Creating an ADP Center of Excellence: A Miltonian Approach of Looking through Two Decades of "Visible Darkness"

Peter Parker, U.S. Dept. of Commerce, Washington, DC

Abstract

After almost two decades of working with large and then small computers, primarily with SAS® software, I step back from my monitor and ponder- what makes my ADP shop a Center for Excellence. In this paper, I'll inquire on what most becomes a computer guru, an unusual type of professional. As well, I offer my truths on how to handle users, management, and the constantly reinvented technology. I will use notes that I have amassed throughout my career, now at its midpoint. Rather than taking a dry totally technical approach, I prefer a more literary stylistic one, drawing on John Milton's Paradise Lost and other classic literature as descriptive aids to my arguments. My observations may help you keep users satisfied, and mostly plan a prosperous career in information processing and office automation.

I have been programming since 1979, mostly in SAS software. I'm proficient in most hardware and software. I run an ADP shop in the U.S. Department of Commerce. As well, I run a consulting firm, Perfect Programmer Enterprises, where I provide programming (again primarily SAS software), Office Automation and Internet support. I have BA and MA degrees in Economics.

The Argument

"Receive thy new possessor; one who brings
A mind not to be changed by place or time.
The mind is its own place, and in itself
Can make a Heaven of Hell, a Hell of Heaven." (Milton, Paradise Lost, Book I, lines 242-245)

I fell into computers (though not from Heaven) almost two decades ago. My first languages were SAS software and FORTRAN and my computer was a large IBM® mainframe, the Big Iron. At the time, I was an Economist, right out of college, with little understanding about what a computer can do, or what a computer programmer does. I never expected to craft a career out it. Now eighteen years later, with plenty of time spent behind a monitor, writing and testing code, building interactive applications, and making things work, I managed to pull myself away from the "*visible darkness*" *(Milton, Book I, line 63*) of my virtual world. I have culled together my own observations, as well as others', on creating an ADP shop that is a center for excellence. Many of my ideas were formed from conversations with SAS programmers that I have met at SUGI conferences for the past thirteen years. These conferences emphasize the same tactics that I used to create my paper; one needs occasionally to pull himself out of his work environment to reflect on where he has been and where he's going. That's hard to manage with deadlines looming, phones ringing, and users continuously pounding at your door. Like SUGI should be, this paper is one long sustained reflective conversation in a thought conducive atmosphere.

My paper will focus on the main aspects of an ADP shop, from the programmers, the users, and the management to the technology. The ultimate goal of any ADP shop is to make the users fulfilled (re: Full Service Fulfillment Center of Excellence). Even more important, is making the management look good. However, as a career computer person, you need to maintain your employment viability. Having satisfied customers is not enough. Sometimes, taking too good care of them is detrimental to your shop. I'll comment in greater detail on these major errors that *the flesh is heir to (Shakespeare, Hamlet, Act II, Scene I)*. As well, I have an abundance of anecdotes on management blunders.

Finally, I'll speculate on the future and the need for a well thought out vision on where the information management industry is progressing. Lack of vision will destine one to life in a tiny cubicle punching out COBOL code, while squinting at the glowing embers of a monochrome monitor. From my perspective, that would be career Hell on Earth. Even Milton's Satan, being cast out of Heaven into the fiery pits of Hell, managed to make the most of his new situation. Likewise, one should emulate his positive scheming approach and take charge of his career. The demon, Mammon, stated it well:

Let us not then pursue
By force impossible, by leave obtained
Unacceptable, though in Heaven, our state
Of splendid vassalage; but rather seek
Our own good from ourselves, and from our own
Live to ourselves, though in this vast recess,
Free, and to none accountable, preferring
Hard liberty before the easy yoke
of servile pomp. (Milton, Book II, lines 251-257)

The Programmers

(computer gurus, hackers, techno-geeks, soulless technocrats, bon vivants, code-heads, and social misfits):

