

Pull-Down Menus: Out of Sight, Out of Mind

— published in *The X Journal*, November/December, 1995



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.

This second front-line report from the GUI revolution focuses on a particularly wily cryptodesign foe: pull-down menus. Recall our definition of the developer's eternal foe: cryptodesign. These are decisions that worked for certain situations, but are often misapplied in different, inappropriate situations. Pull-downs are the "guerrilla" combatants of GUI design—so named because at one glance they look like good-guy civilians, but in another moment, they've wreaked havoc on ease-of-use. Let's explore how to neutralize these design sapper bombers.

PULL-DOWNS ARISE FROM A METAPHOR Where do pull-downs come from? What is their *raison d'être*? The good-guy civilian persona for pull-downs arises from their stalwart role in our word processors, spreadsheets, and our drawing tools. Pull-downs resolutely support GUI applications that require large empty areas on the screen! In document processing, drawing, and spreadsheeting (!) pull-downs support a metaphor—namely the blank page. It's the *tabula rosa* for your creative work. To provide space, we hide commands from view under pull-downs. Yes, we don't mind the extra visual, intellectual, memory, and motor work that pull-downs require. Because in return we get a lot of nice, clean, empty screen real estate.

CRYPTO PULL-DOWNS SNEAK ATTACK The guerrilla persona of pull-downs sneaks into the works when we forget that the blank page is a metaphor. Obviously, a blank page is not appropriate for every application. Yet, we have seen many corporate data-entry applications open up with a "blank" first screen! The sneak attack starts here. The developer and product manager must have decided that pull-downs should be used regardless of the task. Why else would we ever start with a blank screen? Look at the extra work: to get started, the user must access the File pull-down option, then access the Open option to get going. We just got sapper bombed! Cryptodesign infiltrated and did its nasty work.

That was just the beginning. In most corporate data entry or data viewing applications, pull-downs make navigation difficult. For example, an international firm called us in after suffering a competitive beating because clients got lost in their pull-down menu structure. Their users were highly paid stock traders who used their PC to locate and follow real-time stock prices. And they didn't want to take time to "learn" the application—they were too busy making money. Cryptodesign easily did its nasty stuff. Here's how.

In the heat of work, users had to chose from six major functions for the application, selecting one from the left-most pull-down menu (Activities). Certain other menubar names changed according to the current Activity selection (see Table 1). Did users get confused? You bet. It felt like operating a slot machine; each of the six Activity options displayed a different menubar of cherries, apples, and dumbbells, etc. in no easily predictable fashion. Even worse, note that pull-down options under the Choices and Setup pull-downs changed from one Activity to the other, even though the menubar names remained the same. Users asked “how do I get back to where I was?” They had forgotten how they created the menus they had just used. The application placed significant demands on memorizing the relationships among the menu and pull-down options. Table 1 shows what users saw for each of the six options under the Activities menu.

TOOL BARS AS CRYPTOSOLUTIONS Research shows that users dislike searching for hidden com-

mand options. Therefore, word processors use toolbar icons to represent options in an attempt to make the hidden pull-down commands visible. (This is direct evidence of the pull-down problem!) Meanwhile, we must avoid the cryptonotion that a toolbar ribbon in your transaction-oriented applications solves the problems of pull-downs. While toolbars serve well in applications that need a blank page, it creates problem for other applications. First, the icons are difficult to interpret for new or casual user of a transaction application. They at least need labels. (Have you memorized all your word processor icons yet?) Second, in corporate applications, we often find navigation icons mixed with other icons that merely change settings or provide ancillary functions. Thus, users lose contextual clues for understanding functions at a glance.

THE TASK IS THE THING Can we distinguish the guerrilla pull-downs from civilian pull-downs yet? Let’s look at some guidelines for correct use of

Table 1: Example of a pull-down menu that confused users and lost market share.

Activity (Prices "on")	View	Layout	Pricing	Setup (options M, N, O, P, Q)	
Activity (Pages "on")	View	Layout (options A, B, C)	Choices	Inserts	
Activity (Charts "on")	View	Layout	Screens	Time	Choices (options U, V, W,)
Activity (News "on")	View	Layout (options D, E, F, G, H)	Choices		
Activity (Tickers "on")	View	Layout (options I, J, K, L)	Choices		
Activity (System "on")	View	Layout (options R, S, T)	Setup		

pull-downs in corporate applications that handle data entry or data viewing. Let's use our VIMM model discussed in last month's column.

Reduce Visual Work

Issue: With pull-downs, users must play hide and seek with the option they want.

Solution: Keep frequently used actions on the screen, probably as push buttons, secondarily as labeled toolbar icons. Users can let their eyes do the walking faster than their fingers can explore pull-downs! If you are really tight on screen space, the best candidates for pull-downs are system-wide functions that are used infrequently. Furthermore, since users navigate frequently and need to stay oriented to where they are, really really really try to avoid pull-downs for navigation.

