

The Morphing Waldo:

An Adaptive User Interface

by Colby Chambers Howell



Is there anyone out there who doesn't recognize the coyly delivered message "You've got mail"? How about the boinks, bongos, chimes, and other audio cues that have become as familiar as the Oscar Meyer weiner song? Our language now reflects the omnipresence of the computer; *crash*, *code*, and *program* have extended their definitions and new words and phrases proliferate—software, download, RAM, WYSIWYG, on and on. As a society we have become joined at the hip to computer technology. The operation was successful, but some of the patients wish they were dead.

The technological revolution has swept over us all in a remarkably short period. Even those people who do not own a computer are indirectly affected every time they purchase something at the store, make a phone call, or use an automated teller machine. We are all being pulled along, willy-nilly, by an automaton that we don't understand, that won't answer our questions, and that is incapable of listening to reason.

Techno-rage is everywhere; frustration with computers is the rule, not the exception. For example—

Joke emails circulate about "Computer Tourette's." This is a play on the disorder known as

Tourette's Syndrome, where sufferers engage in uncontrollable bouts of swearing. The joke is that you can walk down the hall of most modern office buildings and hear otherwise-normal people sitting in front of their monitors, jaws clenched, swearing repeatedly in fury. Who knows what triggered such an outburst; a misplaced file, an inaccessible image, or a frustrating interaction. Or maybe the program just blandly erased the user's only copy of a 500-page manuscript because he responded with a "Yes" to a confirmation dialog box, assuming that it had asked him if he wanted to "save your changes?" when it actually asked if he wanted to "discard your work?" (Cooper, 1999, p. 14).

Four months ago I went to the county administrative offices to pull a permit for a garage my husband and I were getting ready to build. The most interesting part of the process occurred at the end of the multistep transaction, when it became necessary to record pertinent information in the county's database. I stood on one side of the counter, with my maps and permissions, while a very nice lady on the other side of the counter began to enter the data in her

computer. I could tell that it was an uphill battle because of the intense expression on her face. She even muttered a deplorable word under her breath. I got interested at that point, explained that I was interested in interface design, and asked about the problem. That opened the floodgates. She swiveled the monitor around so that I could watch and for five minutes demonstrated just how eccentric and unreasonable a new, high-dollar, top-of-the-line application could be. It was so difficult, she said, that two of her coworkers had opted for early retirement rather than work with it anymore. That is what I call a powerful application.

There is no going back to the good old days. Like it or loathe it, technology is here to stay. It is no accident that the gang-buster global economy and the technology boom have occurred simultaneously. “The difference between having a software solution for your problem and not having any solution is so great that we accept any hardship or difficulty that the solution might force upon us” (Cooper, 1999, p. 27).

The Cause of the Problem

“The human mind is exquisitely tailored to make sense of the world. Give it the slightest clue and off it goes, providing explanation, rationalization, understanding” (Norman, 1990, p. 2). We are learning organisms. From the cradle we take in sensations and information to create and build up a knowledge base. How this occurs is left to the cognitive theorists, but the results are what enable us to deal with life and ourselves. The actual trial-and-error learning process can be painful and frustrating, but humans have selective memory. We tend to recall the good times more clearly than the bad. I do not remember learning how to ride a bicycle, only the enjoyment of going a million miles per hour, free as a bird. That there were problems during the learning process is evident by the presence of certain scars on my knees and elbows.

But what if, during that bicycle-learning time, some evil genie had operated on my bike every night while I slept and changed it so that there was a new machine and a new set of riding rules each morning? The outcome probably would have been quite different. That scenario is not so different from the realities of today’s business world. We face devices that often do not make good intuitive sense and that also have the nasty habit of frequently changing into something new and even more complex. We are the often the victims of poorly designed software and a business environment that is constantly in flux.

No one will deny the existence of a problem. Gloria Gery addresses the difficulty in her book on electronic performance support systems, but her comments can be expanded to cover the larger business context:

The reality is that, regardless of how much or how well we do, the problems we are attacking are accel-

erating at an even faster rate. Or our development efforts are simply taking too long.... We are applying radically different technological alternatives to old frameworks without reexamining their underlying assumptions and structures. In our pursuit of solutions, we have assumed that the future should be an extension of the past. We have not taken time to step back from the situation to reexamine whether the old approaches should or must be the best solutions. We apply sophisticated technology to an obsolete paradigm of human performance development. And, as a result, we are not making the difference we should (Gery, 1991, p. 17).

It is said that if you can clearly define a problem, you are three-quarters of the way toward determining the solution.

