From: James R. Olsen. April 28, 2022
Hamilton, Montana

To: Board of County Commissioners, Ravalli County Clerk and Recorder

Subject: Discussion and decision on letter to Secretary of State regarding local post-election audit, Agenda Item 5, April 22, 2022, Meeting of Board of County Commissioners.

I watched subject meeting on Granicus. I understand from that meeting that there is going to be a presentation on the voting machines used by Ravalli County and election security in May.

Request for Information. I hereby request a copy of all information provided to the presenter so that I may prepare myself for the meeting. I have some technical background in this subject.

Preserving the Republic. I was working my way through school, living on the economy, when I went to a college I could afford, which happened to also be the center of gravity for protests in the late 1960's, the University of California at Berkeley. I didn't have a lot of spare time, but, on my way to classes, I would stop at the tables in Sproul Plaza to find myself talking to real live communists — their rap was that the government is corrupt through and through; we can't trust our government; we needed to tear it down. I acknowledged the flaws; I saw some of them on the street. But, when I asked what it would look like when it was rebuilt, I didn't like the answer. I wasn't going to fall for the "tear it down" story, unless I got a good answer for how our Republic would thrive afterwards. I have applied that rule ever since. Luckily America didn't fall for this story — but some other countries did, going from a Republic to a Dictatorship.

Anyone can come up with a scenario for how to steal an election, manipulate machines, put secret agents in places of power, but that does not mean the story is true. If it "might" be true, then I would feel obligated to call it what it is, "maybe."

So, when I see someone saying the election was stolen, I wait to hear some facts, some reasons. It swings both ways, stories suggesting voting machines swinging Republican.[1] Maybe it's true, maybe not; so I wait. If some believes it because someone else is telling a story about it, letting us know who it is would help, so I can run it down.

A claim that our Republic is that broken deserves more than simply taking someone's word for it, accepting cherry-picked snippets that get one's blood boiling, maybe a story-teller with something to sell or an axe to grind.[2] It goes too far when we start treating our neighbors who happen to work for the government like they are some card-board character created by some outsider with a story to sell, instead of who they are. I hope we are careful to not fall into this.

In the '60s I owned a car. It was vulnerable. I used to keep a couple of wires handing in the back seat when I went to the beach so that if I lost my keys in the sand, I could hot-wire it.[3] Just because it was vulnerable, doesn't mean it was stolen — it never was.

---

[1] Republicans Have a Friend in the Company That Counts Their Votes, https://www.dcreport.org/2020/12/31/ess-voting-systems-a-friend-to-republicans/

[2] According to the *Epoch Times*, a movie, *2000 Mules*, based on an upcoming book by Dinesh D'Souza is coming out in May. The theme is about how the election was stolen — he has written and made movies along antisocialist lines in the past. The trailer suggests widespread "mules" bring sealed ballots to drop boxes, also known a ballot harvesting, which it claims is illegal in all states, which does not appear true in 2020, though some states have since made it illegal. We'll have to see if this has merit in other states or Montana one the complete book and movie come out. The trailer does not claim the votes in question were not valid votes, however — so did logically lead to the claim that it "proved" the election was stolen — since they appear to be valid votes which presumably would have made their way to the voting box one way or the other. But, we'll see when the whole story comes out. One wonders why Mr. D'Souze, who has alleged proof of illegally dropping off ballots has not filed criminal complaints.

[3] How To: Hot Wire A Car! (classic car): https://www.youtube.com/watch?v=rbUIwPjqWwM.

As I walk through the vulnerabilities of machines with software — nearly ALL of which require very sophisticated techniques and equipment, these of only some of them. It is old news. Industries have known for decades that machines and software a vulnerable, machines the run the power grid, dams, nuclear power plants, cars, trains, planes, and cell phones.

Examining the security of our elections and the machines that count votes is worthwhile on the other hand. I believe improvements can be made, and that the machines do not raise to the level of security that I see in Defense Systems and Air Traffic Control — and they should get a lot closer. I will discuss each of these points in more detail, but the summary is:

- All systems with large amounts of software have bugs, simply because they are such complex beasts.
- There are designs and protections that can be implemented in a system to help protect it from purposeful malicious manipulation. But in the end, the defense industry insists on *people* they trust — it is simply impossible to have a trustworthy system without trustworthy people — even if the technology is only pen and paper and you hand count.
- The methods for verifying election machine described by the clerk and recorder are based on the same methods used by the Defense Industry, Homeland Security, and Air Traffic Control.

