ABSTRACT for the 2014 ISA WWAC Symposium

Managing Complexity - Minimizing Risk

Balancing system growth against destabilizing risk

Blair Sooley, M.B.A, P. Eng.[1]

[1]Trihedral Engineering Limited, 1160 Bedford Highway, Bedford, Nova Scotia, Canada, B4A 1C1,

KEYWORDS

Utilities, Water, Wastewater, SCADA, Risk Management

ABSTRACT

As managers of the world's most valuable resource, SCADA systems in the water and wastewater industry present a need to be properly fortified against both internal and external risks. While there is great buzz around cyber-security, precious little attention has been given to properly ensuring the stability of a system against the sheer weight of its increasing complexity. A system that once stood as a standalone PC dedicated to managing a few assets, is now a highly customizable collection of modules, a node on a network, a server within a greater architecture of servers. As vendors, developers, system integrators and consultants, it is our job to ensure the system remains stable and unnecessary risks are mitigated.

HOW SYSTEMS BECOME COMPLEX

Supervisory Control and Data Acquisition (SCADA) Systems are inherently complex. Control logic, communication protocols, user interfaces, software-to-software interfaces and, most recently, cyber security, have significantly impacted the evolution of SCADA. Further, many utilities are tasked with additional monitoring requirements as a result of population growth, which has led to an increase in demand for public services and the need for increasing water distribution, wastewater collections and treatment of both. While immigration leads to increased tax revenues, increasing power costs and geographically expanding infrastructure often exceed the additional revenues, resulting in greater emphasis on monitoring and controlling large power users, such as pumps and other treatment equipment. Further, customers expect a higher level of service than ever before, requiring always-on services and guaranteed uptime.

All this growth can lead to a reactionary quick-fix, resulting in a bolt-on mentality regarding SCADA expansion. Many SCADA systems have been in service for decades, and started with meager beginnings as a single

Type	Typical Components
Control devices	PLCs, RTUs, I/O blocks, smart modems
Networks	Field busses, business networks, routers, modems, radios
Protocols	Polling, poll/report-by-exception, data-loggers
User Interfaces	Servers, workstations, laptops, tablets, phones
SCADA software	Application servers, I/O drivers, alarms notification, historian
Business Interfaces	Reporting, data analysis, CMMS, GIS, LIMS, WIMS

Table 1 Variety of components in large SCADA systems.

computer monitoring a local treatment process over a low-bandwidth network while a disparate system monitored the status of a few remote pumps over a radio network. Commonly, these systems were proprietary, due to specific benefits that were only available via such systems. As new requirements arose and new assets were added, the proprietary systems were expanded, communications networks saturated and polling frequencies and slow bandwidths started to impede the functionality of the system. Further growth meant new components and changing technology, yet with such an existing investment in existing infrastructure, the easiest option was to retain the existing proprietary system and start a new one to handle the additional requirements. Alarm notification systems (i.e. Alarm Dialers), once completely reliant on analog phone lines and pagers, began to support email and text message options. Instead of migrating technologies, often the original technology remained while new ones were added. Users became more mobile, requiring support for mobile client connections. Data analysis and business tools offered new ways to identify efficiencies, and so the control networks and business networks were merged, requiring firewalls, virtual private networks and dual-homed servers. Administration recognized the value that SCADA offers in regards business decision making, and so SCADA became a tool to consolidate data from multiple sources (e.g. multiple plants.) And amongst all this growth, the population continues to grow and the system continues to expand, affecting all of these interconnected components. Table 1 shows the wide variety of components that a typical large SCADA might include.

For those who have adopted a bolt-on methodology, the result is a perfect storm. Leased line communications are now exorbitantly expensive, as much as $600 per line per month, analog lines are disappearing, users of wide bandwidth radios are facing fines, proprietary fieldbus networks are incapable of high-speed bandwidths, and the closet of backup DOS and XP computer to support older SCADA software is now rendered useless due to incompatibility with new security mandates. The system is complicated with no easy path forward.

While the evolutionary process is typical of how complication creeps into a SCADA over time, some systems simply start out complex; many dedicated servers, modular software with many sub-modules, each with separate integration developments and security, terminal services, complex software licensing, 3rd party connector programs, multi-level server redundancy, differing communication protocols, overly complicated security paradigms. These systems take a great deal of resources to design and implement and have high life-time maintenance costs, issues that could have been avoided using more efficient components and with future scalability in mind.