It is said that if you can clearly define a problem, you are three-quarters of the way toward determining the solution. The problem appears to be that the old ways of designing software solutions don’t cut it any more; hence the phrase “We need to think outside of the box.” But it is easy to slide back into the old way of problemsolving and programming; it is a comfortable rut. So exactly how do we design now?

It appears that the process starts with a description of the problem or challenge at hand. There is analysis, then design, development, implementation, and finally evaluation—the ADDIE model if you are an instructional designer or the scientific method if you are a scientist. Whichever

Computers	Humans
incredibly fast	incredibly slow
error-free	error-prone
deterministic	irrational
apathetic	emotional
literal	inferential
sequential	random
predictable	unpredictable
amoral	ethical
stupid	intelligent

Figure 1. Different Traits of Computers and Humans (Source: Cooper, 1999, p. 87).

name you give the process, it still follows the linear sequential product-centered model. Alan Cooper, in his book *The Inmates Are Running the Asylum*, looks at the difficulty specifically from the software-development angle. He suggests that the programmers and engineers are doing most of the decisionmaking and designing for the computer, not the user (see Figure 1).

This comparative list says it all; we are designing to accommodate the machine but it is the poor user's task to get the job done. Because today's business world is all about information gathering, handling, and manipulation via the computer, it is no wonder there's on-the-job stress.

A General Solution

Performance-centered design (PCD) offers an alternative to the old methodology of software development. A representation is created by using elements from total quality management, human factors engineering and process, and diversity and usability modeling. These elements are chosen to work together to produce a synchrony of process, content, and context. Maximized performance comes from the confluence of giving the user just enough information at just the right time so that he or she may carry out the tasks at hand efficiently and effectively. This intersection or overlap is named the Performance Zone (see Figure 2). The important point is this: PCD concentrates not on the components (user, information, system) but on how they work together most productively. "PCD...establishes a performance goal and an organizing framework prior to systems design, (rather than just hoping for a good system judgment made after the design and development fact)" (Raybould, 1995, p. 11).

One of my friends has a huge kitchen and lots of electric cooking gadgets. She has a blender, a grinder, a mixer, a food processor, crockpots, a rotisserie, and a lot of other plug-in equipment. It takes her a long time to cook a meal, and the mess is widespread and large. I also enjoy cooking, but I have a small kitchen and a short list of utensils. The point is this: When all is said and done, I can create a good meal in half the preparation time, making one-third the mess. I have designed, through trial and error, my own performance zone. I only use the tools necessary to do the job and thus I avoid the clutter and confusion. Cleanup is much quicker, too. My family knows not to give me kitchen gadgetry for gifts. Now garden tools...that is a different story.

The same situation exists in the software industry. More is better, larger is better, multifaceted and multitasking are better. It is this type of thinking and the products it engenders that create the problems experienced by the county clerk. She has tools that are too complex and confusing, and her job performance is constantly hindered. The "solutions" take their toll in diminished efficiency and increased anxiety. Who enjoys being made to feel stupid and incompetent?

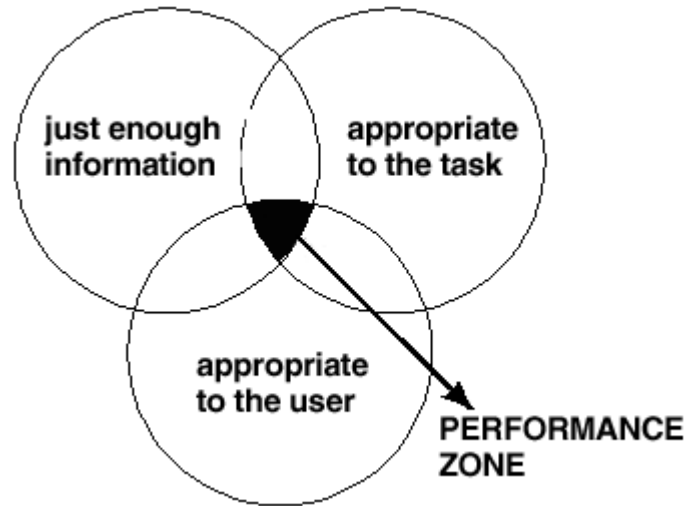


Figure 2. The Performance Zone (Source: Dickelman, 1996).

A Specific Solution Using PCD

"I may not know much about art, but I know what I like." This well-worn phrase could be suitably modified to fit software users. Except for those daredevil power users, most of us would like an application to just get out of the way and let us get on with the job, whatever that may be.

But we are individuals and our approaches to a task, our likes and dislikes, our foibles and twitches and thought processes are quite different. Is it reasonable to assume that a single version of a software product can be made to accommodate us all? Probably not, at least not perfectly, but it should be possible to hit a comfort level or create a performance zone for a larger number of users.

