Estimating Software Reliability for Space Launch Vehicles in Probabilistic Risk Assessment (PRA)

Steven D. Novack (1), Mohammad Al Hassan (2), Adam Harden (1), Steven Kossow (1)

(1) Bastion Technologies Incorporated, 17625 El Camino Real #330, TX 77058, USA, Email/Phone Number: steven.d.novack@nasa.gov/(256) 544-2739, adam.j.harden@nasa.gov/(256) 544-5767, Steven.o.Kossow@nasa.gov/(256) 544-1607

(2) NASA Marshall Space Flight Center, Huntsville, AL 35812, USA, Email/Phone Number: mohammad.i.alhassan@nasa.gov/(256) 544-2410

ABSTRACT

It is acutely recognized in the Probabilistic Risk Assessment (PRA) field that software plays a defining role in overall system reliability for all modern systems across a wide variety of industries. Regardless of whether the software is embedded firmware for working components or elements, part of a Human-Machine-Interface, or automated command and control logic, the success of the software to fulfill its function under nominal and off-nominal environments will be a dominant contributor to system reliability. It is also recognized that software reliability prediction and estimation is one of the more challenging and questionable aspects of any PRA or system analyses due to the nature of software and its integration with physics based systems. Irrespective of this dichotomy, any incorporation of software reliability methods requires that the contributions are accountable, quantitative, and tractable.

This paper provides a brief overview of software reliability methods, establishes some minimum requirements that the methods should incorporate for completeness, and provides a logic structure for applying software reliability. Model resolution will be discussed that supports current testing plans and trade studies. We will provide initial recommendations for use in the National Aeronautics and Space Administration (NASA) PRA and present a future dynamic option for software and PRA.

Space Launch Vehicle software is recognized to be reliable in static conditions, yet relatively vulnerable to a set of failure modes in changing environments/flight phases. Two quantitative methods were chosen to incorporate software reliability into a Space Launch Vehicle PRA accounting for phase adjustments. One method predicts latent software failure using statistical methods, and the second provides estimates of coding errors and software operating system failures based on test and historical data. Software uncertainty will also be discussed. It is determined that recommendations for PRA software reliability should be modeled at the software module level where multiple software components compose a module and combinations of the software architecture can lead to a functional failure.

INTRODUCTION

Software risk estimates and predictions are difficult to quantify and incorporate into a formal risk assessment not only because software has inherent characteristics that differ from hardware [1], but also because software can fail in similar modes and generate significantly dissimilar unpredicted effects on a system. Yet software plays a critical part in all space launch systems. This part is only going to increase as cost and schedule pressures mount in a budget-tight space industry. Software is inherent in every space launch electronic element in some manner and often participates in actuation; in particular, it controls the avionics system during ascent and can contribute significantly to system risk during this flight stage.

An apparent and often erroneous reliability approach to estimating software reliability is the treatment of software like a standard hardware component. Software and hardware contain some dissimilar features that pertain to reliability and risk. Software does not wear out and is often very reliable in static conditions. This can be seen with orbiting satellite historical data that exhibit minor dynamic changes after startup and upon reaching designated orbit. Software is not subject directly to environmental stressors such as temperature, pressure, and vibration and there are typically no warnings or indications prior to a software failure. Hardware has physical connections or interfaces such as power lines and communication cables while software connections are virtual and often obscure. Software can potentially repair itself with a reboot and software redundancy and standard parts are truly a different concept than with hardware. Therefore, although modeling software in a PRA can be accomplished in a similar manner to hardware via deductive reasoning, the quantification and application of software failures needs to be accomplished within the constraints of software characteristics.

This paper will provide an example application of modeling and software reliability estimates that may be applied to other industries. This approach was established when avionics software was in the development stage, yet the basic structure and specifications were known well enough to establish preliminary design, documents, and functional requirements. The paper is organized into the following sections: the general approach, quantification of the approach, an estimate of uncertainty, design modification, and testing scheme discussions.