UNDERSTANDING COMPLEXITY, WHERE IT'S BAD, WHERE IT'S GOOD

There are many reasons for the complexity in such systems, though the most common one is that achieving the desired functionality at the moment is often given higher priority that taking a holistic approach, whereby introduction of new functionality is evaluated from the perspective of the effect (and long term cost) on the system as a whole. The holistic approach may identify a way that the intended functionality could be achieved by replacing an existing pinnacle component rather than adding another one, thereby keeping the level of complexity from increasing. Further, the replacement approach allows an opportunity to move forward technologically, which often leads to significant cost savings in the long run.

Software is one area where this type of opportunity often arises, due to the fact that software technology changes faster than any other. For example, for security reasons a utility may want to minimize its exposure to outside interference to ensure system uptime. As a result, it employs a new security software to minimize tampering, then adds a backup software to create copies of the system configuration, then develops a procedure for ensuring new changes are backed up once deployed, then employs a version control software to maintain each incremental backup, and finally sets up an automated process to ensure the version control system is replicated to more than one geographical location. Complexity has increased greatly with three new software components, one new manual procedure and one automated process. Further, it relies on the dedication of humans to follow procedures in order to be successful.

Often the bolt-on mentality is the result of stagnant thinking, also known as the "It's always been done that way" approach or "more is always better". An extension of this thought process in the context of cost is the belief that because something is more expensive, it must be better. This is likely due to our constant exposure to savvy marketing for cars, houses, restaurants and various other consumer goods, and we bring these entrenched biases to bear on business decisions.

Let's revisit our example using a holistic approach. The SCADA software product provides none of the additional, required functionality. However, a newer SCADA software product exists that not only offers similar functionality as the current software, it offers an integrated version control system that automatically records configuration changes to an encrypted repository and creates a redundant backup to a geographically separated server. All we need provide is the additional security perimeter software and we have only increased our complexity by one component.

The overprotective "more is always better" mentality can is common, as seen in SCADA systems that have multiple layers of software redundancy, redundant PLC CPUs, large server racks full of dedicated components, and high-capability field devices providing low-level service functions.

A far less valuable, and thankfully less common, reason for complexity is simply for bragging rights. Extensive drawings, wall mimics and SCADA displays illustrate the details of the SCADA system's interdependencies and layers, largely for show rather than to provide value. As evidence is the utility manager who pulled out his SCADA architecture drawings to brag about how bullet-proof its security perimeter was, only to have its water plant's control algorithms taken over an hour later by a simulator program inadvertently left running in its internal training room.

However, not all complexity is bad, and in some circumstances may be a necessity. For example, redundancy for critical system processes is highly beneficial in the water and wastewater industry. A second layer of redundancy at a geographically-separated server location may even be beneficial if flood, extreme weather or political unrest may occur. But three identical systems sitting side by side, capable of failing over to one another is likely more valuable in an airplane at 30,000ft, where nuclear meltdown is a concern, or on the moon, and even here the cost of excess equipment weight may exceed the likelihood of cascading failures, and therefore be overruled.

One interesting situation is that of planned complexity. An attempt to move forward can sometimes require such a situation, where two disparate systems, the existing and the replacement systems, are to run side by side for a period in order to validate the replacement before decommissioning the existing.

These situations require tight coordination to ensure that only one component at a time takes an active role, such as controlling a process.

One large utility found itself in just such a complex situation. It had invested in a large, inefficient SCADA software and now required a large number of additional real-time algorithms. With the software incapable of easily incorporating these computations, the decision was made to add the computations as a custom bolt-on module, running on a separate server. As further algorithms were added, more custom modules were developed, resulting in a heavily loaded server running a number of custom modules with no future supportability path. Moreover, the system included complete system redundancy at two geographically separated locations with further redundancy at each, resulting in quadruple redundancy and more than 20 servers. This design provided little flexibility and no easy upgrade path for the system as a whole. With looming operating system supportability issues, the existing system faced full-scale replacement of all software components and potentially large interruptions in availability during the process.

RISK ANALYSIS PROCESS: SOLICITATION of CONCERNS

The aforementioned system would have benefitted greatly from a redesign much earlier in the process. In this case, it is reasonable to suggest that a replacement of the software at the time the first algorithm was introduced could have offered a chance to transition to a software with support for such algorithms. Indeed, efficient algorithm processing methodologies would also have eliminated the need for more server hardware. Such opportunities, as well as many less obvious ones, can be identified via the development of a risk analysis profile.

