Notes on My Past Employment History -- The ZZZ Version (v2022.07.27.16:20)

This is a work in process. I've tried to capture snippets of my amazing journey through the lens of technology as it comes and goes. No doubt there will be additions, subtractions and clarifications in the future.

An old Pastor of mine, who had attended the deathbed of many people, recalled that not even one wished they had spent more time at work. 😊. Yet ….

As of today, I've done 420+ job inquiries/submits/interviews since starting my job search in September of 2018. I attribute the lack of hire-ability to age discrimination and competition from the other side of the planet. An old friend of mine from my startup days needs a Dev to write some code.

COVID unemployment benefits kept me afloat for months. Exhausted now. Back in the sunny Huntington Beach condo. Painting long overdue. Looking better in while. Seattle meanwhile is getting pounded with rain. Timing is everything.

Meanwhile: Taking a long view of the water I've plowed. Enjoying life. Kayaking the shores of Lake Washington among other places. Hiking. Camping. Biking. Trading Equities for profit.

Trying to keep my technical chops. Use it or lose it. Lately relearning embedded development in C. Built a native JavaScript app that reads Bluetooth power levels, stores them locally with IndexedDB and posts them to MongoDB Atlas via a simple webhook. Given my love of PreactJS, I'm thinking about learning React Native.

There are five projects in Node, Go and Preact on the Github. Published a series of jigsaw puzzles in PreactJS version the other day. And in the Sandbox, in Chrome, one can right click Inspect and click on the Sources button to see the source code. Lots of ways to verify my skill as a modern software developer.

I've written source code for maybe 20 platforms in my life as a programmer … none has the beauty and elegance of Go. Feels good to have thrown off the ball-and-chain of the languages that can only run inside Virtual Machines. Goodbye and good riddance Java and C#. Let utter simplicity rule the day in the future. Ditto for PreactJs.

I eagerly await the dawn of tomorrow, where Spiking Neural Networks become as ubiquitous as the ultra-cheap but hard to get 4-bit micro-controller.

Happiness Equals Reality Minus Expectations

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

I started working at age 11, delivering newspapers; then, a year later became the manager in charge of "The Shack", where newspapers were doled out to each newsboy and changes to delivery service was managed. Mostly it involved throwing bad news on the doorstep (1966 - 1973). Vietnam War. Boeing-driven unemployment in Seattle. End of the Apollo program. No automated Bill Pay … had to collect door-to-door monthly ... phone and face to face customer relationship management. Retained about forty bucks a month (for 30 days of labor) after paying the Times.

In those days, printed Bulk Detailed Information was peddled from place to place. Also used to start fires. And clean windows. Can't do that with a smartphone.

The smartphone. A junk email streaming machine. Can't stop it, they'll send more! My idea: put a ChargeBack button on the app. If the email is appreciated or expected, No Charge. Otherwise: 5 cents please. 1 cent to the system. Would require a second confirmation an hour later. I'd be rich ….

My Mom remarked one day that I was happiest when I was busy accomplishing something. Making stuff happen. I am not much of a spectator. Therefore I Code. I Test.

My Dad would note much later that I was also a Quick Study. Like a stem cell, capable of adapting.

Achieved Eagle Scout status when I was 14. Had to travel into the long-gone Seattle Times building for a photo.

During the college summers I worked for the Washington State Highway Department. Worked the Survey Crew: cutting brush, positioning plum bobs, and holding signs. Eventually got to run the Level and the Transit. Learned about Right of Way, Centerlines, Curves, Slopes, Drainage, Super Elevation, and developed a practical appreciation of trigonometry. Decent money, almost enough to pay for school tuition at WSU.

I had crafted an engineering degree that was half Pre-Med and half Electronics. Aced FORTRAN and learned how to program the now-extinct Analog Computer; with the correct transform function an Analog Computer can be programmed to simulate a neural net. Then there was the Pre-Med aspect: got to spend a semester with the Cadavers. Had an interesting experience interning on an early LVAD project involving McDonnel Douglas.

Challenged out of Calculus and English Comp. Graduated with lots of extra credits but a mediocre GPA. Too many hard classes at once.

I So Grokked the 6502. Miss it.

After graduation, my Dad suggested that I take a course in a brand-new technology called Microprocessors at nearby Seattle University. I recall hand-entering 6502 op codes into a keypad. A life-changing decision.

Then, one day, saw an Ad in the Times regarding work at the Hughes Aircraft Company in Fullerton, California. At the time Hughes was the most advanced technology company on the planet.

Hughes was owned by HHMI (Howard Hughes Medical Institute) which was technically a non-profit. Every penny was plowed back into the business. I worked at the 17,000-person Fullerton complex (now a housing tract) writing Technical Manuals on sophisticated Electronic Systems.

