Tales from the Programming Trenches

gene m. stover

created 6 April 2004
updated Sunday, 2013-02-10T14:11:23-0800

Contents

1   Introduction

I've been programming since 1981, including some work for pay in my teens. I turned pro in 1992. I've mostly worked as a contractor, about 9 months on each job on the mean, so I've seen a lot of places, a lot of teams, & a lot of projects since then. Here are some true stories from what I've seen.

The stories are in no particular order, though I've sometimes tried to group related stories into the same chapter.

2   First Job

They say you learn the most on the first six months of your first job. I learned the single most surprising & impressive lesson of my career on the first six months of my first job.

It was 1992, & I was at Notable, a start-up company in Oakland, California, writing application software for Go's Penpoint operating system.[1] The company's flagship application had a team of about five programmers. I didn't get to know them well (I wasn't on that team), but they were nice people, appeared to be good programmers, & definitely worked really hard. They weren't wasting time when they were at work.

Nevertheless, the project fell behind schedule. I think it was originally slated for six months, & we were on about the sixteenth month. Management got so concerned with the schedule slip that they called a meeting of the entire company (18 people – I was number 18) in which they stated their concerns about the schedule. Then they asked the programmers what they needed so they could finish the job in less than six months. Management really did ask this. The CEO was standing in front of the entire start-up & said "So what do you guys need from me so we can ship in six months?" He was serious, & he listed some possibilities: more hardware, development tools, work from home, cots in their offices, catering, another one or two programmers, unusual hours so they could have the entire building to themselves when they were at work? He made it clear he'd get them anything he could afford if it would mean the product would ship no more than six months in the future.

The programmers looked at each other & talked a little, & said there really wasn't a problem, it's just how software is. I think they said they could use a faster build machine, but mostly there wasnt much the CEO could do. Development was proceeding as most software development does: more slowly than anyone expected.

There was one thing the CEO wanted from the programmers. He wanted each programmer on that project to sign an agreement that said he would do his best to get the product to ship in six more months, & if it didn't ship, he would be out of a job & with no benefits. In other words, if they didn't ship in six more months, they'd be fired.

The programmers all signed the agreement, though I wasn't present when they did.

Six months came & went, & we didn't ship on time. No one was fired. Nobody even mentioned that special agreement. It was the most astonishing lesson of my career.

If I remember correctly, we shipped about two months after the six-month deadline. So the project which was originally scheduled for six months took $16 + 6 + 2 \rightarrow 24$ months.

2.1 Pen Computing

As I said, that start-up company in 1992 was writing software for pen computers. There was a whole pen computing industry at that time. We went to conventions that were exclusively for pen computing – not a keyboard in the place.

---

[1] As of 2004, Microsoft has introduced a tablet computer – I don't recall its name – as if it were new, but in 1992, there was an entire industry built around pen-based computers.

So what happened to that industry? Why don't we have a memory of pen computing continuously from the very early 1990s until today (2004)? I'll tell you.

The first pen computing company I knew was *Go*. I believe it was started by AT&T, or maybe AT&T was a financial contributor.

Go made a pen operating system called Pen Point & the hardware that ran it. Both were pretty cool. Pen Point was fully object-oriented & multithreaded. They had two hardware units: the 440 (Thor) and the 880 (Odin). The 880 was about 8.5 by 11 inches by about 1.5 inches thick. (The screen was on the 8.5-by-11 inch part, & part of that 8.5-by-11 inches included a boarder about 1 inch thick, maybe a little thinner than that.) Both units came with a built-in cell phone/modem. All this in 1992.

As the pen computing industry picked up speed, *Go* spun-off a comany called Eo^2 to make the hardware so Go could concentrate on the Pen Point operating system.

That was the time of the Apple Newton, & as the industry picked-up speed, other companies entered the picture. There was the Casio(?) Zoomer. There was some kind of tablet that showed an unappealingly cutsie city scape & had you walk to different buildings to do things like send e-mail or balance your checkbook; maybe it was called "Pen Magic" or "Magic Cap" or "Magicap". There were laptop computers that had flip-tops so you could use them with a keyboard or as a tablet; IBM Thinkpads were one example & Grid's Gridpads were another. There was even a pen-based version of Microsoft MS-DOS. There was our little start-up, making applications for Pen Point & some other systems. The pen computing industry was really moving. Like I said, we went to conventions that were exclusively for pen computing.

Then one day, we got news that Eo aquired Go. It seemed ironic because Go had created Eo less than a year before, but I didn't think it was significant. Within a day or two of that news, a coworker of mine toured AT&T and saw a tablet computer than ran Pen Point & was small enough to fit in your hand; my coworker called it a "Newton killer".

About one week after Eo aquired Go, we heard that AT&T aquired Eo & squashed it the next day. I'm pretty sure we heard this on a Thursday. That was the end of the pen computing industry for a few years.

So what happened? I have an idea.

In those days of pen computing's first incarnation, one of the main advertised features was that pen computers would recognize your hand-writing. Of course, the recognition accuracy was never 100 percent. It was about 95 percent by my gut-feeling estimate. Lots of reviewers who wrote about pen computers said they were neat but that they wouldn't be useful until hand-writing recognition accuracy reached 100 percent.

I have news for you: Hand-Writing recognition accuracy will never reach 100 percent. Do you recognize your own hand-writing 100 percent of the time? Nope. Recognition accuracy can get really good (already has), but it will never

---

[2]Pronounced like "ee-oh" or like the "io" in "cheerio".

reach 100 percent. So if those reviewers who wanted 100 percent recognition accuracy represented average consumers, nobody would be buying pen computers.

The situation probably wasn't as bad as zero sales, but I'd bet money that the actual customer base was turning out to be smaller than predicted, & marketers were blaming customers' desire for 100 percent hand-writing recognition accuracy.

I think AT&T saw this happening, & they decided that the cure was to wipe pen computing from the collective customer awareness & re-introduce it later with less emphasis on the hand-writing recognition feature. So Eo bought Go, & one week later, AT&T bought Eo & stamped it out.

Pen computing virtually disappeared for a while. I saw a Casio Zoomer in a consumer electronics store in 1996 I think, but it had been renamed. Other than that, pen computing disappeared until the Palm Pilot.

Nowadays, the pen computing industry is healthy & growing. Pen computers still recognize your hand-writing (probably better than ever), but notice that it's not one of the advertised features. The feature isn't a secret, but it's not emphasized. You don't want a pen computer because it recognizes your hand-writing. You want one because you can take it with you, & it keeps you in touch with your calendar & the network at all times. The hand-writing recognition feature is de-emphasized so customers will not demand 100 percent accuracy in the hand-writing recognition.