Risk analysis is the process of identifying real concerns with a system and applying proven tools to determine contributing factors and the level of risk if that concern is not addressed. A current example of such a concern would be a utility still using Windows XP, given the recent end of security updates for this operating system.

Concerns are most easily identified in conversation with groups of people involved with the system, otherwise known as stakeholders. In the context of SCADA, these fall into one of three categories:

- Design/Enhancement - Consultants, System Integrators, Internal System Configurations Team
- Operations - Operators, Maintenance, Lab Technicians
- Administration - Management, Information Technology, Finance

Soliciting concerns can be completed either as a series of events, such as focus groups, as one-on-one meetings to provide anonymity, or as a continuous improvement activity, such as a feedback reporting system. For the purposes of this analysis, we will work with a set of common concerns identified by this writer in conversation with many utilities. These are:

- Escalating training requirements
- Lack of proper maintenance
- Server racks filled to capacity
- Stagnant system functionality
- Disorganized software configuration

RISK ANALYSIS PROCESS: CAUSAL ANALYSIS

This section applies methodologies developed by the Canadian Association of Management Consultants to a sample list of concerns. The completed analysis is included in its entirety in Appendix A - Sample Risk Analysis of this paper.

Each concern identified above is first evaluated independently to determine how it is manifested, the contributing causes and what are the potential outcomes if not addressed. This analysis is included in Appendix A, Table 1.

From this analysis is developed a summary of the underlying causes. The total number of concerns associated with each underlying cause is weighted in combination with the significance of potential outcomes to determine a priority level. This analysis is included in Appendix A, Table 2. Causes with a high or medium level priority are considered candidates for risk reduction. Low priority causes can also be considered for risk mitigation but only after higher priority causes have been dealt with.

Based on the analysis, the underlying causes of significant priority identified using this method are:

- Many different components (medium priority)
- Complex components (medium priority)
- Few people with system expertise (high priority)
- Inefficient software (high priority as it directly affects the number of components required)

Each of the higher priority causes is then further analyzed to determine the implications if not addressed and the opportunities if addressed. These opportunities help to support management initiatives for funding efforts. This analysis is included in Appendix A, Table 3.

RISK ANALYSIS PROCESS: OBSERVATIONS and CONCLUSIONS

From the preceding analysis, four high-level initiatives are defined. This list is included in Appendix A and summarized below as follows:

1. ***Standardize*** - Migrate toward common supportable components and industry standards
2. ***Reduce software inefficiencies*** - Migrate to efficient software with greater flexibility and fewer interdependencies
3. ***Reduce total number of hardware components*** - Eliminate components that provide greater risk than benefit.
4. ***Consolidate hardware use*** - Leverage software and hardware simplification

Each of these analysis points may then be examined to identify actionable objectives for implementation, by evaluating the initiative in context of the SCADA system. This summary is provided in Table 2 below.

Standardize	Reduce Software Inefficiencies	Reduce total hardware components	Consolidate hardware use
Suggested Actions			
Replace low-cost, short service life components as soon as possible (e.g. replace older computers running outdated operating systems) Select appropriate common-use technologies. Avoid proprietary technologies if possible (e.g. use Modbus for wide availability, DNP3 for efficiency in pay per use communications networks, ODBC, Web Services for software-to-software interfaces, Ethernet networks) Focus on migration from high lifecycle-cost components (e.g. use high reliability servers, software with large integration network)	Migrate to software without hidden requirements (e.g. Microsoft CALs, terminal services) Use event-driven software to reduce computer resource requirements and increase response times. Select software with integrated components (alarm management, historian, native drivers, scripting environment) to reduce version incompatibilities during upgrades. Eliminate bolt-on programs where possible (e.g. OPC drivers, 3rd party alarm dialers, custom functions such as external algorithms) Use a software with a simple development environment for internal users and flexible tools for expert system integrators Select software with object-oriented toward efficient integration methods to support reusability of graphic and tag templates. Remove functionality that is no longer required.	Remove components associated with excess redundancy (e.g. triple redundant servers, redundancy in systems with high Mean Time Between Failure (MTBF)) Remove components that can be achieved using existing alternative means (e.g. base-station PLCs for telemetry networks where no remote asset inter-dependency is required)	Redistribute loads to lightly loaded assets (e.g. servers) Use new network technology (e.g. higher bandwidth networks, VPNs to separate different network functions on same network rather than separate networks)

Table 2 Actionable objectives to meet initiatives.