"Him the Almighty Power hurled headlong flaming from the ethereal sky, with hideous ruin and combustion, down to bottomless perdition, there to dwell in adamantine chains and peral fire. . . " (Milton, Book I, lines 44-45)

Who makes the best programmers? Not necessarily those with Computer Science degrees. Not that intensive study in information technology is undesirable. But through my experiences I have found that Liberal Arts majors (like in Economics and History), who fell into programming make the best[1]. Programming is more than sopping up technical texts and classroom learning. It's more of a personality sensitive occupation that calls out for the independent types who may not fit in well with a conventional bureaucratic or corporate structure. Before the computer was invented, these people may have been stuck in clerical jobs similar to young Einstein's stint in a patent office, miserable with their lot in life. The closest common trait I can identify in many of them is that they're nonlinear abstract thinkers, comfortable with mathematics. Their minds don't think in a straight line from beginning to end. They may ponder on a problem in the middle, work on the end, then go on to the beginning, like in writing a program. If one were to drop my paper and randomly rearrange the pages, it would probably make more sense to them. To a linear thinker, this method of reading would be highly irritable.[2]

I'm not suggesting that computer science majors make poor programmers, rather I'm alluding that many of these students may be drawn to this occupation purely by market signals. There are many well paying jobs in the computer field. There is more certainty in a financially rewarding livelihood than say, studying English Literature. One should consider the glut of Certified Network Engineers (CNE). The schools are grinding out these paper certificates at an amazing rate, but how many of these neo-CNEs only have classroom knowledge? How many have hands-on real life training? Many businesses are disappointed in the current quality of CNEs who understand LANs in theory, but have no concept of what to do in the real world. I'm not

[1] Professional integrity requires me to admit that I have two degrees in Economics, but none in Computer Science.
[2] I have always suspected that left-handers tend to be non-linear thinkers. I also admit to being left-handed.

claiming that certification is a bad idea. At least it weeds out those with only a cursory grasp of the technology. I'm merely noting that certification by itself is not enough. Certification, be it through CNE or CSE (Certified System Engineers), is an excellent incentive for professional computer types to enhance their skills. It is also an effective way to determine how serious the neophyte hacker is.

Additionally, you need a personality that can accept repeated failure. Programs likely will not work the first time. One Economist that I knew, who dabbled with computers, once estimated that you have a 30% chance if getting it right the first time. Coping with mistakes is an apt metaphor for the life of a programmer. As well, you need the ability to close out the rest of the world, and exclusively focus on a computer problem for hours. And you'll spend much time debugging your programs. And time will fly, "*but at my back I always hear, Time's winged chariot hurrying near (Marvell, lines 21-22)*." Many are the times that my mornings become evenings without my being aware. Time will move differently as you work on programs.

Programmers are stubbornly independent, sometimes beyond that of practical good sense. We don't like user manuals. We're programmers! We should be able to figure the software out. Peeking into manuals is almost like cheating. This attitude may be too headstrong, but it also forces us to learn the software in depth. We will usually start off by playing with it, maybe even creating non-job related applications to test on it. We're children in the sense that we like to play with computers like they're toys. We even refer to them as "our toys." Yet, that is how one becomes an expert. Many concepts are not attainable by just reading a book. We call it "getting our hands dirty", not unlike children making mud pies.

The Image- Once you determine whether you have the mettle of a computer warrior, you need to cultivate the proper image. While normally I wouldn't promote stereotypes, this case is the exception. While you may not be eccentric, eclectic, or dress unusually, you should! Otherwise you'll be treated like any other office personnel. This status could cause you to endure strict micro and time management, and a lot of meetings. That's not why you became programmers. If you appear as a techno genius, you will have more freedom to do projects, more comfortable working conditions, less meetings, less managerial oversight, and much more time to think. People will leave you alone (except when they have hardware/software problems for you to fix). Imagine that every day is casual Friday. Imagine that you can work on projects alone, rather than within a committee that will talk the data to death. The ideal situation is to be handed a project and being told, "make it so."