My Background. I wrote my first line of code in 1968. I have written software code, written requirements for software, designed software, managed large software developments, tested and verified large software systems, maintained configuration management of source code for software, and delt with security of software and hardware that included software. I did this for over 40 years.

My education includes not only the practice of software coding, but the underlying theories of computers, software languages, operating systems, and modeling. The size of code I was involved in defense, air traffic control, and homeland security ranged from 250,000 lines of code to over 1 million lines of code. I have also produced, single-handedly, software for commercial products, including one patented product — two were licensed and were on the consumer market.

A good number of the defense programs have what is known as Security Accreditation and Certification requirements, which include an analysis of the software to accidental and inattentional security breaches. This is a specialty; I do not claim to be one. I have worked with specialists and used their analysis as part of the design of the software. My company bid and was awarded a fixed-priced contract to do such an assessment on the UK Astor program. I ran the program and had experienced software security specialists working for me.

I have worked in Air Traffic Control programs which include what is called Flight Safety-Critical software and hardware. Again, there are specialists; I do not claim to be one. Again, I have worked with them and included their designs and analyses in software and hardware.

Appendix 1 contains the details from the relevant experience from my resume.

All Large Software Products are Delivered with Bugs. I have worked on them, helped design them, been the customer who tested, verified, and accepted them, been the contractor who designed and build them — systems with large amounts of software. While there are a number of design methods to mitigate the consequence of errors, in NO case were these error-free. If the voting machine does have over 600,000 lines of code it would be very, very rare if it did not sometimes function other than as intended — a bug. For a delivered system these are often inconsequential, possibly causing the software to stop, the locked machine you have to reboot – or a loop, the swirling a beachball on your laptop. But serious errors can occur.

It is impossible to try out all of the possible paths in any program of any size. In Appendix 2 shows that trying out all of the possible paths in most software would take longer than the life of the

universe, even if the "try-out" process was automated. Errors that are sitting there waiting for some combination of paths to occur can and do remain latent for years until one day the software does not do what is expected. (Fun computer science question – what does the code the end of Appendix 2 do?)

There are ways to mitigate the consequences. In an Air Traffic Control radar, the FAA doesn't mind so much of a failure, having to take the system down, but they are absolutely paranoid about putting a dot for an aircraft on an Air Traffic Controller's screen in the wrong place. So, the radar is actually two radars and has two complete copies of the hardware and software in two strings with a Comparator on the end, simple hardware, small software package that compares the results these two strings of hardware and software to make sure they are the same, turning off the radar if they are not.

So, a voting machine 600,000 lines of code is almost sure to have errors in the code. The question is what are the safeguards? They are not nefarious, purposeful errors — they are mistakes unfound — most of which unlikely to effect on the voting count, but some other feature of the machine or the accuracy and precision of its performance in detecting a mark on the page.

Proprietary Code. The Department of Defense buys copyrighted commercial code all of the time. Servers with Microsoft Windows Operating Systems are common — though the operation systems has been subject to security scrutiny and provides a mechanism for security modules to be added on; Commercial mapping systems, commercial laptops are common. While there are often contractual prohibitions from modifying, reselling, etc. I know of no case where making your own backup, examining the executable 1's and 0's. However, access to the source code of an operation system is not the norm.

Access to the source code (in a language use as "C" which is "compiled," that is transformed into the machine instructions that are stored in memory and executed by the machine). For a large system for audit is unlikely to find malicious code unless the auditor already has some idea of the type of malicious activity to look for.

Few systems are secure from malicious attacks. The only computer that is almost-perfectly secure is in room with a safe door, no windows, in a faraday, vibration isolated, with no powerlines running into the room unless they are isolated. A simple metal box doesn't usually work. With no perfect solution that isn't expensive, DoD deploys systems that are not perfect but have been subject to a rigorous risk assessment called Security Accreditation and Certification.

