Unfit for Use

(Unless You Perform a Lot of Maintenance!)

I work best under a deadline. Which is why Heather Giacalone, the awesome CrossTalk Article Coordinator, has learned to not let me know that a BackTalk article is due until right before she needs it. Give it to me too early and I end up writing an inferior article, but then beleaguer Heather with numerous "updates," "minor improvements," and frequently, an "ignore earlier submission, here's a different one." Wait until it's due, add the last-minute stress, and out pops a BackTalk column.

You would think, however, that I would sort of know when two months pass on a calendar, and start thinking of what to write. Not this time. I was at a total loss for a topic until the phone rang yesterday while I was driving. It was my pharmacy, letting me know that my prescription was ready for refill. Except the process was less than smooth. First, the automated messaging system asked me to, "Verify my identity by entering my birthday in the format MMYYYY." At least, that's what I think it said. I was busy juggling my cell phone, trying to put it on speaker mode, and bring up the keypad to enter the numbers. Because I started this before the automated message finished, I didn't hear (and couldn't remember) if I was supposed to hit the pound key afterwards. Obviously not, because after I entered my birthday, nothing seemed to happen, so I hit the # key just in time to hear, "Invalid entry. Please enter the date again." I reentered the date, did not hit the # key, and eventually got a message saying, "Your automatically renewed prescriptions are ready for pickup."

Why couldn't the message just have said, "The individuals associated with this phone number have prescriptions ready to pickup"? If I was sitting at home or work, using a landline phone, entering birthdays using the touchtone pad is pretty easy. On a smartphone? Not easy at all, especially if you are driving.

I bet this system worked great about 15 years ago when nobody had a cell phone. Not now. The system has become obsolete. It's "unfit for use." It's not unfit for use due to bad programming, but due to a changing environment and changing user needs. The system has become outdated— just one step away from obsolete.

As most good software developers know, maintenance is the hard part of keeping a system running. It's hard work that is typically unappreciated. It is, however, probably the most important part of any long-lived system.

I teach software engineering, and the maintenance paradigm I teach involves three components:

1. Adaptive Maintenance (keeping the system current for changing environments and user needs—hopefully in advance, but usually at the last minute, and in a panic)

2. Corrective Maintenance (fixing errors)

3. Perfective Maintenance (a mythical time wherein developers have some spare time, and they take a working system and make it perform faster or more reliably)

Let's ignore perfective maintenance (most systems do!) as it's discretionary. Perfective maintenance is for "when you get it working." As in "…let's just use this quick-and-dirty algorithm, and we'll go back and fix it and do it right when we get it working." Nice idea, but it seldom happens. Adaptive and corrective maintenance are more important.

I would estimate that up to 75% of all maintenance is corrective, but the word "corrective" covers a lot of ground. Didn't have time to put in some functionality during development? Mark it as an error, and call it corrective maintenance. It's not right, but it's the way we get software out the door within a reasonable time frame.

Unfortunately, lots of corrective maintenance is just plain fixing errors. I ran across an excellent online article called "The First Rule of Programming: It's Always Your Fault" that states part of being a good programmer is recognizing your mistakes, and fixing them. Unfortunately, just fixing errors does not give you a long-lasting system. For that, you sometimes have to take a working system and change it to meet rapidly evolving environmental changes (think new Operating System or new hardware) or user needs. This requires adaptive maintenance.

Adaptive maintenance—now that's the money-maker. The difference between a system that lasts 1 year and one that lasts 10 years is the ability to modify the system to meet changing user needs and changing environments. Used to run on DOS? Then Windows 3.1? Then 95, 98, ME, 2000, XP, Vista, 7 and now 8? You must have some good maintenance programmers. Has your radar systems survived 10 different airplane bus upgrades? Kudos to you!

The aforementioned pharmacy system is unfit for use due to lack of adaptive maintenance. User needs (and operating conditions) changed, and they did not adapt the system to match. Most people today use a smart phone of some type. You can't assume it's easy to have a keypad available without some fumbling. Voice recognition, while frustrating at times, would work better.

Some systems just stumble along, never really working well, and users sort of pretend to not notice the unfitness of the system. My pharmacy's telephone automated refill system is horrible—but then, I use the online system and avoid the telephone system at all costs.

Still, how hard could it be to put in a simple voice recognition system? (And understand, I have no idea how hard that would be. That's the problem with maintenance. Changes sound so simple to the naïve non-developer, don't they? But that's another column!)

David A. Cook, Ph.D.
Stephen F. Austin State University
(and former maintenance programmer)