2.2   Pen Point Quickies

The Go/Eo computers that ran Pen Point used a CPU called the Hobbit, which was from AT&T.[3]

Pen Point presented an object-oriented file system to application programs, but it was on top of the FAT file system of MS-DOS.[4] When I say it was an object-oriented file system, I mean it. Every file was an object with its own thread of execution. It's more like the file system was where document objects went when you weren't using them.

Pen Point allowed the user to install third-party hand-writing packages. Some company made such a package called Lexicus, which gets my vote for one of the coolest names ever. I might have to write a software library or application & call it Lexicus just because I want to name my software Lexicus.

3   Right Hand, Left Hand

Does the right hand always know what the left hand is doing? Not in business!

At AirTouch in 1995, our team of application developers thought of a really cool feature we could add to our application which analyzed the wireless phone network's performance. To implement the feature, we'd need to collect some

---

[3]Thanks for Mark Peever for the detail about who made the Hobbit.
[4]Pen Point did not run on top of MS-DOS.

data in real time from the switches.[5] We asked for such a connection. We were told it was flatly, absolutely, completely, technically, totally, & forever more, impossible. Not just "we can't afford to do it" infeasible. We were told it was *impossible.*

In 1997, I started working at Metapath in Bellevue, Washington, which made exactly that type of generic, data-gathering interface to switches. I learned that AirTouch had been a customer of Metapath for years. So even as AirTouch said it was impossible to make a generic switch interface, they already had exactly this type of switch interface.

4   Comments in the Code

At AirTouch, we used header comments in our files, like this:

```
/* History
 *
 * yyyymmdd  who       what
 * --------  --------  ----
 * 19940203  joe       created
 * 19940215  joe       fixed bug #101
 */
```

A lot of the files – dozens of them, maybe a hundred, had a comment like this:

```
 * yyyymmdd  who       what
 * --------  --------  ----
 * 19950312  mark      removed all references to the "f word"
 */
```

Cool.

5   Got Root?

At AirTouch, we had a team of about 35 programmers. It was policy to have the same root password on all the workstations & on the server(s) our team used & to give that password to all the programmers. I refused to accept that password, & I made sure they knew that I hadn't even looked at it.

Why did I refuse the password? Was it wisdom in the risks of security? Nah. I had my own network at home where I had been able to be root all I wanted & learned that, instead of being god, root is the guy who (a) is most likely to delete things that shouldn't be deleted & (b) has to do all the boring

---

[5]In wireless telecom, a "switch" is one of the computers which hold your account information & which control the SS7 network which carries your voice data. The name is historical.

configuration tasks. So I didn't want the root password at work because I didn't want anyone to ask me to do any configuration tasks.

I later learned that four other programmers had also refused the root password.

So about 30 programmers, a couple of managers, & the system administrators had the root password.

One day we arrive at work to find... no source code! Where did it go? The network admministrators did some research & said that at about midnight the night before, root had logged onto the server which held the source code control system & typed "`rm -rf *`". Notice that it was *on the source code control system*, & we used ClearCase[6], & it was configured so that even the source code you had checked out & locked was still on the server. So it was gone, all gone.

We were willing to believe that whoever had done it had done it in error[7], but no one came forward & admitted to it.

Not to worry, we'll just narrow the list down to the handful of people who had the root password... Doh!

Instead of narrowing the list of suspects down to a handful of people, they were only able to eliminate a handful of people from the list. I was glad I was one of them. You get to feel pretty smart & smug in moments like that because you, through foresight[8], have been proved innocent.

For what it's worth, we didn't lose any (or much) source code. The nightly backups had already run when root ran its bungled `rm`, so the network administrators just had to restore the backups. We lost a full day of work for 35 programmers, a couple of managers, & a handful of system administrators, though.

And that is why you don't want to have the root password on any system you don't own & are not paid to maintain.

6   Unfortunate Design Decisions

I have a story for this chapter, but I need someone's permission before I post it. (6 October 2004)

7   Really Stupid Design Decisions

7.1   Making Classes

At one company about 1995, a programmer & friend who was on another project came to me for a second opinion. I can't remember his name exactly, but I think it was Ken or Kent. I'll call him Ken.

---

[6]ClearCase, the mother of all source code control systems. Bad assed & cool, though proprietary & so complicated that I usually just prefer RCS.

[7]At least I was able to believe it; I guess I don't know what others thought for sure.

[8]Wisdom? Yeah, right. I already wrote that I had refused the password more through laziness than security concerns.

Ken was having a design argument with his team lead, so he asked me for a second opinion.

Their problem related to mobile phone units & the number of antenna registers in it, but I'll translate it to cars. If their program had manipulated cars instead of mobile phone handsets, here's what the team lead would have designed:

- Define a class Car.

- Define a class Car-Containing-People & make it a subclass of Car.

- Define a class Empty-Car & make it a subclass of Car.

Now, your program simulates car traffic. When a car object has people in it, you use an instance of class Car-Containing-People.[9] When the people park the car & get out, you change the car to be an instance of class Empty-Car.

If this seems stupid to you, that's because you are a good object-oriented programmer & you can recognize complete & total nonsense when you see it.

The team lead had designed something like this. Ken hadn't liked it from the start, but the team lead had pulled rank, & Ken had gone along with it. During implementation (using C++), they had come to a place in the program where they needed to change the class of the car objects. They were using car objects by value, & they sent a Car-Containing-People object, by value, to some function that created a new Empty-Car, copied whatever fields it could from the Car-Containing-People, & then returned the new Empty-Car. Then they stuffed the Empty-Car's value into the original Car-Containing-People object. I don't need to tell you that the C++ compiler didn't like it, that the team lead tried all sorts of tricks to force the C++ compiler to allow it, & that it still didn't work.

Ken hadn't liked any of this but had worked with the team lead in the hopes of getting the damned thing done & moving onto something else, but at this point in development, they were encountering a problem with almost no technical solution.

Ken & I tried to explain to the team lead that this simply would not work; they should at least allocate the car objects dynamically & pass pointers. We also tried to explain that some attributes of an object should be class members. For example, there should be a Car class, maybe with an "is empty" attribute or a count of its occupants, but not an Empty-Car class & a Car-Containing-People class.

The team lead wouldn't listen. He insisted that he needed the two subclasses of Car. He said to me "What's the point of object-oriented programming if not to make classes?" At that point, I gave up & left Ken to fight the team lead on his own.

---

[9] Hyphens aren't allowed in C++ symbols, but I'll include them here so LaTeX can break these words across lines. It'll be more readable by humans, & we're not going to feed these class names to a compiler, so it won't hurt.

