Diagnosing Sick Screen Syndrome

The same thing that makes VB so attractive lowers the developer's resistance to the 12 Deadly Developer Diseases. Education is the most potent vaccine against this scourge that results in gooey GUIs.

BY KEN SCHIFF

These days you have to be careful. New development tools give us tremendous flexibility, ingenious capabilities, and fabulous power. But they also enable us to create more garbage—a lot faster and in higher resolution. A graphical user interface carries no inherent guarantee of usability. Many developers seem to believe that the GUIer, the better. But caution is necessary. From processor time to programming time to configuration time, misguided GUI designs frequently end up wasting system and human resources rather than enhancing productivity.

To the end user, the interface is the software. A less-than-usable front end can totally scuttle a brilliant back-end design. Historically, interface design and usability have been poorly understood, and as a result, have had the fewest resources allocated to them. Driven by the high expectations and demands of today's users, interface design and usability are becoming critical components of the development process. Visual Basic is a powerful tool for the creation of graphical user interfaces. However, there is no substitute for understanding visual design principles and Windows conventions.

In fact, after several years of intensive research, I have identified twelve Deadly Developer Diseases. These contagious maladies are found throughout the worldwide developer population. Although symptoms vary widely, these diseases all seem to be caused by the same thing: a lack of training and experience. The disease manifests itself ultimately by hindering the developer's ability to design a usable, logical, and attractive user interface, regardless of how hard he or she tries. The best cure is prevention, through self-administered awareness, discipline, and education (ADE).

Awareness is frequently innate or instinctive, but it can be acquired. Discipline is self-imposed, although feedback from others can help keep you on the right track. Education is an ongoing process and can come from classes, seminars, and reading. I'd like to provide you with some background you'll need to start recognizing the symptoms of the Deadly Developer Diseases. Consider this article an intense lab session, a sort of DDD 101. In the name of patient confidentiality, I've removed the names of all the applications used in this article's figures.

Ken Schiff is president of Productivity Through Technology Inc., a consultancy for creating obvious, discoverable, and usable software. Ken's clients include Fortune 100 corporations, large commercial software developers, small ISVs, VARs, and system integrators. Reach him on the Internet at kenguiguy@aol.com, or by fax at 510-471-2402.

HYPERSTRATAMANIA

This disorder's principal symptom is the excessive use of 3-D controls on the screen, to the extent that available space for data is considerably reduced. Sporting panels with three layers chiseled in, four layers raised out, and buttons with half-inch high elevations, some screens can be mistaken for Arizona's geological formations. Some applications even have menus, drop-down menus, and labels with text that rival the *bas relief* of the ancients (see Figure 1). Using 3-D sparingly will cure Hyperstratamania.

HYPERLAYERMANIA

Have you ever seen a screen with five or six layers of *modal* dialogs? Some programs use so many modal layers that you forget what you were doing when you started opening the dialogs. Because they're modal, you can't move any of the previous layers out of the way to see the parent window (see Figure 2). That's Hyperlayermania at its best (or worst). The cure is simple: limit the levels used on screen.

CLAUSTROPHOBIC OBJECTOSIS

This syndrome is characterized by too many objects on the screen, giving the user that crowded feeling (see Figure 3). White space (or in the case of contemporary Windows applications, gray space) is important. Ease of use varies inversely with the number of widgets on the screen at any given time. Don't be afraid of using blank space: it can open up the claustrophobic interface dramatically.

ACUTE ICONITIS

The most prevalent of the diseases, Acute Iconitis is manifested by placing too many icons on the screen. From different toolbars for each child window to multicolored graphics on each command button, this disease is frequently accompanied by a major misunderstanding of visual design (see Figure 4). You can cure Iconitis by limiting the number of icons displayed at any one time.

HYPOTHERMAL PSEUDOUSABILITIS

This disease has reached worldwide epidemic proportions in the developer community. Programs are being designed to look "cool" or "sexy," with total disregard for usability. While the UI may look great, it may not make tasks any easier. In the words of interface designer Don Norman, "The best interface is a transparent one." Rather than saying, "Hey, what a great UI," the best compliment from a user is no comment at all. Just seeing a user sit right down and get to work without any overt awareness of the interface is the sign of a healthy application.

