Improving Software Project Outcomes through Predictive Analytics

By Gina Guillaume-Joseph

B.A. in Computer Science, May 2004, Boston College
M.S. in Information Systems, May 2008, University of Maryland Baltimore County

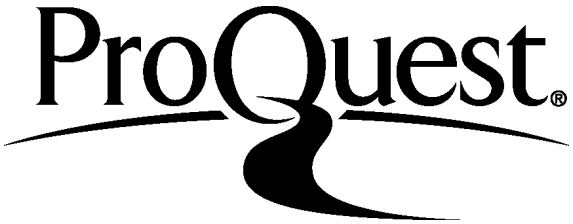
A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

January 31, 2016

Dissertation directed by

James Wasek
Professor of Engineering Management and Systems Engineering



The School of Engineering and Applied Science of The George Washington University certifies that Gina Guillaume-Joseph has passed the Final Examination for the degree of Doctor of Philosophy as of September 25, 2015. This is the final and approval form of the dissertation.

Improving Software Project Outcomes through Predictive Analytics

Gina Guillaume-Joseph

Dissertation Research Committee:

Shahram Sarkani, Professor of Engineering Management and Systems Engineering, Dissertation Co-Director

Thomas A. Mazzuchi, Professor of Engineering Management and Systems Engineering, Dissertation Co-Director

James Wasek, Professor of Engineering Management and Systems Engineering, Committee Member

Paul Blessner, Professor of Engineering Management and Systems Engineering, Committee Member

E. Lile Murphree, Professor Emeritus Engineering Management and Systems Engineering, Committee Chair

Dedication

I dedicate this work to my intelligent, inquisitive and beautiful daughters, Sarah, Hannah and Rebekah. You have taught me that anything is possible; you are my world. Thank you for your continued love, support and kind words during this journey in our lives.

Acknowledgements

To my wonderful family and friends who motivated and supported me in my research. I owe a debt of gratitude to my Grandmother Ternicile Guillaume, you are my hero and inspiration. To my advisors for your consistent support, motivation and candid honesty throughout this journey. Dr. Wasek, your reports relating to my progress in the program always closed with these words "keep up the great work!" Each one prompted and motivated me to meet each successive program milestone. Dr. Campos, your predictive analytics courses prepared me to embark on this research. Your patience with me in class as I often struggled to grasp and apply the concepts provided just the push I needed to learn the fundamentals to allow me to base this research upon those principles. Dr. Fomin, your advice in navigating the research and staying the course proved to be sound. To my Temple Baptist Church Family who prayed for my wisdom and fortitude to complete my research. To all my MITRE Colleagues and including Nicole Misek, Dr. David Kuperman, and Fred Robinson who provided either tutoring support, inspiration or editorial feedback in both the journal articles related to my research and this dissertation. To my INCOSE Colleagues who provided guidance and moral support along this journey. To my George Washington University Colleagues who became dear friends and confidantes through our PhD journey together. And lastly, to dear friends and PhD Colleagues Ralph Labarge and Razi Dianat, your legacy lives on even after your passing. I miss your kind and gentle smiles.

Abstract of Dissertation

Improving Software Project Outcomes through Predictive Analytics

The complex and emergent behavior of software systems makes information and data key components of this unpredictable environment. The use of a data-driven approach to identify and to accurately predict the sources of software project delays, cost overruns, failures, or successes may prove a significant contribution to the fields of systems engineering, software development and project management. Software project failures are pervasive and despite the research, failures still persist. The research presented highlights the systems mindset in addressing failure to introduce a software-specific predictive analytics model that accurately predicts software project outcomes of failure or success and identifies opportunities for incorporation in the federal and commercial space. The research describes how to systematically learn from historical software project failures to train a confidence level model to make project failure predictions. The research demonstrates how to train a generalized model with a range of failure factors combined to efficiently and accurately predict software project failures. The results of the model would be used during acquisition, prior to project initiation, and throughout the software development lifecycle. It is a decision analysis tool to assist decision makers in making the crucial decisions early in the lifecycle to cancel a project predicted to failure or to identify and implement mitigation strategies to improve project outcome. The use of an evidence-based approach to identify software project failure factors will result in better understanding of these phenomena that will ultimately improve software project success rates and minimize risks in systems engineering efforts.