7.2 Singletons & Threads

I was once on a project that had been designed by its first two programming team members. They were both new programmers who had recently read *Design Patterns*[**?**], & as people besides me have said, new programmers & design patterns are a bad combination.

These programmers had been dazzled by design patterns. In their mind, if you could claim a particular design decision was an implementation of a particular design pattern, then your design decision was a good one.

They had been especially dazzled by the Singleton design pattern. The program was littered with singletons. They were everywhere. Need a list? Make a Singleton subclass of class List. Need a button? Make a Singleton subclass of class Button.

What's more, they threw threads at everything in the name of run time efficiency. It seemed like every object had one or two threads dedicated to it, & all the threads busy-waited. The program's designers didn't know how to block on a mutex; they only knew how to poll it.

Here's an example of what was in the application:

The application communicated with a custom hardware device over a 9,600-bps serial port. They had created a class, instances of which were responsible for said communication. Did they create an instance of that class parameterized by a file descriptor to the open serial port & be done with it? Of course not. They made a Singleton. If I remember correctly, they for some reason kept the real functionality out of the base class & inserted it only in the Singleton class so you would have no hope of ever removing the Singleton nature. They put a dedicated, busy-waiting thread within that Singleton. They had a dedicated, forever-looping thread just to handle a 9,600-bps serial port.

That Singleton which monitored the serial port had to communicate with the rest of the application somehow, so they made two queues. One queue was for messages to the serial port object. The other was for messages from it. By itself, that would have been reasonable, but of course each queue was a Singleton. Each queue had two threads. One thread handled requests to retrieve objects from its queue. The other thread handled requests to insert objects into its queue. All the threads busy-waited.

Notice that already we have several Singleton objects & five forever-looping, busy-waiting threads to handle a single, 9,600-bps serial port. A busy-waiting thread always consumes all available CPU cycles. These guys had five of them just for one 9,600-bps serial port. They claimed this arrangement was necessary for performance, but the application had horrible performance, & they ignored me when I pointed out that my old IBM PC XT running MS-DOS & Minix was able to handle a 9,600-bps modem without busy-waiting & without threads at all.

There were other Singletons in the application. Like I said, they were all over the place, usually with one or two dedicated busy-waiting threads inside them. All this for a program which was mostly a GUI that hid some communication with a 9,600-bps serial port.

It could have been a simple application that could have made few demands on the CPU, but because of all the threads, it would always consume all available CPU cycles on any computer, even the fastest piece of hardware in existence.

And the Singletons were just plain stupid.

7.3 Report AG34

At one company, there was a program called "AG34". It was an hourly report that required 65 minutes to run.

Think about that.

My friend, Horton[10], was writing a program that needed the output from AG34 before it was old news, so he needed to make AG34 run in less than an hour. He dug up the source code for AG34. Then he showed the source code to me because I needed a laugh.

The main tasks of AG34 were to read records from a text file, remove some of the records, & concatenate others because logically they were one record though they were stored in the file as two consecutive lines. The records were stored as one line of text for each record (standard unix stuff).

AG34 handled this simple task in over 6,000 lines of C code with no comments. As I read through the code with Horton, I many times had trouble deciding whether to laugh or hurl.

AG34 would read some records, then sort them. Then it would copy the un-sorted list over the sorted list. At other places, it would allocate memory to hold a bunch of records, then copy the records into the new chunk using a manually written inlined chunk of code that copied byte-by-byte – & sometimes didn't copy enough or copied too much, then free the chunk of memory or throw away the pointer to it without freeing it. It would call a function that would scan, sort, copy, or mangle an array of records, then that function would call yet another function which would do more of the same, or would throw away what its caller had done. Somewhere in all that mess, the program or its programmer had forgotten to do its main task.

Remember, this program has a simple task that got lost in 6,000 lines of crap.

AG34 had been written by a programmer named Darmage.[11] I had a cruel streak in me in those days (about 1994), so I found Darmage at his desk & told him, with my most friendly tone of voice but with very unfriendly words, what crap he had written. It's amazing how much insult you can dish to a person if you do it with a smile.

Meanwhile, Horton re-wrote AG34. He did it in plain C with nice, readable comments. About half the lines were comments. His new AG34 was 60 lines of code & ran in less than 5 minutes.

Nowadays, I probably could have rewritten AG34 in even fewer lines using Perl or Awk.

---

[10]His last name really was Horton.
[11]How do you pronounce Darmage? It sounds like Damage.

8 Microsoft

I've done a couple of contracts at Microsoft. Both contracts ended badly, & I disliked the work environment. My friends have since said that they will not allow me to work at Microsoft again. Friends don't let friends work at Microsoft.

8.1 Programming Conventions

At Microsoft, if you want to know how to use some function within Windows, you are expected to read the source code to that function to learn how it works & how to use it.

Excuse me, but this is called "the code documents itself" & has been known for decades to be a terrible software development practice. Frederick Brooks mentions it in the first edition of *The Mythical Man-Month The Mythical Man-Month* ([**?**]).

8.2 Longhorn

As of 2004, the next unreleased version of Microsoft Winders is code-named Longhorn. I worked on it.

There's a rumor that the source code for Winders is 35 million lines. That's almost right. The 35 is right. The actual size as of 2002 is 35 gigabytes of source code, though "source code" in this case includes resource files, configuration files, & maybe some binary object files because they tend to check their binary files into their source code control system.

Once you build all of Longhorn (as of 2002), it occupies almost 100 gigabytes of disk space, if I remember correctly.

When I started my Longhorn contract, the build labs could do a clean build of the entire operating system overnight. I guess that was about a 12 hour build. When my contract ended six months later, the build required about 18 hours. From the hints I gathered, they usually upgrade their build hardware about every 9 months, which halves the build time, returning it to less than 12 hours.

The main entry point into the build system is called "build". It or one of the programs it runs & which does most of the work of compiling the system, is a Perl program as of 2002. Perl really rubs people the wrong way at Microsoft because Perl wasn't invented at Microsoft, so I'm sure that by the time anyone reads this, the program will have been replaced. But for a while, for more than a few years, Microsoft used a Perl program to build Winders.

8.3 Xbox

Game consoles are sold at a loss. The companies make money when customers buy games. At least that's how it is with Xbox. For every Xbox sold, Microsoft needs to sell about 2.5 games to break even. I presume it's about the same for Sony & their Playstation 2.

Because consoles are sold at a loss & the companies who make them do not want you to buy a console & use it as a low-cost general-purpose computer. Xbox contains lots of modern security features to prevent you from running software on it other than games signed by Microsoft. I'm not nocking their decision; it makes economic sense.

