My 50 Years in Information Technology

Introduction

I feel that my career has been one of wonder in the lands of Information Technology and Computing, and I want to share how the breadth and depth of experiences led me through life, and the hard earned, hard learned lessons I can leave behind for others to learn from.

There are many quotes and sayings about IT, and they exist for good reasons. I hope you haven’t had to learn any of them the hard way, but then again, I don’t trust anyone in IT that doesn’t have a drawer full of bloody t-shirts. I have a steamer trunk full by now.

One aspect of IT that has always confounded me is the lack of second or third level use of technology. I have long been a fan of using the computer in front of me to assist me in doing my work. Automating tasks, building code and data generators, and even automating documentation.

This has now come to the fore as the use of ChatGPT and other AI tools has enabled many of the engineer level techs to get their work done far more rapidly than ever before.

As I go through this story, I will mention 4 instances of Code Generators that I have used or written, and if you want those gory details, they are in my “History of Code Generation” story also available in my repo.

The classic definition of IT (from 1958) is “techniques for processing, the application of statistical and mathematical methods for decision making, and the simulation of higher-order thinking through computer programs”.

How prescient that is considering the state of AI these days!

The computing power used has ranged from Mainframes (which keeping cropping up across the decades), to the era of the Minicomputers, to Desktop computing, to “Blades”, and now Cloud and all the options there for services and platforms.

IT has been one of the “revolutions” that changed society. Somewhat for the better, somewhat for the worse as we are now seeing with the dis- and mis- information campaigns being waged on social media sites these days.

The advent of ERP systems for Just-In-Time manufacturers was an absolute game changer across the entire ecosystem of manufacturing. Online banking and the 24 hour ATM network has certainly been a boon to many people. And provided much humor in the early days when the ATMs were called “Yuppie Food Stamp Dispensers”. Yeah, that was a thing for a while.

Think about the automated systems you use every day. From banking to online purchases and even government services like auto tag renewals.

When’s the last time you had to write a paper check?

## Who am I?

I am a 71 year old nerd. Proud of that. I’m also a private pilot and budding machinist. Learning all about Metrology and finding out that our math teachers misled us – ½” and 4/8” are only the same in math class. No where near the same in Metrology.

Way back in the 1950’s when I was a child, we didn’t have all the diagnoses we have today, so I was just known as “precocious”. My warning label has always been “Sarcasm Ahead”. My parents knew that Christmas toys with “some assembly required” should just be left in the box as I would immediately take tools to them to disassemble everything to figure out how it worked. No reason to assemble something I was just going to destroy.

My parents also learned to buy fuses in bulk, but that’s another story. Mom used to tell people that my super curly hair was from “sticking everything he finds into wall sockets”.

She also went with me on the first day of school every year. She would just introduce herself to the teacher, go ahead and schedule for the 1st parent-teacher meeting, and then wish the teacher good luck.

Christmas of 1963 brought me a Digi-Comp 1 – described as “first real operating digital computer in plastic”. It was a 3-bit binary computer that could be “programmed” with plastic pins for True and False, with more pins acting as the “set” or “reset” for each bit. It came with a programming manual showing you where to put the logic and set pins for each program.

You can read about it on its Wikipedia page at <https://en.wikipedia.org/wiki/Digi-Comp_I>, and you can sometimes find them on e-bay. Minds on Toys sells a hardboard version and can be ordered from <https://mindsontoys.com/kits.htm?dc1_main.htm>

My love of all things computers was born. I did every experiment in the manual, then got my mom to order the “extra programs” book.

I still have my Digi-Comp 1 – it is at the office, programmed to count in binary from “000” to “111” and I use it to show new developers how binary logic works.

In The Beginning

I was fortunate to be in advanced classes with wonderful teachers and early in my High School I had a teacher/mentor in Mr Ferguson. He got me work as a math tutor at a local university for pocket money.

Somehow, he had gotten the school to purchase a Wang 300 “programmable” calculator for the science lab. There is still information on them at

<https://www.oldcalculatormuseum.com/wang360.html>

They used a pre-scored “IBM card” that you punched out with a paper clip, dropped into the “toaster” card reader, and the calculator would “run” the punched instructions. That was another fun introduction to programming.

Being a Floridian, I love the note on that website:

“…

These cards are similar to the (in)famous "hanging chad" punched cards used by Florida in the 2000 Presidential Election.

…”

Remember – this was before the “hand held” calculator era. I aced my “Slide Rule” class from him as I had my dad’s 50cm K&E that he used in the Navy in WWII. All the other kids had the little 8” or 10” ones from the bookstore. Years later, I donated that slide rule to a friend’s museum display.

As I did well with coding for the Wang 300, he then got me to enroll in a computer-based Explorer Post hosted at a local computer college. That is where I learned ForTran (written that way as the actual name is Formula Translator) on an IBM 360/20 and my first program went live in 1969. The love of all things computers was cemented. I knew I had to work in IT. And I did well enough in their post that I was able to take classes at night to start my formal IT education. So I started taking college level classes at night while still in High School.

And yes, that was state-of-the-art way back in the early 70’s. The mainframes I started on still had magnetic core memory – and discrete components – at least they were past the vacuum tube stage.

This was also the Punch Card era of computing. Developers wrote their code on coding sheets, then sent them to the Keypunch operators. Then submitted their compile jobs – decks (or boxes) of cards with Job Control Language and their source code to the computer room to be run on one of the mainframes.

And the only “network” we had involved sneakers. We did have early modems – they were “Acoustic Couplers” ( <https://en.wikipedia.org/wiki/Acoustic_coupler> ) that worked by you dialing a rotary phone, placing the handset into a pair of rubber cups on the top of the modem, and hoping the sound came through clearly. But that was point-to-point communications only. And ran at a blistering 150 baud. New units doubled that to 300. The final version was 1200, but as Hayes modems were released, the Acoustic Coupler died out. Modems rapidly improved from 1200 to 2400, and grew through 4800, 9600, 14400, 19200, 28800, and finally the massive 56K. My desk has a 10 GB modem in it. My house had a 1GB fiber internet connection.

“baud” refers basically to digital bits per second. Don’t think that baud/8 equals bytes, as it most certainly did not. It was really a timing clock, and a byte of data required 1, 1.5, or 2 “start” bit times, then the 8 bits of data, and 1, 1.5, or 2 “stop” bit times.

And that byte of data had to be surrounded by other timing events and possible data bit.

Effectively, “silence” for 1 bit time; start bit, 7 or 8 data bits, 1 parity bit, stop bit. The parity bit would either be the 8th bit of the byte, or an additional bit. Meaning about 12 bit times per byte.

That’s 50% overhead.

So baud/12 is closer to the number of data bytes per second throughput in one direction.

And of course, the 2 modems have to talk to each other, and be in sync, so they add more overhead.

In 1975, my friend Lee was able to purchase one of the sweet Altair 8800 kits. That was a blast to play with and knowledge of it helped me just 4 years later in my real job. Yeah, our first “home computers” were from kits you had to solder together and troubleshoot with electronic test equipment. We all owned soldering irons, voltmeters, and oscilloscopes. Still do.

And the “technology” side was pretty small. We had Assembly Language, COBOL, and ForTran. And the language was chosen based on the needs of the application. And the sad part is that even back then the developers were segregated by language. You were either a business COBOL developer, or a scientific ForTran developer.

I’m sorry, but I believe that a mechanic should know every tool in the toolbox.

Starting in the 1970’s

When I left school in 1972, I lucked into a job at the IT division of a telco, working in the section called “Data Control”. This was the interface between the developers, customers, and the computer room. Developers submitted their rubber-band wrapped, boxed, or even trays of cards to be submitted as “test” jobs, or “production compiles” through the front window, we logged them in, delivered them to the computer operators, and they (eventually) returned the output listings, new punch cards, and jobs back to us. We returned them to the developers.

You guys with IDEs and instant compiles know nothing of waiting hours or days for a compilation to come back with a syntax or keypunch error.

We also handled all the production reports -removing the carbon paper from multi-part forms, separating them into pages, cutting the customer phone bills down into the envelop sized “pages”, and every other task to get from that input through the process to the output.

On one fateful night, I was working Data Control by myself. A guy comes up to the window, says his name is Jack (remember him!) and announces he is our new IT Director and wants a tour. I asked him for his company ID. He said he didn’t have any. I asked him how he got past the guards at the guard desk. He said he told them he was the new boss and they had to let him pass, and they did.

I said that was their problem, but my instructions were to NOT let anyone in without proper company ID or prior authorization. He argued with me for quite a while, but finally left.

The next morning, after leaving at midnight, I was called and ordered to report back immediately. In I went and was ushered into his office.

My Boss:

“How dare you not let him in?”

“He had no ID”.

“Do you know how much trouble you are in? You could be fired for this!”

Jack:

“Shut up. He’s the reason ***you*** aren’t fired. The guards are, I want a new security company by tonight. Louis can teach physical security to everyone else”.

Add a “Security IS your job” bloody t-shirt to the collection.

Now, the funny part is that he came at night. Had he come during the day, and walked around the building, he would have learned something scary. To explain, the facility was in an old building that had been an auto dealer/repair shop. The rear of the building had a one-bay “Loading Dock” with a 3 foot drop “freight elevator” to get parts/equipment out of trucks and onto the shop floor. With the raised floor in the computer room, it was only 2 feet from that surface to the outside loading dock.

