks on corporate and governmental Web and GUI interface design issues.



John Sorflaten, Ph.D., CUA, CPE, teaches and consults as a Project Director at HFI. With Eric, he initiated a usability curriculum at a local university in his home town of Fairfield, IA.

©1996, Human Factors International, Inc.

Brevity. Thomas Jefferson, the writer of the U.S. constitution, once wrote a letter to his friend John Adams. At the conclusion, Jefferson apologized for the lengthy discourse. He wrote that he lacked the time to frame his thoughts in more concise prose. While we in our so-called modern age may complain that “time is money,” Jefferson was in effect saying “less time is money” when applied to reading ease. It costs the writer much effort to package ideas so his reader would spend less time. How do developers handle this in their GUI designs?

We find that many, if not most, screens fail to communicate well. Screens often display too many words or too few words. Or the words fail to meet the users needs. It costs readers time (and money). There are reasons for these problems. They all stem from the wily work of our eternal foe, *cryptodesign*. Recall that cryptodesign manifests when a developer uses a design solution suitable for one problem, but misapplies it to another, quite different situation. Soul design takes different situations into account. Soul is the wit of brevity (to turn a phrase). Here’s how developers go wrong. We’ll use a metaphorical story.

DRIVING YOUR RENTAL CAR TO LA GUARDIA Imagine that you are driving your rental car in Manhattan. (Yes, we know most of you would take a cab, but let’s enjoy the metaphor.) Now imagine that you have only 90 minutes to get to the La Guardia airport. This is your first time to drive the route. You must struggle through lower Manhattan, race to the so-called “freeway” (clogged with cabs, trucks, and other hapless traffic victims), survive crossing the Triborough bridge, choose the correct exit to La Guardia, and turn in your rental car. Good luck, because 90 minutes is short. (Life is short, too.) If you miss the exit, you must continue on the freeway another 10 miles before you can turn around. **ONLY ONE CHANCE TO GET THE CORRECT EXIT!**

While driving, you look down to your map to identify the freeway. (You then look up to view the highway exit signs.) Down, up. Down, up. Down, up. Finally, you see the exit. Yes....here’s the one you need! You make the commitment and take the turn. You depended on clear highway instructions.

Then, after making the turn-off, what do you do next? When we ask this question in our GUI seminar, we invariably hear that drivers want to see confirmation that they chose the correct exit! Very interesting. What good is mere confirmation? You already made the turn. But nev-

ertheless, all of us need emotional closure on that demanding decision process. We not only need to know what to do, but also whether we did it correctly! What lesson have we learned? Good road signs both instruct and confirm our actions (see Figure 1).

The faster we travel, the clearer we want our signs. Whether on the highway or on the screen, we need to make decisions fast and without error. Cryptodesign fails to give us the signs we need!

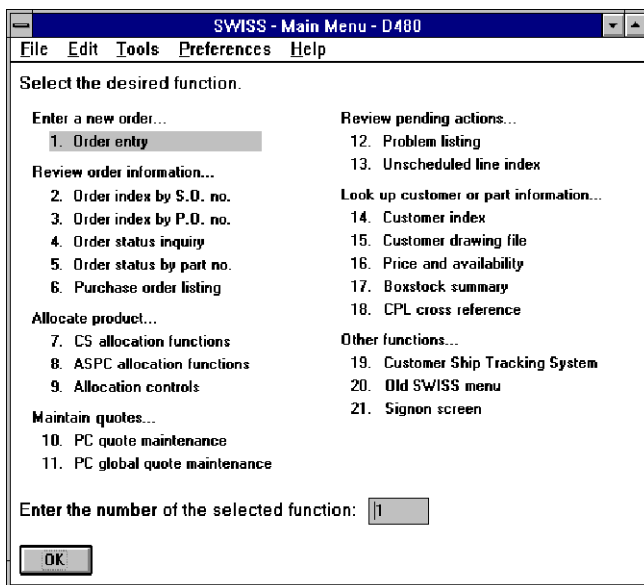


Figure 1. In this classical hierarchical menu, we orient the end-user with an initial instruction at the top. (Its a good road sign.) The 7 groups consume attention for reading. Therefore, at the bottom of the screen, we remind the user what to do. Notice that the grouping headers use a verb form to help clarify the available actions. (More good road signs.) We sequenced the options by the most probable task flow. These points all represent good screen writing.

CRYPTOWRITER SNEAK ATTACK Sometimes designers say they don’t want to include instructions on a screen because they “clutter” the screen for expert users. The designers say that new and infrequent users will get training, or figure it out and soon become an expert. Our reply--how many times do you want to miss the La Guardia exit before getting there on time? Why penalize the new and infrequent users at all?