RANDOM HYPERMULTIBUTTON SYNDROME

This disease is manifested by an excessive number of command buttons, grouped and positioned in various different alignments, and confused by the use of 3-D panels to simulate secondary command buttons (see Figure 5). This is typically cured by using tabbed dialogs and other types of containers.

POLYSYLLABIC TECHNOJARGONITIS

Programmers speak a different language than the average user. When faxing, they *rasterize* rather than *prepare* the cover page (see Figure 6). Remember who your users are. Regardless of the conversational language they use, keep in mind that they are not usually propeller-heads. Use words that *they* will understand.

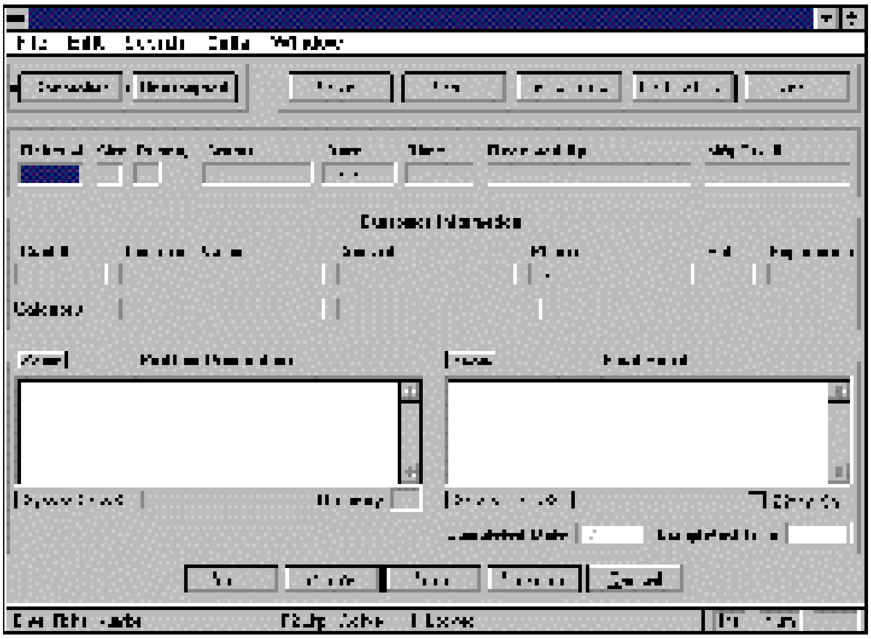

FIGURE 1 ***Symptoms of Hyperstratamania.*** *This dialog has three layers facing "in" and four facing "out," with nearly no room for data. Overusing 3-D creates a Grand Canyon of form fissures; use 3-D only when it enhances usability, not simply because it's available.*