During the day, both inner and outer elevator doors were usually open, with the deck parked half-way between them. This enabled the operators to hang out and enjoy smoking and working. Yeah, back then, smoking was still that common. So the front door had a guards desk and the back where the computers were was wide open. Yep. Security Theater, even in the early days. Far too much of this new-fangled “Cyber Security” is still theatric in nature.

I soon moved from Data Control to “Junior Computer Operator” on the test machine - an IBM 360/50 with a whole 512KB of core storage, 4 tape drives, and 8 disk drives. Those hard drives were a massive 29MB each. What a beast that machine was. So I was the one loading the programs into the card reader, processing the job, getting the reports off the printers, and handing it all back to those new folks in Data Control. Learned what all the hardware parts were, how to load disk packs and tape reels, read and punch card decks, and all the Operating System commands to manage the mainframe.

This was also the machine that the revered “systems programmers” used to generate new Operating System images. “SYSGEN” was the process, and being the inquisitive type, I started by asking “Can I watch”? sure. As the engineer worked through the process, he would read the reports, modify the configuration deck, and try again. I watched, and I asked questions. After a few, I asked “can I try”? He laughed but gave me a chance. I didn’t get it right the first time, but he realized that my error was because they were trying something new, and he hadn’t told me about it. One simple change, and I made my first OS image!

I was soon promoted up to the Production Operators crews. Running on the big IBM 360/65, 768KB, 12 tape drives, and 16 disk drives. Monster of a machine. We worked 12-hour shifts, but only 3 days. And we rotated from day to night, and from Mon-Wed to Thur-Sat. This meant that we frequently worked 6 days straight, but also frequently had 6 off. The company at the time also had a weekend warrior schedule of 2 16-hour days that had the requirement of being ONLY for full-time college students to put themselves through school to become developers for that division. Imagine that – a company recruited students, supported them through their education, and gave them developer jobs upon graduation. Doesn’t happen anymore.

At one point, there was a major blow up with a developer and his manager about the computer room refusing to run his test job. It had been submitted a week prior, and nothing had been done.

Yep, the job was sitting in the systems job queue, it just wouldn’t start it. So the Systems Programmers got involved – they dumped the job out of the test machine, and loaded it into the production machine. Ignored. They “drained” (told the system to stop running new jobs, let everything finish) the system, put all other jobs in the queue into “HOLD” status, and released only the test job. Ignored. System would NOT run it. So they printed it off, and went to figure out what was wrong. They came back with the developer and his manager, and brought them into the computer room.

“This is our production machine. As you can see, it has 12 tape drives. This is the test machine, and as you can see, it has 4. Add both together, we have 16 tape drives. Your job requires 20. We don’t own that many, and since this is for the Hawaii Data Center who only own 4 tape drives, it won’t run there at all. Go redesign it.”

Yeah. Developers (and users!) thinking systems are magic and have infinite capabilities go a LONG way back…

Add a “Computers are NOT magic” bloody t-shirt to the collection.

An important lesson I learned from that time was based on what happened, but not exactly how it happened. As the IT division of a Telco, one of the most important jobs we ran was the bi-weekly payroll. These were for union employees, so you can understand how critical it was to get it done, on time, and absolutely perfect.

The good news was that a couple of developers had to be on-call, preferably near by the night of payroll to fix any issues. The bad news is that they usually chose to spend the evening in the bar across the street.

I had the pleasure of reading the core dumps to them, and helping to keypunch corrections to the jobs and code at 2am with drunk-on-their-ass developers.

The lesson I learned was to ***write dumb as dirt readable code*** that you could understand and fix at 2 am in that condition. Didn’t care if it wasn’t pretty, wasn’t elegant, wasn’t obfuscated – the simpler, the dumber, the easier to fix.

Only 2 years later, we moved to a new facility on the 7th floor of the First Florida tower. The move was scheduled over a weekend, and the IBM team asked for volunteers to help with the move. Taking mainframes apart, moving them, re-assembling them? COUNT ME IN!

What a learning experience that was. This part should probably be in the classified stories, but I’m putting it here anyway. The number of trucks required was slightly miscalculated, and there was one large channel frame (6’ high, 6’ long, 3’ wide) that wouldn’t fit. So 3 of the IBM guys and I rolled it 4 blocks to the new building, up the ramp, and into the new facility. Yeah. Rolling a $250K computer frame down the streets of Tampa as the bars were closing. Probably not a smart move, but we did it.

The facility was huge, and we grew by adding an IBM 370/158 with a sea of disk drives and a room with 2 dozen tape drives in it. And it had a whopping 1 MB of core storage! We were hot stuff at that level. 4 high-speed printers added to the print farm. And I kept learning and was taking Assembly language courses at the time. The operating system was new and required more educational classes. And of course, I was teaching new computer operators how to do their jobs and all aspects of working in a major data center. Always learning, always sharing knowledge.

This is also when I turned the mainframe into a PC kit computer and got thrown out of the course, but that’s another story. You can read that in my Computer Emulation story.

However, it did get me moved from Computer Operator to Systems Programmer. I started being the “go-to” guy for Cobol Developers who needed someone to read “core dumps” to them to help figure out failures and bugs. Got into it a few times with a couple of the more arrogant ones “My code is perfect, the computer is wrong”. Uh, no. Let me prove it to you.

The Rise of Minicomputers

Technology changed in the late 70’s as minicomputers came into mainstream usage. I had heard about several of the vendors, but the company had not purchased any yet. That was about to change.

### The R&D/Prototype project

Along came 1979, and I was summoned to the office of one of the senior VPs on a Friday right after lunch. I figured I was in serious trouble, as did everyone who knew I had been summoned. I was informed that I was a known troublemaker and also known for making things happen and fixing things when others couldn’t. I wasn’t in trouble; I was about to be challenged.

This is basically my conversation with the Dan, the Sr. VP of our IT division:

“…

We have a new project – it will be running on a DEC PDP 11/70 and no one knows anything about them, but Jack recommended you for the job – so are you interested? Sure. Ok, well, you start tonight at 5:30. There is a semi due to deliver it after hours. We need it up and running by Monday morning, and there are engineers from DEC scheduled to work all weekend with you. Make it happen. Keep me informed. [ that was a mistake on his part ]

…”

Yeah. We did it. Got it powered up and the engineers signed it off at 0300 Sunday. Being an asshole, I followed the “keep me informed” instruction and called Dan figuring he had been out partying Saturday night and would be nice and hung over and really appreciate a 3AM wake up call. Then I went home to crash hard. Went in at 0500 on Monday, grabbed the manuals, and installed the O/S. The team arrived at 0900, and the software engineer from DEC asked me what I was doing typing on the console.

“Learning this Operating System and how to work it”.

“But I haven’t installed it yet. That’s why I’m here”.

“I already installed it”.

“How?”.

“You left me alone with a computer, a box of tapes, and a box of manuals. No problem”.

“How did you know how to work the front panel switches?”.

“Oh, my buddy Lee built an Altair years ago. The front panel is just like it”

And yes, the PDP 11/70 front panel was very much like the old Altair 8800 (or IMSAI) kit computers.

Set the switches to an address, select, set to data, store, repeat until code loaded, hit run. Amazing where you can learn something that will help you later in your job.

There is now a Raspberry Pi based replica that you can buy and build your own:

<https://obsolescence.wixsite.com/obsolescence/pidp-11>

Search a bit and you can even find VT-100 terminal replicas that you can print the case on a 3D printer and get all the electronics to emulate.

The software we were running was a global experiment, but due to legal issues, I can’t give details. However, it did get me a chance to work with a very different type of software development shop in Europe.

They didn’t have a computer.

Read that line again. Software Development without a computer.

And the code we got from them was ***flawless***.

How the hell?

They wrote the code in long-hand (a holdover using coding sheets). When a developer was ready for his code to be peer-reviewed, this is what happened:

The entire team got into their “test room”. Covered in white boards, wall-to-wall, floor-to-ceiling. The test lead wrote all the constants and variables on the first board, and they walked the code, line by line, writing on the boards, tracking loops, modifying variables, basically hand-executing the entire module. For every condition. Could take hours to days. Only when they came out in agreement that it was ready did the code get sent to the computer center to be entered on a computer and compiled.

I didn’t see quality like that again until I was doing DoD work after 9/11. A sales weasel asked one of the DoD guys about bugs in production – the answer was “18 years ago we had one”.

[ I really wish I could tell that whole story, but it is classified and really embarrassing to the weasel ]

How do you feel about code quality these days?

We upgraded from the PDP 11/70 to a brand-new VAX 11/780. Moved to our own offices, the team was transferred to the parent company (with guaranteed jobs in the IT shop afterwards), and I was running the PDP and VAX, acting as the data center manager, architect, developer, network engineer, and everything else.

As one of the partner companies had just acquired Telenet, we installed a massive 9600 baud modem on a B-1 line (2 pairs of telephone lines) and put our server onto the X.25 network. It could be reached from anywhere via dial-up access. [ <https://en.wikipedia.org/wiki/Telenet> ]

One day, I’m up at the Corporate HQ, working directly with the Chief Scientist of the parent corporation. An incredible man. In the room are me, him, my manager, and a VP pest.