Table of Contents

List of Figures

List of Tables

Chapter 1.0 Introduction

Software Engineering, Systems Engineering and Project Management are intertwined and interrelated disciplines (Selby & Boehm, 2007) (ISO/IEC/IEEE, 12207, 2008) (SWEBOK, 2004). When properly executed, they together can be instrumental in successfully developing software-based solutions that meet the business and technical needs of the project stakeholders with the goal of delivering quality software products and systems (INCOSE, 2010). However, the research shows that software projects often fall short of quality and successful implementation goals. According to a 2011 Government Accountability Office (GAO) report on software project failures, planned federal information technology (IT) projects with investment costs totaling approximately $81 billion frequently incurred cost overruns, schedule slippages and contributed little to mission-related outcomes (GAO-12-7, 2011).

1.1 Problem Summary

When improperly executed, the systems engineering process can result in complex, incomplete, and inconsistent systems where variables of design, reliability, quality, user satisfaction, budgets, and schedules are often poorly considered and estimated throughout the Systems Engineering Lifecycle (SELC) (Eisner, 2005). Project failure often stems from the fact that software systems are becoming increasingly complex as Systems Engineers strive to integrate multiple, disparate subsystems through the use of software (Ryan, Sarkani, & Mazzuchi, 2014) (Madni & Sievers, 2014) (ISO/IEC/IEEE, 12207, 2008).

Software project failure was a concern in 1968 when the North Atlantic Treaty Organization (NATO) held its first conference on software engineering to address the problem. The term "software engineering" was coined to point out the need to instill engineering rigor in the development of software. Experts from 11 countries convened at the conference to discuss many of the same failure factors that systems engineers are still facing today: changing or unclear requirements, insufficient technical knowledge, problematic technology, project cost overruns, and project schedule delays (Stevens, 2011) (Ewusi-Mensah, 2003) (Eisner, 2011). The conference was organized to shed light on these very issues in software engineering and to discuss possible techniques to solve them in order to improve software project outcomes; the conference generated a report to define how software should be developed. An attendee at the conference recommended that software projects develop a feedback loop to monitor the system, through which performance data could be collected for use in future improvements (Naur & Randell, 1969).

The research presented here sets out to understand the underlying causes of software project failure in order to develop a software specific predictive analytics model to accurately predict future outcomes for software projects using data from past software project performance. The predictive model will point to some of the root causes of past software project failures to assist in identifying strategies that can steer projects to successful outcomes in the future.

1.2 Significance

Software is very difficult to quantify because it is intangible and has no physical form. Software, and investments in software, cannot be defined as an economic factor of labor,

land, or capital. The literature places it in a separate domain of knowledge because it is the manifestation of human know-how into bits and bytes of computer code. When buying the software product, the consumer does not acquire the software per se, but the rights of use. Software's functionality is perceptible only in its end or near-end state during user interaction or by way of automated controlled transactions. Despite the ambiguous, uncertain, and intangible nature of software, investment in software represents one of the largest proportions of information technology spending. Forrester Research, Inc. (2013) forecasted global information technology (IT) spending of a staggering $2.06 trillion across IT services by enterprises and government for hardware and software. Figure 1 identifies software spending as the largest investment category, with $542 billion in spending in 2013. The biggest changes and greatest innovations in technology are software-intensive, and include cloud-based Software-as-a-Service (SaaS) implementations, mobile applications, and smart computing such as Analytics and Business Intelligence (Lunden, 2013).