Every time an electron flows, it creates a magnetic field that can be sensed. Every time a magnetic field is in flux it can affect an electron in a wire. Any presence of electromagnetic energy — called by names such as radio waves, x-rays, and light — performing functions such as wifi and cell phones — can affect the electrons in a computer. Since the data and code are created using electric circuits, if you can affect the electrons, you can affect the data. Of course, normal electronic circuit designs mitigate a nearly all of this, otherwise they wouldn't work. But, malicious actors, with sophisticated equipment, work with the left-overs that the "nearly" does not catch.

There are standards in DoD for a very secure computer and software system, beginning with a standard called TEMPEST, though most of the systems I have worked are not perfectly secure — most have a formal security risk assessment made considering both the vulnerabilities and consequences of a compromise. Without diving into this, the bottom line is that a voting machine that is connected to wifi is very vulnerable and those that are not are still hackable by a sophisticated actor, especially one nearby.

So, as we skim along the edges of what is unlikely but possible, we get to people. We don't know who wrote the code and if they are trusted agents. This is one of the key features of software security. In DoD, that is what Security Clearances are for.

The bottom line is that I have seen no evidence that the voting machines were nefariously manipulated in the 2020 election so far, and some of my friends had me looking at a number of claims. But, the machines ARE vulnerable to sophisticated attacks. A touch screen voting machine seems to be most vulnerable (except those that print out a paper ballot that is counted by another machine). An optical paper ballot counting machine that is not connected to a wifi or the internet seem less vulnerable. Not connected even to ethernet is better.

How to test and verify. The way to test a system, accept it for initial operation, verify that a maintenance action or update is valid, and confirm it is still ready for operation, is to run it through its paces with known inputs that emulate the real world as best as is reasonable. Thus, if one has a voting machine, one would run through ballots that have been verified by hand to see it the count matches. That verifies a lot.

We then get to the idea of testing the boundaries and exceptions. This may already be happening in Ravalli County. In the case of a voting counting machine, it would be wise to run invalid ballots, improperly marked to make sure they are rejected. It would be wise to run various patterns through of valid ballots that have different combinations of votes, and missing votes. This is the testing principle I would propose to DoD, Homeland Security, or the FAA.

If one wanted to go further, it may be good to hand count a sample of actual ballots, then, at a random time, feed them through the counting machines and verify the count — this may counter malicious code that times its actions based on the clock or level of activity. Of course, a vote of 60/40 in the actual ballots comes out 60/40 in the county could be two error that go in opposite directions, compensating for each other. So, running a single ballots in addition and verifying it works would be an additional check.

In order to capture a baseline, a copy of the storage media containing the executable code should be made before starting and after — though this is no 100% guarantee since it is possible to insert malicious code during the counting process if one can hack into the machine that will do its worst and erase itself, restoring the original code.[4]

But, in the end, if the machine was not built by a trusted agent, it will be difficult to prove that it is trustworthy to a suspicious mind.

Hand counts are subject to error and malicious intent. Checks and balances are required — many of which I suspect already happen. People make mistakes.

It would be interesting to bring a copy to several packs of 100 sample ballots to a public meeting and have the audience take a crack at counting them and see how they do after they have all turned in their results to be compared.

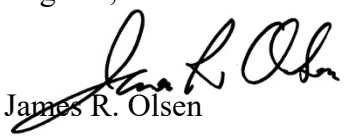
At this point, if we go back to 100% hand counts, given the political environment, a malicious actor or actors could volunteer to count. Of the 45,000 or so people in Ravalli County, it would only take one to want to do it. If a call for volunteers is made, maybe 1/2% of the population would come forth. But there would be a 100% chance the malicious actor would volunteer. Thus, the process would be set up so that it would remain valid even if one or more malicious actors were present.

A long run solution. 600,000 lines of code seems a lot to me based on the function of the machine. But, assuming that is what it takes, along with a hardware design, Montana could put out a Request for Proposal, with security and accuracy requirements stated, for a machine and code, to be owned by the state. If you pay for it to be developed, you own it. It would need to include an option of

[4] There are techniques, such a check-sums, that provide a an accepted probability that two data bases are the same without needing to do a bit-by-bit comparison. https://www.howtogeek.com/363735/what-is-a-checksum-and-why-should-you-care/

maintenance and updates. I suspect it would take about 3 years. Of course, the side benefit if Montana pulled it off, they could sell it to other states.