As I’m working with the CS building the demo he will be giving, the VP is getting on everyone’s nerves. He makes the mistake of asking me what he can do to help.

I respond without moving, or stopping my development of the demo, I keep typing and deadpan say:

“Sure. On the table by the window is my little green toolbox. Open it up, and in the top tray is a glass cutter. Use it to cut yourself an egress in the window”. Damn if he didn’t reach for the toolbox before realizing what I said and stormed out in a huff.

The CS stands up, turning away from me, hugging the wall, and exits the room. My manager goes into panic mode “YOU IDIOT YOU GOT US BOTH FIRED”

5 minutes later, the CS’s secretary comes in, closes the door, and says “What happened? He is upstairs in his office laughing his ass off. Said that was the funniest thing he had ever seen. He’s on the phone with HR getting the VP removed from the project. He’ll be back in a bit”

Upgraded to Intelligent Systems 3600 series color terminals, and I got to attend training on setting those up, adjusting the high-voltage in the CRT, aligning the 3 color guns in the CRT, and troubleshooting the hardware. Yeah, back then, we just had to handle it all.

We then went down to Miami for the Interface ’80 expo and demonstrated our prototype. Got quite a bit of good press and reactions to it.

Anyway, as I said at the start of this story, “legal” happened, and that project closed. Back to a normal IT job.

Nope.

### The Employee tracking/project costing system

As the research project came to a close, Dan and Jack called me. That duo always had challenges, so here we go again…

Jack says to me that he suspects supervisors are “hiding” employees to help get them through projects and causing cost over-runs but he can’t prove it. He wants to restructure the entire division, but doesn’t have the true details of employee headcount and hierarchy.

Jack asks if 2 of us (Frank and I) from the project would take on building a new system that would track all employees, their cost (salary + overhead), the hours assigned to every project, and the actual projected costs of building them. He says there is a manager he trusts, and he’ll have us moved into his group.

Sure. Sounds like fun, and Dan says “Just take over the DEC VAX and terminals. They are surplus now”. So I set up a data center room, dismantle and move the system, reinstall the system, get it up and running, and Frank and I start building the employee/project tracker. I wired 2 dozen terminals in a work room for the Supervisors to use for data entry. Yeah, I soldered together the 25-pin serial cables by running 50 pair telco cables through the ceiling. We did everything back in the day.

A little bit about Franks coding skills. His first 2 questions about any new language were always:

“How many dimensions can arrays have, and how are they addressed?”.

The man could visualize data and slicing/combining data across multiple vectors in ways I never understood. He would read every bit of information into the dimensions of the arrays, and instantly produce reports that seemed to have nothing in common but the final values.

So Frank and I architect a solution that would define every employee, the reporting hierarchy, project assignments by hour, and the requisite reports. I designed a workflow that would calculate the fully loaded employee costs, overhead (supervisor/manager/director/VP) of each division, then apply those to the hour based project plans, and deliver the data to Frank for his magic arrays.

That workflow had 7 major operations for each departments data to go through. The program had a rudimentary GUI (still using mosaic graphics!) that displayed the # of departments in each step and the ID of the department currently being processed in that step. The users could enter their data for phase 1, then watch the console for that data to complete, then enter phase 2 information. Remember – we were running on a minicomputer with a whopping 256KB of memory. Had to run programs in small steps.

We built it in 90 days, and Jack announces to everyone in the division that they are to contact us, schedule time to enter their data, and work with us to complete the annual budget and project proposal cost sheets.

Sure enough, several supervisors take us aside and ask how to hide extra resources. They are pissed to find it that blocking that is why the system was built.

With the TRUE cost of every team and project fully projected, the client was able to select the ones not quite worth the cost, and eliminate them.

Jack got the data he needed and was able to do the clean-up and re-org he wanted to accomplish.

And this is another lesson learned – I have architected and engineered many systems to track WTH is going on as senior executives didn’t have the views into the operation that today’s tools give them.

This was during the time of “Management By Walking Around” and “Management By Objective” which were (failed) attempts to change how projects and teams were controlled.

Jack had fun with it as he assigned my manager the objective of “Get Louis to stop calling assholes “assholes” at least to their faces”.

And one of the supervisors tried writing me up – actually was stupid enough to put into writing – “Louis is hard on incompetent co-workers”.

Maybe a normal job now.

Nope, another call from Jack.

### The “How Not To Do It” project.

“Hey… since you are available, I got a project in California that just can’t seem to get into UAT. They can’t get it to install at the client site. They have failed 3 times. I need you to go figure out why. But the team can’t know you actually report to me”.

“Not a problem, I still have my ID from the parent company, I’ll use that”.

Fun. Sure. So I go to LA as an “HQ auditor” for the contract and installation test. The project team loads up their tape, copies the installation file, then attempts to run it. Fail. They don’t know why.

***THEY DON’T KNOW WHY***

I ask to look at the console, and ask them all to step outside. As I’m currently carrying an ID from the company that owns their company, they do so, not sure of who I actually am. I had my tools from the users group in my bag, so I installed their version of the missing software, tested it, removed that page of the console log, and asked them back in.

“Try it again now”.

The script ran. Product installed.

Client can begin their testing, I again ask them to step outside, I remove the tools I used for the installation, and the team returns to the development shop.

Manager asks me what I did to fix the install. “I didn’t fix it, I just let it run one time” I ask to see the contract for the project development. The contract specifically states the project is to be written in COBOL. The developers had built some of the code in ForTran and the client system didn’t have that compiler, so of course their install failed. Worked in Development as that system did have COBOL, ForTran, and Pascal installed.

Another bloody “But it works in Dev” t-shirt.

Asked for the requirements manual. Read the requirements, and wrote up my findings. I still remember my opening statement:

“The process is well documented, the details of every report seem to be complete, however, the input specs are conspicuously absent”.

Said my audit findings would be submitted, and I would be leaving.

Copy of report delivered to the director who was expecting it, and to Jack with my notes on the ForTran fiasco. Jack was not pleased. He asked if I could go figure out how the hell the developers wrote code in a language that was against the contract and how management was not aware of it.

The director was then told that the auditor had been asked to do an in-depth analysis of the project, and would be working embedded in the project for 90 days, and had been moved into his group, reporting to him.

I won’t go into all the gory details, but just the front and back problems.

Front – the data to be acquired had to come from dial-up 1200 baud modems. (this was long ago…). Finally found that the reason the input specs were not documented is that the amount of data to be collected every 5 minutes on these modems would take about 10 to 15 minutes to transmit. That’s pretty much a project killer. No wonder the input specs were conspicuously absent.

Back - Then, the highly complex math-based reports, written in COBOL, (not a complex math language) – well, I had the head of the PMO mock up an hours data, and run it through the reports.

Note that this was to be a “real-time” traffic analysis system. ***REAL-TIME***

The mock-up of ONE HOURS data took 7 ½ DAYS to get through all the reports.

At the same time, I was going in on Saturday nights doing forensic analysis on their computer, which happened to be another DEC PDP 11/70 (which is why I had the compilers) and I knew that one inside and out. I found a smoking gun hidden on a disk pack at 3AM one Sunday morning. I was SO pissed I called Jack. It was 6AM for him. I told him that I found the smoking gun, and he said he and the director would be there 1st thing Monday morning.

They get there and the Director asks me what I think I’m doing and who do I think I am to demand their presence??? Jack says “Shut up, he doesn’t work for you, he works for me”.

Jack asked what I had. I told him that was a surprise for later, but they needed to learn a few things for themselves first, and I had it set up. I gave them a list of questions, sending Jack to the manager, and the director to the project supervisor to get their answers.

They came back, and I said, now watch this – I called in the Project Tech Lead, and said, now ask him and compare the 3 answers. You guessed it, nothing matched. What the team was ACTUALLY working on was not what the supervisor was reporting, and not what the manager was reporting.

The supervisor and manager were both in full-on CYA mode reporting to keep their empire intact.

This is why transparency and sharing of project status is SO critical!

EVERYONE needs to be using the same tracking tool, or have them tightly integrated so the stories are NOT different!

Jack had an early lunch appointment, so I said I was saving the best for last, and we’d get together when he returned.

Got into the conference room, and brought in the PMO guy who ran the reports tests for me. I laid out on the board the math showing the data couldn’t be collected as required, and then had him present the timing to run all the reports.

Jack thanked them for their help and cooperation with me, dismissed everyone but the director from the room, then turned to me and asked “What about the smoking gun you called me at 6AM for?”

That part is tied up in legalese, sorry.

After giving them the details, Jack sits there stunned for a few minutes, then looks at me and says “Get. Me. To. The. Bar.”. I knew that look, and that tone.

2 martinis later, he finally speaks to the director. “I expect your retirement letter on my desk when we return”. “Louis – is there anyone I shouldn’t fire over this?”. “Yes, the tech lead who was open and honest, and the PMO guy – he reported the problems to me, and did all the research I needed”.

And that folks is how an entire IT division can be shut down and everyone lose their jobs.

And the lessons to learn from this one:

The technology could not support the physical transfer of the required amount of data.

The language was wrong for the mathematical calculations required for the reporting.

