Editor’s Gossip Column

The School of Recalcitrant Users

Whil Hentzen

It’s early July in Milwaukee, and that means we’ve only got about a month before summer arrives. I spent the Fourth of July weekend out by the pool, bundled in a parka and battery-heated socks, thinking big thoughts. (That’s French for “thinking without any real purpose or goal.”)

Lessons I’ve learned:

1. User Group Lesson No. 1: People like to be asked. If you’re running a user group or if you’re involved in any organization, don’t ask questions in a group setting. Call the person up, or pull him or her aside at the next gathering. Most people are flattered that you think enough of them to ask. A few will be too busy or feel unworthy, but a surprising number will respond.

2. User Group Lesson No. 2: People will let you down. This is a volunteer organization, after all, and when someone is put between a rock and a hard place, you’ll soon find out which gives first. So when that customer of theirs asks for something ridiculous, or their family has troubles, or whatever, don’t despair—instead of getting all upset, plan for it. It’s part of the deal that you accept when you become involved.

3. Ten percent of the people on the planet should never be put in front of a computer. But you still have to deal with them. You’re probably thinking, “And most of those people are my customers/users,” right? You’ve all heard the story of the technical support desk that received a call from someone who complained that the coffee cup holder on their computer was broken, and you’re probably getting tired of it. I have one user who’s been the primary contact for one of our larger systems for four or five years now, and this person still reads the entire “Do you want to save changes?” dialog box in Word each time it comes up. I guess it *is* possible that the message could unexpectedly change.

Again, instead of getting frustrated, try to play to users’ strengths. We spend a lot of time making sure that people aren’t made to feel uncomfortable. Computers are scary things, and programmers are even scarier. It’s important to let even these 10 percent save face. “Press the left arrow key.” (pause) “Uh, no, your other left.”

4. “Trust, but verify” is a stupid idea. People make mistakes. And while most people can be trusted all the time, some people can only be trusted some of the time, and a few people can never be trusted. How are you going to tell the difference? Just pick up a copy of *Inc.* magazine any month and you’ll read a story about some small business that went under because the owner trusted the bookkeeper and the bookkeeper was keeping three mistresses on the company payroll. You have to check up on others because people make mistakes, and once in a while you’ll run into that weasel, and those people generally don’t wear signs. This phrase seems oxymoronic. Why not just say, “Verify”?

5. Employees will attempt to do a good job under even the worst circumstances, and will fail only if you don’t let them do their job. Same thing for users. Step back and think about the last few problems you had at your shop. Was it because the other person was a total nincompoop, or could it have been—given the fact that people make mistakes—that he or she was put in a difficult or impossible situation? It’s your job to construct the situation so that it’s not impossible. You are in the unique position to reconstruct the situation so that the roadblocks go away or can be avoided. And that’s why you get paid the big bucks—to solve those hard problems.

6. There is no such thing as a computer problem. They are all people problems. Think about it. People don’t get fired because they’re incompetent (except in rare situations). They are asked to leave because they’re not playing well with the other boys and girls in the sandbox. Same thing with computers.

7. People remember the darndest things. ’Nuff said. It would be a lot easier if you just kept your mouth shut the whole time, wouldn’t it?

8. Stay on the edge of the chair, but not so that it’s uncomfortable. I’ve always told every potential hire that my job is to keep them sufficiently challenged so that they’re never in danger of falling asleep, but also so they're never so uncomfortable as to be unhappy for more than a short period of time.

During periods of heavy training, I always use the following rule of thumb to gauge how my workouts are progressing: Out of every five runs, two should feel great, two should feel terrible, and one could go either way. If I had more than a couple of great runs, it meant that I wasn't working hard enough. If more than a couple felt really bad, it meant I was working too hard, or that I was getting sick. Of course, you'd be ready to quit in a month if you had two miserable days each week, but that's where the analogy falls part—my philosophy is that the purpose in a workout is to make blood ooze out of my eyes. Work isn't quite the same. Still, feeling heavily challenged two or three days a week is probably a good percentage.

9. Programming requires an individual with an amazing resiliency to failure. Computers are difficult creatures to deal with because they virtually never make a mistake. This is a problem because we—humans—tend to make a lot of mistakes. You know the old saying, "You always find something in the last place you look"? Well, of course—because once you find it, you stop looking. Same thing with computers, software, whatever. You can beat your head against a wall all day until you once, finally, get it right. Final score: Computer 238, Human 1.

I've always wondered why programmers tend to make a really lousy salespeople. Salespeople also need a tremendous capability to withstand rejection. I'm guessing that the rejection a programmer faces is different, because it's logical, and you know you're going to eventually be able to beat it (or, failing that, have the option to switch to a simpler language.) You may never sell to a particular customer, and never know why, because people are irrational. And this brings me to my last thought:

10. Assuming that people will be rational is the biggest mistake you can make in business. Face it—adults are simply children with bills. But, of course, this is also an advantage, because it relieves you of the burden of acting grown-up as well.

Have an excellent day, and don't take yourself too seriously. If you were in Milwaukee right now, I certainly wouldn't let you!