I would like to suggest one design possibility that could be used to create a more universally satisfying interface for existing applications. In general it would be an adaptive user interface; I would call it a *morphing waldo*. It would be a miniprogram that "sits" between the larger application and the user. It would be capable of changing relative to the specific individual (i.e., of morphing) and act as a protective skin to separate the user from the seldom-used and less-desirable functions of the larger application. Today the term *waldo* is used primarily in the field of robotics; it is a remote-controlled prosthetic device that becomes an extension of the controller so that he or she can manipulate objects in a distant and potentially dangerous environment. Hence, the *morphing waldo*.

Physical artifacts can be designed so that they are easy to learn, easy to use. The same is true for cognitive artifacts, although here some new principles must be provided. We need to consider the nature of the task to be performed and the powers of the human.

We humans seek understanding, causes and purpose. We are good at remembering experiences, good at stories and events, bad with minutiae of modern life. We are attentive to our surrounds, remarkably quick to notice changes. And we see patterns and meanings even when they are obscure and hidden. These very same characteristics, however, can conflict with the demands of the modern industrial, technological life. The conflicts are made worse by the technology that is imposed upon us on its terms instead of ours. The conflicts could be minimized or even eliminated if the technology came to us on our terms (Norman, 1993, p. 103).

Like 90% of the free world, I use Microsoft Word® for writing reports, letters, outlines, or whatever. It is not a great relationship. The dropdown menus and toolbar contain items and icons that I do not recognize or know how to use. And the program does spooky little things...capitalizes letters that I don't want capitalized, prevents me from setting up lists the way I want, constantly questions my spelling and grammar usage. It doesn't have a very large vocabulary, and its thesaurus is impoverished. After all the miles I've put on this application, it still doesn't recognize me. My car at least knows my favorite seat position and radio stations.

To create the waldo for Word, I would use elements of PCD diversity modeling and process modeling. I would gather information from observations of individuals using the application and from surveys and interviews. Then I would create personas from the information, with particular attention to personality types and cognitive processing styles (in the mode of diversity modeling). I would also create a rubric listing most frequently used functions, icon styles and sizes, menu and tool bar preferences, types of word processing done (i.e., business letters, reports, briefs, outlines, etc.) and other items describing tool use and interface appearance (in the mode of process modeling). By melding personas with preferences in the rubric, it should be possible to come up with examples of interfaces and actions from which individuals could choose in order to find the one most appropriate to their particular style and use. This interface would be the one that would automatically appear whenever they opened Word.

To protect the user, I would make sure this new interface could be replaced by the original Word interface at the click of a button. But if they are constantly returning to the original interface to reuse one specific function, the waldo should be capable of importing that one button or menu item and adding it to its own skinned-down interface. It would be capable of learning more about each user through use and of adapting to each one's particular wants and needs.

Problems

There are difficulties in the implementation of this application. Proprietary issues would need to be addressed. How difficult is it to create this type of program? Would it be possible to have one universal waldo that could work with most applications, or would it be necessary to have a small battalion of them, one for each application? How many personas would be needed to adequately provide for the greatest number of users? Would the waldo create more problems than it resolved? I do not know.

I do know this. While I was writing this article there were lots of little machine wrestling matches. Some I won. But somewhere out there, floating around in the cyberspace archive of lost items, there is a nifty-looking rubric that I worked hard to create. It is in a copy of this document that I can't get my hands on because it is already being used by someone named Colby Howell. 🏠

References

- Cooper, A. (1999). *The inmates are running the asylum: Why high tech products drive us crazy and how to restore the sanity*. Indianapolis, IN: SAMS.
- Dickelman, G. (1996). Gershon's law. *CBT Solutions Magazine*, September/October.
- Gery, G. (1991). *Electronic performance support systems*. Tolland, MA: Gery Performance Press.
- Norman, D.A. (1990). *The design of everyday things*. New York: Doubleday.
- Norman, D.A. (1993). *Things that make us smart*. Cambridge, MA: Perseus Books.
- Raybould, B. (1995). Performance support engineering: An emerging development methodology for enabling organizational learning. *Performance Improvement Quarterly*, 8, (1).

Colby C. Howell is currently completing work for a master's degree in Instructional Technology through the EDIT Immersion Program at George Mason University. She is focusing on electronic performance support systems and performance-centered design as applied to online learning environments and computer-based training. Colby has taught chemistry and physics for the past 12 years in both public and private school systems. During this time she codesigned a process-oriented science course for ninth grade students, "Science Method and Analysis"; wrote the manual for the course; and presented the material at the National Science Teachers Association national convention in Anaheim, California, in 1994. Colby may be reached at ccham@erols.com.