No analysis would be complete without considering budgetary constraints, and any identified action should be assessed with respect to cost/benefit before it can be undertaken. However, the analysis herein is a starting point to identify and mitigate risks, rather than an exercise in budgetary planning. As such, no budgetary analysis is undertaken herein.

INTRODUCING CHANGE

Organizations change constantly. It is through change that organizations stay competitive and relevant. In regards water and wastewater SCADA , this statement holds deeper meaning. It is through change that these organizations provide safe, reliable service. Managing risk forces change and provides a clear picture of a system's current status and its level of flexibility to quickly adjust to a rapidly changing operating environment.

Large scale change can be expensive, time consuming and difficult for all involved. However, sometimes it is necessary. For instance, changing one legacy component of a tightly coupled, proprietary system may have a domino effect that requires several major, and significantly more expensive, components to be replaced at the same time. Such monumental undertakings can be overwhelming and have a cooling effect on the entire project, resulting in procrastination that eventually ends in catastrophic failure with no easy method of recovery.

Recently, one very large utility decided to make large scale change to avoid such a situation. For the past decade, it had been using a proprietary hardware and software technology for a system with greater than 200,000 tags. Their infrastructure investment was huge, with many hundreds of field controllers in service which had been migrated from one proprietary model to the next. Support for the system was provided by a large internal team, yet still required very expensive support services from the supplier. The utility was now facing an uncertain future as the supplier had been acquired by a larger organization with other interests and the proprietary nature of the utility's investment had provided few support alternatives and no easy path to open architecture. Further, the software had reached end-of-life status and the only choice from the supplier was to move to a newer proprietary offering. A pinnacle decision was at hand.

A way forward presented itself with the acquisition of a much smaller utility that had been using an open architecture software and field device hardware platform. The open architecture nature of the two product's offered flexibility unavailable through the proprietary platform, so much so that the smaller utility's SCADA software could be scaled continuously to include a careful migration of all of the larger utility's assets and functionality. Moreover, the small utility system's object oriented technology provided for definition of structured tag templates and matching graphic templates to match the proprietary system's existing station configuration. As a result, the migration of each remote station could be completed in minutes, virtually eliminating human error. The new software was

Migration plan - proprietary to open architecture

Short-term

- Migration to open architecture software.
- Introduction of two widely supported RTUs (1 low, 1 high capability)

Medium-term

- Side by side software burn-in period to validate functionality and develop familiarity

Long-term

- Migration from existing RTU network to newly introduced
- Conversion to Ethernet radio
- Migration to industry standard protocol
- Integration with business systems

flexible enough to initially read data directly from the existing system's historian, leaving control functionality with the existing system and allowing the two systems to work side by side during commissioning. Once operators were trained and comfortable with the new system, the proprietary system could be decommissioned and all communications with the proprietary field controllers switched to the open architecture system. This would be accomplished via the use of the new SCADA software's direct protocol driver support and a simple protocol redirection from the historian to the proprietary central terminal unit.

With this new openness would come new opportunity to tie in business systems, such as the utility's new computerized maintenance management system (CMMS.) Additionally, the utility could consider adoption of greater numbers of the smaller utility's field devices, which offer standard communications protocols, are widely supported and include options for efficiency monitoring. Finally, Integration work could be provided either internally or by a large network of experienced local integrators.

DEALING with COGNITIVE BIASES

Rational analysis is not always enough to invoke change. Sometimes, we clearly comprehend the analysis and make decisions that are unreasonable regardless. Such decisions are often the result of bias.

Bias is an inclination toward a specific thing, person or group[1] and cognitive biases are tendencies to think in certain ways. Cognitive biases can lead to systematic deviations from a standard of rationality or good judgment[2]. Such biases exist in some fashion in us all, and if not recognized, can interfere with the business decision making process and sideline a well designed change process. Wikipedia lists ninety-one cognitive biases, among them many we may recognize in ourselves. Table 3 provides a subset of these that are very common excuses provided for not considering technological change.