Regards,



James R. Olsen

Appendix 1 — Resume of related experience

Human Interactive Products, Inc.
James R. Olsen, President and majority owner

2006-2018
Great Bear Restoration
Provides restoration native plants primarily to mines, but also road and public land restorations throughout the Rocky Mountain West and Retail Nurseries in a three-state area.

Jim developed the most accurate Excel **software cost model** in the industry using data from Quickbooks, Time Cards, and inputs for growing cycle for 400 species.

West House Mental Health Crisis Center

Worked with a Clinical Social Worker. Jim worked on a problem important to the Sheriff as well as people, some of whom had been in custody, that Jim met at peer meetings through the Local Advisory Committees — people recovering from mental illness. He engaged in research and conversations to understand the police, justice system, mental health therapists, case workers, and people's experience with mental health and with the justice and mental health systems.

Jim created a business plan for the development and operation of a local crisis center to assist the county in competing for a grant funded by the state legislature. The plan showing both the costs and impact on police and people with mental illness and included results from **a software model** (coded in the General Purpose Simulation System) Jim developed using statistics from county attorney mental health crises events to determine the number of beds required. Presented the plan to a joint meeting of the County Commissioners and a representative of the Montana Dept. of Public Health and Human Services, who said, "You guys get it." The county got a grant; Jim then joined a committee to develop a defined inter-agency process for dealing with a mental health crisis from custody to West House; the facility was built and is in operation.

Bison Internet Café

Teamed with a Mental Health professional and non-profit to implement a place were people recovering from mental illness could safely interact with the community. It was supported by a grant from the Montana Department of Public Health and Human Services – which required by special accounting standards and HIPAA compliance for mental health peers/customers. The model included hiring people recovering from mental illness as staff. **Designed and set up IT and computers for retail use.**

CONSULTING SERVICES
2001-2010

The following programs involved large software content from 250,000 to over 1 million lines of code, with substantial real time response time. Some involved safety critical software subject to FAA certification and approval. All programs required software security, some requiring a small provable modules that interacted across the internet or disturbed over-the-air networks. These software for these systems is written in a variety of software languages including C, C+, C++, JavaScript, Ada, FORTRAN, JOVIAL, MATLAB generated code, assembly language on a variety of computer platforms, from Windows Servers, hardened mil-spec computers, minicomputers, and mainframes most interacting with circuit cards containing embedded software and microcode using a variety of computer chips, some designed and manufactured by Raytheon and other vendors in the team. All

programs involved real-time code, secure communications. Several involved widely geographically dispersed secure networks and system control.

Because I got increasing responsibility as I went along, I did not code in all of these languages. My code is in the earlier programs. My designs and architecture choices were coded by someone else in the later systems. To keep my hand close to the machine, I did some commercial products on the side. Those that I remember — I have written code in C, C+, Visual Basic, Basic, PL-1, FORTRAN, Pascal, GPSS, LISP, APL, and assembly language (machine code) for six or seven different computers. I invented three domain specific higher order languages myself and wrote interpreters for them, one for modeling hormone systems and two for analyzing data recorded by real-time systems, post mission.

Jim developed a team of engineers from around the country and non-engineers from the Bitterroot Valley to participate in defense programs — a training program for proposal management, proposal writing, and data management for local recruits – those who passed muster were deployed on jobs throughout the country integrated into a multi-company team — pay ranged from $35 to $65 per hour for non-engineers and over $100 for engineering positions – and deployed to the programs below.

The company developed a fast-response deployment system — deploy staff to a customer site within 48 hours, including apartment and condo rentals for longer deployments – setting up a proposal center in the customer's location if needed – at locations throughout the United States and the United Kingdom.

Foreign Sales. Jim led teams deployed to support foreign sales of airborne Intelligence, Surveillance, and Reconnaissance (ISR) hardware/**software** systems to foreign countries including countries in North Africa, India, and Jamaica.

Airborne Standoff-Off Radar (ASTOR)/Sentinel R1 Aircraft - This system includes five Bombardier aircraft fitted with a synthetic aperture radar, on-board intelligence analysis and operations center, and line-of-sight and sat-com, along with defensive aides. ASTOR includes a ground segment of ground-mobile operations centers, a realistic training center, and all required logistics, documentation, and certifications (a complete system which is common for the systems discussed below). "Astor can fly over Kalispell and watch us drive on the streets of Hamilton in real time as well as create image streets and buildings." It contains **one million lines of software.** The United Kingdom has operated the system in Afghanistan, Iraq, and Libya as well as search and rescue operations.