The idea was to hire degreed engineers who could read schematics and wiring diagrams; smart people who put the whole picture together without benefit of the Design Engineers. They were way too busy on too many projects. Learned the technology behind RF radios, Microcode, Bit-Slice processors, LCD projectors, passive sonar, command and control systems and track-while-scan radar. My job was to tell people how they worked in great detail, how to trouble-shoot, and how to maintain. They preached: work Smart, not Hard.

Theory of Operation was the most challenging part. After understanding it, I was told, draw a picture or diagram. That would go to the Art Department. Then write text on a yellow pad. That would go to into the Redactor system for input. Editing often involved a Copy machine, scissors and Scotch Tape.

I recall writing up the Theory of Operation on the AN/PRC-104 antenna unit. It was a marvel. Had the ability to do automatic impedance-matching between the radio and its antenna. That resulted in better signal reception and transmission. Allegedly could “tune-up” to a chain-link fence.

The PRC-104 was a single-sideband radio. Normally when one mixes the carrier with the signal, two sidebands emerge; in this case, one was filtered out.

All this this was before microprocessors. Digital technology was implemented in ROMS which drove flip-flop laden state-machines. Screens were painted in what's called stroke video. UYK-40 was state of the art.

Hughes built Command and Control systems. The US Navy was one of our primary patrons. Especially the ASW folk. Imagine, being able to detect a distant 50hz audio signal via a giant towed hydrophone array … and getting it back to HQ via satellite.

During that time at Hughes my boss nominated me to be a Hughes Fellow. That allowed me to receive a stipend while I worked on my MSEE. I did a Master's Thesis on what I figured would be the next generation of computing: Data Flow with its Actor-based execution proposition. Dataflow also a smart way to implement neural networks; especially after the transition to Optical interference-based systems.

My expertise analyzing digital circuits made it possible for me to illustrate and describe that data-flow computer down to the block component level, where it could be fabricated using VSLI technology. The Thesis task caused me to buy an Osborne computer to save on publication costs. I won an $500 award for that Thesis at graduation. The Osborne allowed me to learn another set of assembly language ... this one based on Intel.

The industry, however, never got away from the Von Neumann computing paradigm. So now we consume vast amounts of power in vast Cloud systems based on that antique technological lynchpin. Totally avoidable.

But recently I saw something. Spiking Neural Networks (SNN) are being implemented in silicon where the memory is local.   TrueNorth and Loihi will be the chipsets of the next age I suspect, given their lower carbon footprint and faster adhoc learning.

Back in 1981 I used the Osborne to do project estimating. Osborne ran on CPM. CPM was a popular hobbyist operating system. It ran an early spreadsheet called SuperCalc 2. I became a spreadsheet guru, even making a simple Kalman filter out of it a few years later.

At the end of my tenure at Hughes I finally broke out of Tech Writing and did 8051 programming to make screens on a newly invented raster-based monitor. My first in-circuit emulator. My first UI. For the day, it was a complicated system: loosely and tightly coupled multi-threaded microprocessors. At one point we ran into a snag with the silicon and the critical Test and Set instruction ... that was the troubleshooting adventure of a lifetime.

Hello Small But, Brilliant Tech. What is Truth?

The commute to Hughes was daunting. So, in 1985 I managed to get a job at a closer software company called CAST which did GPS software in FORTRAN. Mostly PhDs, I was junior with my MSEE.

I learned about the Kalman Filter there from the guru himself Don Elder. A Kalman filter is commonly used in the aerospace industry to determine **"truth"** from multiple sometimes inconsistent inputs.

That project's Kalman modeled 17 states to precisely estimate where the GPS antenna was in ECEF coordinates. The Kalman Filter combined GPS data and inertial (gyro) data. Surprisingly, not that computationally heavy. An eight-bit CPU of the day could handle it. Source code was written in FORTAN.

Aside. Out on the kayak ride today on Lake Washington, I saw jet after jet fly overhead into SeaTac. And a number of Boeing 737's at the Renton plant. Every 737 takes its first flight over Mercer Island. I pondered the MCAS system of the 737 Max.

I wondered why the MCAS apparently didn't involve a Kalman filter and inputs from multiple sources, including the on-board gyro, 4-antenna differential GPS and the two-standard angle of attack sensors. Seems to me that the MCAS should have done a much better job of modeling truth. A terminator

It didn't. So now apparently, we are in age where Software Kills. Skynet not even involved.

There's a wormhole appearing on the horizon for the 737 that would take it to the other side. Somehow, a bunch of renegade Boeing Blue Badges and Gray badges –the Makers—get permission to do a Vintage Voltage scenario on a 737 Max. Management gives the team license to fail as fast as it can. No email. Just Slack.

A 737 has its engines and fuel tanks removed. 30 Chevy Bolts are disassembled. The batteries are stood up in cargo, the motors and differentials are strapped to the wings. A few go to the wheels. Each motor is connected to its own battery pack. 30 DC1 charge ports for fast charge. Somehow the air vehicle rotates and cruises at 250 kmph. Wormhole closes.