Common cognitive biases affecting decision making	
Status quo	Tendency to like things to stay the same
Bandwagon effect	Belief in a technology because others do
Sunk-cost	Continued investment based on previous investment
Mere exposure	Tendency to like things merely due to familiarity
Normalcy	Refusal to plan for a disaster that has never happened

Table 3 Cognitive biases in context of decision making regarding technology

CHANGE INTRODUCTION PROCESS

As important as it is to determine the correct changes to apply, selecting the proper method to apply change is critical to its successful acceptance by stakeholders. The stakeholders must be an integrated part of the process and must be able to see that the changes will have intended, beneficial effects. If not, further change will face stiff protest. For this reason, any change initiative that is relatively inexpensive and can get a "quick win", that being something

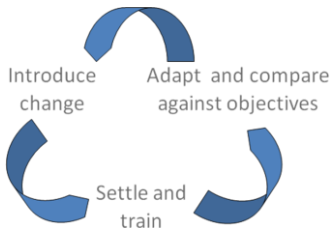
Change Introduction

Before starting any change initiative, ensure you have clear expected outcomes and responsibilities

Step 1 - Introduce change

Step 2 - Settle from change and train stakeholders

Step 3 - Adapt and compare actual against expected outcomes



Return to Step 2 with next change initiative

which is largely beneficial to all (e.g. replacement of a particularly troublesome component), will lend credibility to the process and ease support for more difficult initiatives later.

A proper change process can be simply stated as 1) Introduce, 2) Settle, and 3) Adapt. The introduction step is the project phase, the settling step introduces stakeholders to the change and provides them training to mitigate fear-based biases, while the adaptation step is a period where stakeholders become comfortable with the new environment and have a good understanding of the new component's capabilities and limitations (i.e. the burn-in period.) This is also an important time for stakeholders to provide feedback on the process and evaluate expected versus actual outcomes. No further change is introduced during the adaptation step.

Following a reasonable adaptation period, the next change initiative may then be undertaken, following the same three steps.

PLANNING for OPTIMIZED GROWTH

SCADA systems will continue to grow, regardless whether we plan for it or not. Today's pressures come from regulatory authorities, energy markets and security concerns. Tomorrow's may be significantly different. Since we don't know, we can only stay prepared by keeping our systems healthy and flexible enough to adjust to the changing operational environment.

Holistic approach (affect on system as a whole) for component selection to optimize growth and minimize complexity

- **Can functionality be accomplished within the functional limits of the existing system?**
- **Eliminate bias**
- **Select technologies with common-use interfaces**
- **Select technologies that are easy to configure, use and support.**

With this in mind, planning for optimized growth should use a holistic approach. Consider the effect of introducing new system components on the system as a whole. For instance, if a new telemetry product is to use cellular communications and DNP3 protocol, but the existing system uses neither, is that a good choice? Maybe. Maybe not. If you're currently entrenched with both a proprietary telemetry device and unable to mix protocols on a serial radio system, this may be an excellent choice. Or maybe the asset's location is unreachable without great investment and the DNP3/cellular option is a cheap way forward. However, if you're using an open, supportable system that can easily accommodate additional functionality without departure from established common components, there may be no compelling reason to make this choice. In fact, the added complexity may result in a less supportable system.

A few simple rules can help keep decision making in perspective when introducing change in a SCADA system. First, can the required functionality be added with the existing system? If not, should custom functionality, and the addition of new components, be considered or is the problem systemic and indicative of a stangant component architecture? Second, is this decision being made without bias? Third, new components should be selected for long-term system stability, specifically with respect to common interfaces for ease of future integration. And fouth, select technologies that are easy to configure, use and support.

SUMMARY

Regardless how complexity is introduced into a SCADA system, it is important to fully understand where the complexity provides value, and where it can be eliminated. A complete understanding of envirnonmental operating conditions, combined with aggregate concerns from all systems stakeholders provide a clear picture of the potential for system failure. Risk analysis offers one method for turning concerns into actionable initiatives by identifying and prioritizing core cointributing causes.

Change can only be implemented successfully with the involvement and buy-in of system stakeholders. Success early in the process helps to eliminate biases and create an atmosphere of acceptance for further initiatives. Change should follow a regular process that allows operational users time to familiarize and grow comfortable before further intiatives are undertaken.

Systems will continue to grow and building flexibility through open, simplified architecture greatly enhances the ability to absorb this growth. This is further bolstered in an environment where risks are clearly understood and mitigated through continuous change.