Here are some fashion/social pointers to keep in mind. Most of these suggestions should come naturally to you. First, you don't need to wear a tie! Ever! Don't overdo it on casual. You still want to look like a professional, in order to maintain some respect from your co-workers. So, don't dress like a teen-ager, hill-jack or Unix programmer. Especially if you're a temporary consultant, you don't want your users to be embarrassed to be seen with you. Use discretion. Work on the ambiance of your office. Play classical or jazz music and talk radio all day, and carefully create piles of CPUs with their guts hanging out, and heaps of disheveled books lying around. Tidiness is not crucial if you know where things are. Besides, you're too busy to clean up. Remnants of doughnuts and cookies look good, but be wary of becoming a rat haven. Working strange hours is smart, like first coming in at lunchtime. The bottom line is to reinforce the computer type stereotype. Don't disappoint your co-workers. An office kept properly eclectic is a comfortable office that gets a lot of work done. As well, you'll justify the users' beliefs that computers are too esoteric to understand. They'd rather rely on social misfits like you to worry about their ADP concerns. In effect, you'll be creating jobs for you and other creative types.

Too much of a comfortable image can backfire on you. One large office that I worked in was in terrible shape when I arrived. None of the programs worked well. The databases were poorly designed and the reports were unreliable and untimely. In no time, I became their "Perfect Programmer" and could do no wrong. However, I made the mistake of helping move furniture. Along with my manner of dress, I became known as the "moving-man." That status helped me stay away from the notice of high level management for years[3]. Yet nothing is more irritating than to be totally immersed in deciphering a problem involving intricate cutting edge technology (like SAS software, of course) and to be interrupted by a request to move some boxes.

What We Really Are-

"His tongue dropped manna and could make the worse appear the better reason" (Milton, Book II, lines 112-114)

What we really are in one word- plagiarist. "Copycats" might be a milder epithet. Any programmer worth his salary copies code from others. That's the main reason to use SAS software. Let someone else write the bubble-sort code. You need to know when and where to use it. Whenever I start working on new projects, I first look for a similar program either written by me or by someone else. I revise it only as necessary. You don't have time to write from scratch. I use this rule in all aspects of my computer usage. If I have to enter data into a database and it exists on word-processed document, I delimit the information to fit into the database. Never, never retype information or code unless you must. Rather than "copycats", perhaps "efficiency-expert" would be more appropriate. Consider that if some of our work was done by hand, it would take weeks, perhaps lifetimes, rather than seconds and minutes. I have a major report that I update every month with new import data. The database production and report generation takes about of two hours of clock-on-the-wall time to produce. We used to joke that it would take 700 secretaries with fast hands weeks to do it. Perhaps we deserve their salaries, as well.

While we are imitators, remember that imitation is the highest form of flattery. If someone copies your work, you should be honored, not offended.

Other Unusual Distinguishing Features-

While we may dwell inordinately in our self-contained cyber worlds, and at times, may appear to be out of control, we are doers. We produce tangible product, be it databases, reports, queries, or user-interfaced applications. We dread meaningless diversions such as meetings. We hate them. We like playing with ideas and mulling over alternatives, but most meetings don't provide the atmosphere necessary for an exchange of good ideas. Rather, they become scenarios for heads lolling about, an "office-synchronous nap." Here, one can expect lots of talk. Everyone puts in two-cents even if don't even have that much to say. Some bureaucrats have a compulsion to jabber and contribute sterile words. The best meetings are either only fifteen minutes max, or occur as you take a stroll with one or two of the main people.

A programmer must yield to change. If one resists, he'll become technologically obsolete. To me, constant change is a major plus. Otherwise, I'd soon become bored with my job. See the section on "Technology" for more details.