Back to the Employment Story.

I got my first exposure to anti-shared-memory coding CAST programming. The rule: pass every dependency via the parameter stack. Avoid shared memory. No objects yet, just Common blocks that were unexpectedly overwritten. Met Greg Seibert there, who was slinging FORTRAN code there at the time. His Dad was a Hughes alumnus.

Next, I was recruited for a job at Infotech Development. They had won a contract to re-engineer a track-while-scan radar controller; aka, Precision Approach Radar: PAR. The PAR was used to guide aircraft to the airfield in the event of low visibility or loss of navigation equipment. An operator sat in a shipping container of electronics and monitored the raw radar signal on both azimuth and elevation axis. The operator could move a cursor to the radar signal, press the Track button, and PAR would acquire a Target, periodically revisiting it to update its location when it wasn't otherwise scanning.

Built originally by Raytheon, the Target Data Computer consisted of magnetic core memory and a CPU that had a three-way branch. Code written in a unique assembly language loaded via paper tape. There were some interesting subtle mechanisms encoded within.   Especially in the Beam Steering Unit. And the Tracker.

Realtime Mission Critical Software.

I was the software team lead. It was a formidable project. We build our own boards using a brand-new technology called Field Programmable Gate Arrays. We build our own real time operating system from scratch. Learned 68000 assembly and C. The industry insiders were sure we would fail. We succeeded.

I recall using a planning tool called Micro-Planner on a 512Mac. Learned how to orchestrate schedules to either optimize resources or time. Sometimes it made sense to serialize tasks, sometimes it made sense to execute them concurrently.

Aside. At that time, I was keen to understand how unplanned work or "fires" effected schedule. So I made a simple sheet of grid paper. I asked each developer to list the tasks he/she thought they would be able to do in the next month. Then draw a line below that. And each day list the fires. Then pencil in the amount of actual effort for either planned or unplanned work. At the end of the month I tallied the result. Determined the same result across three companies – huge, medium, and small – the amount of unplanned work was always about 25% for the junior developers and more like 50% for the senior developers. This knowledge helped me hone my ability to estimate software jobs accurately. A valued skill in that day.

Most of the developers and had never seen assembly or C code before. But I knew they were smart and convinced them that they could succeed, and they did.

After the radar job was complete, we were tutored on the process of writing DoD proposals. Business Development, it was called. Start with topics, which lead to storyboards, which lead to diagrams and sales-type prose. Everything was critiqued, constantly, by both the lead and the peer team members. Polish. Polish. Polish.

The company won a big GPS-related procurement for USAF. GPS ISF (integrated support facility) [I wonder if that's going into MSFTs recent Azure win].

That enabled us to go after in-cockpit GPS Navigation integration studies. Did that for the A-10 and C-141.

I was the Chief Engineer on that A-10 CDU upgrade proposal. Got to meet the A-10 management team at Grumman. Vetted at least four CDU vendors … design, capacity, manufacturability, reliability. Visited sites in the US and Canada. To me it made sense to push the CDU compute capacity up to a 32-bit platform. There's nothing worse then putting 6 pounds of software in a 5 pound bag; it seems to work fine up to a certain point, then fails.

My Ship Came In

I eventually wrote the technical proposal to re-engineer a piece of antique Test Equipment that calibrated Instrument Landing Systems (PIR). Small potatoes, just 6 million bucks. We gave them 13 million bucks of value back (that was the TRW price).

The PIR measured the accuracy of the navigation tones used in Instrument Landing System (ILS) beacons. The radio in the aircraft used the 90 – 120 hz audio tones to detect their relationship from azimuth centerline using DDM (difference in depth of modulation). There was another similar band 3x up that detected the relationship to the elevation centerline. This allowed the pilot to "fly to the needle" on the analog cockpit display.

The unit had to work from -40 degrees C to 100 degrees C. Had to measure signal accuracy at very low power levels. Had to work on a battery.

The unit we were replacing was a very simple crystal radio that could only measure a single frequency set at a time. One per airfield.

Our design sampled the RF at a high frequency and used both hardware and software digital signal processing techniques to get to the ILS signal. Doing that on a Ni-Cad battery was no mean trick. The UI was simple, essentially a bank ATM interface instead of an array of knobs, switches and dials.

Got the Program Manager/Lead Engineer role. Had some great people (29) supporting me. Gary Johnson and Steve Smith worked the RF. Rich McCormack took on the Antenna and LCA aspects. We had to innovate the design from mostly analog to mostly digital in order to get the required performance. Earned a US patent for digital signal processing. Hard on the battery sub-system. My old Hughes buddy Mike Case worked that aspect. My old Hughes buddy Mike Courtney did the heavy-lifting of writing the code, in Ada, as I recall.