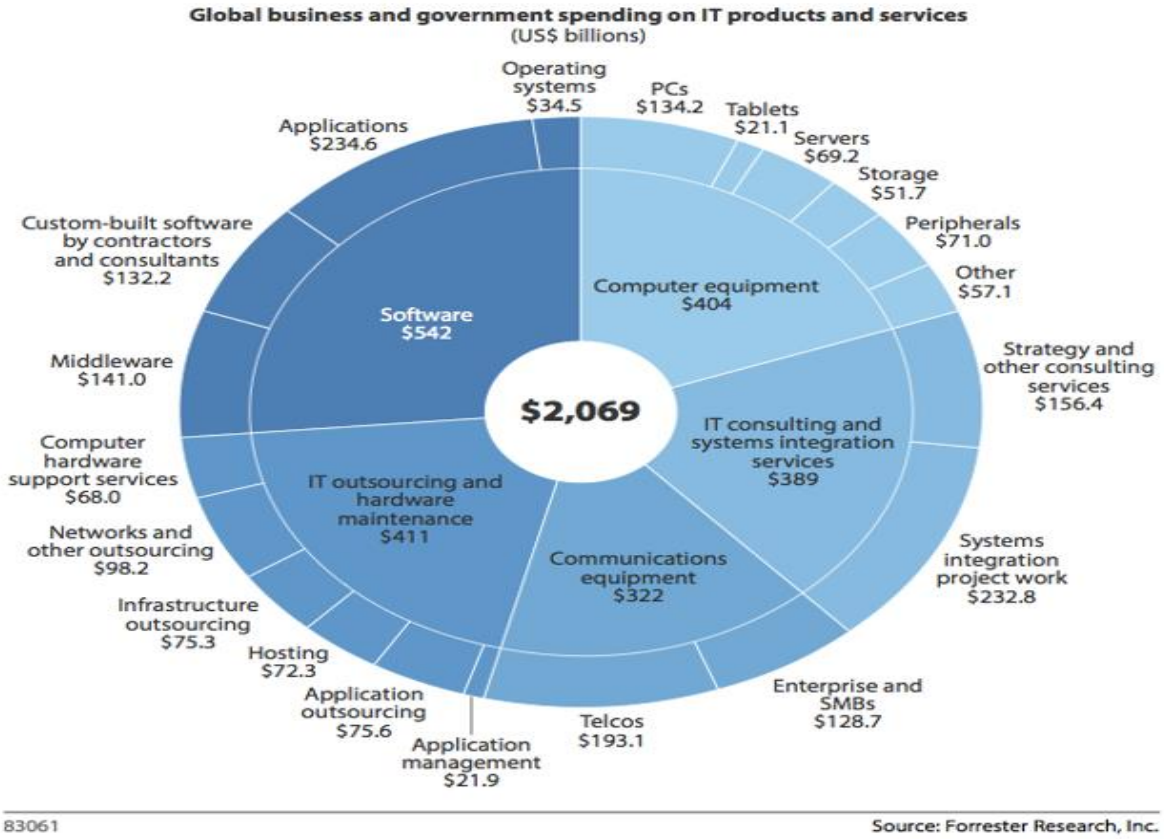


Figure 1: Purchases of IT Products and Services in 2013

(Forrester Research, Inc. Global Tech Industry Wheel)

The subject of this research as well as Gartner's top technology priority list of IT spending in 2013 is noted as Analytics and Business Intelligence. The Gartner "survey highlighted the need for CIOs to set aside old rules and adopt new tools" (Pettey & van der Muelen, 2013). Analytics and business intelligence were named as top priority because CIOs understood the importance of not repeating the past by leveraging these IT innovations to harvest data and information to extend the past into the future. A McKinsey & Company survey observed that IT leaders all agree that the need for better data and analytics is a challenge that has grown in importance in recent years as support of managerial decision making has become top priority (Arandjelovic, Bulin, & Khan, 2015).

Barry Boehm (1991) observed that no tools were available for assessing the probability of the risk factors associated with project outcomes. Eisner (2005) observed that complex systems present technical and management challenges that "require stronger and more insightful thinking patterns to be successful" (Ryan, Sarkani, & Mazzuchi, 2014). Additionally, a panel of experts on Systems Engineering Science at the International Council on Systems Engineering (INCOSE) International Symposium (IS) held in Henderson, Nevada in 2014 discussed leveraging systems thinking to improve project outcomes. A member of the panel indicated that the "systems engineering toolbox needs a mechanism that predicts software project outcomes. Understanding why natural systems fail should be in the toolbox for systems engineers. Unfortunately it is not … it is a research problem that should have effort put into it" (Singer, 2014).