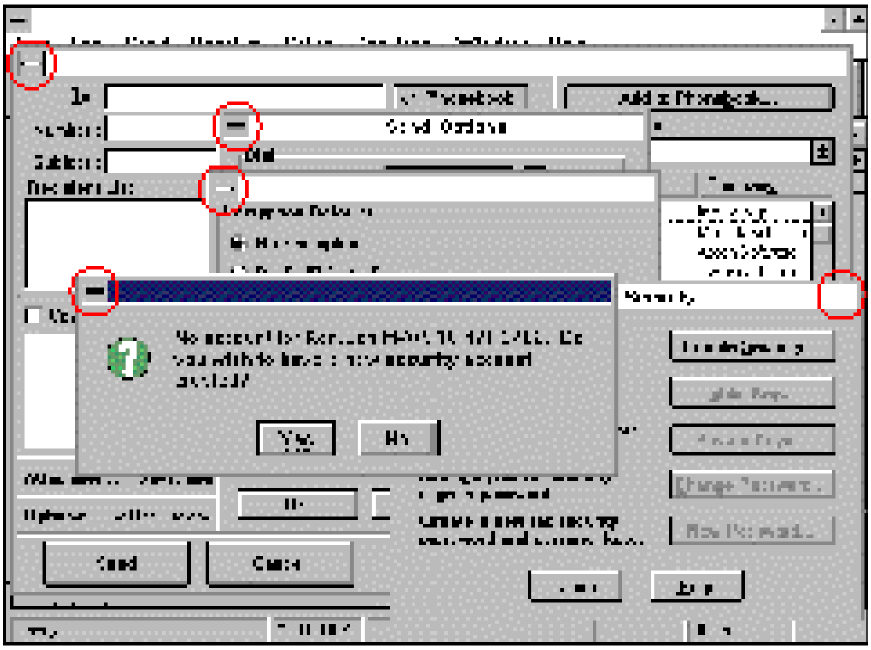

FIGURE 2 ***Trapped in a Modal Fever.*** *This is an example of the downward spiral resulting from an overuse of modal dialogs. Five different modal dialogs must be closed in reverse order to get back to work. Change to property sheets or tabs for better organizational health.*

INFECTIOUS FEATURITIS

Users use applications to get work done. Infectious Featuritis is typified by adding features and complexity that users don't really want or need, but developers love. These are usually solutions in search of problems rather than ways that will allow users to accomplish a task sooner or more efficiently. Note that if too many features are available, users won't know where to start. A valuable feature is one that is used.

TOXIC DOCFIXMANIA

This highly prevalent disease is named for the frequently heard phrase among

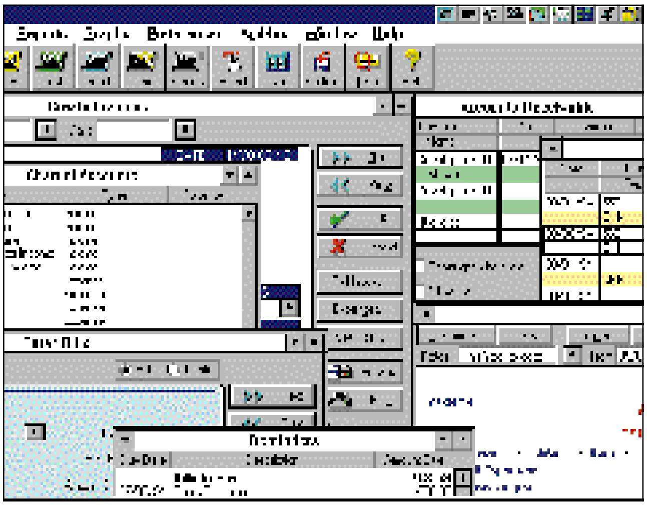

FIGURE 3 ***I'm Breaking Out in Windows!*** *Enabling a user to open many overlapping windows simultaneously is symptomatic of Claustrophobic Objectosis, not to mention "Out of Memory" errors. Again, tabbed dialogs can free up memory, as well as your user.*

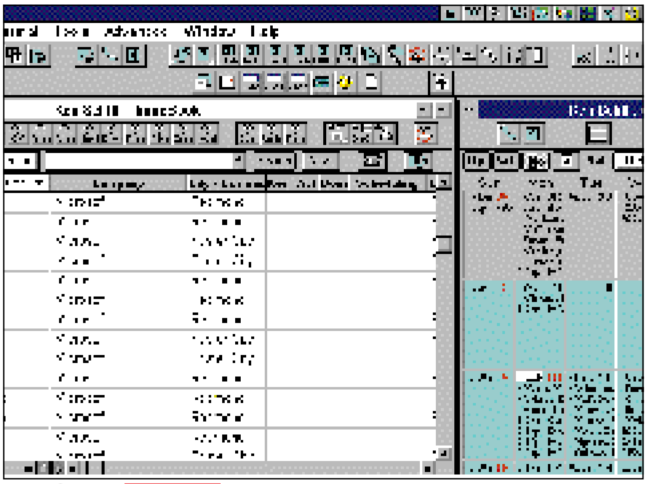

FIGURE 4 ***An Outbreak of Acute Iconitis.*** *Every child window on this screen has its own toolbar of related functions. Besides confusing the user as to the functionality of each button, the only white space to be found is in the data area. Use restraint to avoid the ailment: go lightly with buttons.*

developer teams: "Let's fix it in the documentation." While we can occasionally temper the symptoms, the cause is still unknown. Here's a preventive measure: if it doesn't work right, don't include it. Never use documentation as a crutch.