There were no open and true communications in the project.

And another “Communication is Key” bloody t-shirt added to the collection.

And another “Architectural Prototypes are MANDATORY” bloody t-shirt added to the collection.

And another “You need ALL the requirements” bloody t-shirt added to the collection.

Since we mentioned requirements….

# Ambiguity and Requirements

One thing about requirements is the need to remove all ambiguity from them. One of the lessons I used to give Business Analysts that I was mentoring was focused on this, by demonstrating it. I have frequently done this in a room with the BA’s and the client in attendance. The lesson goes like this:

===============================================================================

In order to find the real requirements of a system, we need to talk to 2 different groups of users:

1. Those that know the job
2. Those that know the job

Did you all notice that I used the exact same words for these wildly divergent groups of users? I have completely different meanings for these 2 – and this is an example of getting to the bottom of the real requirement by eliminating the ambiguity.

The first group of “Those that know the job” are the ones who know it from the point of view of WHY it exists, the PURPOSE of its data and reports, and how it serves the company and its users.

The second group of “Those that know the job” are the ones who know it from the point of view of:

I click on this bookmark

Navigate to this page

Enter the customer’s name here, the address in this field, “12” in this field, the phone number here….

See the problem here? See the loss of institutional knowledge demonstrated by this response?

This can be traced to the “worker” users who learned the system from the guy that learned it from the guy who learned it on his own when the guy that knew it walked out one day.

That damn “12” response stops everything. Go back to group #1 and find the purpose, meaning, and usage of that field. No system relies on a CONSTANT being entered by every user. So you research it, you have the DBA analyze the database and find out the last time that field wasn’t a 12… and help the business regain its lost knowledge. Then properly document the requirement for that data and its entry.

==============================================================================

# Rise of the network and creative users crashing it.

During this time, “networks” became a thing. From the old “Windows for Workgroups” to Novell Netware (installation from hell) to the modern LANs of today. The adoption of the internet in the early 80’s and the adoption of languages like PERL and PHP in the 90’s for web work were changing the face of the IT landscape.

Of course, along with new technologies comes creative users who crash it in unexpected ways. My favorite 2 from those early days:

### Network to Crawl Speed

I get a call from one building that their network was barely functioning. We jump on our monitoring tools (such as they were) and see 100% traffic. Trace it to a very creative user. He wanted more space on his local hard drive, so he moved his SWAPFILE to the LAN drive. Enough tech knowledge to be dangerous.

### Best Bottleneck Ever

The second time, the traffic load dropped drastically and was not performing at all. There was almost no traffic. Traced that to another user who decided to define his PC as a “network router” so the entire building’s traffic was trying to go through his 2400 baud modem.

## Serving Department needs

As networks grew, so did the usage of “Department” based servers. In the beginning, each vendor had pretty customized operating systems, and we all went to their classes to learn to use and program on each system. Data General, DEC, HP, Tandem – all the same, just different. Even the mighty IBM got into the “mini” game with their System 34 and A/S 400 series machines.

Another bloody t-shirt. Learn to learn, implement, and forget.

Languages started growing again. Pascal became popular due to the Wirth Pascal compiler, which was a 2-phase implementation. The “Compiler” produced a p-code file, and the “run time” would then process that file. The purpose being that code could be compiled on any machine, and then run on any other that had the run-time for that host. Sounds a lot like Java and even newer languages.

Even the mainframes had added new languages – PL/1 (Programming language 1), RPG (Report Generator), APL (A Programming Language) and many scripting languages as well. Clist and REXX for online scripting using the Time Sharing Option (TSO) with terminals directly connected to the mainframe.

The rise of Unix came along, and all the vendors (well, the ones who managed to stay in business) migrated to using it on their minicomputers.

The proliferation of software solutions was born. A real “Wild West” of development days.

Formal Mentoring

Back to a normal job helping departments with issues. Nope. On to a new phase of my career – Mentoring, Tutoring, and teaching missing fundamentals.

“Hey Louis – I got a division that needs you. Expect a call”. Thanks Dan. I think.

Sure enough, I get the call that one of the VP/General Managers has a group that has a few needs.

## Management Mentoring

First up – “We have been told that you will NEVER serve as a manager, but will you mentor our new DBA Manager? She has never had a group this large, and needs guidance. Jack and Dan both said you could do it”. Sure. Thanks guys.

So I met Patti. I sit with her in her office, and get to know her. Immersion mentoring is a fun thing to do, and involves a lot of analysis of how and why a person works the way they do, and how to guide them and hone their skills into what needs to be done. I was fortunate enough to get to do that again years later with a Senior Software Engineer who was being groomed to become a true Enterprise Architect.

One early observation was that she allowed herself to be overwhelmed. This has the easy answer of getting an assistant. If you are in charge of a 100+ resource group, trust me, you need help. So we reached out to our college support team and requested an intern. We got an incredible resource in a young lady named Pom. Pom eagerly learned to aide and assist Patti, dealing with nuisance emails and messages, and left the real managerial work for Patti. She quickly became adept at wrangling the cats that make up a Data Base Administration group (a lot of Divas and Snowflakes).

We built the group into a powerhouse and crown jewel in the VPs division. Took a few months, but we got there.

## Shared Resources Require a Shared Plan

Second – “Louis, the document management group seems to be having problems delivering on time. Go fix it”. Sure.

Here we go again, dealing with a lot of confusion and a manager, 4 supervisors and about 50 tech writers. After a few days, I realize they don’t have a central project plan documented anywhere, even though they share resources on projects. “Get me an intern”.

With a nod to “The Association”, “Along Comes Mary”… a wonderful young lady studying CS at the local college who I take under my wing and teach the fundamentals of Microsoft Project Plan. I task her with helping to document each of the supervisors’ groups and what they are working on, and what they have assigned to various shared resources, then we merge it all into a master project plan showing the actual usage of those shared resources. It winds up as a printout 5 pages wide, 20 pages long. As we are talking about getting it printed, Mary asks if she can take it home and her little ones will help her tape it all together into one giant rolled-up printout. Absolutely.

I go reserve the Presidents Conference room for the following Monday morning from 0900 to 1200. I inform the supervisors and manager that we will be meeting there and to not be late. They asked how I got the room as no mere mortals are ever allowed there. I just laughed and said “his secretary knows me and was glad I only wanted the room”.

So we get into the room, Mary rolls the project plan out on the huge table (hence the need for this particular room) and I ask if any of them has a calculator handy. One says he has one at his desk, and he will run to get it. I grab a marker, and stand at the white board.

By the way, just to remind you, I love being an arrogant bastard and asshole at times. This is one of them.

He comes back, and I make a show of asking him to calculate 24 times 7. I make a point of writing it large on the board:

Hours in a Day: 24

Days in a Week: 7

Total hours in a week: 168

I then point to the plan.

“You just told me that there are only 168 hours in a week, yet you have resources scheduled to ***work*** up to 352 hours each week. Care to explain? Think that might be why you are missing every deadline? I will leave Mary with you all to work on the project plan. Get EVERYONE down to no more than 50 hours per week. Then tell your projects when they can realistically expect their documentation.”

I reported my findings to the VP/GM. She was neither surprised nor happy.

And another “Communication is Key” bloody t-shirt added to the collection. You can NOT share resources without a shared plan!

And another “learn the tool before you try using the tool” bloody t-shirt added to the collection.

## SQL Fail on reporting

Third challenge - another call from the VP/GM. “Hey, we bought a company out in Phoenix, and they are having problems with running reports. Go check it out”.

I find that every SQL query for a report starts with the same massive 4 pages of selection code (where field1=b or field1=c or field1=d and field2 in (…)) with hundreds of compares and hundreds of entries in the “IN” list. THE SAME CODE. Yeah, that’s not how you do it folks. You do that on data insertion and set reporting flags so the reports have a one-line selector.

We spun up a new project team to re-design the database, add the requisite reporting flags, and re-work every query and report. Fortunately, we had one hell of a Data Scientist (before that term became widely known) who had architected and built an amazing reporting tool that “learned” the database and provided a GUI to easily generate reports.

And in the “IT is a small world” realm, he was a classmate from high school. He and I were in a few of the same AP classes.

## My reputation can be leveraged

My manager (Bob) calls me into his office at 0900 one morning, says just sit there while I make a call. He calls a group and basically says (well, screams):

“You guys have been delaying this fix for weeks. When I get back from lunch today, one of 2 things will happen. I will have confirmation from you that it is fixed, or I will send Louis with orders to fix it and leave dead burnt bodies in the hall. Do you understand me? Good.“ Slams the phone down. Laughs. “You get that?”. “Yep, and happy to fix it. I will keep the body count down to those that need it”.

Damn if it wasn’t fixed before he went to lunch.

## A seed gets planted

As this was the mid 1990’s, the internet was starting to grow, and everyone wanted their own websites. I found a really great product called Internet Creator that was an MS Access application to generate websites from filling out data entry forms. You selected the theme and entered the text or data for the pages you wanted created. Then clicked the “generate” button and out popped HTML files, CSS files, and images. All ready to go. The product included an FTP client and would even publish your site for you. This product planted a seed in my fertile imagination.

I built quite a few websites for various departments in the company over the next few years.