Jim was recruited by a Raytheon Segment President who opened the conversation with "I have a $6 billion job for a $4.5 billion fixed priced contract. Ratified by the CEO, Jim got to work figuring out how to fix the failing program that was in the midst of system integration – a multi-company, multi-division team with operations in Texas, California, Arizona, and the United Kingdom.

He worked with the Raytheon Program Manager to reorganize the program. He **reworked the schedule for hardware and software and Integrated Management Plan** to create the roadmap for completing the program. Acting as Raytheon's agent, Jim **renegotiated the major subcontracts** so that their terms and conditions were aligned with the needs of the job. He participated and helped develop strategies for United Kingdom Ministry of Defence (MoD) renegotiations, sometimes acting as Raytheon's agent.

Jim also worked to fix engineering management. He organized a **requirements management** process and served as the interim Raytheon **System Engineering Integration and Test Manager** while Raytheon was looking for an internal candidate for the job.

Jim fixed the **software management**, repairing broken company-to-company and Raytheon-to-UK MoD relationships.

As the system neared deployment Jim was put in charge of production of a sales video for the Farnborough Air Show.

Airborne Standoff-Off Radar (ASTOR) Security and Accreditation Subcontract. At Raytheon's request, Jim negotiation a fixed-priced multimillion-dollar subcontract with Raytheon for his company to perform the **Security Accreditation** in accordance with MoD requirements. It was implemented with profit sharing arrangement with the staff and executed under budget. He used the metrics based Earned Value Management to good effect.

Mobile User Objective System (MUOS) Concept Definition contract and Development Pursuit – A world-wide military secure satellite communications system for handheld and terminals. Jim was both the Proposal Manager and Demonstration Manager during the concept definition contract and follow-on proposal for the development program. Raytheon was in a "come from behind" position never having been prime contractor for a satellite system, although the company had been a payload subcontractor. Jim helped drive the Integrated Management Plan and key system architecture issues as well as strategies to drive down technical risk.

He proposed a multimedia **software-hardware-in-the-loop** demonstration to Raytheon and led the effort. He developed a story-telling approach of presenting the hardware demonstration in the context of multimedia vignettes. One of the customer's called it, "The best demo I've ever seen."

Aegis Combat System Target Data Fusion – Jim led a proposal for the development of advanced target **data fusion software** from multiple sensors on multiple platforms. Won.

MILSTAR Terminal Upgrades – Secure, nuclear hard satellite communication system. Jim provided senior strategy support and authorship for both an Air Force and Navy upgrade of **hardware and software** for MILSTAR secure sat-com terminals. The Navy upgrade was won and is complete.

Cobra Judy Upgrade – A ship with an imaging and tracking radar for monitoring missile tests for foreign countries. Jim provided win strategy support and integration strategy for the Cobra Judy Upgrade Program, including **software integration and reuse**. This was ordered sole source and was deployed in 2014.

Joint Tactical Radio System (JTRS) – **A software defined radio with 1 million lines of code**. Jim provided win strategy support and integration strategy for a JTRS proposal as well as the Executive Summary and other parts of the proposal.

Raytheon Company

Engineering, Software, Program Management, Director 1982-2000

Program support and business development ½ time, based in Hamilton, MT
1993-2000

Developed win strategies; conducted teaming negotiations; led proposal teams, demonstrations, senior management review teams; and proposal authoring for several new business opportunities. Programs included Joint Precision Approach Landing System (JPALS), National Aerospace Simulation and Modeling (NASM), Joint National Test Facility, Patriot upgrades, Battlefield Adaptive Data Distribution, Air Force worldwide logistics management, PRISM (Command and Control Rapid Prototyping), and the National Missile Defense System.

Consulted on **computer and software architecture** SIVAM (Environmental Monitoring and Expert System for Amazon Basin). Developed a movie concept and script for Ballistic Missile Defense for Taiwan.

Led and participated in recovery initiatives for active programs experiencing schedule, cost, and performance problems. These include CAATS (Air Traffic Route Planning **software**) and MK57 (Navy Ship Defense). CAATS required several months of effort in Richmond, BC, Canada and a work permit.