On the other hand, many people do want to buy a game console & use it as a low-coast computer to run Gnu/Linux. And you know that someone, somewhere, somehow, will figure out how to bypass any amount of modern crypto-fascist security features & run whatever they want on it.

So how did Sony deal with this situation? They realized that they can't stop it, so they sell a Gnu/Linux kit for Playstation 2, presumably at a price that covers the loss at which they originally sold the Playstation 2 console. This could be considered a Taoist approach. Instead of expending more effort to prevent people from using a console as a low-cost computer, they made it easy & legal for you to do it by selling a kit.

In a meeting, I suggested Sony's solution to Microsoft's problem of Linux on Xboxen. Microsoft can't do that with Xbox because they hate Gnu/Linux & other forms of unix so much. Their not-invented-here pride prevents them from taking the practical way out (selling a Gnu/Linux kit), so they have to spend more effort to put more security features into Xbox & even more security features into Xbox 2. In this case, pride prevents practicality.

While I was on Xbox, I found an article on Slashdot or maybe elsewhere on the Web that explained how to download Linux onto your Xbox. Even after that, people at Microsoft would insist that "nobody has hacked an Xbox" and "nobody has run Linux on an Xbox".

8.4   Xbox Hong Kong Special

Somehow, I ended up with a special Xbox on my desk. A legitimate Xbox will only run software that has been cryptographically signed by Microsoft & which is also on a DVD (not the recordable kind; the pressed kind). So if a piece of software isn't signed, an Xbox won't run it, or if you copy a signed, commercial game onto a recordable medium, the Xbox won't run it because it's not on the correct media. I also heard that the legitimate medium was tweaked (bad sectors or highly refletive regions or something) to make it difficult to make an accurate copy, anyway.[12]

This special Xbox had been hacked in Hong Kong & would run pirated software. With this special Xbox, I had also been given two or three pirated games, & it ran them fine. (A normal Xbox wouldn't run them.) The special Xbox would also run legitimate Xbox games.

I talked to different specialists in my quest to figure out how the box had been hacked. What was weird was that the DVD specialists insisted that it was physically impossible for an Xbox to read a recordable DVD. They said the

---

[12]Developer Xboxes are less strict about the signatures, but they won't run a game that is cryptographically signed as a normal game for consumer consumption.

color or frequency or reflectivity of a recordable DVD was wrong so the DVD drive in the Xbox couldn't scan it; the drive couldn't read the data to give to the Xbox's CPU in the first place. Their point was that even if the firmware had been hacked, an Xbox couldn't read the software on a pirated disk. Yet I had this hacked Xbox, & I could demonstrate it reading a pirated game disk. Even then, some of the specialists insisted it was impossible.

By the way, by mixing & matching parts, I demonstrated that the DVD drive in the special Xbox had not been modified at all. The hack was purely in the firmware.

8.5   Java Script in Xbox

When you boot an Xbox without a game disk in it, you see those green control panels. That's called the *dash*, & it's a Java Script program. It's not pure Java Script because the Java Script interpreter can utilize Xbox graphical & sound features via a foreign function interface (FFI) & because Microsoft insists on calling their Java Script interpreter "Jscript".

8.6   Santa Crude Operations

A week or two after SCO announced its ludicrous claim that it owns Linux & that the whole world is in violation of a license agreement, Microsoft gave a bunch of money to SCO (whether as a donation, as buying stock, as a loan, or whatnot, I don't remember). This occurred in the first half of 2003. At the time, I mentally filed it was Microsoft stirring up the ants, & I thought it was actually kind of a good little joke on Microsoft's part.

At Microsoft, I talked to a large hand-ful of employees who speculated that Microsoft was originally behind SCO's silly claims. These are people who work for Microsoft & like it, yet they speculate that Microsoft had inspired SCO to raise their ruckus.

8.7   Compiler Directives

Early in one job at Microsoft, I had inherited a small application written in C with Visual C++ & MFC. I compiled the program from scratch to make sure I could, then uploaded the `*.cpp` files to my home unix system so `indent` could beautifuy them for me. Then I downloaded them back to my Microsoft Windows computer at work & compiled, but the compiler puked all over them. I couldn't figure out what was wrong since I had only indented the code. Go figure. (And `indent` did not mess up any "`::`" C++ tokens; there weren't any such tokens. The program was mostly plain C.)

The error messages from Visual C++ were not helpful in tracking down the error.

A coworker helped me out. Where I'm a unix guy who admitedly doesn't have much experience programming Microsoft Windows, he was a long-time Windows programmer, so I thought he'd be able to find the problem quickly.

Even so, it took the two of us almost an hour, but he did find it, which impressed me, considering what the problem was.

The problem was that Visual C++ (or maybe the resource compiler or a preprocessor) recognizes directives within comments. I don't remember the exact directive, but it was something like "//{". My `indent` program had inserted a space in those, & the compiler had croaked on it (but had produced a misleading error message). The old programmer explained that Visual C++ (or the resource compiler or some other compiler, whatever) used directives within comments, & it was not very forgiving about formatting.

As he left my office, the old programmer said "Any system that puts compiler directives in comments is fundamentally fucked".

8.8 Robocopy

Do you know of the `xcopy` program on PC-DOS? On unix, we don't need it because we have `cp`, but on DOS, where the alternative is `copy`, `xcopy` is pretty nice. I don't think it's used often on Windows, where people more frequently copy by drag-&-drop[13], `xcopy` probably isn't used all that often, but it's still there[14] Like I said, within the PC-DOS world, `xcopy` was a respectible little program.

At Microsoft, they told me not to use `xcopy`. It seems there is a subtle bug in it. I don't remember the exact bug, but it was something like if you copy a large number of files across a network, `xcopy` might forget to copy some of them, or maybe it forgets to report an error if some of them can't be copied.

The bug had been around for over a decade. Did someone at Microsoft fix it? Nope. Instead, someone at Microsoft wrote `robocopy`, & they use that.

Did they fix `xcopy` itself? No; they continue to ship a known bug to customers. Sure, it's a minor bug, but it's easily fixed. It's already fixed in `robocopy`.

What business logic could have made this decision? What does it say about a company? Is this an example of how free markets are good for the consumer?

By the way, I don't know the answers to these questions; I'm actually asking them (but you don't need to send me your answer).

8.9 Windows Services for Unix

Microsoft has a software package called "Windows Services for Unix". I think it's an internal-only product, but maybe it's distributed externally, too. I don't know for sure, & the details don't matter.

If a software package is called "Windows Services for Unix", you'd think it would provide for unix features that are in Windows? For example, maybe Windows can cook eggs, & a unix can't, so "Windows Services for Unix" might provide the egg-cooking feature to your unix. Right?