Recently found a few people selling used Portable ILS Receivers (PIR) on eBay. They include a shot of the nameplate: Infotech Development right there with the contract number. Will link those to the main website.

Learned a lot about RF circuits, and their ability to work at extremely low and extremely high temps. At one point had to switch to polyamide circuit boards.

Later recruited Gordon Osterhues to work the RF Module on a Radar Altimeter product derived from PIR. Swapped out some software and the RF module and we had a Radar Altimeter.

Towards the end at Infotech, I began working on an Expert System for USAF. Expert Systems are based on rules. But, given my Bio background, was more fascinated how the brain did it. How did systems self-organize? How did they naturally retain and forget information?

I recall driving up to UCLA in 1992 to take a graduate extension course on Neural Nets. Given my previous training in neural biology, analog computing and Kalman filters I sensed the next Event Horizon. Today it’s called machine learning, usually as TensorFlow running on some special silicon in the Cloud. The antique Von Neumann cloud.

At that point in the early 1990's the Peace Dividend arrived. Meaning: 250,000 aerospace jobs had to go away. Mine went too. Was clear to me that no one would be hiring high tech program managers.

One day the axe fell. Softly. Kindly. With respect. With a few minutes to clean out my desk, say goodbye. Turns out you can’t always expect that.

Tetra Information Technology Origin and Explanation

My friend Steve Aaronson took me out that night to the HB Beer Company. Then we headed over to Longboards for more. Started to discuss the idea of my own consulting company. My father had done a similar thing when he started his consulting company, Collier Controls. Steve was a consultant and knew the ropes.

We were pondering what to call it. Then at that very moment, as if by fate, a woman with substantial, unavoidable cleavage walked in the door. I recall, at that instance, we simultaneously uttered the acronym: T.I.T. Beer Fog?

And so, Tetra Information Technology was born. Class Mammalia's common denominator. Ever heard that saying "sucking off the governments tit?" That was the angle. Some are offended by the reference … perhaps they are reptiles.

Infotech retained me as a Consultant. I went up to USAF Sacramento and delivered the Expert System to support maintenance of some radar circuits we had built for them. On a Mac Laptop. HyperCard UI to a Nexpert rule base. [Note: later stolen]. Mother's milk.

I got a few more Air Force projects. One attempting to resolve the undesired speed increase in the FPGAs we had used. I brought in Chad Nikoletich who was the design expert. Another handing off PAR system knowledge to SAIC.

Maybe the Coolest Thing I Ever Did – from Scratch.

I had a short but fun consulting project working for Steve Smith on a TV to FM converter. The idea was that you are in your car, stuck on the 405 freeway, and want to hear the Lakers Game that's being broadcast on Channel 7. This under $50 gadget would convert Channel 7 to FM 95 that you could hear on your car radio.

Used an all-in-one-chip of 8051 linage that had built-in EEPROM, RAM and CPU. After erasing it under a UV lamp, I'd plop it in the EEPROM burner for programming.

I recall now drawing and etching the circuits into the proto board. Inserting the 2-inch long socket.

Made a custom assembler/linker/S-Record translator out of Apple HyperCard. Copied S-Records from the Apple clipboard to the EPROM burner. Voila. A real micro-controller. Bit banged the LCD on a two-wire serial protocol. Handled the input. Bit banged various RF synthesizers and loop mechanisms. I had built the entire thing from the silicon on up on my own. This time: no in-circuit emulator. I got a breadboard, bought some parts and built my own serial readout from scratch.

During my tenure at Infotech Development Inc, I had developed an expertise in a new handheld technology from Apple called Newton. I worked with Stan George on a proposal and we got to know each other. Later he asked me to work on a project to display Electron Medical Records using it. We developed a way to connect an Oracle DB on a VAX to the handheld over an ordinary phone line. I came up with a super-efficient serialization protocol.

NewtonScript was the basis for one of those rare, beautiful software platforms. Message-based as I recall. Pascal like. Object-oriented data, in a format that looks like today's JSON. In retrospect, programming in it was a privilege. Was the first time I heard the term: Endpoint.

Then Stan and I embarked on what we now refer to as The Newton Debacle. We did a lean startup (PDA Solutions); Our super-efficient, super-fast protocol was key to success over our competitors. We traveled from potential customer to customer, showing them the product. They'd tell us what it was missing. I'd code it up on the flight home. (Airline seats used to be big enough so you could type).

We began to get orders. And re-orders. Stories of how the nurse's Newton distracted kids with cancer. Felt like I was on a holy mission.

Our competition was funded to the tune of $12 million dollars. We had a tenth of that. Cash was super tight.

Then one day, after being notified that we had won three large (100K+) orders for the system, we gathered in the conference room to get funding to service the order. This meant all the $85,000 of AR owed to me would be repaid.