## The 21 day to 3 project

Still in the mid 1990’s and Bob tells me that our group has been tasked with taking over this report that must be run every month, and that the current team reports that it takes 3 weeks to put it together each month, and they have new project to work on. So we inherit it.

I dig in and find days of manual copying of files, entering data, running reports, summarizing… all menial work being done by hand and code that one of their “experts” wrote.

So I get busy automating their process. Next month rolls around, I call Bob on the 3rd. “Reports ready, who do I send it to?” “WHAT? Don’t do ANYTHING until I get down there”.

He comes down to my work center, and asks the usual WTF Did You Do? So I explained that their manual effort was stupid, and that their “expert” was anything but. I showed him one routine that had to be run over 100 times to get the final report. That routine took ONE HOUR to process. I re-wrote it in the correct language and implementation, and my version ran in 63 SECONDS.

Another “use the right language for the job” bloody T-Shirt added to the collection.

Another AUTOMATE THE PROCESS bloody T-Shirt added to the collection.

He said, well, be sure to send it to them on the 21st. That’s what they demanded, so that’s what they get. Why don’t you work on the R&D projects you have been wanting to do? Sure.

Moving from Employee to Consultant  
  
Left there in 1998 when a buddy and I decided to branch out on our own.

On our first contract we were tasked with helping 4 telcos merge their IT into one. This effort started with selecting the billing system of one company, and turning it into a product that all 4 could use. They were having issues, so we were thrown in the deep end. He took on the management side, and I dove into the technical.

One of the guys was tasked with pulling data together for the mainframe based build process, and his process was a mess.

* Manually submit and run a job on the mainframe
* Log on to view the report online
* Copy/paste each page of the report into a file on the PC
* Save the file to a LAN drive
* Import the report into an Excel spreadsheet
* Massage the data by hand
* Process the resultant data

Nope. Not on my watch.

I asked the IBM engineer on site to get me a JCL step for doing FTP to the LAN as I hadn’t been on a mainframe in 2 decades and didn’t know the current programs.

I then wrote a master MS Access process flow application to execute the dozen steps in his process, tracking each step on a workflow menu page. Initializing the workflow deleted the prior files from the LAN and let you begin. Asshole tendencies come into play again, as I name the database “ATAMO”. Yep.

A key of software development: “And Then A Miracle Occurs”.

I modified the initial job to write the report to disk on the mainframe, then run the FTP step to copy the report to the LAN drive, then run a step to create a “Trigger” file. Coded the MS access app to submit the job, then check every minute for the trigger file to appear.

The data import was then automatically run with VBA code, and the process would continue unless it got an error. All the manual steps were converted to macros, triggers, and VBA code.

The job became “submit, sit back and enjoy!”

This automation saved hours of effort and eliminated the potential for errors from the manual copy/paste of the report file editing and excel data manipulation.

***Always use external tools to automate and eliminate manual efforts.*** Never rely on copy/paste.

After that effort, I moved back to Tampa to relax a bit.

Maybe rest a while.

Nope. Shortly I was recruited to work with a company on several projects in Florida and other offices. One morning, the lead PM comes in and says that one of the teams is failing to meet every deadline, and he wants me to figure out what is wrong with the technical team.

After interviewing them, I head to their project managers office, and ask him to show me the project plan. He opens it, and the chart looks normal, except for the numbers. So I ask him to open the resources tab, and he does so. “See, everyone is defined. They just fail to make their deliverables”.

I look at it, look at him, look at it. Blink. Blink. Blink. Look around for a camera as I’m sure I’m being pranked. Nope. Back to the lead PM.

“No problem with the technical team, the PM is an idiot”.

Yeah, 30 years later and tact still isn’t in my skill set.

“Prove it”.

“Sure, come look.”

“Please show him your plan”. “Open the resources tab”.

“Do you see what I see?”  
“No”.

“Yeah, your mind just won’t process it, but look at the allocation percentile”

“FOURTEEN HUNDRED PERCENT?”

“My work here is done. I’ll leave you to it. I’m giving the team the rest of the day off to rest while you figure out a new plan.”

Remember that 352 hours per week plan? Well, 1,400% allocation is worse.

Folks, you have to know how to use and how to read the tools you use to manage projects. A little knowledge isn’t enough in many cases. Unfortunately, that “24\*7=168” is one of those unbreakable limitations. And then people want that “sleep and rest and family time” stuff.

One morning, the managing director comes and asks me to accompany one of the business development guys (aka sales weasel) to a potential client meeting as they have yet to make a sale to them.

Sure. So we meet with them, and he is trying to sell them on a security analysis, which they absolutely reject as they know what they have and don’t need it. They are perfectly secure. Well, I recognize that statement for what it is. “No problem, I totally understand. Do you guys know Steve Gibson? The guy that invented Spin Right? Well, he has branched into computer security, and has a website that has the tools to probe and test your site. Just hit it and it will confirm that everything is secure”.

We get in the car, and the sales weasel is PISSED. Bitching at me all the way back to the office that he is getting me fired for giving them a solution for free. We walk in, and he leads me to the resource managers office. The Managing Director stops us on the way in.

“They called and want a contract to fix their network. What did you do?”

The sales weasel is stunned.

“Oh, I gave them a site that would prove they had issues, but wouldn’t tell them what to do about them. I knew they just had to prove it to themselves before they would listen to anyone else”.

I turned to the weasel “Still want me fired?”. He slunk back to his office.

Learn to recognize when clients need to prove to themselves that they need help, and what tools you can recommend they use to do so.

***Don’t try to force a sale when you can create a need.***

Another day and the MD then comes and says “We have a major problem in Chicago. Can you go fix it?”. Sure.

I get there and the developer is frazzled. He is just sure that his code is the problem, but can’t figure out why. It needs to perform at 10K operations a second, but takes 2 to 4 seconds to run.

“Have you done a harness test?” “What?”.

So I worked with him to write a test harness to call the code and time it. Just over 31K transactions per second. Not his code. He was so relieved.

“Show me the hardware”. Looks ok, but looks can be deceiving.

“Ok, I’m going to watch here while you run your test”.

“It is running, see how slow it is”.

I watch the hardware, and see the problem. One hard drive goes SOLID active, then the other. The lights alternate as his 10 transactions attempt to run.

“Find anything?”

“Yep. Get me a screwdriver and needle nose pliers”.

Power it down, remove each drive, move the jumpers, replace the drives, remove the controller card, move the jumpers, re-install the card, power up.

“Try it again”.

Didn’t even see the drives blink. Done.

“Try 10,000 transactions”

BLINK. Done.

“Try 20,000”

BLINK. Done.

All fixed. The controller and drives were set wrong for high-speed transactions. They were set for dedicated streaming with zero loss.

Folks, sometimes it is the wrong hardware, or it is configured incorrectly. And please write and use test harnesses to determine the speed of your code in high-transaction environments.

Learn to write “performant” code when it is needed – learn how compilers build the underlying code and how it manipulates data – particularly for calling APIs or subroutines. Passing by Reference or Passing by Value has a huge impact on execution time in critical scenarios. Use the language that supports the performance and application needs.

We were then sent to a client who was trying to add a new application to their operation. Unfortunately, they were in a position of needing to integrate 5 different systems – and in attempting to hook each one to every other one, they had a mess on their hands.

Yes, you can tightly couple 2 systems. 3 even. 4 becomes a nightmare, and 5 is a Sisyphean task.

Don’t do that.

Integration of multiple systems directly to all the others is not the way to go. Our solution was to build a separate “virtual” application on top of them all, and integrate it with each. That way, there was only one point of integration for each application (much easier to do) and the logic was contained in the virtual. Check inventory via one call to the ERP managing the warehouse; one to the AS/400 JD Edwards financial system; one to shipping. It also means that any one application can be easily upgraded or even replaced by changing only the one set of integration points. Orchestration of APIs is always a better option.

Unfortunately, that company ran out of clients needing my level of expertise, so on to the next adventure!

My work wife Alicia said she knew a company that was looking for senior engineers, business analysts, and architects and would I be interested. Sure.

The client was another telco, and with my extensive telephony experience, I was immediately hired and thrown in the deep end. The telco was migrating from their old mainframe monolithic billing system to a new Unix server based solution.

Their problem was that the 2 were not compatible. So the “web store” front-end would have to be modified to be able to talk to both, and the workload of constantly doing that as they migrated areas of the country was a huge task.

Made even more so as the new system did not have an API layer.

Lots of meetings and discussions of options. The legacy group said they had already developed an API layer in front of their mainframe system, and it was ready to go.

In another meeting with a different group, they mentioned that they had several applications they had built an interface to that did not have APIs. I asked for details, and their engineer showed me the “software robot” they had built that fetched a screen, filled in the forms, “clicked” the submit, captured the messages.

That was the key I needed.

[ Why did their own folks not know this? Because the 2 managers hated each other and would not allow their groups to work together. There’s a bloody t-shirt or three ]

I proposed a different solution. Route the front-end messages into an MQ Series message layer; have it translate the incoming message into the right message format for the desired back end; fire the message to the back end; capture, coalesce, and return the result. Between the home-grown API in front of the legacy mainframe and the software robot for the new system, we had what we needed.