---

[13]...or by loading something into Microsoft Turd, then saving it to a new file. It makes me cringe.

[14]Still there in Windows XP as of Saturday, 12 June 2004.

Nope. "Windows Services for Unix" provides features such as `ls` & `grep` for Windows.

Let's see: Programs such as `ls` & `grep` are standard for unix-like operating systems. They aren't common among unix-like operating systems; they are present on every unix-like operating system. And "Windows Services for Unix" installs these programs on Windows.

So does "Windows Services for Unix" provide Windows' features to unix? No. It provides some of unix's features for Windows.

Why is it called "Windows Services for Unix"? To mislead, or was someone at Microsoft very, very confused?

What was really amazing is that I met programmers at Microsoft who insisted that "Windows Services for Unix" added some much-needed Windows-like features that unix had for ages neglected to offer. They would tell me this even though they had "Windows Services for Unix" installed on their Windows computers. Did these programmers begin life in a state of double-think, or had Microsoft taught them?

8.10   Microsoft & Free Software

(By Free software, I mean Freedom. It is not inaccurate to use Open Source as a synonym.)

Microsoft hates Free software. This is not news.

Why does Microsoft hate Free software?

You might think it's because Free software is often better software & lower-cost software so it'll take some of the market share from Microsoft's products. Maybe this is one of the reasons Microsoft hates Free software, but it's not the main reason. After all, Microsoft products captured their huge market share even though they are inferior products – even inferior to other proprietary products. So fear of superior products does not scare Microsoft all that much.

When I have worked at Microsoft[15], my coworkers & managers knew that I was a Free software programmer. They would ask me almost daily if I had inserted some Linux code into Windows. When they asked, they were mostly joking, but there was a tiny amount of seriousness in their voices, too.

I'm almost certain that the main reason Microsoft hates Free software is the Gnu Public License agreement, also called the GPL. Microsoft is afraid that some programmer will insert some GPLed code into Windows or some other Microsoft product. If that happened, & if it was found, Microsoft could find itself in court. If they could show that it was a true accident or the act of a saboteur, maybe they could leave court without paying a fine, but what if they couldn't prove either of those cases, & the amount of misappropriated code was large, & maybe they got a little unlucky when the judge was appointed or the jury was selected?

The GPL scares Microsoft.

---

[15]I've worked there twice, hated it both times, & refuse to work there ever again.

9   Linux vs. Windows Costs

When I started working at Metapath in 1997, they put a Linux box under my desk. That computer probably cost less than 1,000 US dollars.

That was my development machine at Metapath. I wrote & tested a fairly large software suite on it. I turned it into a Web server for the documentation, & it was the distribution server for my software suite. I also used it for e-mail & Web-surfing. During this time, that computer required no system administration at all. I didn't upgrade the operating system or the hardware, & it worked just fine.

In 1999, the company upgraded their Windows users to Office 2000.[16] Office 2000 bogged-down all of those computers so much that they became unusable. I was skeptical when a coworker told me her computer was unusable, but as I watched over her shoulder while it took fully five minutes to login, & even more time to launch e-mail, & then the mouse cursor moved jittery, I realized that "unusable" was not an exaggeration.

Half the desktop computers in the company were suddenly unusable.[17] They rolled-back some of those Office 2000 installations. On other systems, they upgraded the memory to the max, then the CPUs, then the motherboards. They contacted Microsoft to help with the problem. Two weeks later, they still hadn't solved it, so they rolled-back the rest of the Office 2000 installations.

Most of those Windows computers were less than six months old, & their operating systems were up-to-date. They had a dedicated I.T. department to manage them. That I.T. department worked overtime for two weeks, which ran up costs. Almost half of the employees (i.e., those who used Windows) were performing sub-optimally because their computers were unusable. New hardware was purchased, installed, reconfigured, & reinstalled. Oh yeah, & there was the cost of purchasing all those copies of Office 2000.

On the other hand, my own Linux box – & I know it wasn't the only one – had been a one-time cost of 1,000 US dollars, & fully two years later, it was a perfectly functional system. It didn't have light duties; it was used for e-mail, development, & as a server. It hadn't required new software, hardware, or maintenance costs. It really was a one-time cost of 1,000 dollars, amortized over two or more years.

Compare this to the cost of the Windows machines. The hardware may have cost 1,000 initially, just like my Linux machine, but the software had cost more, the hardware had required upgrades, the machine had required professional maintenance, & the Office 2000 upgrade had cost not only the license fee for Office 2000 but a two-week productivity drop across almost half the employees.

When I noticed that, I was amazed.

I am still amazed that the accounting department didn't notice, & accounting departments at other companies don't appear to notice, either. People say that accountants are bean counters. Well, isn't this the kind of cost difference that

---

[16]I think it was Office 2000. Maybe it was Office 1999. Whatever.

[17]The other half were Linux & other unix-like systems.

you want your bean counters to notice? You'd want them to notice it & start asking questions to figure out why the difference was there.

Anyway, so Linux (& probably other unix-like systems) cost a heck of a lot less than Windows systems.

10 The Dumbest Project Manager in the World

All of these stories are about a project manager I encountered at AirTouch from 1995 to the middle of 1996. I'll call him Baltar, though that wasn't exactly his name, but it's pretty close.

In spite of the unbelievable amount of stupidity & irrationality in these stories, & in spite of how I tell it as though I was practically innocent of the whole thing, I will swear to any god of your choice that these stories are literally true. I might remember some of the dates & durations incorrectly, & I know I don't remember the order of events very well, but other than that, it's all true.

10.1 First Impressions

Originally at AirTouch, I was working on the "frequency assignment problem" which was inherent in early cellular networks. It was the flip-side of the short-range benefits of the short-range transmitters that made cellular phones possible. I recognized the problem as probably being NP-complete.[18] No one else knew what "NP-complete" meant, so they told me to write a program that would solve the problem. We called it the Frequency Assignment Program (FAP).

After working on FAP alone for about a month, they paused FAP & told me to research & recommend a development framework. That was a learning experience all its own.[19] Four months later, I delivered my recommendation & returned to FAP.

FAP Resurrected now had a project manager, Baltar. He said I had been working on it for six months without delivering, so I needed someone to manage me. He actually said that; the boy needed a class in human relations, if you ask me.

I pointed out that I had only worked on FAP for a month. The other four months were on the project to research & recommend a development platform. (I had only been at AirTouch for five months, so his claim of six months was bad math at best.) Baltar said "You still didn't deliver after six months". I sighed as I realized that I would be sighing a lot in the future.

My program was rolled into a larger effort managed by Baltar, though I never, ever was able to learn any other features of that larger effort.