What is the soul design antidote? For starters, psychology experiments show that experts can easily “gate out” or ignore those instructions. In fact, a definition of “expert” includes the skills to use a screen, often “heads down,” i.e., they may not even look much at the screen! They memorized the tab sequences. When you know the road, you ignore the road signs.

Next, and more important, the instructions serve as “just-in-time” training to new and casual users.

This cryptowriting sneak attack occurs because the application tools used by developers and analysts fail to provide good examples of just-in-time training. Word processors and GUI builders are terse, intellectually demanding, and focused on a limited set of tasks. Cryptodesign fails to appreciate the fact that most end-users will face a much larger variety of screens that service a multitude of different tasks. These end-users need on-screen road signs through insightful labels, enlightened grouping headers, and tactical instructions. Help systems fail to meet this need! (Research reveals that few end-users resort to help, even though developers may use it readily.)

We call good screen design “just-in-time training”. The soul designer solves the problem with brief instructions at strategic points. When space constraints get really tough, you might have to cut back on instructions. But quick usability tests verify how well you have placed road signs for the fast-driving user.

In our screen writing world, we need a balance. We must avoid too few words (it takes time to figure out the screen). But we must also avoid too many words (it takes time to read them all, see Figure 2). Let’s explore how the soul designer avoids being gored on these twin horns of the writer’s dilemma. In brief, as we’ll see: “Know thy users, for they are not you.” (Ask us for a free button with those words at HFI’s Web site.)

Let’s check out the types of work we can eliminate with good screen writing. Remember our VIMM model: Visual, Intellectual, Memory and Motor work can all be reduced.

THE TASK IS THE THING We often forget that writing constitutes part of “GUI” design. While writing is not “G—graphical,” it certainly is “UI—user interface.” Let’s review some crypto-pitfalls that we see many times.

Figure 2. This confirmation message demands a lot of reading. It’s overkill writing. Better solution: provide an “Undo” capability to your application. Yes, it’s more work, technically. But “Undo” saves the effort of writing and reading numerous confirmation messages.

Reduce Visual Work

Issue: As in Figure 2, “visual work” implies too much reading. Often this results from lack of experience in “tight editing.” Also, unfamiliarity with the language can make it difficult for the ESL (English as a Second Language) developer to use short, clear, and precise terminology.

Example problem: “As a member of the board of directors, you may wish to evaluate the feasibility of the merger proposal. The following section headings can be read and you may select any heading for more in-depth review. Then enter the num-

ber that is displayed to the left of the heading. Then press the key marked ENTER.”

Solution: Eliminate unnecessary words. If the reader knows what to do, leave it out. Example: “To evaluate the merger proposal, enter the number of the heading. Then press the ENTER key.” Some corollary soul solutions to visual work in screen writing:

- **Keep words and sentences short and simple.**

Research indicates that sentences over 20 words cause loss of reading comprehension with each additional word. Longer sentences raise the “reading grade level.” Use your grammar checker. Keep your writing at the 8th grade level or less. (These sentences are less than 20 words each!)

- **AVOID UPPER CASE.**

ALL-CAPITAL PARAGRAPHS TAKE 14-20% LONGER TO READ. TABLES ARE DIFFICULT TOO. ALSO, E-MAIL PROTOCOL SAYS ALL CAPS IS “SHOUTING” AND IMPOLITE.

- **Eliminate repeated words.**

They provide “noise” that submerges your message (see Figure 3).

- **Avoid paragraphs.**

Utilize modular writing techniques, step charts, and bullet lists (see Figures 4 and 5).

Reduce Memory Work

Issue: Cryptowriting makes end-users remember too much. What about “acronym acrominy?” Acronyms are the AKA (Also Known As) for a phrase or product. When the definition remains TBD (To Be Determined) by the user we have a communication GAP (Greatly Aggravated Problem). And what about CMOS, BIS, CMS, and ICBM?

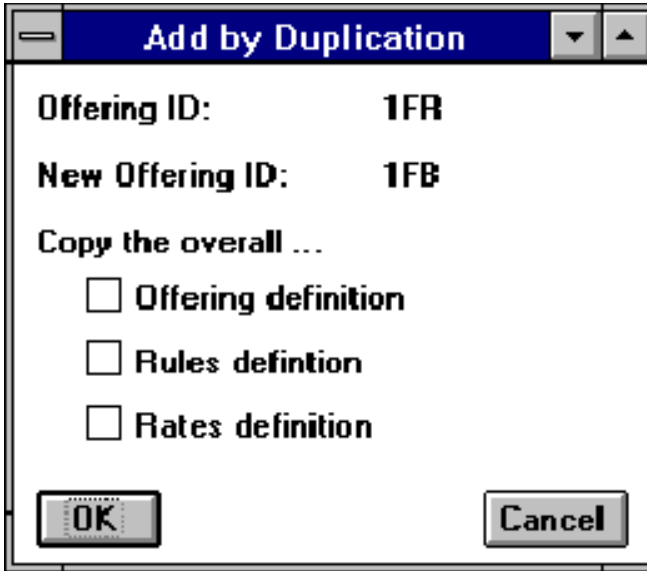


Figure 3. If you read the three check box labels, 50% of the 6 words is “noise.” Rewrite the header to say “Copy the overall definition for...” Repeated words are candidates for grouping headers, as well.

