CV

1983 - Data Salvation

I salvaged the company's core asset; their mailing list. It resided on a proprietary computer with a proprietary 8" floppy storage. The company behind the computer had been sold and this model was no longer supported. More problematic was it inconsistently powered off and/or froze. The computer's reporting capability was adequate to dump the data as a report to a printer. I, therefore, constructed a cable to effectively fool the obsolete computer into thinking the new IBM PC was a printer. The resulting single file had the necessary information, but was malformatted. I wrote numerous mailing list management programs which properly formatted the data, eliminated redundancy, and imported it into Dbase II.

1984 - Pagination

I wrote a pagination program to automatically paste up classified ads in a magazine, thus saving several man-hours per publication. It accommodated kerning, line spacing, font size/type, categorization, and other factors..

1985 - Matching

Working with several psychiatrists, they created a paper questionnaire from which I created a software application to match people. Though heavily processor intensive, it was very successful, increasing company profits by over 500%. To this day, the algorithm likely stands above the rest. Unfortunately, the success of this program was also its downfall as the company could not sustain the necessary number of single people to adequately pair. The algorithm had a diminishing point of return by sacrificing matching criteria. Eventually, the company succumbed to this regrettable success. Had it been 10 years later, there's no doubt in my mind there'd never be a match.com.

1987 - Fiber Optic Network

Under a sensitive security classification, I evaluated technology and designed a star topology fiber optic network to support heavy bandwidth requirements for the Defense industry. Included in the recommendation was operational network software. The recommendation factored in amortized costs to justify its longevity and ROI.

1990 - Campus Wide Network

I designed, purchased, and implemented an inter building, campus-wide, network for Brown & Root in Los Angeles which included 14 departmental servers co-located in a controlled environment. The system was not only hardware and software, but also policies, procedures and a support infrastructure.

1991 - Distributed Tiered Support Program

I implemented a unique support matrix to tap knowledge within the existing company infrastructure. This reduced support costs by providing a graded level of support. What that means is I identified employees who

were experts in certain software and implemented a departmental bill back mechanism justifying part of their time as high end application support.

1992 - Custom Furniture

I designed a furniture line and had it built. As part of this I created a real-time document by integrating MS Word with AutoCAD. If an AutoCAD drawing changed, the document changed. This streamlined publication, allowing for quicker design to production.

1993 - System Integration

I integrated 3 disparate computing platforms into one homogeneous system for the 'fraud' group of Los Angeles Cellular using multifunction VMs for Windows, Linux, and 3270 Emulation software. This involved testing, design, and procurement of a highly sensitive 21 workstation LAN.

1994 - Remote Access

I engineered a networking solution allowing remote workers to securely log into the host system using FOB authentication.

1995 - IETech

• I established my own company to assist small businesses with their technology needs. On a macro level it included any/all office technology previously only available to larger corporations.
• As part of these pseudo CTO roles, I established company-wide policies and guidelines to reduce employee transition costs by structuring their technology. For example, login IDs were by role and the file structure was by job description instead of individual.
• I researched and subsequently replaced all of my client's back end Microsoft servers with Linux servers, thus reducing life cycle costs nearly 65%. I replaced Active Directory with OpenLDAP, IIS with Apache, file services with Samba, Email with Postfix, Backup with Bacula, and SQL Server with MySql.
• I redesigned my client's networks to improve security, integrity, and expand telecommuting. I identified and isolated key computing functions to minimize the impact of failure. I created a backbone to support maintenance. I developed redundancy for mission critical applications.
• I researched and installed an AJAX enterprise email system which processed over 50,000 emails per week. I then built an email gateway to filter bad traffic by qualifying senders and delaying SMTP traffic. More comprehensive scanning continued to be done for legitimate senders and recipients on the main mail server.
• I evaluated and tested multiple virtualization environments before settling on Xen as the environment to simulate client configurations.
• I designed and installed cable plants allowing flexible workgrouping and location of resource stations (eg printers, fax, copiers, etc)

1996 - LTree

I wrote an in-house full stack application to represent server data in a browser as a tree. This effectively eliminated internal web development costs. Extensive thought was given to the most efficient interface to minimize data transfer yet provide a flexible, secure, and powerful file tree. The system was designed to separate content from format, thus allowing it to be used as a menu system as well. Its purpose was to display

a directory structure in a browser, but with many extended features. For example it had the ability to incorporate 'virtual' directories, resulting in limitless expandability. In the end, maintaining a single file system was all that was required to create a company website.