---

[18]I read a few years later that it had been proved to be NP-hard. I don't know if it's been proved to be NP-complete.

[19]It wasn't a *bad* one. It was a useful learning experience, & maybe I should write about it here.

10.2   The User Can't Fork

My program would treat the frequency assignment problem as an optimization problem. It would eat a description of a cellular network. It would grind away at the problem with a genetic algorithm, searching for better frequency assignment plans. Whenever it found a plan that was better than the previous plans it had found, it would output that plan. Sure, at first it would spit out lots of crap, but if you let it run for a few days or a few weeks, it would produce good plans at less frequent intervals. That's how genetic algorithms work.

The user interface was simple: A small window with a Go button. Press the Go button, & it would optimize frequency assignment plans, displaying the improved plans & writing them to an output file. While it did this, the Go button would be a Pause button. The program would run until you pressed the Pause button.[20] With Baltar as the project manager, I started working on this.

I told him about the program's user interface. He asked how the program would be able to do its optimization while monitoring the Pause button. I said that when you pressed the Go button, it would `fork` a subprocess that would do the optimization while the main program monitored the GUI.

Baltar said that the user wasn't technical enough to deal with multiple processes, especially ones created with such a low-level facility as `fork`. I pointed out that the user wouldn't be aware of multiple processes, much less that one of them was started with `fork`; the user would only know about the Go/Pause button & some configuration items. Baltar repeated that the user would be non-technical & unable to deal with multiple processes. I explained again & again, each time in a different way, that the user wouldn't be aware of multiple processes at all, but every time, Baltar said exactly the same thing: "Ah, no. The user will be non-technical & will be unable to understand multiple processes". I tried explaining things at least ten times, but he always said the same thing. So I gave up trying to explain.

10.3   I Didn't Read It

Baltar asked me to document my proposal. I didn't mind; I had documented it five months before. He also asked me to add a place where he, the development manager, & I would sign it to show that we all agreed & that I was obligated to deliver.[21] I spruced up that document & gave copies to Baltar & to the development manager. All three of us signed it.

Weeks later, when I was experiencing a form of Creeping Featuritis Creeping Featuritis ([?]) worse than any I could have imagined, I reminded him that he was making changes to a proposal that he had approved by signing. Baltar said he had never really read the proposal, so his signature "didn't count".

---

[20]If I were doing it again, I wouldn't even give it a graphical user interface. It'd be a command line program that printed improved plans to `stdout` in a machine-readable form – CSV.

[21]I felt like I was being accused of something that was never specified. I still feel like that.

10.4   Creeping Features

Have you heard the programmer's apocryphal stories about the dangers of adding features to a program's specification as you develop it? It's sometimes called creeping featuritis. Frustrating as it is, I would have been relieved if the creeping featuritis I had to deal with were that simple.

As I described in The User Can't Fork section The User Can't Fork ([**?**]), my program would have a really simple user interface. It was the first time anyone had tried to deal with the frequency assignment problem this way, so I felt that a simple interface was good enough to allow us to test the computational idea.[22]

I swear, I swear, I swear that for the time during which Baltar was my project manager, *at least* once a day, he would tell me that the concept for the program had changed. Sometimes, he would want it to be a spreadsheet. Other times, he wanted some kind of free-form natural language interface that I never did understand. Sometimes he wanted it to display the results on the screen. Other times, in a file or in a database (whose schema changed each time he described it to me). Sometimes it was to be a frequency assignment plan optimizer (as originally planned), but other times, he wanted just an editor or programs that displayed frequency assignment plans in different ways. I think he once wanted some kind of alarm system to warn if someone tried to change a frequency assignment plan in the database.

These weren't features that he sometimes wanted included or excluded. These were entire concept changes. It's as if you were programming a spreadsheet & your boss told you that he wanted it to be a word processor instead. And I swear, he did this every day, sometimes more than once a day.

I couldn't talk him out of these wacky changes, so I reminded him that they would set us back weeks because I had been writing one type of application, & now he wanted me to write another. He said that a good programmer wouldn't lose any time because he could re-use code.

10.5   If I Need It, Read It

The most ridiculous, hilarious, & painfully irrational incarnation of the application in Baltar's mind was a distributed, cooperative frequency assignment plan editor.

Baltar's idea on this day was that a user at our offices in, say, New York, might be editing some frequency assignment plan which was stored in the database. At the same time, a user at our offices in, say, Los Angeles, might edit the same frequency assignment plan. Baltar's idea was that if both users changed the same thing in the frequency assignment plan, then the program would figure out which change was "better", & it would put that change on both screens.

Nowadays, we have collaborative software, so the idea doesn't seem totally stupid (but it still seems mostly stupid), but collaboration wasn't Baltar's idea.

---

[22]Nowadays, I wouldn't have suggested a graphical user interface at all. It would read & write files, with configuration data in the form of command line options.

Let me put it into perspective with an example. Let's say you are editing a monthly budget spreadsheet that is, for whatever reason, stored in a database instead of a file. At the same time, another user is editing that same spreadsheet. If you make a change to the spreadsheet that improves the monthly budget, the program displays that change on the other user's screen. If the other user makes a change that improves the budget, that part of your spreadsheet magically changes to reflect that change. When cells on your spreadsheet apparently change on their own & without warning, for how long will you remain sane?

I tried to talk Baltar out of this unimplementable idea, but he wouldn't listen, so we went to the development manager. He was a sane guy, so I figured he'd tell Baltar to stop changing the application daily & to let me get some work done. He didn't. After listening to Baltar & I each describe the situation, the development manager looked at me & asked "So what's your problem?"

I tried to explain how dumb the idea was (not in those terms), but he didn't get it, so I pointed out that it would be even more difficult to implement than the program I was writing & which was already behind schedule. In particular, the interprocess communication (IPC) would take too much time.

Baltar agreed that IPC would take too long for me to implement[23], but he said it had to be done.

I asked if I could do it by having the processes communicate through files.[24] Baltar said I couldn't use files because he didn't think files would be a good way to have two processes communicate.

So I suggested a TCP socket between the two processes. Baltar vetoed that because it would take too long to implement. (I agreed.)

So I suggested communication through the relational database. Baltar vetoed that.

I couldn't think of any other way for two processes to communicate, so I asked Baltar what he suggested. Here's what he said, as near to a word-for-word quote as I can remember ten years after the fact:

> "We have two users editing the same frequency assignment plan, right? So if one user makes a change that improves the plan, then the other user needs to see that change. He needs that change. So if the other user makes a change & I need it, read it."

I couldn't believe this non-sequittir, so I asked for clarification. He repeated himself word-for-word, especially that "if I need it, read it" part.