Avoid Paragraphs

Complex instruction:

If the order is equal to or under the customer's limits, approve it. If the credit limit is exceeded and the pay experience is good, approve the order. Otherwise reject it.

Figure 4. While developers may find this paragraph trivial to comprehend, end-users will find it difficult. “Know thy users, for they are not you.”

Avoid Paragraphs

If the order is...	And pay experience is...	Then...
Equal or below credit limit	→	Approve order
Above credit limit	Good	Disapprove order
	Not good	

Figure 5. Documentation, help, and even screens can benefit from diagrammatic presentation, or modular writing. Yes, it’s more work than sentences when creating it. But the time saved across hundreds of users when reading it can be worth the investment. Decision tables reduce reading errors, too.

Be Consistent

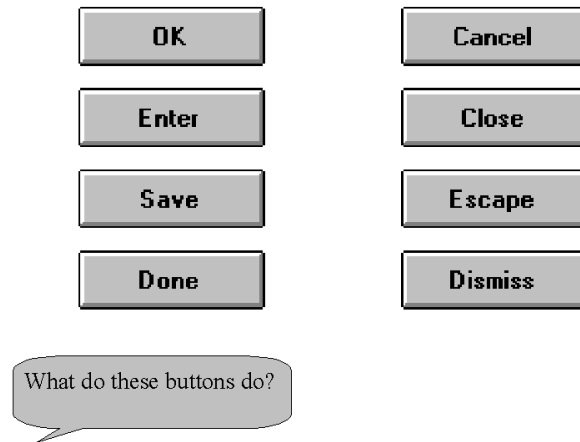


Figure 6. Your design team should stick to a single button name for “OK” and “Cancel” functions. Pick your favorite. Here are alternatives we have seen. Note that “Close” has a special meaning in Microsoft World. It indicates that the window disallows any edits, therefore, nothing can be “Canceled”. Subsequently, Close is not a universal substitute for Cancel.

Wording Inconsistency

Column 1 is called Column 3
 Column 2 is called Column 4
 Column 3 is called Column 5
 Column 4 is called Column 11 and labeled 1
 Column 44 is called 41 and labeled 4

Sheet on the screen is called Page on the Form

On the screen, the field labeled “FORM” does not refer to the field “Form Number” on the form. It refers to an unlabeled field (1). The error message refers to this as a “Form Retrieval Number” which is not the same as the “Retrieval Number” on the form.

Figure 7. An actual “read me” file. The screen terminology failed to match the associated paper forms. Therefore, the developer had to inflict this tutorial on the hapless end-user. Could you remember it when using the application?

Solution: Avoid acronyms. On the other hand, seeing “International Business Machines” may be equally surprising to the user who expects to see “IBM.” Therefore, stick with familiar acronyms like DOS, IRS, and BVDs!

Some corollary solutions to memory demands in screen writing:

- Maintain consistent terminology, e.g. use standard button names (see Figure 6).
- Avoid “read me” files! (They report discrepancies in the application.) Who remembers the details when trying to use the application? We have an actual case that defies comprehension (see Figure 7).

Reduce Intellectual Work

Issue: When you try to “figure out” something, you’re doing intellectual work. Computer jargon places intellectual demands on end-users. What does the system mean when it displays “Spooling to selected peripheral device?” Does spooling imply “threading” the printer output on the spiral

tracks of the hard drive? (Most of our GUI seminar participants think this is the correct interpretation.)

Solution: “Spool” is an acronym. Unix developers may know this more than others. It means “Simultaneous Peripheral Output On Line.” But note that most end-users have invented an inaccurate “mythology” to create meaning when they had none. What creative interpretations have you heard for “I/O Error,” “Syntax correction,” or even “Start date of the range?” Therefore, if end-users are not computer specialists, avoid computer jargon. It’s too demanding, and can result in superstitious mythology.

Here are some corollary solutions to intellectual demands from the written word:

- **Acknowledge the user’s jargon.**

“Knowing thy user” means understanding any special meaning for given words. Know the user’s mental model. For instance, “tank” means “gas tank” to an auto service station attendant. However, to an infantryman, “tank” means an armored, tracked vehicle with a big gun. Recall that General Motors failed to get a following in Spain for its top rated car, the Chevy “Nova.” Executives failed to realize that “Nova” in Spanish means “no go.” Would you buy a car called the “No Go”?