What an amazing journey. What an amazing software platform. World class handwriting recognition. Especially when trained a bit.

Real time integration over point to point telephone lines and that new brand-new thing: The Internet. We connected with Cooper Lee who was doing VOIP ATM over the Rio Grande, poking a whole in the Mexican phone service monopoly.

Then, fatefully, the receptionist walked in with a dispatch from AOL: Apple Kills Newton. Apparently, Steve Jobs never used my healthcare app. Had he, it's entirely possible he would have made better decisions and be with us today.

We visited Apple, hoping we could take over Newton Systems Division. Steve was gone that day. Apple wanted 75 million bucks. For a company they were tossing out. I thought 10 million was more reasonable at that point.

I had spent five years becoming a specialist on that Platform. Lived and breathed it. Authored a tongue-in-cheek novella titled "Condos On the Moon" on it. Was my joy. Now extinct. Now more broke than ever. Another Dead Platform.

We tried porting the App to the General Magic platform. Some of my best developer times were with Rick, who helped me convert the code base from NewtonScript to C++. Stan and I would travel up to San Jose during the week and come back on the weekend.

Sadly, General Magic's PDA lacked the speed necessary to work the application. They had cash problems too. They disliked my business-like tabular UI and wanted to go with something like Microsoft's Bob. Ultimately things didn't work out.

Fortunately, I had developed a side-expertise in Lotus Notes and Domino at both Infotech and PDA Solutions. After PDA Solutions declined to retain me, I did an interesting Notes project management application for the City of Chicago. Probably not used. Fun to build, however.

Eventually wound up at Boeing Anaheim doing Domino work. Working the Web. Initially for the Missile Defense System, then on a Software Quality Toolset. Then an Aircraft Modification system, then a Project Information Management system. Then a Risk Management System. Then a quick replacement for an existing, and very costly contract data system. Finally got to work from home on a Supplier Quality System that spanned 18 different data domains. It was web-based but could cache local data according to a formula and work offline. Made sense if the rep was working from a place where there was no internet.

Domino allowed us to create data-centric web applications at a breakneck pace. I built many. The platform however did not provide a way to create views combining data from multiple data systems in real time. The lack of decent reportability was cited repeatedly in various FUD attacks from the Microsoft enthusiasts.

This vexed me. I learned Java servlets and authored 35 pages of code I called J5J Search. A user could now visit a UI; they could add data sources and select the query and properties to be extracted. I managed to exploit memory (as opposed to Domino's use of the disk) by storing and sorting the result in a Big Array of Strings. I recall writing my own quick sort, because Array. Sort didn't exist yet.

At one point I had 23 data sources that could be joined, categorized and sorted. Ran fast.

Was fun to watch on the Performance Monitor. For a change Memory and CPU spiked twice: once during data-extraction, once during output. The search was useful. It pointed out voids and problems with the data. Huge result sets came back quickly.

I read a story in the Wall Street Journal one day about AJAX. Figured out how to do that with Domino, which made the applications smooth and responsive instead of jerky server page updates. Domino had "agents". The front end sent an off-page AJAX query to the agent, which marshalled the data and returned it as JSON, a novel idea at the time.

IBM acquired the Platform. Microsoft development SharePoint. The platform lost share and my job prospects to boot.

Fortunately, I had developed enough expertise in JavaScript and HTML to work the web in other ways.

A short gig at Trane provided an opportunity to learn Object-Oriented JavaScript. The script ran in an embedded IE5 browser. The processor was not quite up to speed however and at times ran over 100%. Also learned way too much about browser caching.

The Trane folk were pleasant and smart. Trane junior and senior management was probably the smartest Engineering Management I ever saw; they continued to improve the product until they got what they needed.

I worked at the Technology Center in La Cross. Epic winter snows. Chillers (think a round tube, half the size of a freight car) were their specialty. They ran so smooth you couldn't hear them.

Got me remembering the physics of Heat Transfer and Air Conditioning. Three modes: Conduction, Convection and Radiation. Modeled as a 3-dimensional grid over time. Thermal energy on one side of the System is transmitted to the other side. Exploits changes in gas pressure. Move the Set Point and one side gets hotter while the other side gets colder.

Hops a Member of the Hemp Family

After that ended, I went back to Yakima to work some Domino Projects at Roy Farms. I had done Newton and Domino work for them over the years. Learned a lot about all the stuff that goes into making great Hops. Soils. Spray applications. Drying applications. Processing applications. Well-tracking applications. Payroll. Contracts. Property Taxes.

Hops are phototrophic plants. They start growing out of the ground on the first week of May and are 20 feet high at harvest in late August. One set of hops was grown for flavor; the other for bittering. The bittering type were extremely flammable.

The hops business was unique, so I was essentially building tools for a small number of people with a familiar, intuitive UI for each application. Many variants on a single theme.