Programmers love technology. We yearn to fiddle with these latest in electronic boxes. We tweak them to push them to their limits and then take them out for cyber joy rides. Our computers are not merely tools of the trade. We take great pride in them like racecar drivers and their souped-up vehicles.

I've always been amused by the titles we confer upon ourselves. Whenever someone asks me what I do, I give them a different name every time. Without thinking, I'll call myself a Sr. Programmer, but often I'm inclined to beef up my "*imperial title (Milton, Book V, line 801).*" Here's a short list of names that I have used or have conspired to use:

1. Chief, ADP Operations

[3] This deception worked until an official was wondering why my boss was bullying the moving-man (me).

2. Chief Information Officer
3. Computer Guru
4. Computer Guru Supreme
5. Perfect Programmer
6. Head Hacker
7. Information Resource Manager (very high falutin')
8. Information Technology Specialist
9. Doctor of Data
10. Lucifer the "Light-Bearer" (shedding light on information)

The Users:

(abusers, analysts, clients, customers, the befuddled, hellhounds)

"*… a knave; a rascal; an eater of broken meats; a base, proud, shallow, beggarly, three-suited, hundred-pound, filthy, worsted-stocking knave; a lily-liver'd, action-taking, whoreson, glass-gazing, superserviceable, finical rogue; one-trunk-inheriting slave; one that wouldst be a bawd in way of good service, and art nothing but the composition of a knave, beggar, coward, pandar, and the son and heir of a mongrel bitch: one whom I will beat into clamorous whining, if thou deniest the least syllable of thy addition . . . (Shakespeare, King Lear, Act II, Scene II)*"

Like *"locusts warping on the eastern wind" (Milton, Book I, line 345.)*

Old hacker saying- G-d must have loved the befuddled user, because he made so many of them.

The users are our reason for being. Without them, we are nothing. Otherwise, there would be no point in creating databases, web sites, or data analysis. We exist for them.

Although we are indebted to the users, we still must be careful with our dealings with them. They can also be public nuisances. In fact, most are. How often do the users interrupt you, while you're up to your knees in number crunching bugs? The number one rule when dealing with them is "Don't Make Yourself Too Accessible." Otherwise, if they have any problem with their computers, they will have two choices. They can either figure it out themselves (like plug in the computer and turn it on, or restart it) or they can call you. If it comes down to either thinking for themselves, or having you think for them, they'll call you. If they get used to quick service, they may become upset if you can't run immediately to them. At times, they're like children in temperament. Their basic assumption is that if you're not doing something particular for them, then you must be doing nothing. Usually they only have minor PC problems (lost icons, printer paper jams) but these interruptions can quickly add up during the course of a day and prevent you from focusing on your major projects (the ones you're getting paid the big bucks for)[4]. You can easily get behind. I knew one programmer who assumed that every request (no matter how small) will take at least two weeks. He may have been on to something.

Besides being pesky, the users have no sense of vision. They hate change. If they had their way, they'd still be using clunky DOS® software. To them a PC is merely a tool, like a telephone or toaster. To some people, a car is for getting you from point A to point B. You have to push these people into the Electronic Information Age. Moving them to Windows95® software was relatively painless, because it is user-friendlier and I have created many software icons on their desktops. Still, it being different, some of them whined about having to work slightly differently. They don't like learning new things. If anything goes wrong, like the computer crashing, they won't let you forget how good their PC was before. They have forgotten that Windows 3.1® crashed constantly. Their attitude reminds me of the newly freed Hebrews complaining to Moses in the desert on how good they had it while they were slaves in Egypt. *"If only we had died by the hand of the Lord in the land of Egypt, when we sat by flesh pots, when we ate our fill of bread! For you have brought us out into this wilderness to starve this whole congregation to death. (JPS Tanakh, Exodus 16:3)."* Ignore them, they'll get over it. If their PCs have problems, they'll complain. If their PCs work too well, they'll still complain:

- "My PC is too fast! When I save my spreadsheet, I can't tell whether it worked or not."
- " I can't use a mouse. Can I use Windows without a mouse?"
- "Will my new laser printer still print DOS output?"
- "Can I still use my DOS word processor in Windows95?"

