musings

I do not mean to bash engineers. I am part engineer myself and have often been employed by people needing an engineer interpreter. I found that I had a talent for listening to engineers, then explaining what they would say to me in terms that others, the mere mortals of the world, would understand. I even learned how to ask questions in an appropriately humble and self-deprecating fashion so the engineers would deem me worthy of their attention and time.

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V.*

<rik@spirit.com>

We live in a world made marvelous by the doings of engineers, from the high tech of computers and cellphones to the subtle but all-important underpinnings of our society, such as sewers and water filtration systems. Without engineers, we would be living in the stone age. But until engineers learn how to design machines and software for mere mortals, their successes will be limited. What stands in the way today is an element of design known as the state machine.

Please keep in mind as I ramble that I consider software designers (programmers) as engineers, and this bit of prose will start making more sense and eventually get to the point.

A Different Path

As you might have gathered, I believe that engineers live in a slightly different world from the rest of humanity. Like a bit of science fiction, the world in which engineers live is in a slightly offset space, where everything can be expressed more precisely and every event has a totally predictable outcome. When the engineers interact with other people, their skewed worldview (from the perspective of others) can get in the way of smooth and empathetic communications. It also leads to many surprises for the engineers, because people are not nearly as predictable as physical models or programs.

For example, my first software engineer boss was a brilliant guy. He could bum lines of code out of anything I wrote. We were working on an embedded system, and we really interfaced well when it came to single-stepping through some gnarly assembler. But when it came to working with other people, well, it was interesting to say the least. I remember one time when I could hear him thinking aloud about giving the secretary (the one who worked for all the engineers) a makework task that would take her days. When he presented her with the task, he began (with no trace of sarcasm), "You won't mind doing this . . . ." I merely cringed. She later wept, then quit.

I am sure you have met or worked with people like this. These people are valuable, or we would have shot (or stoned) them long ago. And they can be quite fun to be around. I really enjoyed the engineering majors when I lived in a college dorm. The chemical engineers would describe exactly what was happening to them as they got drunk. The electrical engineers would stick a knife precisely into a toaster, then ask someone to hold it. The unfortunate volunteer would get a non-fatal shock because the engineer was using the heating coils inside the toaster to create a resistor bridge to reduce the voltage to a "safe" level. The crepe rubber soles of the engineer's shoes protected him.

Real Fun

While I am on the topic of household appliances, consider for a minute the humble, but much abused, VCR. Very likely, if you are a member of USENIX, you are somewhat of an engineer yourself. You can determine this by taking a quick test: do you find VCRs difficult to program? If you answered yes, you have very little engineering blood

The entire desktop design metaphor is a maze of states.

in you. If you wonder why anyone has trouble programming VCRs, please keep reading.

The secret to the programmable nature of VCRs is a simple microprocessor, quite commonly a Motorola 6805 descendent. By pressing a key on the remote control, you make a menu of choices appear. By pressing the correct sequence of keys, you can instruct the VCR to turn itself on, switch to channel 9, and begin recording at 10:00 pm on August 13 at the extended play speed. The remote control (or front panel) keypad is the input device, and the menus displayed on the TV screen are the output.

The VCR's microprocessor is executing a simple programming algorithm known as a state machine. Up to a point, each key pressed displays another menu, representing another state. Then the keys can be used to set parameters, such as starting time. When the user finds him- or herself in a state, pressing a particular key has a meaning pertinent only within this state. The programmer has made this so. The unfortunate user may not have made the deductive leap (never having implemented a state machine) that each button is not like a light switch, but can be used for many different purposes. It all depends upon what state the system is in.

This brings me back to my first programming boss. His triumph of human-machine engineering included a 16-button keypad, where most keys had four different purposes, depending upon the order in which you pressed the keys. Being a programmer and part engineer myself, I quickly caught on to this (yes, I find VCRs trivial to program, and digital watches, too). It wasn't until I was sent out on the road to teach customers about this incredibly simple user interface that I discovered how really inscrutable state machines could be.