In light of the large financial investment in software and the staggering rates of software project failure, this work contributes to the field of software development because the model is software-specific and was developed using software-intensive project data. The data contained a combination of failed and successful software projects to ensure that its predictive capabilities were substantiated by both sets of outcomes. Further, the authors identified specific software development legislation, processes, and government directives in which the incorporation of the software predictive model in the development of software might be mandated in order to enable sound decision making and to improve overall project outcome. The predictive model developed in this research allows organizations to quantify a software product's probability of success or failure before, during, and after its development. The model predicts whether or not the software functionality will be successfully implemented prior to getting to the end state and before large financial

investments have been made. It helps decision makers in making early and informed decisions about which projects in the portfolio to proceed with and which to cancel. The predictive model is a tool for planning, decision making, and strategic thinking.

1.3 Scope of Research

This research set out to understand the underlying causes of software project failure in order to develop a predictive analytics model to accurately predict future outcomes for software projects using data from past software project performance. The predictive model will point to some of the root causes of past software project failures to assist decision makers in developing strategies that steer successful outcomes in the future.

1.4 Organization of Research

The researcher first defined and classified software project failure, building on discussions in other studies found in their research. Then, as shown in the Literature Review, research on predictive analytics, software project failure, and logistic regression was conducted to identify other tools developed to predict project failure. The Methodology section describes the development of the Predictive Model and how software project failure data were captured and organized. In the Findings section, the results of the research and the potential uses of the predictive model are discussed. The Conclusion provides the recommendations for future exploration and expansion of this work.

2.0 Literature Review

2.1 Software Project Success

In order to fully understand the phenomenon of software project failure, the author identified both successes and failures in the literature, analyzed them, and incorporated them into the predictive model development. Among the successful software project investments the author identified in her research were seven project investments identified by several federal department officials and found in Table 1 (GAO-12-7, 2011).

Table 1: GAO Identification of Seven Successful Software Projects

Table 1: IT Investments Identified as Successful by Federal Departments

Dollars in millions

Department	Managing agency	Investment	Total estimated life-cycle costs
Commerce	Census Bureau	Decennial Response Integration System	$1,050.0
Defense	Defense Information Systems Agency	Global Combat Support System–Joint Increment 7	$249.9
Energy	National Nuclear Security Administration	Manufacturing Operations Management Project	$41.3
Homeland Security	U.S. Customs and Border Protection	Western Hemisphere Travel Initiative	$2,000.0
Transportation	Federal Aviation Administration	Integrated Terminal Weather System	$472.5
Treasury	Internal Revenue Service	Customer Account Data Engine 2	$1,300.0 (Transition States 1 and 2)
Veterans Affairs	Veterans Health Administration	Occupational Health Record-keeping System	$34.4

According to the GAO report, stakeholders from the seven project investments indicated active program stakeholder engagement as a critical factor contributing to

success. Officials stated that program stakeholders attended program management office meetings on a regular basis, were working and active members of integrated project teams and were consistently notified of problems and concerns as soon as they arose. Active engagement with stakeholders created an environment of transparency and trust. Table 2 lists nine common factors critical to the success of those seven software programs (GAO-12-7, 2011).

Table 2: Commonly Identified Critical Success Factors across Seven Software Projects

		Investments						
	Critical success factors	**DRIS**	**GCSS-J**	**MOMentum**	**WHTI**	**ITWS**	**CADE 2**	**OHRS**
1	Program officials were actively engaged with stakeholders.	X	X	X	X	X	X	X
2	Program staff had the necessary knowledge and skills.	X		X	X	X	X	X
3	Senior department and agency executives supported the programs.	X	X		X	X	X	X
4	End users and stakeholders were involved in the development of requirements.	X	X	X		X		X
5	End users participated in testing of system functionality prior to formal end user acceptance testing.		X	X	X	X		X
6	Government and contractor staff were consistent and stable.	X	X		X	X		
7	Program staff prioritized requirements.		X	X		X		X
8	Program officials maintained regular communication with the prime contractor.	X		X	X			X
9	Programs received sufficient funding.	X			X		X	

Homeland Security Western Hemisphere Travel Initiative (WHTI), identified as one of those successful programs, exhibited seven of the nine success factors. WHTI was developed to facilitate inspections at the air, sea, and land ports of entry in the United States, Mexico, Canada, and Bermuda. The program implemented Radio Frequency Identification, License Plate Reader, and Vehicle Primary Client technologies to provide Customs and Border Protection (CBP) officers with border crossing history information for each traveler and his or her vehicle. The program was fully operational in June 2009 at 39 ports of entry with 354 traffic lanes supporting 95 percent of land traffic. The Critical Success Factors and their descriptions are listed in Table 3.