To double-check that I had understood what was going on, I asked one more time, something like this: So these two programs must communicate with each other, but I can't use files, the database, sockets, or any other kind of IPC?

Baltar said that was correct.

Without sarcasm, I asked if he could suggest some technique by which the programs could communicate & of which he would approve. Baltar said I would have to figure that one out.

---

[23]He had some way of saying that which implied I was incompetent. At least that's how it sounded to me.

[24]I've since used that technique many times & never regretted it.

I looked at the development manager for help, but he said that I complained too much.

I sighed.

10.6 How Long Will It Take?

At one point, Baltar had me in a meeting with about ten or fifteen upper-level managers. I don't remember who they were or the topic of the meeting, but Baltar spent most of the time presenting some new feature of our application suite, or maybe it was a whole new application in the suite. I don't remember.

During most of the meeting, I wondered why I was there, but then from the front of the room, Baltar asked me how long it would take to implement this feature (or program or whatever it was).

I thought about it for a few seconds, then said it would take eights weeks.

Baltar said "Ah, no. We need it in two weeks."

I shrugged.

Baltar asked if I could do it in six weeks. I thought for a few seconds & then, with reluctance, said "Maaaaaybeeee".

Without missing a beet, Baltar said "Then we'll split the difference & call it four weeks." The meeting was over.

10.7 Plagiarism

At some point, my program needed to connect to the database that described the cellular network. In fact, I had gotten the program to this point originally, before Baltar became the project manager, but they wouldn't let me see the database schema. I didn't even need to see the production database. I just wanted to know what the schema was like so my program could read data from the database. They said no.

During the time that Baltar was the project manager, I stopped asking to see the schema & designed my own, including a way to map mine to the existing schema, if I was ever allowed to know it. I made a good start at documenting it, & then I described it to Baltar. It was probably the only idea I had & which he liked.

I said I could start programming it as soon as I finished documenting it & got the development manager's approval. I figured that could be done by the end of the day. Baltar said there was no time to lose. I should start programming right away, & he'd put the finishing touches to the documentation, then e-mail it to the development manager & some others. I said I could finish the documentation myself, but Baltar insisted.

So Baltar asked me to e-mail him the documentation I had so far. Then he took the whiteboard, where I had drawn my ideas, from my cube wall & walked down the hall to his office with my full-sized whiteboard under his arm.

About an hour later, Baltar sent an e-mail to the entire department announcing the new database schema he had designed. He included my report,

verbatim, with no finishing touches, except that he replaced my name with his. It was a textbook case of plagiarism.

I didn't blow up. Instead, I went to Baltar & politely suggested that he add my name next to his in the credits, since I had written the original draft. In fact, he hadn't made any changes at all to my draft, so had written the entire thing. I phrased my request to avoid conflict.

Baltar raised his voice & told me in no uncertain terms that I was a credit-hog. He talked about how the entire thing was a team project, so nobody's name should be on any of our publications.

Minutes later, Baltar sent an e-mail to the entire department in which he announced an updated version of the schema report. The only change, which he did not point out, was that it was credited to "The FAP Team".

A week later, he changed the credits back to Baltar.

10.8   The Schedule Dictates Reality

Baltar loved schedules. He had a detailed Gant chart, & every morning, the entire team of six-or-so people would spend an hour listening to him explain the detailed dependencies in the schedule. Every morning.

My part of the project was divided into something like ten main phases. They were things like GUI, database interface, computation, & event loop. I don't remember all of them exactly.

Each of those phases was scheduled for a certain amount of time, something like one week or two weeks. Whenever a phase was scheduled to end, Baltar would ask me if it was done. Between all the meetings, interruptions, & irrational design changes (Section 10.4), of course it wasn't done. Nevertheless, Baltar would tell me to stop working on that phase & start working on the next. If the schedule said that someone was done & that something else was in progress, then I stopped working on the first something & started working on the second something. Period.

10.9   Why I Quit

Every week or two, as part of Schedule Worship, Baltar would order me to drop the unfinished phase I had been trying to work on & start working on the next phase. Eventually, I was on the final phase, & I needed to make use of what I had started but never finished in the other phases. None of the previous parts of the program were finished because Baltar had always ordered me to drop them & move to the next phase so that I would be doing what the schedule predicted I would be doing. I needed those things now, but none of them were done. Baltar's habit of sweeping problems under the rug had come back to bite us.

I had long since learned not to begin to take the iniciative, so I decided to ask Baltar how he'd like me to handle this problem. I didn't want to talk to him directly because I knew we'd fight. So I went to our new non-technical

development manager[25] I told her the whole story of how Baltar always had me stop working on one phase & go to the other so that what I did matched the schedule even if I wasn't finished. I explained my current problem & how I needed to know what Baltar wanted me to do about it, & how I didn't want to ask him myself because I didn't want to get in a fight. I didn't say what I thought of his management skills & strategy; I told the story in as neutral & innocent way as I could because I knew what was coming.

So the non-technical development manager asked Baltar & I to meet at my desk. There, she explained the situation to Baltar. It basically came down to this: Gene is at a point where he needs to go back & finish some of the phases that have been unfinished, & he'd like to know which phase you want him to finish first.

Baltar replied "Gene's plate is already full, so let's not give him more to do".

She explained again, emphasizing that I could not proceed until some of the earlier phases of the program were finished first.

Baltar replied "Gene's plate is already full, so let's not give him more to do". Yes, he repeated himself, verbatim.

She explained to Baltar a few more times, each time with him replying that "Gene's plate is already full".

Eventually, she looked at me & shrugged.

I dropped my badged, blew some raspberries at Baltar, & walked out for good.[26]

10.10   Summary

In retrospect, that whole FAP project was a death march, & the signs were there as early as the first month, when I had the program ready to use real data from the database, but they wouldn't allow me to access that data. I mean, if you are told to write a program but that you can't access the exact data that the program will use, not even a dilluted version of that data so that you can test, then you have an impossible task in front of you. Baltar just made the worst of it.

11   Brilliant Idea

On one project, I was asked to write a program that would eat data of some type, do some simple correlations on the data, & stuff the results into a database. The team leads said I had to do it in C++, & I needed to do it post haste because we were already two weeks behind schedule.

I thought the C++ part was a less-than-optimal idea. C++ isn't the fastest language for development, we were already behind schedule, & the program

---

[25]The development manager in the If I Need It, Read It section had become the technical development manager.