1. GENERAL APPROACH FOR SOFTWARE FAILURE RATES

1.1 Methods

Methods for estimating software failures vary from complex algorithms to qualifying organizational effectiveness. Software failures, unlike their counterparts in hardware are due entirely to human error rather than being influenced by physical or environmental factors. Qualitative methods may be based on project management and identify organizational and management influence on the code outcome and reliability. Another method focuses on the process of code development and relies on a function of upgrade and software improvements to reach a steady state of reliability. Quantitative methods include reviewing fault and failure data and estimating reliability trending from defect density. The better methods for use in PRA are quantitative and based on historical information that is close to the target if available. Some software applications, such as SoftRel use qualitative factors such as testing time, organizational experience, and quality requirements, as well as a wide historical defect density database. This approach is very good for answering questions beyond the specific PRA question of failure rate. For example, it can be used to provide to a safety program an estimate of the time required for testing to establish an acceptable failure rate.

1.2 Historical Data

This approach quantifies failure rates extracted from similar historical data rather than actual data or heuristic reliability predictions for several reasons. First and foremost, this approach is for software under development and not completed. Specifically, where official testing of the software is under initiation and defect density does not have the historical basis to estimate current failure rate data for the space craft software available. Secondly, the use of heuristic reliability predictions was considered to be less accurate than using similar historical data. Lastly, there were ample similar historical data available from space launch vehicles such as the Space Shuttle. Similar is a relative term; however, for example the Space Shuttle avionics system may have some striking similarities to more modern vehicles despite the time differences when it was designed and built from some modern space craft, including a voting computer group, similar software languages, similar main engines and boosters, and many similar functional interfacing avionics boxes. Nevertheless, there were some obvious differences with some modern space craft. The question regarding use of the Space Shuttle historical data was whether the major differences could be adjusted to better reflect space launch craft. Even if this adjustment could not be accomplished, the authors felt that the historical data would better represent space launch craft software failure rates than other prediction approaches. Fortunately, after reviewing the data and calculations used in the Space Shuttle Program, there were adjustments that reflected the space launch craft such that the historical data would provide reasonable failure predictions. Some adjustments may need to be incorporated such as removing the effectiveness of the back-up computer system considered in the Space Shuttle since some space launch systems do not have a back-up computer, using the percentage of flight time in ascent and in space since the failure rate was based on total mission time, and adjusting the number of critical software failure reports (DR-1s) to reflect potential failures during the associated phases.

The historical data is based on the Primary Avionics Software System (PASS) report [3]. The quantitative basis for the failure rate in the PASS report is the Source Line of Code (SLOC). SLOC represents a reasonable, although debatable, measure of code complexity and quality especially when using similar coding rules, languages, and specifications such as the PASS and some space launch craft coding practices. For the PRA, the level of coding expertise and organizational aspects that can affect defect density is considered to be similar to the Space Shuttle. The PASS also takes into account coding and context errors; the latter represents latent errors that occurred during off-nominal flights that made it through testing and on to Space Shuttle flights. Some of these errors (critical and non-critical) were on an operational increment (OI) of the code that flew multiple Space Shuttle missions [3].

1.3 Software Failure Modes

A logical progression of predicting software reliability starts with the anticipated Failure Mission Scenarios (FMS) and progresses through a set of software failure modes that are determined based on what could cause the designated scenarios. Then the portion of the code that could cause each of the failure modes is determined and a historical failure rate for the apportioned code, adjusted for space launch craft, is applied to the failure modes. The numeric SLOC was determined to be the common measurement to relate the Space Shuttle failure rates with predictive space launch craft values. Software contributions to these FMS can account for distinct failure modes and are considered to be mutually exclusive.

Most space launch vehicles' software uses some form of ARCINC 653 specification [1] that provides some resiliency in safety-critical application by partitioning major functions. Each partition acts separately and can fail without harming other partitions. Next, the Flight Software (FS) structure is segregated into functional partitions that are made up of one or more modules. Each partition is a functional construct that is also directly related to a computer process (i.e., each partition has its own process) that is monitored by the operating system. Software failure modes can be caused by a specific failure of either code in a set of partitions or modules depending on their function. Using this approach, a total SLOC can be determined for each failure mode. Fig. 1 represents a representation of different failure modes and their definitions. As can be seen in Fig. 1, some partitions or modules may cause different software failure modes to occur. For example, a management partition includes the time and mode manager modules. Therefore, an early command software failure (error in the management partition) and loss of communication (requires an increment in the time stamp from the time manager module) could potentially be caused by loss of functionality of the management partition. When the SLOC for the same partition is accounted for in multiple

failure modes it should be noted that the total software failure rate needs to be normalized such that the total SLOC of critical code equals the total software failure SLOC to avoid double counting.