PERNICIOUS VERFIXMANIA

This ailment comes from another phrase often heard from developers, "Let's fix it in the next release." Why introduce problems? If it doesn't work right, don't include it in *this* release. If there is a problem in the first release, your program probably won't live to see a second chance.

HYPERCOMPLEX HIGHRESOSIS

This is a relative newcomer to the Desk Reference. It erupted when relatively inexpensive high-resolution monitors and display devices became readily available. It is best explained by this theorem: the propensity to complicate user interfaces and the challenge to keep things simple increase as the screen resolution increases. The only known cure is restraint.

CHRONIC MARKET PRESSURITIS

This disease comes from the pressure to release products before they are ready. It is usually contracted by contact with individuals from other departments within a developer's organization. Its only cure is good judgment, which comes from experience. Unfortunately, experience usually comes from poor judgment.

If you encounter maladies that should be included on this list, send descriptions to the National Effort for Research on Developer Diseases (NERDD) in care of myself to kenguiguy@aol.com. Fax: 510-471-2402. ■

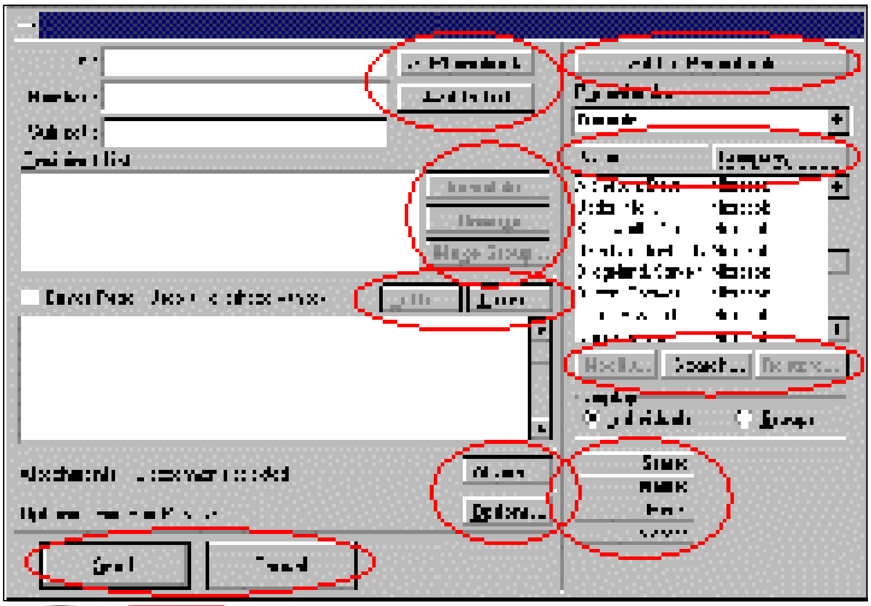

FIGURE 5 ***Call in the Control Chiropractor.*** *This victim of Random Hypermultibutton Syndrome is in severe need of an adjustment to straighten it out. There are five different alignment schemes on this crooked dialog, and raised panels are easy to confuse with functional buttons.*

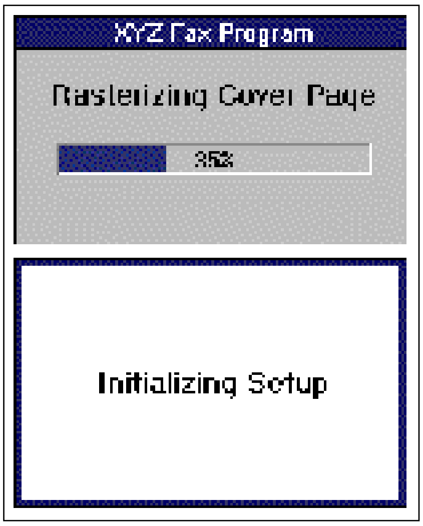

FIGURE 6 ***C'mon Doc: Give it to Me Straight.*** *Specialists in any industry throw jargon around like it's going out of style. When you are updating a user about the progress of his program, tell him in language he'll understand.*