Reduce Intellectual Work

Issue: Pull-downs use shorter phrases than can be placed on push buttons. This makes them harder to understand. Also, since pull-down or toolbar options are lumped together, they are not associated with the screen task flow. You may have to figure out for yourself where the action fits in the task flow.

Solution: Put the options on push buttons. Buttons and menus can handle longer phrases than pull-downs. You get clear communication for casual and new users. You can position the push button on the screen at the place the user needs it, matching the task flow as it goes in a left-right, top-down pattern. If a command appears on several screens, try to design a task flow that allows a consistent position for the buttons.

When using pull-downs, ever change menubar names! (Play the slots with a slot machine, not a menubar.) If an option doesn't apply in a particular situation, deactivate it by dimming it rather than "offing" it.

Reduce Memory Work

Issue: Since pull-downs hide options, users must remember to remember they exist! Research demonstrates that users forget rapidly and often fail to search under menu options for ideas!

Solution: Keep your functions visible. (Call us for our inspirational button "If the user can't find it, the function's not there." 1-800-242-4480.) At the same time, hide functions until they are needed. Don't overwhelm the new or casual user with choices not required yet. (Experts can handle them better.) If you're plumb out of screen space, candidates for pull-downs include system-wide functions. Functions used least frequently are your first choice for pull-downs.

Reduce Motor Work

Issue: Using a pull-down menu requires two mouse clicks. One to open the menu, another one to select the option. Furthermore, exploring a set of menubar options requires a lot of "mousing around."

Solution: A push button, or a classical menu option only requires one mouse click and no mousing around.

SOUL DESIGN STRIKES BACK We haven't given up on pull-downs! Here are some ways to set the cryptocritters straight. Let's assume you are designing an application that has no need for the blank page metaphor. Instead, the task has an identifiable task flow with task modules. Here are some options that follow the VIMM model given previously.

Context Switch In Figure 1 we see that buttons make navigational options obvious to the user. This approach guarantees your users will know where they are and how to get there at all times. Yes, we kept some pull-downs for functions that

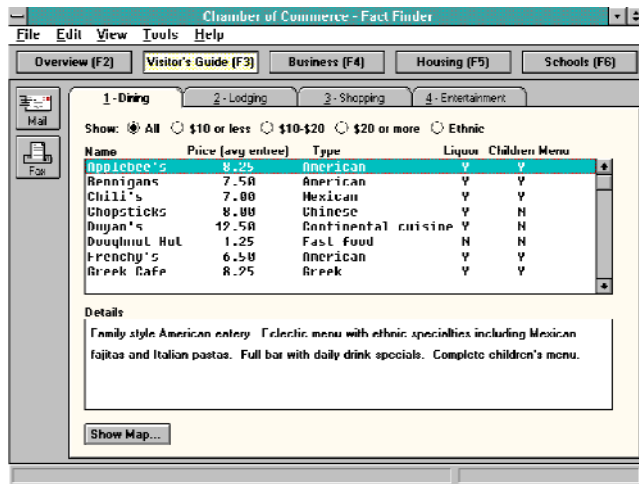


Figure 1. Context switch.

were system-wide as well as infrequent. Note that the screen below the buttons “swaps out” one display for another. This solves the problem of window thrashing and window pile-up that we discussed in last month’s column. The user avoids the motor work of selecting “OK” or “Cancel” to close one window prior to opening another. The folder metaphor extends these same benefits to a second level of menu.

By the way, Windows 95 uses a “task bar” similar to the context switch. It’s not a bad idea. However, cryptotemptations abound. For example, don’t imitate the cascade menu that sprouts off the “Start” button. Cascades are physically hard to use. Microsoft even recommends against them (see page 134 in *The Windows Interface Guidelines for Software Design* by Microsoft, 1995). Also, keep all the buttons visible and in a fixed sequence (unlike the Win95 taskbar). Users develop a gut-level feel where items show up. Don’t disappoint them.

Controlled Task Panel In Figure 2, we show a modification of the context switch for cases where all transactions belong to a “key object”. In this case, the key object can be referenced as a name, telephone number, or an account number. (The

entry field changes according to the selected radio button.) We call this a controlled task panel because the task buttons and important data associated with the key object remain visible on the panel throughout the application. The panel data must be justified by its usefulness to the user.