Software Failure Mode	Associated SW Module 1	Associated SW Module 2	Associated SW Module 3	Associated SW Module 4	Associated SW Module 5	Associated SW Module 6	Associated SW Module 7
Software conversion / corruption data for control	Data Out	GNC Module					
Last good data on BUS (stale data) causes a fail in place	Input Data Exchange	Output Data Exchange	Application Partition	Data Out	Data In	Infrastructure	Command Partition
Loss of communication of flight computers	Communication Module	Management Partition					
Software data causes a loss of control environment	GNC Module						
Software does not execute a command	Application Partition	Management Partition					
Software command is early or late	Management Partition						

Figure 1. Notional Software Failure Modes and Associated Partitions

1.4 Maturity Ranges

It is recognized that software tends to mature over time and when it is applied under different scenarios errors often get recognized and removed. New functionality or modifications that represent a new release also introduce new errors. There is often an accounting between finding errors and introducing new errors to determine the current defect density of the code at any given time. This can be clearly seen and calculated in the PASS report [2].

The PASS report divides the failures and ultimately the defect densities into three broad ranges that represent the PASS reliability growth curve for the Space Shuttle. These include an immature, mid-mature, and mature failure rate of the software. An immature range would represent first flight, whereas the mid-mature range/defect density may characterize reuse of some systems for flight. In keeping with the Space Shuttle methodology and providing software reliability predictions for different needs of reliability development, three levels of maturity are calculated for the avionics software. In effect, the ranges represent the uncertainty of the total reliability curve over time.

2. UNCERTAINTY BOUNDS

Ironically, the testing data collection was more defined and structured at the end of the Space Shuttle's mission life and represents the best data (least uncertain) collected. The probability of a SLOC leading to a Severity 1 error includes: the total defect probability; the probability the error makes it through three different testing schemes; the probability the error is on the flight; the probability the error is a Severity 1 error; and the probability that the backup computer fails to be effective in the scenario. Errors and operations found at the three different levels of testing are inputs into a beta distribution in the PASS report and portray an increase in uncertainty as the testing levels increase (initial testing to final testing) due to the decrease in total operations and errors respectively as one would anticipate. These parameters and their uncertainty are shown in Fig. 2.

2.1 Software Distributions

The avionics software prediction uses the same approach to define uncertainty about the mean. A representative depiction of the uncertainty for avionics software is seen in Fig. 3 for the three different ranges.