1997 - Cable plants

I researched, evaluated, tested, and installed multiple cabling plants ranging from small offices to entire campuses. All wiring was professionally terminated RJ45 (T568-B) and integrated with existing RJ11, and standard coax where required. All outlets were professionally finished, labeled, color coded, and documented.

2003 - Legal Data Cataloging

I designed and implemented a cooperative web based information gathering application which allowed multiple people to aggregate evidence for a law firm. I used Backbase as the client side platform and JSON as the delivery format. I used PHP on the back end to process all stored data.

2005 - Slideshows

I wrote and/or integrated several AJAX/PHP slideshow programs. Each program delivers data differently to best suit the calling device (eg large screen display, computer monitor, tablet, or phone). This was prior to current CSS modeling availability. Navigation, search, and pagination buttons were configurable. XML and JSON were supported delivery formats.

2012 - Report Generator

I designed an AJAX/PHP SQL report generator to support unique customer requirements. I wrote a query parser which formed the foundation of subsequent grid movement. I used sessions and AJAX calls to minimize data transfer. I created a key based authentication system.

2009 - Client Feasibility Study

I tested the feasibility of Linux as a client platform. I built test computers running SuSE, Ubuntu, and RedHat. I used VirtualBox to maintain Microsoft awareness and run proprietary Microsoft applications. I used REST to access shared calendaring applications. My objective was to recommend a platform to lower life cycle costs, increase security, improve integrity, and extend the hardware life.

2013 - Enterprise Monitoring System

As part of my support role, I identified the need for a monitoring solution. I was solely responsible for requirements analysis, investigative research, system architecture, system development, system deployment, system support, and sustainment of this enterprise monitoring system to facilitate adherence to SLAs and provide all forms of operational QA data to developers (ie unit, component, end-to-end, performance).

Benefits

- Open source - this entire solution uses open source tools. It, therefore, is modifiable and expandable with the only cost being man-hours. Additionally, there are no licenses, support contracts, or recurring costs whatsoever for the resources used to build this solution. Keep in mind employee time is required

regardless of what monitoring solution is deployed. When considering this *it is difficult to find an alternative with a better ROI.*

- Autonomous - The primary objective was a solution which was autonomous from the product. This, effectively, *eliminates the potential conflicts of an integrated system.*
- Single code base - Due to the rapid deployment mechanisms and design, there is only one code base. It is always the 'production' code and *is always up to date at all customers*.
- Flexible data formatting - The 'results' generated from the agents are all in Nagios format. This allows the agents to port to several backend monitoring solutions. After reviewing several aggregator backends, I opted for Centreon. At that point in time it, simply, had the best ROI. Furthermore, knowing that construct, if needed, it could be integrated with alternative backends.
- Community of help - Nagios is the 3rd largest monitoring solution in the world and the largest 'open-source' monitoring solution. As an open-source product of such scope *the qualified scope of available talent is large*.
- Store and forward model - Much like 'postfix' we opted for a store-and-forward concept in the design of this product. This detached process *allows for 'updates', 'improvements', and 'fixes' to be rapidly deployed*. 'Agents' extract metrics, 'analyzers' assess and package this data, 'channel mechanisms' deliver the data, monitoring 'presents' the results, the email engine enhances the messages and the additional information library enables a more consistent support response.
    - 'Agents' - These are all bash scripts called using a hierarchy of cron scheduling. The 'results' are stored as text files which are later analyzed for further processing.
    - 'Analyzer' - This script assesses all the data created by the agents every 'cycle' (default is 20 minutes). The analyzer evaluates metrics against thresholds, 'rolls up similar metrics' to combine metrics in graphs (thus reducing the number of rrd files generated) and groups based on category to combine similar metrics into fewer emails (this dramatically
    - 'Channel' delivery - The 'analyzed' results are either 'pulled' or 'pushed' (depending on the customer's security model) to a core DMZ site. There are multiple options for delivery of the data which more readily adheres to customer security choices. From there they are automatically security vetted and relayed to an internal destination for 'presentation' to the team.
    - 'Presentation' - The core monitoring component 'presents' the data. This means it graphs, reports, and where applicable submits an alert to the email engine.
    - 'Email engine' - This process augments the message with color coding, adds content specific instruction and validation links and formats for multiple delivery mechanisms (eg email, sms, etc)
    - 'Library' - The links embedded in email alerts redirect the team to 'next' steps. These are called 'click-throughs'. By having these with each alert, the learning curve to onboard personnel is substantially reduced. It further reduces errors in 'assumptions' and/or diagnostic methods as this library of instructions is constantly maintained by the support team with the optimal procedures and/or alternate procedures unique to customers.
- Self governing - This *solution monitors itself* too. There are several maintenance scripts, and monitoring agents designed to make sure monitoring does not impact production. Again, the 'modular' architecture further reduces this risk.
- Unique agents - Though much of monitoring can be replicated in a commercial product, some subsets are *unique and unmatched*.
    - Log monitoring - The global community of log monitors, for the most part, was inadequate. A custom script was created allowing for custom thresholds, custom strings, and rapid deployment of new strings to watch.
    - Sqlclient - This is a custom java sql script for 'read-only' DB queries. Since it covers, MSSql, Oracle, MySql, and Postgres it is transparent to the monitoring scripts using it.