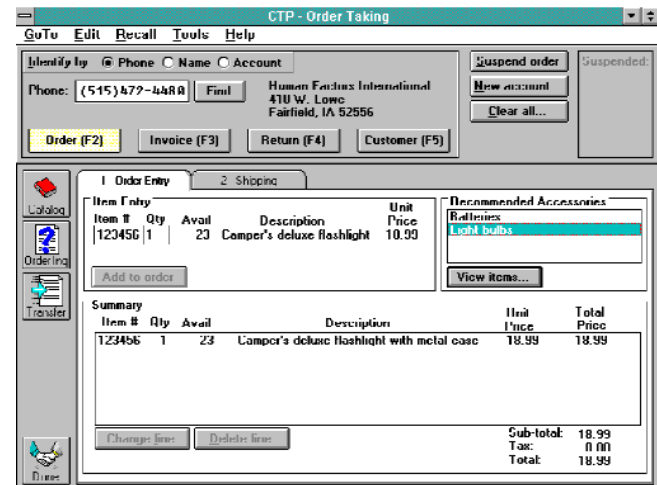


Figure 2. Controlled task panel.

Examples of use include telephone customer service and order-taking applications. Note that the icon buttons on the left can change according to which button module is active.

Hierarchical Menus Alive! In contrast to the two UI architectures given above, some task flows are very predictable. The user may enter data into a structured sequence of screens, such as three screens used for taking a bank loan application. When that task is finished, the user may want to accomplish a different task, with a different set of screens, and so forth. In such cases, we provide a “hub” for accessing the first screen of the various sequences. See Figure 3.

The soul-design alternative to pull-down navigation is the “classical hierarchical menu”, shown in Figure 4. Notice we’ve given it a modern flavor by including business statistics in the various bar chart status indicators. The user keeps in touch with the pulse of the business every time he or she returns to

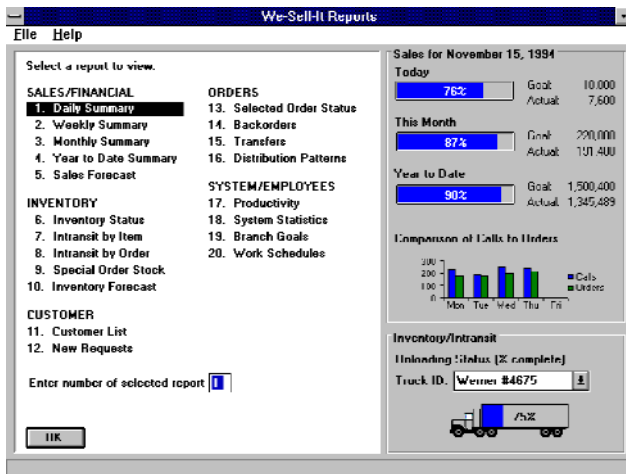


Figure 3. Predictable task flows with “hub”.

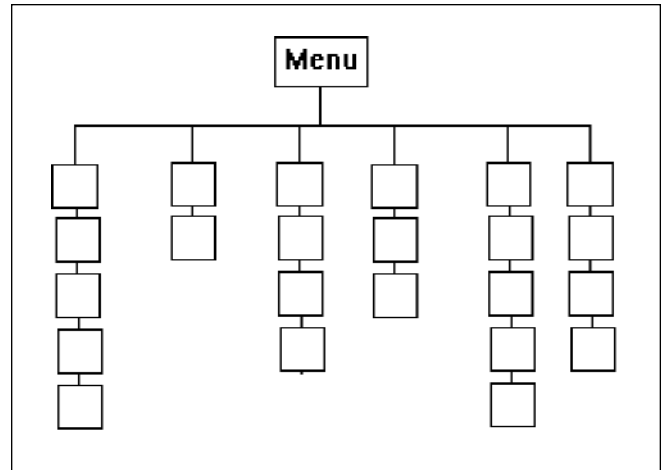


Figure 4. Classical hierarchical menu.

the hub, or menu. Human factors research gives us some rules for such menus. No more than six groups. No more than ten items within a group. And no more than 18 to 24 menu items.

WRAP UP AND CONCLUSIONS Beware of pull-downs. Don't use them indiscriminantly because you'll sapper bomb your users with extra VIMM work. If you are able to design a clear task flow, you can decide on a decent alternative such as the context switch, controlled task panel, or a classic hierarchical menu. These design alternatives may support tasks that your organization uses a lot. If so, consider using these as templates or guidelines in a standard.

Are there emotional issues surrounding pull-downs? Well, yes. Some developers ask why we

recommend hierarchical menus when we're in a GUI pull-down world. We reply that we want to keep life simple for users. We can justify a classical menu when the task flow uses linear sequences of screens plus the “hub” concept. Pull-downs are not simple to users. Meanwhile other developers may demand that pull-downs and icons carry the major weight of navigation in the name of “object-oriented interface design. In reply, we suggest that designers want to speed user comprehension and minimize learning time by providing clear UI structures. Thus, soul design seeks UI structures that combine objects in manner that suggests a useful task flow or object relationship. If we introduce these structures, then we automatically move in the direction of the design alternatives illustrated above.