Lotus Notes Domino worked on the premise of documents, views and agents. The agents are essentially scripts that manipulate the data; sometimes autonomously. The scripts start with a Session variable from which many other objects were derived.

The data was mostly collected via Spreadsheet. Then imported as batch. The resulting data was aggregated into summary records and checked. Then converted for output to the back-office system.

This time around we worked with Google Maps to estimate world-wide hop production acreage. Heather combed Maps for indications of Hops. I entered the KML into a Domino DB containing other meta data.

As that ended, I unexpectedly got a job as a junior J2EE architect at AT&T. I thought the job was supposed to last 6 months, but after completing 18 inches of documentation I was let go at the 6-week mark as the Recession of 2008 started. Good news for me frankly. Was spending 8 hours a day on a group call. Not fun. Project doomed to failure. Was predicated on the idea of Apple allowing SWF to run on iPhones. Supervisor apparently wanted to bring in an old friend.

I returned to California in December of 2008. The Great Recession had taken effect. No work to be found. Learned Dojo, an early JavaScript library. Learned PHP and MySQL. Time well spent upgrading my toolset.

After 9 months, landed a low pay contract gig at Verizon Wireless. Not enough for positive cashflow, but enough to stave off The End for another day.

Great people. Great culture. Great products. The polar opposite of AT&T. Learned yet another JavaScript toolset, this one called MooTools. Learned CSS at whole new level.

This time around I got to build an object-oriented JavaScript widget that lived at the top of every page. A dynamic, cross-domain menuing system. On load, it read a huge JSON file and pushed it into its Model. The Controller than orchestrated View construction. Ran fast.

Later did more of the JSTL backend page development that was popular at that time.

Eventually moved on to a well-paying 1099 gig in San Diego. 85 miles down. 85 miles back. Fortunately, I-5 moves at 80 mph most days. I remember being gridlocked in north San Diego after a shooting one day.

In San Diego I first worked for Intuit. And jQuery. Part a team minting a tax rebate card. Then at Qualcomm a few miles south. Then back at Intuit. Developed a light-weight dynamic object-oriented page style. One dependency: minimal jQuery.   Dependencies minimized.

I got tired of driving back and forth to San Diego every day. Eventually found a gig out of El Segundo that was mostly remote, requiring only a few days a month enduring the crushing 40-mile drive from Huntington Beach to El Segundo.

MaestroDev was an open source startup. They had developed a DevOps tool and its UI needed to run fast and fluidly. My specialty. I had learned from Trane that size does matter in JavaScript. Less is more. Minimize repaints. Minimize dependencies.

The team was distributed all over the planet. CTO in Sidney. DevOps gurus in Spain, Dallas, New Mexico and Santa Monica. Used FlowDock to keep us all in sync. A pleasure to work with a group of really really smart co-workers. Learned git, dev ops, chrome dev tools.

I wanted to work with MongoDB, which was similar to the Object Orient Data System I had enjoyed in Domino. MongoDB looked like JSON to me; JSON is simple yet powerful. It is processed directly by backend JavaScript systems.

I ended up with a gig close to home at Sperion. Learned ExtJS, Groovy, Mongo, SQL Server and a host of other tools.

This outfit tracked autos in real time. A couple million at a time. It also controlled autos in the sense that a kill switch would be activated if the auto payment wasn't made. An embedded IOT GPS device would reveal where the repo man should go.

Big real time data. Too much, too fast for conventional SQL databases. Redis was used to capture and state the auto position and velocity. Backend was Groovy. Developers wrote tests.

Was pushed out of that a year later in a mass purge. Apparently, the company was going public and required a better Revenue to Employee ratio.

So, I had a few months to kill. Learned Google's Angular. A completely different approach to UI development. YouTube videos and books were key. Also installed Visual Studio Code and began to make stuff.

The next gig at Thales involved both Angular and Enterprise Java. Was the first time I heard of Kafka. Only used Scala for backend UI templates. The rest was Angular 1.x. There were lots of APIs to code in Java ... began to get the hang of it as I repeated and repeated them.

I decided I really liked Angular 1.x. The most intuitive framework I'd seen. Code often seemed to work the first time. Worked with Bootstrap to produce a pleasant, calming user experience.

Aside. I have recently discovered a platform called PreactJS. Has a similar programming API to React but is done closer to the metal! The opposite of today's bloated platforms. Uses regular web elements. 3KB load size. This is brilliant. Instead of adding bloat, this makes the pages lighter and more responsive. I enjoy front end programming in Preact.

But, back to the story at 2014. The last gig in Southern California.

I worked the next Gig on a 1099 to my personal business, DBA Tetra Information Technology.

Pay was great. It just took 60 days to arrive. I worked most of it at the Aliso Viejo tech center. Lexipol's developers needed to be separated from the business folk so that they wouldn’t be bombarded with questions or feature requests.