[26]Two weeks later, I started a new job at Celcore in Tennessee. It was good.

```
#! /bin/sh
# $File: silly-sql$
for I in 0 1 2 3 4 5 6 7 8 9; do
    echo insert into plus10 values \($I, 'expr $I + 10'\).
done
echo go
```

Figure 1: A silly example program that sends SQL statements to standard output

```
$ ./silly-sql |sqlplus -l gene -p password
```

Figure 2: Example of using `silly-sql`

didn't need to do much work at run-time, so I wanted a simpler technique that would require less development time.

As it happened, I had been experimenting on my own with Bourne shell programs that manipulated a database by sending plain, vanilla SQL statements to standard output. They were easy to write & had really good performance in spite of what other people predicted. Figure 1 shows a silly example script that uses the technique,

Figure 2 shows how you might use it.

I thought I could use this technique for this new program.

I asked the development manager for permission to try this technique instead of writing a single C++ program. He asked how long it'd take, & I said it would take about two hours. He told me to give it a try.

I was wrong. I didn't take two hours. It took seventy-five minutes, including the time to get one of the testers & the resident database guru to give it a pretty thorough, unofficial check & run a performance test.[27]

In the weekly progress meeting about an hour later, the team leads congratulated me in front of the rest of the team. We had been two weeks behind schedule, they said, & now we were back on track thanks to Gene.

The project admin asked how she would record it in the log. I said to call it the Brilliant Idea. Everyone nodded.

After the meeting, I asked the team leads what I'd be working on next. They said I'd be writing the C++ version of the program. I was stunned. The Bourne shell program worked just fine. I asked why I'd be re-writing it in C++. They said it was because the Bourne shell script couldn't possibly run fast enough.

I sighed.

---

[27]If my memory serves correctly – & it might not – the script processed over 30,000 records per hour. Whatever the number was, it was plenty fast enough.

12 Newspeak

("Newspeak" is a term from the novel *1984*, by George Orwell.)

I wrote the technical whitepaper for one of our products, called PageCount. Months later, the marketing department asked me to customize it for non-technical peeps. (Let's ignore the question of why non-technical peeps should even look at a technical whitepaper. And yes, there was already an advertising brochure.)

I expressed my disapproval, but I agreed to do it. Here is the very beginning of the "dumbed down", non-technical, technical whitepaper that I produced:

> **Executive Summary**
> The PageCount System (PCS) consists of two applications – PageCount & PCLUA. PageCount monitors printer usage on a network & sends the usage report to Print, Inc. PCLUA helps Page-Count monitor a non-networked printer as if that non-networked printer were on the network.

(I made other changes, but they don't matter now.)

I delivered it to the marketing department, & they sent their more-or-less final draft back to the I.T. department for review. Here is how they rewrote my introductory sentence:

> **Executive Summary**
> The PageCount System (PCS) allows Print, Inc. to continuously monitor asset usage within a customer's Document Management Infrastructure (DMI). On an ongoing basis, it enables deployment optimization and continuous improvement to ensure that the DMI is fully supportive of changing business needs and requirements.

Be sure to read the second sentence carefully & think about what it means. Notice the split infinitive.[28]

Exercise for the reader: Try reading the second sentence out loud in a single breath.

13 How Fast is Your Compiler?

Back in college, about 1989, Doctor Nico gave some kind of a presentation in the computer lab. I can't remember what the presentation was except that it wasn't all students from a single class. There were grad students & newbies in the room. Maybe it was some kind of "Introduction to unix" lecture.

---

[28] I missed one point on the final exam (an essay) in a high school English class because I split an infinitive. I received a 99 percent on that exam, which was a respectable second highest grade in the class, but I will forever bitterly remember the rule about split infinitives because my friend & constant competitor, Troy, got the best grade, which was 100 percent.

The presentation was on a Sequent Symmetry with 8(?) Intel 386 cpus at 25 or 36 MHz.[29] It was running Dynix, which was a Sequent flavor of unix that offered both System V & BSD semantics, sort of at your choice.

Anyway, so Nico shows us the source code for some Pascal program. I don't remember what it did, but when I saw how big the source code was, I thought that we'd be there for a while if he wanted to show us how to compile it.

He ran the compiler on the large program, & before you could know what happened, he had a prompt. Instead of the minutes of compile time I had expected, it took no more time than "`echo hello world`".

Nico said "I'm convinced the Sequent Pascal compiler compiles by look-up". Heh heh. It still cracks me up.

14 Dead applications never die; they just fade away

At Print, until about a year ago, we had this program called sas. No, not the famous statistics application. Our sas was an in-house web app. It helped salespeeps track, predict, & report their progress through the sales process for their prospects.

There was a management turn-over at the end of 2004. Naturally, the new management threw out everything done by the old management. The salespeeps encouraged the new management to toss out sas because it was too restrictive or something like that. They never had liked it.

Since then, we've gone through a complicated process (which did not involve me but did involve my friends here) in which we reviewed & selected a new sales automation system. We chose Sales Force.

The sas project manager & one of her programmers became Sales Force customization experts. They did not complain about this. From what I've seen, they've put full effort into their new assignment & are now pretty much Sales Force administrative & customizing gurus.

So the company has been using Sales Force for about three months. We purchased the system; this was not an audition. (We auditioned systems months ago.)

You see where this is going, right?

Today[30], the sales department, backed by management, said they wanted some non-trivial changes to Sales Force. They want it to look & operate totally different. They want it to work "like sas did".

15 Users are so cute

Paraphrased from a Help desk request on Thursday, 2006 May 11:

---

[29]Such a machine sounds silly today, but it was plenty fast back then.
[30]Today as I write this is 2006 February 24.

> I've only worked here for a week, & already my inbox fills with spam daily. How do spammers get my e-mail address so fast? I've only given it to HP customer support, some of our customers, Amazon, e-Bay, Google, & some web sites I read every day.

Another Help desk request (on another day) said "Can you please give me full access to every site on the Internet? Immediately?"

16 Why test?

I was swapping test tool hatred stories with a team mate. He topped all of mine with this one:

> He tried to run some tests against a new of the application. It turned out that the test tools themselves were incompatible with the patch & prevented his tests from running. He reported this to the test tools department. They were surprised & asked "Why are you testing the new version? It's not even released yet."

As the blood drained from my face, he said "I swear to god it's true!"

17 Random quotes & short notes

I overheard a (very sarcastic) analyst yesterday as she said "We're the company of scalable, repeatable processes which are specially tailored to each customer".

A Glossary

A.1 creeping featuritis

There are definitions at

- http://catb.org/ esr/jargon/html/C/creeping-featuritis.html[31]

- http://c2.com/cgi/wiki?CreepingFeaturitis[32]

References

---

[31]`http://catb.org/~esr/jargon/html/C/creeping-featuritis.html`
[32]`http://c2.com/cgi/wiki?CreepingFeaturitis`