You can be the best SAS Coder ever to punch a keyboard but if you can't communicate with the users, you'll be almost worthless to an office. You need to speak their language, because they won't learn yours. The user is more concerned with doing his own job, be it financial analyst, economist or trade negotiator, and usually see no need in learning "Computerese."[5] They only know what they need from you, and sometimes not even that. Having a familiarity with the data, and having a financial, economic, or statistical background is invaluable in translating their poorly articulated and often lamely conceived ideas. You may have to explain to them what they really want. Some large organizations have intermediaries between the users and programmers, just for this purpose. In office automation support, the users rarely convey adequate details besides informing you that their computer "don't work." Usually, they require hands-on assistance in order to resolve their problems. Having a rapport with the users is a marketable skill that all programmers should strive for.

Users have this strange idea that programmers will become obsolete. Perhaps, this is a sign of animosity towards us. I first heard this suggestion about fifteen years ago that with the new technology, programmers will become obsolete. One just presses a button and it goes. You know where we were then, still fumbling with the IBM XT. Every few years I hear the same comment about the "end of programmers." You'll need programmers for many years to come. While I can grind out SAS software code 10x faster than FORTRAN or COBOL, I still need to be a system analyst to determine how the information system will be constructed efficiently (how databases will be updated, maintained and queried). Even with SAS software, one needs to know programming logic. Higher level programming language mostly saves you typing and debugging time with its sophisticated Macro-like utilities. Until we have major advances in artificial intelligence (still far away), our jobs are safe. They need us even more, as technology becomes more accessible to the public. Thirty years ago, the automated office hardly existed. Now, it's the basis for a standard modern office.

The Management:

(bosses, Bobo, Mr. Charlie, "The Man" , "The Great Satan", "Lord of the Flies", "Moloch", tyrants, totally befuddled, "*solitary, short, nasty and brutish*"):

Always bear this truth in mind. They have no concept of what you're doing- keep it that way! Otherwise they may try to help you. Of course, always offer to inform them. They likely won't ask but if they do, always have a massive list available, at least several pages long, just in case. Besides, they should trust you. It could be to your disadvantage if they start understanding what you do. If they're computer savvy, they'll have a greater concept of what the software/hardware is capable of doing. You will be held to stricter standards. You'll also need to explain your excuses better. Saying, "It can't be done" will not suffice. They may not buy the old scam, "it'll take at least two weeks." On the

[4] For the really bad help requests, like their printer doesn't work because it's out of paper, we give out a weekly "Art Vanderlay Award" , named after the most computer illiterate person in our office. Note that his name was changed in this paper to protect the guilty.

[5] Once with a discussion with an user, I was telling him that since HTML files are only ASCII files, one can tweak up these files with a decent ASCII editor. He gave me a perplexed look and ask me what an ASCII file was. At first, I couldn't answer him, having for years taken the term for granted. I stumbled back with an weak answer, "You know, a vanilla file."

other hand, a boss who only understands a little of what you do can be dangerous. He may believe he knows more than he thinks he knows. He may come up with some crazy ideas, like using a word processor to replace a database. Casanova said it best. *"What he professed to know, and did not know, made what he did know useless, and the acuity he tried to have spoiled what acuity he had."*

Management are remarkable for their lack of vision. When they purchase PCs, they desire what is needed now, not for the long run[6]. They end up with machines that are obsolete before they are delivered to your office. They believe in locking in a group-buy on PCs, even though the prices drop significantly every month. It was likely someone's boss who decided that 640K ram is all one will ever need in a PC. Twenty-six drive letters will be plenty. Not much more than a decade ago, I had to convince the management to let me buy color monitors for PCs. They can hold mind-wearying series of long meetings, planning every detail and contingency, and still manage to make all of the wrong decisions.