- **Use the active voice.**

For example: “Enter the number of sales contracts.” Avoid passive voice, which uses some form of the verb “to be” such as “is, was, were, will be, should be” etc. For example: “The number of sales contracts should be entered.” Notice that passive voice sounds optional. Plus, it takes a second for you to realize the instruction wants you to do the work! Government regulations written in passive voice contribute significantly

to bureaucratic inertia. (“What, you mean I should take that action?”)

- **Match the message tone to the user and the environment.**

We don’t need “please” and “thank you” in system and error messages. This advice runs contrary to the MacIntosh standard of error messages announcing “Sorry, the xxx just yyy, and you lost everything.” But note that the MacIntosh aimed for the home market. “Please” and “thank you” served a role in bridging the gab between novice and hi-techno gizmo. In our corporate environments, however, such politeness becomes self-consciously cute and “precious.”

In contrast, what about interfaces such as an ATM (Automatic Teller Machine)? As a substitute for a “real” human, we believe that ATM can justify using “please” and “thank you!” Know thy user. Know their situation, e.g, anthropomorphism is great for kids programs: “Gosh, your hands are cold.” But imitating humans remains inappropriate in corporate settings: “Groovy. I just sent a \$15,000 check to your nice client.”

Reduce Motor Work

Issue: What possible motor, or physical work, could screen writing cause an end-user? Consider the work of typing commands or values, such as a menu option. Yes, we could use a mouse click. But consider that reaching for a mouse costs the same time as required for 3 to 8 keystrokes! Thus typing can be useful. However, beware of cryptoabbreviations. In general abbreviations cause typing errors because writers fail to use a consistent abbreviation scheme (see Figure 8 for an example).

Solution: If the end-user must abbreviate, use dependable knowledge. That is, when the user abbreviates input, then require truncation. Require a consistent number of letters. For example, to

Abbreviate With Caution

ADDM	Addendum <u>M</u> od
ANIM	<u>A</u> NI Sub- <u>M</u> enu
BILC	Chgn <u>B</u> ill Info
BILT	<u>B</u> ill Contact <u>M</u> enu
CACC	<u>C</u> hgn <u>A</u> ccounting CD
CHGM	<u>C</u> hrg/ <u>C</u> redit Sub- <u>M</u> nu
CHNG	General <u>C</u> hange
CCTA	Add <u>C</u> ircuits
CKTO	Disconnect <u>C</u> kts

Figure 8. This is merely one-fifth of a main menu from a telecommunications firm. It displays abbreviations so users would not have to re-learn associated menu option numbers whenever new requirements changed the menu. The pattern appears to use the first three characters of the first word, and the first character of the second word. However, the scheme fails to remain consistent. Users had to use brute force memorization! They would have been better off using numbers for the menu options.

enter “append,” the user could be required to enter the first 4 characters “appe.” The user should know the correct spelling, and thus avoid errors that require motor work for their correction. Similarly, to enter “execute,” the user would type “exec.” When the user must read the abbreviation, use a different method. Use vowel deletion, in which “append” appears as “appnd,” and “execute” appears as “exct.” Obviously, keep vowels when needed to interpret the word! Figure 8. This is merely one-fifth of a main menu from a telecommunications firm. It displays abbreviations so users would not have to re-learn associated menu option numbers whenever new requirements changed the menu. The pattern appears to use the first three characters of the first word, and the first character of the second word. However, the scheme fails to remain consistent. Users had to use

brute force memorization! They would have been better off using numbers for the menu options.

Some corollary solutions to motor demands in screen writing:

- **Use auto complete.**

It’s an extension of the truncated abbreviation technique; use drop down lists, combo-boxes, and searching scrolled lists. Auto complete displays the best match to the character string you type within the time-out period. The Win95 control panel uses it. Unfortunately, Win95 fails to provide it for both combo-boxes and lists. Auto complete speeds input up to 100%

- **Keep typing demands within familiar experience.**

It’s better to require a few extra keystrokes in order to make a meaningful connection between the action and the command word. That is, keep the mnemonics meaningful. For example, why learn the UNIX command “grep” to find a file, when the command “search” says it all? Likewise, why risk typing errors with entering “E049a” when typing “error” can get the same response from the system.

CONCLUSION Brevity. Cut crypto-obfuscation. Take the time to write simple, concise prose. Yes, time does cost money. But when it’s spent on good screen writing, your development money saves “big time” for your end-users. And remember, soul design is the wit of brevity. (Apologies to Shakespeare.)