Manager, Engineer, Software, increasing responsibility (Full Time) 1982 – 1993

BALLISTIC MISSILE DEFENSE (BMD) PROGRAM MANAGER. *1992 - 1993*

Led the BMD Operations Assessment (OA) proposal and the first half of the Concept Definition contract. Led an integrated multi-divisional, multi-company team.

Led the development of updated **system and software engineering processes**, including Object Oriented System Engineering and Raytheon's first soft-real-time geographically **distributed software simulation system**, which anticipated a key concept of the current National Missile Defense architecture.

Demonstrated internet-based video communication (pre-skype) with a dissertation on how to improve distributed team effectiveness by matching the communications mode with the communication purpose (from email to face-to-face).

DIRECTOR, DATA ACQUISITION DIRECTORATE. *1991 - 1992*

Led a $500 million per year profit center. Programs included Ground Based Radar (GBR), Patriot Antennas, Iridium (a commercial satellite communications net) Antennas, large phased array radars and several technology programs. Guided program managers and resolved performance, cost and schedule issues. Developed business opportunities in military systems integration, Command and Control, and **software markets**.

Cobra Dane Upgrade PROGRAM MANAGER. *1989 - 1991*

Led teaming, proposal, and execution of the Cobra Dane program. Cobra Dane was delivered ahead of schedule and exceeded profit goals. The Cobra Dane upgrade was the first large real-time Ada program. Developed contract structure, business relationships, and management structure for a closely integrated team from six contractors. By integrating the Raytheon software labs into **a latest-technology software effort**, this program served to show Raytheon the path to growing its software development capacity in the marketplace. The system was delivered to Shemya Island.

Also served on several red teams and proposals including Ballistic Missile Defense Ground Based Radar, MILSTAR Satellite Communications, Displays Product Line.

RAMP TEST DIRECTOR (AIR TRAFFIC CONTROL). *1988 - 1989*

Led integration and testing for the Air Traffic Control radar system for Canada. Turned a failing integration and test program into a success.

BMEWS DEPUTY PROGRAM MANAGER, SITE MANAGER. *1986 - 1988*

Led the integration and test of the Ballistic Missile Early Warning System (BMEWS) radar and managed the site in Thule, Greenland. Recovered much of the schedule slip caused by delayed software by integrating the system and incremental software releases in parallel with software completion back in Massachusetts. Within 60 days from being assigned to this job, we moved the entire electronics, computers, people, and logistics system to Thule Greenland, installed and calibrated the equipment into a prebuilt facility, brought the antenna to full power for the first time, and tracked our first live target ten days from pulling the plug in New England.

This project was caught up in strategic arms negotiations – the job including "to get operational before it is traded away." Greenland is a home-rule protectorate of Denmark. The surprise came when a Danish Admiral arrived to show me and the commanding officers clippings of large scale protests in Copenhagen and the fall of the Greenland Government due to the controversy of my project — I had no idea. The claim by protestors was that our sites was part of a Anti-ballistic Missile (ABM) system, a violation of a treaty, which it was not.

The Base Commander and Squadron Commander wanted to hide behind a wall of secrecy. Remembering Cobra Judy (see below), I took to the Admiral aside and said I would arrange for representative of the Greenland and Danish governments to tour the site (the unclassified portions). The tour happened – the controversy died a quietly.

RELOCATABLE OVER THE HORIZON RADAR DATA PROCESSING. *1984 - 1986*

Led the Data Processor and software development effort for Relocatable Over The Horizon Radar (ROTHR). The ROTHR program advanced Over-The-Horizon algorithms to improve reliable operational capability over prior systems. Though designed as a system for the Navy, they decided not to deploy it for fleet defense. ROTHR now provides coverage for the entire Caribbean and northern South America for drug interdiction.

NATIONAL MISSILE DEFENSE DATA PROCESSING MANAGER. *1982 t- 1984*

Was responsible for $40 million software subcontract and oversight of militarized computer development on the SENTRY and GPALS programs.

United States Air Force, Captain *Jan 1970 to May 1982*

Wishing to continue a career as a Program Manager, Jim resigned his commission when selected for Major along with a Pentagon assignment. Accepted one of several offers from industry.