Management also has a lot of bad ideas about managing. They see how well their programmers and other creative types work and they then try to recreate a similar office environment for all to emulate. The basic flaw in their reasoning of this wholly artificial construct is that is works only for a few people. Imitating Bill Gate's personal hygiene won't make you a Bill Gates. I'm not a computer guru because I decorate my office like a garage. Rather, because I am a computer guru, I have these concepts of interior design. Many of the management ideas are laughable and I'd like to list them before they're swept into the dustbin of history, like scientific management and "Total Quality Management (TQM)" (Brooks, pages 20-23):

- The Open Office (no partition or walls, just one big open room- The concept of the naked office, or just a noisy restaurant atmosphere)
- Desks on wheels- making you more mobile, you can congregate with other groups, with your desk always accompanying you
- No job titles (remember: the boss will always be the boss, even if you can't call him that.) As one wag commented on that, "Try parking in the boss's parking space." Also new meaning to "Who am I and what am I doing here?"
- Water cannon camaraderie- everyone has water cannons, and by everyone being equal targets, emphasizes a classless office. Squirt the boss and let me know how it turns out (what are adults?- just big children)

In spite of my harsh words on bosses, there are some good things about them. We always will need bosses. Someone has to make the decisions and live with the consequences. You may design elaborate databases but decisions on every major step of the way have to be made. Ultimately, he must decide on what is best for the users. Additionally, if the wrong decision is made or if you totally screw up, the good boss will take all responsibility for it. Let him. That's why he has a larger paycheck. Privately, he will and should chew you out. Bosses should have a symbiotic relationship with their ADP staff. The bosses should protect us from the bureaucracy and steer us (otherwise we'd noodle around with our toys) and we make them look good (i.e., good product: databases, number crunching, presentations, etc.)[7].

The Technology:

(software, hardware, toys, "*Rocks, Caves, Lakes, Fens, Bogs, Dens…(Milton, Book II, line 621)*"):

"Better to reign in Hell than serve in Heaven" (Milton, Book I, line 264)

The Personal Computer (PC) has to be the greatest thing that could ever happen to a programmer. Finally, we're free of the tyranny of the great mainframe computer. We don't have to share accessibility to the CPU. It's located on our desk and it's ours to control. There is no more sitting in the dark when the system is down. We just reboot our little electronic box. Of course, with the blessing comes the curse. We no longer have the gods of the mainframe to take care of housekeeping like hardware and software maintenance and data backups. There's a price for this independence, like Adam's eating of the fruit of the tree of knowledge of good and bad.[8] Better yet, the fallen angel, Satan, preferred to run his own kingdom in "*this place, our dungeon (Milton, Book II, line 317)*" rather than spend eternity singing hosannas to his Master.

Working with a wide variety of electronic technologies, I have come across certain truths useful for an enlightened computer guru:

Rule 1- A Good Project Is Forever. Be it SAS or Lotus Domino software, a home page, or a database, you'll never finish it. It's like writing an endless book with no last chapter. You can always improve on it. You can make a career out of most projects if you want. But, you've got to let go. Deadlines are a necessary evil or you'll noodle over a project forever. This is especially true for home pages. With SAS programs, you can "macro-code" them to death, using macro windows, and other automation tools. You will make it user-friendlier, and contain more tricks. It also will be much more complicated and much harder to maintain. Even with good top-down structured programming, you can still "over-program" and turn a simple application into a monster.

Rule 2- Whole Lotta Bad Info out there. There's a lot of conflicting information out there, such as the good and bad points of homebrew PCs, particular brands of hardware and software. The Internet magnifies it tenfold. You'll be receiving much more advice from complete strangers. Rumors may prevail. With knowledge bases, message boards, on-line and off-line discussion groups, will we ever resolve whether Java® is dead or alive?