Lexipol’s business were Police and Fire Policy Manuals. Legal exports provided best practice on a given topic; the chief could either write their own version of that paragraph or use the recommended version directly. Think Policy 3.1, Use of Deadly Force. Best practice is to attempt de-escalation, no shoot first and ask questions later. The force read it and had to pass a series of questions to reinforce the policy.

This was an Enterprise Java shop. I did UI in ExtJS. Occasionally got to modify the APIs for feature requests. XML objects in a Relational SQL Server (SS) Database. Slow as death.

Then, the day after my 6oth birthday, I flew to Seattle to help my aging parents -- in their mid 90’s -- die. My Dad passed quickly. I moved back to Mercer Island to care for my Mom, a privilege. The Lexipol gig meanwhile continued remotely for a few months. Started to beat the bushes.

Got a short 7-month Gig at Agilysys doing one last project in Angular 1.x. Fast, responsive with a secure result. First time I've interfaced with a Credit Card interface ... Freedom Pay in this case. As per my custom, I architected the system to have a Data Service as opposed to embedded fetches. I created a Mock Data Service. Consequently, never got blocked by API malfunctions.

Institutional Stint

Next, after a long 6-month hiatus I took on a job a Fred Hutch Cancer Research Center.

Grateful I am for that hiatus. Time well spent with my Mom before she passed. Couldn't figure out which UI platform to master next. Turns out, it would not have mattered.

The Fred Hutch finance users had chosen a platform that the IT department evaluators loathed. I soon understood firsthand.

Once again: XML object blobs in a Relational Sequel Server Database. Seventh normal form. ETL required a special tool that never quite worked. A .Net Entity Management system. C# was involved to a certain extent.

Good news however: the backend scripting language was essentially JavaScript. Bad news: the objects were in XML. To access an attribute, one had to type the pattern: obj.GetQualifiedAtttribute("some property"). Imagine the pleasure of typing that a few thousand times. Sigh.

Domino, which didn't use Sequel Server, managed to do this and was wicked fast doing it IF the backend storage was solid state. This Click platform, however, was Slow as Death. Everything had to be cached. Pages often timed out unless a high-powered server was deployed. Especially after deployments. I had to "warm up" the packages. Otherwise they’d time out.

Had this been built with Node, Bootstrap and Mongo, it would have made sense.

Part of the job was Developer. The other part was DevOps. Building snapshots of patches that were applied to upgrade the codebase. That part: a bit tricky.

The first year of the engagement was normal. A contract Project Manager, David, who knew his stuff and collaborated with the technical folks to optimize. A distributed team inter-connected via a Slack Channel. Spoiled I was.

But outside of that project Silo, the entire Fred Hutch gig felt like a Twilight Zone episode, circa 1990. They worried about imaginary problems and ignored the real ones: a bad case of **Trump Syndrome**.

I believe I especially antagonized management by suggesting they organize work into Slack Channels to accelerate productivity. Emails, physical meetings, and phone calls. Synchronous work always bottlenecked by lack of a meeting time and space. So quaint. Where was the telegraph?

[These days the whole outfit is working virtual. Even Microsoft is changing it culture from Email to Teams. I wonder if Fred Hutch will follow suit.]

I earned my Keep their however. I figure I saved them half a million bucks by doing a difficult mission critical platform upgrade by myself instead of paying the Vendor.

I wrote a huge set of Protractor Automated tests against the codebase. It caught bugs, notably in Platform upgrade patches sent to us by the Vendor. Best practice, I thought. Very tedious, however.

Shit Happens, Wear a Hat

I didn't mind being let go. I got unemployment. Thrice. Turned out, the new Project Manager in the second year didn’t play well with me; yelling at me at one point in a meeting as I tried to explain how schedules worked, then storming out. Always hostile. Not particularly able to appreciate my contribution.

Then there was the one hour, three bus commute each way was a misery; and it was about to get worse.

Wished they had allowed me to collect my personal stuff and go home. Instead, it came piecemeal, taking two months. Not respectful. Usually endings involve having lunch with your workmates and pleasant goodbyes. Not there.

I’m probably a better fit for a Software Development organization rather than an IT Department anyway.   SW Dev orgs have a pleasant day-to-day work pace that makes me feel good at the end of the day. Only felt that way at Fred Hutch when I had knocked out a killer data dashboard.

The Search for Employment Since

I figured I'd be out of work for a while. During my tenure at Fred Hutch the industry transformed itself into Cloud, Containers and Kubernetes. Apache Kafka was now Lord of the Queue. Apache Spark had overtaken Hadoop in the Map/Reduce space. Backend gigs were different now.

Things were also different on Frontend too. Everyone wanted 2 years of React or some other bloat-worthy equivalent. I longed for elegance.