    - ‘Hive’ and ‘Bee’ - The ‘bees’ is an Android application designed to perform round-trip SMS testing to the product. Their results are delivered to the ‘hive’ in the DMZ. Those results are, in turn, retrieved by the core.
    - Stats - Using sqlclient, this script is able to extract monthly billings statistics used by the Monitise adoption team
- Utilities - Over the years several tools have been implemented to facilitate the services monitoring provides
    - mSync - This synchronizes the code to any single, any group (eg ‘cloud’), or all customer VMs.
    - mGetReports - Retrieves any/all reports from any/all customer VMs
    - Report repository - Quick and dirty ajax page to display the local directory of reports.
    - m - A short cut to display VM information
    - mZipAgents / mUnzipAgents - Zips the entire monitoring code base where the ‘customers’ prefer to deploy updates themselves
    - Backbase - Facilitates the ‘sql’ (repository of SQL commands), ‘servers’ (table of customer VMs), and ‘cheat’ (shortcuts to helpful linux commands) tables
    - mArchive - This RRR approved script zips and archives files matching selected filters. This maintains disk integrity by moving and/or removing aging log files.

Drawbacks

- Granularity - This solution extracts metrics in varying time intervals; one minute being the smallest. As such, it is not as precise as on-premise full scale monitoring solutions. This was a licensing, cost, sustainment and accessibility decision. However, it did have SNMP capability bridging this limitation and due to its modularity, system with real-time log analysis could easily be integrated.
- It requires expertise to ‘enhance’ and/or ‘fix’ code. That said most code has been vetted for nearly 5 years and, as such, requires no modifications.

2019 - Splunk Enhancement

Splunk is a wonderful product. But, as a monitoring system, it is merely a repository with a nice front end. It lacks some essential monitoring system capabilities; the most blaring of which is the ability to put the system into maintenance. I wrote a routine using Splunk’s core Python engine to facilitate putting monitoring into maintenance, by either emailing a properly formatted email to the system or using a custom Splunk page to schedule it. I created an API for the script which allowed future expansion if necessary. And, of course, fully documented it. The biggest hurdle was that I was restricted to using the Splunk Python engine which was 10+ years old.

2020 - HTML Email Alert Rendering

The Splunk email alerting engine was static and limited in nature. I identified and augmented the Splunk notification platform with HTML capability. My objective was to make it easier to educate newly hired employees into the workings, of not only Splunk, but of company operations. By rendering it in HTML, I was able to include links to more comprehensive data, educational links, maintenance links, and response action links. All this using Microsoft Outlook’s HTML rendering engine … which was last updated in the 90’s. That was a major obstacle not previously defined.

2020 - AMS

I was part of a team researching, evaluating, and responsible for recommending an enterprise Application Management System.

2021 - Unemployment

As part of a company-wide policy to consolidate employees back into the office, I was released. Coupled with Covid, I opted to take some pre-retirement time-off. But, if you haven't deduced by now, I can't sit still. As such, I designed and built three things myself:

- Detached office - This office has radiant heat, a mini split, a unique truss system, a unique insulation design eliminating thermal bridging, OLED TV, integrated sound system, windows on 3 sides and a glass door on the fourth side, is completely Cat6 wired and has a walkout deck.
- Water purification building - The existing system is severely worn and a threat to malfunction at any moment. I built a bermed building to house all water treatment equipment before sending it 1/3rd of a mile to the house. Having the building available will facilitate an easier transition of moving the filtering to its new location. The system is designed with valve and equipment redundancy allowing minimal water downtime.
- Well housing - What can I say? The acquired housing was 100% dilapidated so I built a completely new one. It is now much better insulated making freezing less likely and is styled to match the above Water Purification Building.

2023 - The future is now

I have visionary ideas; some of which are groundbreaking with unlimited growth. I'm not exaggerating. Find the proper financier and I will grow their investments thousands of times.