This involved building maps of the attribute names of every field in the web forms and in the 2 back ends. Simple conversions in MQ, driven by a selector on the area code field. This enabled converting one area code at a time from legacy to new.

And we needed a way to create a Metric Kiloton of test messages and the matching expected response messages for testing and verification.

Remember the story of Internet Creator? Well, I figured that if an MS access database could generate HTML, then it should be easy to generate XML as well. A few days effort, and I had a working XML Generator application.

My goal was to define the 3 file structures – the data from the web interface supplying the customer order, and to the different back-ends. Each had its own attribute names, so a simple table with 3 columns mapping them was a starting point.

Then I wrote VBA code to generate sample customer orders, and create the 2 xml inbound files, and the matching response files.

Became a simple exercise to run the tests, submitting the files and comparing the results.

Around this time, I was also helping friends build web sites for various startups. PHP had become really popular, and a framework named “PHPNuke” was one of the leaders in the space.

After examining its structure, I realized that the “Blocks” and “Modules” were pretty standard – and if things are standard, then they can be templated and generated.

Version 2 of code generation – my PHP Generator. I engineered a structure to hold the definition of data entry screens, and generated the code for the front-end and the database access layers.

## Moving to DoD work

In 2001 a very good friend was working at a Federal level contractor, and they had an opportunity to build a solution for one of the military branches. They won the contract, and put the initial team in place to build the little test project they had been granted prior to the major work. However, 9/11 happened, and everything changed. They asked if the team could pivot to deliver a critical solution rapidly. He was asked if he knew anyone to handle the technology, he said yes, and I was recruited to handle the technology efforts. So off I went.

These projects spanned the simplest of raw HTML to full Java applications to run on a Sun cluster. Four SunFire E25Ks, to be specific – 106 CPUs each, maxed out core memory. Super Beasts of machines.

So I get to the development center outside the base, and I’m informed that I will have limited access until I get a National Agency Check (NAC) done, and that even that limited of a background check typically takes 4 to 6 months. I laugh and submit the forms. Approved 2 weeks later. Yeah, easy to do when there are already files on you. I’ve had mil files on me since I took the ASVAB in 1969, but those stories are all classified.

So I was able to get to work almost immediately.

The first project was a quick supply chain “find a part” system that could be used in country, over a low capacity satellite phone link. Limited to 10K of HTML, CSS, and javascript per page. Back to my early 90’s roots of pure web work. Why the JS? Even over a dedicated military band, it still required instrumentation/tracking code to run on the military data network. I taught the team the Rational Unified Process, we wrote UML models and Use Cases, and the military guys loved the form and format and declared “that’s our new way forward”.

Then came a java front-end for the legacy systems, and again, we knocked that one out and it was also well received.

The military guys realized that they needed to automate the views of their supply chain – and thought they would start by getting information from both the “retail” and “wholesale” views of inventory. Of course these were totally disparate systems, the retail side running on Sun SunFire Unix systems and the wholesale on IBM Mainframes. A meeting was called with both vendors and divisions of the military and a pretty open discussion was had. At one point the vendor from the mainframe side mentioned that they had recently developed an API to call for inventory availability, and that was the key piece of information we needed.

I stood up, walked to the board, and introduced them to EAI – Enterprise Application Integration, and showed how we could build the virtual layer that would call their 3 APIs for that view, call the API we had written for the retail view, and return the composite picture in seconds.

It was agreed that this would be a joint venture with both contracting companies and the 2 military groups, all we had to do was make a presentation to the Pentagon to get approval. A couple of incredible guys (Karl and Nat) developed the presentation, and I was tapped to give it.

There was a lot of discussion in the presentation, and after I ended, the lead decision maker from the Pentagon said “Hey Louis – if I add $1M can you do … (classified)”. I thought about it for 0.68 seconds (only geeks will get that reference) and said “absolutely”. After the meeting, the president of our division asked me how the hell I could answer that and what risk had I just given them? “He asked me to make 2 phone calls and one site visit, and gave us $1M to do so. Relax”.

Listening is a skill of not only hearing the words being said, but the hidden meanings and words not said. Like the old quote – “Read between the lines”.

We built the system, deployed it, and even got press on it:

https://www.af.mil/News/Article-Display/Article/133012/new-air-force-portal-tool-gets-warfighters-parts-faster/

# The world of blade servers and moving to the cloud

The CIO of one of the larger state agencies in Florida wanted to move from their old patchwork solutions to using Java as a standard. He contacted the Sr VP of the company I was working for and asked him for someone to come introduce Java to them. The SVP “volunteered” me – so I contacted the CIO and asked him if he wanted it as an easy “everyone gets a pass” or a hard-core coding bootcamp to find out who would really cut it. He chose the bootcamp option, so I wrote a 7 day course. 192 slides on Java and the “framework du jour” of Struts with the Tiles layout manager.

And another “IT is a small world”, my SVP had been the CIO’s first employer when he came to the US. The 3 of us got along quite well.

Technology keeps changing, and that was one option for the next iteration, was easy to teach, and easy to setup, and our team had used it successfully on one of the DoD projects.

After the course, I was offered to the CIO as a dedicated resource to help them move to Java, he accepted, and that put me on the path of state government work.

They acquired a rack of HP Blade Servers, and as an Oracle shop, loaded ‘em up with the databases and Weblogic server.

As they were starting to build applications in Java under the Struts/Tiles frameworks, I thought that I would have some fun by writing my 3rd generator – my Struts Generator.

Building on the data entry screen definitions from my PHP Generator, I expanded the definitions in support of the Oracle data types, and the information required to generate all 5 layers – the Form Bean, Action, Data Model, Data Access Object, and the Java Server Page.

Apparently, teaching an MS Access application to generate the java code upsets developers. They didn’t understand that they could use the program to rapidly do their jobs for them. They insisted on still writing it all by hand. So I relegated my app to generating the “KickStart” for their projects.

I enhanced it to create the POM.XML file when we moved to Maven, and then decided to piss off the BA’s as well.

UML models are standard. Searching around, I found a GNU package that would render UML models. I found an HTML rendering package as well.

As the generator had the definitions for data entry screens, it was easy to add the columns for the help text and for the field definition, and create all the information for both the Use Case Specification and the Use Case Realization. So I taught it to generate “raw” HTML that would represent the data entry screen wire-frame, render it and capture the image, create the “dot” files for the GNU package to generate the UML diagrams, and used the Direct Data Exchange (DDE) feature of VBA to open the Word templates, navigate to bookmarks, and insert the generated text, tables, and images.

This way, I could generate a “KickStart” for a new application with a running set of sample code, and the associated documentation for those features. Remembering that the old Internet Creator had a “FTP to server” button, I implemented “PUSH to Git”. So the project started with ready-to-run code in the git repository, and had sample Use Cases as well.

Developers who wanted to use my Generator to rapidly build their solutions were welcome to. Some took advantage of it, most did not.

I still don’t understand sitting in front of a computer with more power than my whole datacenter and not using it.

I now see that split in Engineers who utilize ChatGPT to rapidly produce solutions, and those who are still using “File/New and type for hours” in their IDE.

A side note here:

For many years, one of the IT “adages” has been “The best way to learn a language is to teach it to others”. I took that in a slightly different direction – “The best way to learn a language is to teach it to generate itself”. Automate the mundane tasks, and concentrate on developing the business logic over the generated code.

In 2007, the company I was working with started to fold, so the CIO asked one of his favorite vendors to talk to me. I was introduced to Piyush who took me to lunch, and hired me on the spot.

And that is how I became the CTO of Kyra Solutions, where I am now proudly serving as CTO Emeritus now that I’m in my 70’s and semi-retired.

I stayed at the state agency for over a decade, moving their tech stack several times, and writing (or generating) a ton of “how-to” guides for the technical teams.

While doing all that just to make my job easier in producing rapid prototypes, I was tasked with finding a solution to a problem.

I began by surveying many of the paper forms that the client wanted to automate. By doing a very high-level, content-agnostic survey, I determined that most had the same few dozen needs. People, Addresses, Qualifying Questions, Raw data entry, upload files, Geo Location, Geo Boundary, Signatures, Payments. Pretty standard stuff. All the same, just different. Easily configurable if we built modules for each type of input. Since they were a Java shop, that led us to use the extended EJB pattern, and create XML “orchestration” scripts that contained the configuration for each different paper form.

Handling the information as the user entered it and make it instantly usable in the following steps looked like a fun challenge – and when faced with fun challenges, as you know by now, I think “how dumb can this be made”. Well, it fell out almost instantly. Each section of the form mapped to an EJB, and to a “Step” in the process of filling out the full application. Each step was named. Every input element of every step was named… so a dumb-as-dirt data structure would hold ALL of it in a nice compact way. That structure was a Map whose primary key was the step name, and whose content was another Map – whose primary key was the field name, and whose data was an array of the answer. This covers nearly every type of input – by making it an array, checkbox values can be inserted, multi-select values – pretty much whatever we needed.

Thus was born one of my favorite objects – the MapOfMaps

A simply defined: Map<String, Map<String, String[]>>

This also leads to simple value substitution with {Step Name.Field Name} – and of course, we immediately added {Object Name.Field Name} so we could map in the user information and any information queried from databases during the process.