Rule 3- True worth of general software- If it's truly well written software, you don't need a manual to use it. You should be able to start it up and understand the rudiments within ten minutes. You should be able to create basic applications within thirty minutes. Microsoft PowerPoint® is a good example of intuitively friendly software. You shouldn't ever use the manual except as a reference for complex applications. You should be able to pick up any word processing/spreadsheet/ data query software and start hitting on it. Of course, there is not much difference between most word processors or spreadsheet software for 90% of your needs. Or as I have told one user, sweating over a new more advanced word processor, "Type, Save, and Print. That's all you need to know." You enter information, run grammar utilities, create tables, copy and paste. The differences tend to be in the advanced features that the general user likely never will need to use. With GUI software running on systems like Windows95, most of the battle of understanding how to use software has been won. That was not so during the dark DOS years.

Rule 4- Software training- For users, why? If it's good general-purpose software (see #3), the user should be able to pick up on it immediately. If you're a good programmer, you should be able to pick up on it yourself. However, for the more advanced applications like SAS software or Visual Basic, one definitely either needs training or much time devoted to self-study.

Rule 5- The Vision Thing (imagination, "*the organs of her Fancy…*") -

[6] My PC strategy is to buy the best PC you can afford now and also can afford to get rid of it within three years to buy a better box.

[7] Once I had a meeting with the Office director, and he eagerly questioned me on the latest in computer technology. He inquired, "So what's new? Are you going to replace me with a machine. Press a button and it does my job!"
I curtly replied, "Actually, Mr. Vanderlay, I was thinking more about replacing you with a potted plant."

[8] Carrying that metaphor further, the Apple PC could be seen as the forbidden fruit by which knowledge can be processed at a phenomenal rate, drastically altering our lives.

The Vision thing is without doubt, the most important factor in your computer career. It should be the driving force behind you- where are you going? Face it. You can't be proficient in all computer technology, although you must be a generalist involving the basic concepts. However, you should be able to pick up any decently written program and regardless of what language it's written in, be able to understand what's it doing. While you should be able to understand computer logic and how to operate a PC, you also must specialize in order to advance your career. You should choose software carefully to become an expert application developer. It must be popularly used and must be around for more than a few years. You're going to spend much time learning this software and you need several years of use to make this investment of personal resources worthwhile. Perception may be the death of software. If the programmers perceive that a software package is becoming obsolete, it will become obsolete. This strategy may be similar to how value is assigned on the stock market. If investors believe that a stock's price will fall, they'll initiate actions such as selling their shares, that will put pressure on the stock's price to fall, regardless of whether it was going to or not. This is known as "self-fulfilling prophecy."

How does one determine what will be hot software and what will be forgotten in a flash? SAS software has been around for over twenty years. I anticipate it'll be here for several more. Yet, once a better SAS-like product is developed, it'll be time to bail out of SAS software. I remember working at an agency about fifteen years ago, where an old man was known to be constantly wandering around the halls, and puttering about his terminal. No one knew what he did. I found out later that he used to be a computer programmer during the early 60's, who programmed in old assembler software called CLUE. When this product became obsolete, he refused to give it up. He knew CLUE and that's what he was determined to program in. A sharp programmer must constantly evaluate what's around and make decisions on where he perceives the industry is going.

Here's a list of technology that I think is worthwhile investing time into:[9]

- SAS software
- Lotus Notes/Domino®
- Dynamic Home Pages
- Windows NT®
- Visual Basic
- Adobe Acrobat®
- DVD drives
- Video conferencing

Here's a list of technology that I 'm not sure where it will go, hearing many different opinions:

- Java
- Active X®
- Active Server Pages®
- CGI
- Perl
- Delphi®

Here's a list of technology that I think is dying or has a limited future:

- IBM OS/2®
- Any DOS based software
- COBOL
- FORTRAN
- CD-Rom
- 9 track tape systems

The Bottom Line- I'm not doing the same thing I did fifteen years ago, or ten or five. Five years from now, hopefully I won't be doing the same thing. I definitely won't be left behind as technology advances. To a computer professional, that's career suicide. The computer technology career field is not like accounting or other established professional trades. It's constantly changing. Keep this fact in mind as you elevate yourself to Computer Guru or Computer Guru Supreme.