After I quit that job and moved to California, I vowed that I would make computers easier to use. Not that I ever did – it was too tempting to create elegant designs instead.

More Than Human

Now let's consider the ultimate, the most popular computer-human interface yet developed: the desktop and the mouse. The mouse is a no brainer, right? It has one button that does whatever the programmer wants it to do, depending upon where the indicator (the cursor) is and when the user pushes the button. Instead of a 16-key input device, we have a single-key input device. But wait, there is that 101-key keyboard just sitting there in front of the user. By combining key presses, perhaps key combinations, and mouse clicks, we can "overload" the mouse button a wee bit more.

Of course, we can include state machines that make the designers of VCRs moan with envy. If we hold down the mouse button when the cursor is in the correct position, a menu appears. Then, if we drag the cursor to a position indicated by a triangle that represents an arrow, yet another menu appears, which might contain still more entries with triangles representing arrows. The perfect user interface: with a single button press and a few deft wrist movements, the user can navigate through several states – the ultimate in overloaded operators.

The entire desktop design metaphor is a maze of states. Move the cursor to the menu bar and click, and one thing happens. Move it inside one window, and clicking changes the cursor into a text entry cursor, and you can enter text. Move it to another window and click, and you can select objects or draw lines. In another window, you can use the mouse to play games. With three-button mice, clicking on the background window brings up different menus. The potential for new states is endless.

Babies

You might think I am daft to be complaining about the mouse/desktop design metaphor. Millions of people, tens of millions, have learned how to use a mouse and to be productive using computers, you say. Perhaps you have seen children, even toddlers, quickly figure out how to use a mouse. They move their hand and a funny mark moves around on the screen, it's natural, right? And the alternative, text-based, command line-oriented control, is only for real engineers. Frankly, I personally doubt that Windows would have ever had caught on without the training program known as Solitaire.

And what about the desktop metaphor itself? The original idea, from Xerox PARC and successfully promoted by Apple Computer, was to display an area with objects on it, called the desktop, which the user manipulated using the mouse. Users could drag things they didn't want into a trash can, drop something they wanted to print onto a printer, or double-click on something they wanted to "open" (another bit of programmer-speak). Notice that the desktop itself is free of state machines – or perhaps it can be considered a machine with one level of state. Still, the desktop avoids the many levels that cascading menus impose or changing to new state machines by simply moving the cursor into a different window.

Notice that the desktop itself is free of state machines – or perhaps it can be considered a machine with one level of state.

Now I don't know about your office desktop, but I use another programming metaphor on mine. It's called stacks. When I put something into a stack, it stays there until I need it. I use a FIFO for bills, a special, fast-moving stack for checks, a very deep stack for things that I should read but will never get around to. It works well for me as long as nobody attempts to clean up my desk. Then I can't find anything.

I would like to suggest humbly that the state machines that programmers and engineers are so fond of may not be the best human interface design ever conceived. The mouse works so well because we humans are accustomed to using our hands to move things. But designs that use a single button to do many different things, depending on the current state, do not so neatly correspond to the outer world, where a light switch only turns on or off a light and a rock is only a rock.

Oops, there are fader light switches. And humans have used rocks for grinding grain and smashing enemies. But I don't know if a rock user would ever have figured out that if he presses down on the rock, slides it up to the arrow, then to the right, down to another arrow, right, and then down again, and finally releases, that something different, but desirable, will happen. State machines just aren't natural.

Virtual Reality

I don't have a solution to this quandary. If I had one, I would have to keep it a secret until I could patent it and become the richest man in the world (just kidding). But I do think that an answer lies nearby, by examining the real-world behavior of our fellow humans (and engineers) and applying what we learn to the human-computer interface.

The spatial metaphor of the desktop works well, as does the mouse-moving-cursor metaphor. Perhaps the interface of the future will require 3D, with tiny displays "painted" on our retinas simulating an environment with depth, and input devices better designed to mimic our manipulation of "real"-world objects. I don't think the answer involves putting more buttons on the mouse. But I do invite you to ponder this. Perhaps you will become the richest person in the world, or maybe just the most generous.