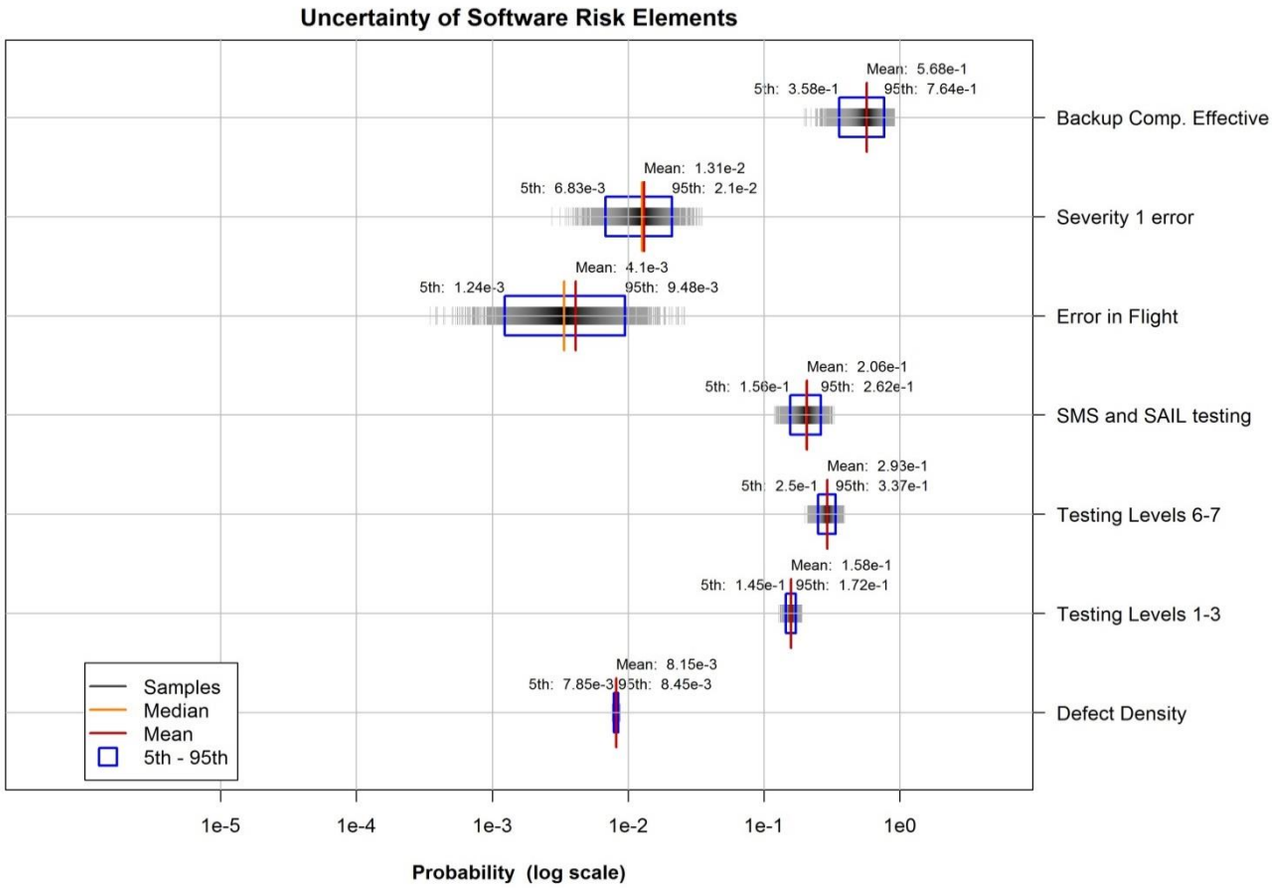


Figure 2. Notional Example of Uncertainty Ranges for Software Risk Elements Determining SLOC Failure Rate