[9] Five years from now, recheck out this list and see how right I was.

Conclusion

Be the best computer professional you can be. Are you cut out to be a programmer? Can you help users? Do you know how to communicate with managers? Do you keep up with the technology? Do you plan for the future?

Computer programming is more than writing code and crunching numbers. It's making things work. Learn to deal with the nuances of occupational personalities and technological change. Especially important is developing well thought out vision. Five years from now the nature of the job will be different. Ten years from now, it will be radically different. Be prepared.
.

Appendix I- The Perfect Meeting

Modern meetings will be the debility of me yet. They tend to be too long, aimless, and too frequent. Meetings are the bane to programmers, who prefer more constructive uses of their time. One should examine the first meeting ever called, described in the second book of Milton's Paradise Lost. After the fallen angels have rebelled against G-d and have been cast out of Heaven into Hell, they meet in the hastily constructed temple, Pandemonium, the dark version of the Pantheon. You have only four speakers of the millions in attendance present their cases. They make their point and then Satan, the leader, after making a decision, immediately prepares to act upon it. First to speak is the bully demon, Moloch. He strongly urges the “hit the head against the wall” approach. Take all you have, including the newly found hellfire and strike back against Heaven immediately. The second speaker, Belial, the sly flatterer, suggest that they bide their time. The full frontal attack is futile. Mammon, blunt and practical, proposes that they forget fighting G-d and concentrate on building up Hell, perhaps as magnificent as Heaven. Finally, Beelzebub, Satan's lieutenant and stooge, argues that they should battle like terrorists, using guerrilla tactics. He has heard that G-d has recently created “man” and they should pervert this new creation as a way of getting revenge on G-d. Satan, who had that idea originally, stands up, manipulates a mandate for the subversion tactic, volunteers for the hazardous duty on Earth and abruptly ends the meeting. He's able to end the meeting on his own terms because he has convinced his audience of his plan's viability and he has nurtured a strong respect/fear for himself. That's how a meeting should be run. The agenda is clear and understood by all (we've just been whipped, what do we do now?). Let a few people express their opinions, saving your eloquent plan for last. The attendants are usually swayed by the last opinion they hear. Rouse up the crowds and then immediately act upon it. Forget the quibbling, debates, and studies. I'm sure that Satan had plan of revenge before the meeting had begun. It did not arise from a brain-picking session. The meeting was meant only to sell his idea, not to fish for ideas. Brilliantly executed! *“Oh shame to me! Devil with devil damned. Firm concord holds, men only disagree of creatures rational. (Milton, Book II, lines 496-498)”*

II. More Amusing Anecdotes

The attendees of this paper presentation will fill in this section. If you have informative stories involving hackers, abusers, the Man, and new toys, bring it to this conference. If they pass the cut, I'll include them in a possible part II to this paper.

References:

Allison, Alexander W., et al., Eds. The Norton Anthology of Poetry, Revised. New York: W.W. Norton & Co.., 1975

Brooks, David. "Cosmic Capitalists" Weekly Standard, 2(43), 20-23.

Milton, John. The Complete Poems of John Milton, New York: Bonanza Press, 1936.

Shakespeare, William. The Complete Works of Shakespeare, New York: Oxford University Press, 1938.

Tanakh, The Jewish Scriptures. Jewish Publication Society, 1985.

Peter Parker
U.S. Dept. of Commerce
Room 3100
1401 Constitution Ave., NW
Washington, DC 20230
(202) 482-1449
peter_parker@ita.doc.gov