Meritorious Service Medal • Air Force Commendation Medal • National Defense Service Medal
Air Force Outstanding Unit Award w/Oak Leaf Cluster • Air Force Excellence Award
Small Arms Expert Marksmanship

AIR FORCE SYSTEMS COMMAND, ELECTRONIC SYSTEMS DIVISION

Cobra Judy DEPUTY PROG. MANAGER (SHIPBORNE INTEL. DATA COLLECTION) *1977 -1982*

Based a Hanscom Air Force Base, Bedford, Massachusetts

Air Force Systems Command Program of the Year Award • Systems Command Personal Commendation

This strategic technical intelligence data gathering program and includes a tracking and imaging radar aboard a ship which had been decommissioned, the *USS Observation Island*. Developed software acquisition strategy, coordinated system requirements between the intelligence community and Army Ballistic Missile Defense Command. Was also put in charge of the ship overhaul program and leading the test program. Cost of change orders was under 2%; went operational ahead of schedule.

This program occurred while the public was questioning the impact of radio waves on human health — the book, *The Zapping of America*, was making the rounds. The Pave Paws radar, being at Cape Cod, was already the subject of controversy, with frequent press. After finishing the overhaul in Baltimore and completing sea trials, we sailed her to the shipyard in Boston Harbor — within sight of downtown Boston. The *Boston Globe* dubbed it a "spy ship" the next day.

When we had prepared the Environmental Impact Statement (EIS) for Cobra Judy, we committed to only testing the radar at sea — which greatly mitigated the public hazard. As I was in the midst of coordinating the test program with the various intelligence agency customers, negotiating ship work requests, and planning operational test and initial deployment to Kwajalein and the North

Pacific, my boss added the task of helping with public interactions. I gave several tours a week to various public groups when the ship was in port – forestalling a controversy.

I had started the program when it was in the pre-award phase. After developing the software and computer acquisition strategy, I was asked to coordinate the development of the System Specification for the entire system with the user-customers, which included the CIA, Army Ballistic Missile Defense Command, Space Command, US Navy Sealift Command, Eastern Test Range, and other intelligence agencies and secure communications agencies.

I was on the Source Selection Board, a Technical Panel Chair, and, because of unusual circumstances, also assigned to the Cost Panel which required a general officer waiver. Led the technical questions and negotiations with the defense contractor/competitors. Gave the Source Selection Briefing to the Decision Maker.

During this assignment, I had the additional duty of leading the startup of an Airborne Telemetry program (APATS) and as Source Selection Evaluation Board Chair and Acting Program Manager while they waited for a Lt. Col. to be assigned to fill the slot.

Developed software acquisition strategies for assuring software stability and reuse that were adopted throughout the command; served on several source selection panels including Pave Paws.

FOURTEENTH AIR FORCE (NOW PART OF SPACE OPERATIONS COMMAND)

COBRA DANE 14TH AIR FORCE TEST TEAM (LONG RANGE TECHNICAL INTEL.) *1975 - 1976*

Based at Shemya Air Force Base (now Eareckson Air Station), Shemya Island, Alaska

Member of the user test and operational transition team for Cobra Dane. Wrote some of the post-processing software and designed and conducted live operational tests. When a contract to rewrite the software fell severely behind, Jim organized a schedule review, the results of which was presented and accepted. The program went operational in accordance with the schedule.

AN/FPS-85. SOFTWARE & SYSTEMS (SPACETRACK, MISSILE EARLY WARNING) *1970 - 1975*

Based at Eglin Air Force Base, Site C-6, Fort Walton Beach, Florida

Lt General Claire Chennault Trophy

Maintained operational real-time software and performed operational readiness and systems level hardware and software diagnostic analysis. Work included programming, software design, system engineering and modeling, test design, and designing and collecting metrics that help pinpoint sophisticated problems and failure modes. Work included a queuing model of new waveforms and test design that used satellites to simulate the behavior of ballistic missiles.

When a contract to rewrite the software fell severely behind, Jim organized a schedule review which was presented up the chain of command and accepted. The program went operational in accordance with the schedule.

Commercial Products

Help Yourself To Natural Health *1995 - 1997*

Organized 20 authors and professionals to create a multi-media software product, comprehensive reference, focused on naturopathic medical practices, but including orthodox medicine. This was not licensed due to the market shifting to the Internet.

Letter Drop *1990 - 1995*

Developed a word game. Licensed to Centron Software and on the consumer market for three years.