ACRONYMS

CAL - Client Access License

CMMS - Computerized Maintenance Management System

CPU - Central Processing Unit

DNP - Distributed Network Protocol

DOS - Disk Operating System

MTBF - Mean Time Between Failures

ODBC - Open Database Connectivity

OPC - Object Linking and Embedding for Process Control

PC - Personal Computer

PLC - Programmable Logic Controller

RTU - Remote Terminal Unit

SCADA - Supervisory Control and Data Acquisition

VPN - Virtual Private Network

References:

1. Definition of Bias, Oxford Dictionary online, http://www.oxforddictionaries.com/definition/english/bias, accessed May 4, 2014

2. List of Cognitive Biases, Wikipedia, http://en.wikipedia.org/wiki/List_of_cognitive_biases, accessed May 5, 2014.

About the Author:

Blair Sooley is an Account Manager with Trihedral Engineering Inc. He holds a Bachelor in Electrical Engineering from Dalhousie University in Halifax, and an MBA from Saint Mary's University. Blair has been working in the controls industry for 19 years and in the water and wastewater sector for 11 years.

Appendix A - Sample Risk Analysis

Table A.1 Operations Concerns Identified

Identified Problems	How is it manifested?	Why is it happening? Cause(s)?	Why Important? Implications if not dealt with?
Escalating training	More time More courses	Increasing number of different components More complex components Few people with system expertise	Insufficient budget for escalating training requirements Poor response capability due to minimal time dealing with each type of component
System not properly maintained	Hardware replacement upon failure Software upgrade due to incompatibility only Complex maintenance procedures	Too many components to maintain Components expensive Complex components Few people with system expertise	System outages Unavailable parts Cyber security breach
Server racks used to capacity	Many high powered server computers for a single SCADA application Increasing air conditioning costs	Inefficient software Attempt to provide redundancy Complex headless servers	Future system expansion introduces additional computers and additional complexity
Stagnant system functionality	Minimal product upgrades Minimal new features Administrators concerned with upsetting system balance	Few people with system expertise Change affects many different components.	System incompatibilities with new technology Cannot meet new requirements Upgrade costs become capital vs. maintenance
Disorganized software configuration	Difficult to configure Access by expert users only No structured tags or graphics Administrators concerned about changing	Inefficient software Few people with system expertise	Inflexibility to meet new requirements Recovery from failures become increasing difficult

Table A.2 – Vertical Causal Analysis

Description of Underlying Cause (Common causes for multiple problems)	Problem(s) Addressed	Frequency of causal identification	Priority and Importance		
			Low	Med	High
1. Many different components	Escalating training System not properly maintained Stagnant system functionality	3		Y	
2. More complex components.	Escalating training System not properly maintained Server racks used to capacity	3		Y	
3. Few people with system expertise.	Escalating training System not properly maintained Stagnant system functionality Disorganized software configuration	4			Y
4. Components expensive	System not properly maintained	1	Y		
5. Inefficient software	Server racks used to capacity Disorganized software configuration	2			Y

Table A.3 – Implications and Opportunities

1. Many different components	
Implications if not Addressed	Opportunities if Addressed (Why important?)
System becomes unwieldy and is eventually dumped and redesigned from scratch. Large process interruption.	Reduced long-term costs and maintainability. Shorter service interruptions.
2. More complex components.	
Implications if not Addressed	Opportunities if Addressed (Why important?)
System becomes unsupported both from personnel and cost standpoints.	Reduced long-term costs and maintainability. Shorter service interruptions.
3. Few people with system expertise	
Implications if not Addressed	Opportunities if Addressed (Why important?)
Management concerned about making enhancements. Functionality becomes stagnant. Prolonged service interruptions. Potentially long outages during catastrophic failures.	Well-designed long-term growth to meet regulatory and functionality growth. System can evolve to meet technology challenges (i.e. XP EOL)
4. Inefficient software	
Implications if not Addressed	Opportunities if Addressed (Why important?)
System expands to point where all subsystems - networks, computers, functionality, are pushed to limits. System 'rusts' and can no longer be supported.	No limit to system enhancements. Reduced costs for network and computer equipment (possibly reuse existing components.)

Table A.4 – Insights - Observations and Conclusions Regarding the Diagnosis

1. Standardize. Migrate toward common supportable components and industry standards.
2. Reduce software inefficiencies. Migrate to efficient software platforms with greater flexibility and fewer 3rd party dependencies.
3. Reduce total number of hardware components. Eliminate components that provide greater risk than benefit.
4. Consolidate hardware use. Leverage software and hardware simplification