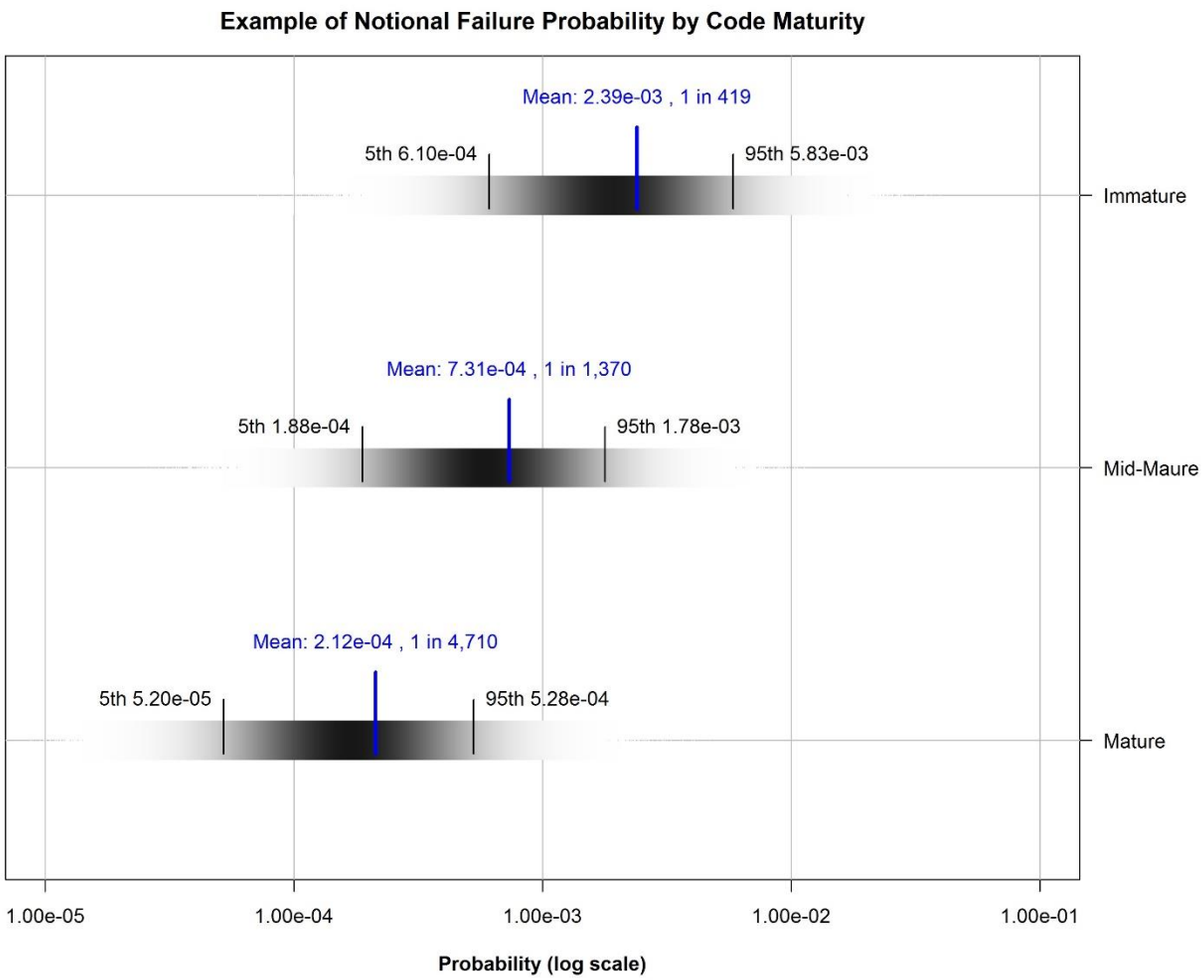


Figure 3. Notional Calculation and Uncertainty for a 100 KSLOC Across Maturity Levels

Actual NASA values are not shown in this paper, although the uncertainty represents an accurate range for the associated maturity levels. The uncertainty charts were developed in R [4] using a 5,000 Monte Carlo sampling routine. The uncertainty, as denoted by an error factor of the mature level, is about 3.2. The uncertainty is on the low side due to the size of the data sets and accounts for additional uncertainty of the adjustment factors (mentioned above) that apply the software failure rates from Space Shuttle to new space launch craft. Uncertainty increased for earlier data represented by Immature and Mid-Mature values because the latter (Mature) data sets in the Space Shuttle were collected more according to the testing schemes. Hence the Immature and Mid-Mature values were derived from the Mature data set.

3. TESTING

When hardware design modifications, such as adding computer redundancy or an independent backup computer, are limited and organizational and team expertise are established, the ability to reduce software unreliability via defect rates relies often on testing. One unique aspect about modern space craft avionics is that all flight computers may act as a voting group and run the exact same versions of the software on each computer. Therefore, an avionics software error will be apparent on all computers defeating the hardware redundancy.

Testing provides a manner to reduce the defect density theoretically to zero using some popular prediction methods [5]. Although theoretical values may be based on an elongated schedule, reality provides limited testing and assumes that some errors make it through even the best testing schemes and practices. The likelihood of this occurring increases with code complexity. Applying a set of metrics to testing and data collection is an initial task. The more challenging effort is in assessing the quality of the tests. Based on the complexity of the space launch craft flight and phases, the number of FMS combinatorial failures are far more numerous than can be tested with-in schedule. PRA can both support the testing schemes and, as test results begin to emerge, use the results to hone software reliability predictions.

With the potential for an inordinate number of potential scenarios for modern space launch craft due to the complexity of the systems and subsystems, the prioritized ranked risk of each avionics FMS scenario is an obvious

starting point for testing. Testing should include these top scenarios for functional testing. Each avionics FMS scenario provides associated logic and identifies specific software failure modes. Therefore, component and integrated testing can also benefit from the PRA logic models.

Course adjustment testing presents a facet in testing that is less practiced. As defects are revealed and tracked, the opportunity to provide feedback to how defects manifest to specific software failure modes becomes possible. Use of a Bayesian updating technique where prior distributions are developed from the Space Shuttle historical data and the posterior distribution includes space launch craft testing results would improve predictions.

4. CONCLUSION

Software reliability predictions are a very difficult and challenging aspect of space flight engineering. Historical data can provide an alternative to demonstrated reliability information for critical systems under development given that the systems and approaches are similar. For the avionics system, the use of Space Shuttle avionics data provides a basis for predicting avionics software and a method for quantifying software specific failure modes. Maturity and uncertainty values provide a probabilistic approach to new, similar, or hybrid systems.