Bridge (the card game) Scoring Calculator (Patent 4130871) *1977 - 1984*

Developed a microprocessor based calculator that keeps score for bridge. Endorsed by Charles Goren (a leading bridge expert) and licensed to Tri Sigma. It was on the consumer market for five years.

Education

University of Florida

GAINSVILLE, FL. MS INDUSTRIAL AND SYSTEMS ENGINEERING *1975*

University of California at Berkeley

BERKELEY, CA. BS ELECTRICAL ENGINEERING AND COMPUTER SCIENCE *1969*

APPENDIX 2 – Complexity of Software

Software of any size over about 1,000 or lines of code *cannot be proven with certainty* to work as expected. There is a theory, see Leung, Hing, "Program Correctness," https://www.maa.org/sites/default/files/images/upload_library/46/Pengelley_projects/Project-15/correct-project.pdf for a brief introduction). I won't go into the theory of proof, but simply examine how long it would take to examine every path a program could take.

This is Pseudo Code that can be written in any computer language such as C, C++, JavaScript, Python.

We use "Do" to indicate executing some unspecified code with one or more instructions, each "Path" doing something different than the others. We assume "Do" records the name of the path.

BEGINNING OF CODE

IF Condition1 THEN Do Path1 OTHERWISE Do OtherPath1

IF Condition2 THEN Do Path2 OTHERWISE Do OtherPath2

IF Condition3 THEN Do Path3 OTHERWISE Do OtherPath3

IF Condition4 THEN Do Path4 OTHERWISE Do OtherPath4

IF Condition5 THEN Do Path5 OTHERWISE Do OtherPath5

IF Condition6 THEN Do Path6 OTHERWISE Do OtherPath6

IF Condition7 THEN Do Path7 OTHERWISE Do OtherPath7

IF Condition8 THEN Do Path8 OTHERWISE Do OtherPath8

IF Condition9 THEN Do Path9 OTHERWISE Do OtherPath9

IF Condition10 THEN Do Path9 OTHERWISE Do OtherPath10

OUTPUT ListOfPaths

END OF CODE

The list of paths for a particular set of conditions could be:
Path1, OtherPath2, OtherPath3 , Path4, OtherPath5, Path6, OtherPath7, OtherPath8, OtherPath9, Path10.

So, the question how many different paths can this code take? In the first line, there are two possible paths, Path1 and OtherPath1.

When we add the second line it can choose Path 2 or OtherPath2.
The first two lines can yield four possible paths

1) Path1, Path2
2) Path1, OtherPath2
3) OtherPath1, Path2
4) OtherPath1, OtherPath2

As the number of IF-statements goes up in this example, the number of possible paths is 2-to- the-10^{th}-power or in math terms, 2^{10}, which equals 1,024. Certainly, a manageable number to hand audit in a week or so.

But, if we do another 10 IF statements, making it 20, it becomes 1,048,572. So now we might have to write a program that walks through all million plus+ paths. Still doable if we are careful.

Now, if we make it 100 IF statements, the number is 1.26×10^{30} number of possible unique paths. The fastest computer world can execute 1.4×10^{25} computer instructions in a year, so it is destined to take about 32 years to walk through all the combinations. Add another 40 IF statements and we get a duration of the life of the universe. If I have math wrong, simply add another 100 IF-statements — you'll get to the life of the universe.

Even with a program of even 10,000 lines of code, which I have written on several occasions, is likely to have more that 100 non-nested IF-statements. If the voting machine has over 600,000 lines of software, its execution is not "provable by trying out all the paths."

There could be an error that only occurs if Condition1 is true, Condition 2 is false, Condition52 is true, and Condition98 is true. Or some table entry contains an incorrect value that affects the performance of a system, but not enough to be noticed for years, until a test was run to measure it — an event that actually happened in my career.

Getting access to the code to read 600,000 lines of code looks something like this.
What code looks like (C++):

```
#include <iostream>
using namespace std;
// Comment that is not compiled to document code
void myfunction(int array[], int size) {
  for (int step = 0; step < size; ++step) {
      for (int i = 0; i < size - step; ++i) {
          if (array[i] > array[i + 1]) {
            int temp = array[i];
            array[i] = array[i + 1];
             array[i + 1] = temp;
          }
       }
     }
  }
```