Looks a lot like the early implementation of “mail merge” fields from the MS products. Yep. Antique technology still has uses today. Never forget it, just remember how and when to leverage it.

Email templates became no-brainers:

Dear {UserInfo.FirstName}

Your application for a {application.permit\_name} has been approved.

There are other fun aspects of the project, but those are outside of this.

Suffice it to say, we built the prototype with the first 4 application forms, and it was an immediate success.

One evening, Bhavik, who was one of my incredibly sharp techs calls me and asks me to come to the Kyra office as they have a critical demo to build and don’t recognize the data stream they need to process. He shows me, and I immediately knew it as it was a decades old technology, we knew way back when as “L’Data” (pronounced ELL-TICK-DATA). The record starts with one or two “length” bytes followed by that number of data bytes. My first usage of that was in the “Intel Hex Format” paper tapes we used to load our kit computers. This was a compressed variant, so it was L’L’Data (length of data bytes, length when unzipped, data). He already had C++ code started, so it only took us 20 minutes from start to processing the data stream. That effort actually planted the seed for a completely new business unit for Kyra Solutions.

And technology advanced into the Platforms we have now, and Salesforce development.

As technology changed again to the \*AAS world, the cloud, and Platform solutions, we began helping our clients to migrate. As government agencies, they were also under a mandate from the legislature to “move to the cloud”. That removed a lot of hesitation and road blocks to adoption.

Although some chose to use “MAAS” (Metal As A Service), many went with hosted solutions. We were able to move source control to private GitLab instances or to GitHub. Moving build stacks, repositories, and production environments soon followed.

Evolution of IT over the years

Having experienced the changes in IT from Mainframes to Mini Computers to Desktop to Blade Servers to Cloud was one of constant adapting to change. And to things that never change. COBOL and ForTran still exist. Java took over. Now GoLang and Full-Stack and RUST.

Tons of new architectures, tools, and utility programs to leverage. DevOps. DevSecOps. Waterfall, RUP and Agile. All new ways of doing the same thing – delivering functionality to companies and their customers.

From project plans to KanBan boards and ticketing systems – new ways of tracking who is doing what and when.

Experiencing the innovation and technological breakthroughs over these 50 years has been exhilarating to say the least. Loving every aspect of IT and what it has enabled us to do in building solutions.

Of course, one of the biggest impacts was the development of Blockchain and its various usages. From doing government work for decades, I was quite intrigued by the implementations around property ownership. All the crypto currencies and tools have had a major impact on many aspects of business. “Smart Contracts” are an incredible new technology making huge improvements in the whole supply chain ecosystem.

And finally, we are becoming more aware of the Cyber Security (and extreme lack thereof) in many aspects of our daily lives. Just saw a story about robot vacuums taking pictures, videos, and audio recordings while operating. That’s a security nightmare right there!

If you aren’t totally familiar with Cyber Security and self-protection, check out the 3 part series “NIST 800: The Home Game” in my repo. It is a fun exercise of learning what your personal environment and risk exposure is.

## Teaching Kids through Exploring.

As I said in the beginning, I was fortunate to be in an Explorer Post that taught me computers, and in 1989 I was asked to join one as an advisor. Thrilled to be giving back to the next generation.

I had a few stellar students, and one serious protégé out of my decade there.

As part of my work at that time, I was one of the guys that did Career Day speaking about jobs in IT. One of the schools I went to each year had 2 kids that I remembered from the 1st visit, and on the 2nd one they tried giving me grief. I challenged them to come to the post and see what they could learn.

They showed up. It happened to be when one of the larger business units was shutting down, and they donated a TRUCKLOAD of old PCs, monitors, keyboards – nearly 100 systems in total.

I told the guys if they really wanted to learn hardware, to start by taking them ALL apart. Pull every cable, every controller card, every hard drive. We got storage bins for the hardware brackets, screws, and cables. Boxed the controllers by type.

“Ok, they are all apart. Now what”.

Inventory what we had, and start building working computers from the parts. We found a dozen of the unique “PC 286” (not PC/AT) and based our builds on those. Ken grabbed a bunch of parts, put them together, and nothing.

Had to teach him that he needed to find it one piece at a time – use a working computer to test the monitors; then test video cards to find a pair that would work. Then use those to test the other chassis, and finally test disk drives and disk controller cards.

And some worked, and some didn’t. Not really. There were at that time, 2 different protocols in hardware for hard drives – Modified Frequency Modulation (MFM) and Run Length Limited (RLL), and the controller and drive had to match. This is important for later.

Ken worked hard – his buddy thought it was too much effort, and slacked off.

So one night Ken comes in and says he got a new job. What? Dishwasher at a better restaurant, I got a $.05 raise! I asked him why the hell he was applying for dishwasher jobs – there must be a computer store near him that needed help. He said he saw a Help Wanted sign at one place, but didn’t think he was qualified. I smacked him up side his head, and told him to go.

He went the next day. The boss and a tech were working on a computer, and not getting it to work. The boss said “fix this and you are hired”. Ken took one look, said “RLL drive, MFM controller. Let me look in your parts bin”. Found the right controller card, popped it in, and fixed the computer. Hired on the spot, and nearly tripled his salary from being a dishwasher.

Ken is now the IT director at a major company here in Florida.

Another activity of the Explorers was the annual weekend at MacDill AFB for a “Scout Show”. As the IT post, the kids developed a system to map out the whole show area and ran the “information booth” where parents and others could ask where a particular troop was, and the system would show them. A guy comes up to our booth and asks to talk to me. Tells me he is starting a medical device company that will build computer controlled devices, and is looking for young developers with mechanical/hardware skills to hire. Did I know anyone? Oh yeah – one of my kids was both mechanically inclined and a hell of a developer, so I put them in touch. Jared is now a VP of a major global medical company.

On a “Shake My Head” note to the Scout Show – when we went to the on-base meetings for the show, the AF folks told us they could show us where it would be, but couldn’t give us a map as the map of the base was classified. So I downloaded it from the US Government “TIGER data” FTP archive, printed a few copies on an E size plotter, and took them to the next meeting. Amazing how one branch of government thought it was classified, and another was giving it away for free.

The AF guys were not pleased.

One thing about the early era of IT that really helped the Explorer Scouts was its absolute “openness”. EVERY monthly magazine included source code. Every book had a floppy (or CD-ROM) with its code. Interface Age even used the “trutone” pressed plastic records to give out running applications in the mid to late 70’s.

Way back in the December 1976 (homebrew computer era) issue of Interface Age was the infamous article on Tiny Basic that gave the world a new definition of sharing code:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; TINY BASIC FOR INTEL 8080

; VERSION 2.0

; BY LI-CHEN WANG

; MODIFIED AND TRANSLATED

; TO INTEL MNEMONICS

; BY ROGER RAUSKOLB

; 10 OCTOBER,1976

; @COPYLEFT

; ALL WRONGS RESERVED

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This was long before the “everything must be owned” mindset of today. At least the Open Source movement has started getting us back to where we started.

Our “Web Server” came from one of those books: “Building your own Website” by O’Reilly press, published in 1996, and included a full server that would run on Windows 95 or NT.

(ISBN 978-1565922327)

Our “applications” code came from a seminal book of that time, “Instant Web Scripts with CGI/PERL” by Selena Sol and Gunther Berznieks. This book included a fully operational web store, a flat-file database, and tons of other really cool resources.

(ISBN 978-1558514904**)**

## A very skilled student

On one of our open houses, my manager (Bob) brings his son Steven. Even though his 14th b-day is a bit away, he tells me he has been coding at home, teaching himself, and wants to learn more. So he starts coming in and studying everything he can get his hands on, becoming a registered member on his b-day.

He works on a lot of the projects in the post, and helps the other kids learn as well. A natural at mentoring and training.

He got into web programming with PHP and fell in love with one of the early social network type services. He actually did some programming for them, and went to work for them after graduating.

Unfortunately, that company lost traction, and he needed to find new work. He reached out to me, and the state agency I was contracting with happened to be looking for Java developers. He applied, I recommended him, and he got hired immediately.

He excelled at developing the solutions they needed, and many of his projects are still in production to this day.

He gained a ton of experience working there, then went into a consulting company, and finally started his own company with a few of his buddies.

Last I heard, they are doing very well.

## An alternate thinker

Another open house, and a guy brings his son who starts wandering around the room a bit. The father tells me the boy is 14, can barely read, is not interested in school, but seems to want to know about computers, and was hoping we could help him. So I started talking to him, and showing him the different computers and what they did. We got to the far corner of the room and his eyes lit up at one machine. It was built on a fiberboard display instead of inside a computer case, so the parts were all out in the open. I showed him the mother board, video card, sound card, and graphics card. It had a 10” Wacom drawing tablet, and was running an early 3D drawing program.

This got his attention, and he asked if he could try it. I showed him a few things, and left him to play.

I talked more with his dad, and kept an eye on him. He was engrossed. 20 minutes later, he calls out for us to come look. I was blown away, and said to his dad, “I didn’t know it could do that”. The kid was an amazing artist – total “image” mind. At that time, he would have been called a “Right Brain”.

He asked how he could learn more about the program, and I said I had the book, but it was a huge book and was going to take him a long time to get through, but he could take it home to study, then come in to do the exercises. He was a little disappointed, but thanked me and took the book.