This approach has the potential to be generalized/ tailored for other aerospace or industrial operations. Applications to other industries would need to establish uncertainty bounds

for the specific application applicability. Critical aspects of this approach are the determination of software failure modes and the identification of the relationship between the software architecture and these failure modes, which provides a basis for applying historical data. PRA can use this approach to conduct case studies and to aid in abort trigger assignments and priorities. As testing progresses and defect data is collected, PRA has the potential to provide course correction to the testing team.

REFERENCES

1. Wind River Systems/IEEE, *ARCINC 653 – An Avionics Standard for Safe, Partitioned Systems*, August 2008

2. Pham, Hoang, *System Software Reliability*, Springer Science and Business Media, April 21, 2007, pg 122

3. Russell Robin, Thompson Nelson, Zhu Shangyi, *NASA Primary Avionics Software System (PASS) Probabilistic Risk Assessment*, SSMA-08-011 Rev. B, August 27, 2010.

4. R-Statistical Software Project

5. Neufelder Ann Marie, Software Reliability Toolkit for Predicting and Managing Software Defects, Copyright, SoftRel, 2006.

Biography

Author – Steven Novack

Mr. Novack received his B.S. degree in Physics from the University of Washington and completed course work in an M.S. Environmental Engineering from Idaho State University. He has worked in the area of Quantitative and Probabilistic Systems Analysis and Reliability for 34 years. Prior to his work as the PRA technical Lead for the Space Launch System (SLS) Safety and Mission Assurance (SMA) contract at NASA's MSFC , Mr. Novack worked as an Advisory Scientist and then Program Manager for 20 years at the Idaho National Laboratory (INL). He also performed probabilistic risk assessments (PRA) as a Principal Analyst at Energy Incorporated, and EBASCO Environmental in Seattle. Mr. Novack has led or participated in multiple PRA software programs including the EI PRA workstation, the Ecological Risk Assessment Program (EcoRAP), the INL Risk Monitor, the Virtual Risk Reliability Availability Maintainability Program (VRAM). He has programming experience in Fortran, C, Visual Basic.Net, and Pascal.

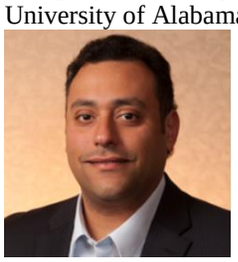
Co-author-Adam Harden

Adam J. Harden currently works for Bastion Technologies, Inc. as a Probabilistic Risk Assessment (PRA) Engineer to support NASA Safety and Mission Assurance for Space Launch System (SLS). His primary responsibilities include PRA analyses for SLS core stage and exploration upper stage avionics as well as the flight termination system.

Previously, Harden was a PRA Engineer for Scientech, a Curtiss-Wright company, in Tukwila, Washington (2013-2017) and for PKMJ Technical Services, located on-site at Davis-Besse Nuclear Power Station, in Oak Harbor, Ohio (2010-2012). Harden graduated with a Bachelor of Science in electrical engineering from Michigan State University in 2008.

Co-author Mohammad Al Hassan

Mohammad Al Hassan currently works for NASA Safety and Mission Assurance (SMA) Directorate as a Reliability engineer under the Space Launch System (SLS) Stages Branch. His primary responsibilities include insight and oversight of the SLS launch vehicle primary contractor, Boeing. Prior to this role, Al Hassan was a Probabilistic Risk Assessment analyst working on SLS core stage, booster avionics and flight termination system.
Previously Al Hassan was an electrical circuit analyst in the Risk-Informed Engineering Department for Southern Nuclear Company in Birmingham, Alabama. Al Hassan obtained a Bachelor of Science in electrical engineering from The University of Alabama at Birmingham.



Co-Author Steven Kossow

Steven Kossow currently works for Bastion Technologies in the NASA Safety and Mission Assurance (SMA) Directorate as a System Safety Engineer. His primary responsibilities include performing and documenting hazard analyses on both systems and software associated with the Space Launch System (SLS) and working with system and design engineering to implement hazard controls into the design to prevent the occurrence of identified potential vehicle mishaps. Prior to this role, Mr. Kossow was a System Safety Engineer for several Department of Defense contractors working on a variety of systems. Mr. Kossow earned his Bachelor of Science in Electronics Engineering Technology from DeVry Institute of Technology in Phoenix, Az and his Master of Science in Systems Engineering from San Jose State University in San Jose, CA.