Flutter, written in Dart (as JavaScript variant), was solid on the IOS and Android handhelds. But had no web-browser port; that piece, Hummingbird was recently released. No Flutter jobs yet though.

Which takes us to Golang. I had taken an extension survey course in 2015. Hoping for the same magic that occurred when I learned Microprocessors in 1977.

Golang is so elegant. So smart. So fast. Scalable on chip.

Feels like the future to me.

At the same time, also getting reacquainted with Akka.   That's a JVM (Java Virtual Machine) format. Akka implements an Actor framework. Messages are sent between actors. Code is generated for both a successful and failed handoff. Often used in a streaming context. Uses a very fast machine to machine data protocol. Works with Kubernetes, Kafka and Spark.

Lightbend.com is a good place to visit for that technology stack. Scala is the preferred programming stack, although Java is also a possibility. Scala is smart and elegant like Go lang.

I'd like to take the Actor model to the next level and create silicon for a data flow computer. Thinking dedicated, plastic node-to-node communication.

Job Tests

Did a whiteboard interview for Amazon a few months ago. As usual, freaked out. I don't do well in whiteboard interviews. In practice, I tend to code a little, then test a little. Over and over again. That's a low-pressure, confidence-building situation. In contrast, whiteboard interviews are intense; I get overwhelmed … the world closes in.

Almost got through one recently with Disney but managed to momentarily forget about the Unique access feature of both Maps and Sets. It seems like they've been getting harder and more complex since I started. As usual, I messed up on the related "how does it scale? Log n, n, n squared?" Big-O question.

All questions about issues that never surface in a real-world environment. Weird. Apparently there's a shortage of people who can answer these trick questions, but a surplus of those who can actually do the job well.

To avoid the torture, I post code examples on my website. I post code examples on my git account. You'd think that would be enough. That's proof.

Let's compare this to the act of hiring say, a musician. You might, for example, ask to hear them to perform something they had rehearsed. That's fair.

This doesn't apply hiring software developers apparently. Whiteboard interviews, in contrast, are more like asking a musician to spontaneously compose a polished song that makes the hearer feel "moved" in a certain way. On a whiteboard or its virtual equivalent. Humming a few of the bits, I suppose.

There is apparently no cognition at today's tech giants that the best indicator of future performance is past performance. Such a reliable indicator. Ignored, in lieu of a random challenge: Magical Thinking At Scale.

Before Microsoft got involved it used to be easier. First there was the practice of importing vast amounts of labor from the other side of the planet. That affected American programmers, causing our rates and opportunities to plummet. Then there was the lawsuit by their Contractors. Since then, contract gigs can't last longer than a year. Anywhere.

This created economic instability.

One aspect of this must be the white hair I now display. These days companies won't tolerate hiring bias based on sexual orientation, DNA heritage, skin color, etc. Certain colors of hair are a different matter. I recently spoke to a peer senior programmer and she concurred, citing the chapter and verse of her experience. A recruiter recalled a client who didn't want any more "senior" Front End candidates sent her way.

From time to time the Employment Security Department called us over for a meeting with our unemployed peers. It's a Sea of Gray- and White-haired folk. Mostly technical types these days. Often programmers. They coach us to remove dates from our resumes; limit the trail of our experience, etc. Apparently, it's a "real-thing".

Back in the day when I was a twenty-something, we respected the folks with experience. Over time, they had proven themselves to be valuable. They foresaw problems and mitigated them in advance. They were always pleasant and engaging. I enjoyed hearing their stories, and the lessons derived from them. Frankly, I was in awe. Was comforting.

Meanwhile, back to the code-bench. I especially enjoy writing Go in Visual Studio Code. Reminds me of the NewtonScript days. Want to port it into a Raspberry Pi at some point.

I'm trying to get trained in The Cloud. Google allowed me to attend a course two years that taught me how to read vast amounts of IoT data, pump it into a cloud store, query it, train a model, then serve that model up via a Flask application via Kubernetes. Google really a First-class Outfit. Big data with multilayer security. Not that hard actually for a user type.

Life Is the Convolution of Fate and Random Chance

My email tagline. Convolution being the Mixing-Bowl function.

One thing I've realized lately is how things are beginning to come full circle in the technology space.

Before the dawn of the Processor, electronic devices were designed so that One piece of the Machinery did One Thing. That's one kind of design head space. It's a different design head space for the Processor, where One Thing is programmed to Many Things as quickly as possible.

The former can Scale, but there are more discrete connections. The later constrained to the execution and fetch cycles of the silicon.

I wonder how this whole thing is going to turn out. People are saying that Algorithms are going to take over all the mundane day to day jobs of trucker drivers and such. But knowing as much about Software as I do, I think that Software Developers of all stripes will also have their share of pain. All that Amazon space in Seattle might be homes for the homeless Developers in the future … Algorithms will take the jobs of their Creators.

-