The first week, he only got through a few pages – but every week, he got through more and more. He was learning to read as he finally had something he wanted to read. His grades also started improving as a by-product.

Matt became an incredible graphic designer and artist.

Look for the skills that may be dormant or un-realized in people. Help them bring them to bear and you may be amazed at the outcome.

## My First Protégé

One night, this almost as arrogant as me kid comes in, and introduces himself – Troy. He tells me he has heard a lot about us and me from his friends that were members, and he wanted to learn from me. After talking a while, we agreed that he would learn what I gave him, and I guaranteed him that it would be worth his effort. Started him on Pascal as in his opinion, “C is the only language worth knowing”. I gave him a book on Algorithms in Pascal, and told him we would need all of them for a project, and for him to get them ready. Learning Pascal taught him a LOT about debugging (which was the goal). Moved on to Visual Basic for developing good looking screens. Did web work with Perl and PHP and CGI scripts. Wrote a multi-user online access system that was the basic implementation of the ISO stack in reduced form.

A few years later, early on a Friday afternoon, I get a panic call from one of the data center managers – he needs a high-end Unix Systems Admin to help fix their issues and get them on the right track. Asked if I knew anyone. Of course, I have just the guy you need, I’ll get ahold of him this evening. “Can you have him call me at home tomorrow? I’ll interview him, and if he is as good as you say, he can start on Monday”. I get a message to Troy to get to the post as quickly as he can.

He comes in late that afternoon, and asks “What’s up?”.

“You have an interview in the morning for a Unix Systems Administrator job.”

“I Don’t Know Unix”.

“You will by morning”.

He aced the interview, started the job, automated the manual updating of the dozen systems across the US in our various data centers, and blew everyone away with his knowledge and skill.

Years later, he was sent to a class to learn networking. He messages me from the class – “You PUTZ! You made me write the core of TCP/IP in Pascal!!! WTH”. For years I would get the occasional “putz” message meaning he had run into something else I taught him without his realizing it at the time.

As total nerds, when I haven’t heard from him in a while, I just send the message

“rodtsasdt 111111report\*” and he responds with the correct “oolcay itay”.

(ISBN 978-0441003600)

Last year, I introduced him to my newest protégé Chris – after the meeting, Chris described Troy as “The baddest bad ass I’ve ever seen in IT”. I wish Troy could write his story, but most of it would be classified and only readable by federal law enforcement agencies and people with TS/SCI + LS poly.

Yeah. Don’t ask me how I know.

Into the Future

AI is rapidly changing the landscape of how the work is done. It can quickly deliver code based on intelligent prompts from our new career skill of being a “Prompt Engineer”. So we have a new skill, a new focus, and a new method for creating code.

And again, adaptability and the ability to learn/implement/forget is critical for a career in IT where the adage “The only constant is change” is spot-on.

### Into the Future with Chris

Sometimes the world is a strange and small place. I decided to again attempt to learn guitar, so found an instructor. After a few months, she said she was not a good fit for what I wanted to learn, but she knew someone who was. She put me in touch with Chris, and he started coming out to instruct me. He happened to come on Sundays when another protégé was coming to the house to do some IT work on a cluster of Raspberry and Banana Pi computers. We built a replica data center on a large piece of pegboard using 30 machines – had one with GitLab hosting the code, another with Jenkins build, a cluster of 4 running wildfire, a banana with scsi drives for the lan. Everything. Learned a lot from it.

Anyway, Chris was interested, and we started talking about IT. He had done some informally in the past, and decided to put himself through a full-stack school in Chicago. When he returned, he asked me if I would look at his resume. Sure.

Shock did not cover it. The schools and background he had were incredible. I told him to not bother looking for work, I was sending it straight to the founder/CEO of Kyra and we’d be in touch. Within 10 minutes of receiving it, Piyush called me. ”Is this for real?” “yep”. “I’ll come up this weekend, can we meet for brunch on Sunday”. “Sure, I’ll let Chris know”. Brunch/interview/hired.

Chris led the efforts on high-end technology demonstrations and Proof-of-Concept development with Blockchain and dynamic data entry, and all kinds of really state-of-the-art practices. He was asked to join the new Transportation group and help them with setting the foundation for moving into that realm. And he excelled.

He decided to branch out on his own to start a non-profit in the Music scene, and to utilize AI as the core of his software engineering methodology. From AI built tools and testing, to generated code snippets – all using the newest ChatGPT services and features. His app for the local music scene in Tallahassee is now available on both Android and IOS as well as his website.

Our usage of ChatGPT and AI have expanded. This document that you are reading now is laid out from a structure developed by ChatGPT from a simple prompt:

“…I want to write the story of my 50 year career in information technology.  Can you suggest an outline? …”

Chris’ dad Neil is an established author with books on the operations of Government at the Federal and State level. I shared my interaction with the AI, and asked his opinion. His response is very interesting:

“… Your question, in fact, probably was the prompt for me to use ChatGPT 4 yesterday for a series of “tell me more” kinds of questions on a writing project. It saved me probably hours of hit-or-miss reading and research. It was weak in actual data (I kept asking, but it had none on the item), but it provided conclusions and analysis citing sources where I could get the data. I also used it to rewrite and condense a cobbled-together passage with some organizing headers, then added some more info that was omitted and got a new version. …”

Leveraging the AI tool to save hours of effort, and to point him in the right direction is incredible. This from an author who, like me, is of “boomer” age.

AI is the latest tool in our tool belts and tool chests. But it is just that – ONE more tool, not “THE” tool to replace all others. Not one damn thing in the last 50 years that has claimed to be “THE” answer has actually delivered on that claim.

Of all my bloody t-shirts, the “use the right tool dummy” one is my favorite.

We’ve come a long, long way.

My first IDE was a set of coding sheets, #2 pencils, razor-blade pencil sharpener, and a Pink Pearl.

Now I use JetBrains IDEA, Visual Studio and Xcode.

Developed code in ForTran, Assembly (IBM, DEC, PC), COBOL, Pascal, Basic (dozens of different ones), C, Java, Go, half a dozen scripting languages, and way too many source control and build systems.

Built computers from the original kits to PCs to server racks to my $20K “desk” computer.

About that… a few years ago I decided to build myself a real desk computer. I got a Hydra Desk Case from Italy, an ASUS RoG Zenith Extreme motherboard, a 16 core 32 thread “Threadripper”, 128 GB memory, and a couple of 1TB SSDs. Added an NVIDIA Titan V video card, water cooling, a 49” Samsung gaming monitor, Klipsch 5.1 surround sound. This thing has way more capacity than the entire mainframe data center I started at in 1972.

All the same, just different.

And now my “Junior developer” and my “tech writer” are an AI. That should scare a lot of people in this industry. How are people going to get in when the entry level positions can be done by AI?

Where is the industry going next? AI is going to be a massive part of it, and using tools a critical way forward. I see the engineers who learn to leverage the AI and tools as the ones who will excel and lead us to an exciting explosion of new facets of the IT industry.

# Axioms, Idioms, and IT truth

***Constants aren’t, Variables won’t.***

Invariably, over time, some developer will change the value of a constant, and everyone will wonder why the system crashed. And careful analysis will find variables defined, set, and never modified.

***There is one and only one outcome of building a “fool-proof” system. No matter how much you invest in doing so, the only result will be the Universe giving birth to a more creative fool.***

IT has always and will always be about IPO. ***Input, Process, Output.***

And after more than 50 years in IT, there are many things that have changed, and many that are still the same. Some come and go; some come and go and come back; and some stay forever.

Look at Key/Value stores – invented decades ago, and returned as new data storage technology.

Compressed data streams are still a thing. Learn to recognize them, how to process them, and just change the tool used to do so.

Languages are the same. Learn to learn, implement, adopt, and box up.

***Never forget a technology, just put it in storage, odds are it will come back.***

Vendors come and go.

***Avoid tools that cause vendor lock-in early in a project.***

Read my “Database” doc in the LiTiRi repo for a hard example of what I mean by that.

***Learn the fundamentals and underlying technology.***

Read RFCs, JSRs, and any other reference material. Learn ***WHY*** something came into usage. That should prevent you from doing the wrong thing with the wrong tool.

Everyone should do the 1st 6 steps of NIST 800 at home for their personal environment to see what your exposure is, and why it is so important for a business. You can get a copy of my “NIST 800 Home Game” from my repo.

Wrap Up

The kids (and workers) that I have mentored over the years are my legacy. The code I leave behind in the repo can only serve to show how this lunatic leveraged technology, and how solutions are derived from needs or weird ideas.

I have wonderful memories of helping my clients succeed and deliver value to their customers.

The military team was comprised of a lot of very sharp people and new friends.

All my kids from the post.

My very special DHBs – Patti and Pom.

My work wives Alicia and Drey.

Lots of workers and leaders from the companies I have worked for and with over the years.

My mentors, Mr Ferguson, Jack, and Dan.

I hope you have enjoyed my stories, and that you build many in your career – the highlights, the bad times, and a large well-earned bloody t-shirt collection.

If you are up for it, upon request I will share my “personal (NSFW)” stories with you. Just ask for access, and promise to not tell anyone…

If you are a “VP” or “C” level, I will share my “Classified/Confidential” stories with the same caveats.