Table 3: Western Hemisphere Travel Initiative Critical Success Factors

(Source: GAO-12-7, 2011)

Critical Success Factors	Descriptions
Leadership exhibited urgency and commitment This supports the commonly identified critical success factor: Senior department and agency executives supported the programs.	According to U.S. Customs and Border Protection (CBP) officials, senior leadership committed to implementing WHTI at land and sea ports of entry by June 1, 2009. The WHTI program manager stated that this deadline resulted in greater involvement of senior Department of Homeland Security (DHS) and CBP leadership. For example, the program manager told us that a former Deputy Secretary reached out to another agency when that agency's efforts to collaborate on an issue were not meeting the schedule requirements of the WHTI program. That official told us that after receiving the necessary support from the other department, CBP was able to more rapidly query that department's data.
Congressional support through funding--This supports the commonly identified critical success factor: Programs received sufficient funding	The WHTI program manager stated that the WHTI program received the requested funding from Congress for the 2 years leading up to the June 1, 2009, implementation date. Additionally, that official told us that Congress provided 2-yemoney, that is, money that could be obligated over a period of 2 years. Officials use that the 2-year money gave the program great flexibility to accommodate the inherent complexities and expenditures incurred in a multiyear deployment, and to adapt to inevitable modifications in deployment requirements (that is, additional sites, lanes, and functionality).
Program office control of WHTI budget	CBP officials explained that the WHTI program budget was controlled by the WHTI program manager. Those officials stated that the WHTI program manager agreed on spending limits with the CBP offices that supported WHTI (e.g., facilities and technology) and monitored the expenditures. In contrast, CBP officials explained that funds are traditionally allocated to the CBP offices that support programs by the CBP Office of Administration. This arrangement reduces business sponsor oversight and control.
Program manager leadership	CBP officials explained that the WHTI program office was led by an experienced program manager. Those officials explained that the WHTI program manager created the necessary environment for the team to succeed. One official added that the WHTI program manager's leadership inspired the WHTI team.

Critical Success Factors	Descriptions
Program office knowledge --This supports the commonly identified critical success factor: Program staff had the necessary knowledge and skills.	CBP officials explained that the WHTI program was supported by experienced staff members. CBP officials stated that almost every member of the WHTI team had a good understanding of acquisitions of (demonstrated by some staff holding acquisition certifications) and program management. Further, those officials told us that the team always had two members who were knowledgeable on a particular issue—one team member was responsible for the issue and the other was a backup in the event that the primary member was not available. These skills contributed to effective program oversight of the WHTI contractors through all phases of the acquisition, not just during contract award. Moreover, one official attributed the unity of the team and the commitment to work collaboratively to the respect that each team member had for others.
Program office staff familiarity and stability---This supports the commonly identified critical success factor: Government and contractor staff were consistent and stable.	CBP officials stated that many team members worked together on previous projects. As a result, those officials said that these team members already knew each other's role, skills, and work style, and this familiarity enabled the program office to quickly perform at a high level. Those officials added that key staff member--such as the WHTI technical leader--remained consistent throughout the WHTI program. The low turnover of WHTI program staff helped to maintain that high performance. Moreover, according to a CBP official, the staff genuinely liked to work with one another and were able to collaborate effectively.
Stakeholder involvement--This supports the commonly identified critical success factor: Program officials were actively engaged with stakeholders	CBP officials told us that the WHTI integrated project team was formed before the completions of planning efforts and well before the initiation of development efforts. According to CBP officials, the WHTI integrated project team was composed of numerous stakeholders such as legal support and representatives from budget/finance. CBP officials added that the team was formed prior to acquisition and development efforts, and weekly and later biweekly meetings were held with high participation rates. Those officials stated that the integrated project team was a decision-making body—not just a mechanism for the WHTI program office to communicate with stakeholders.
Consistent message when communicating about the program	CBP officials told us that everyone in DHS and CBP—including the DHS Secretary and CBP Commissioner—adhered to WHTI's consistent

Critical Success Factors	Descriptions
	message and terminology when communicating with Congress, the media, and the American public. This consistent message was used to describe, for example, the goals of the program, deployment plans, privacy implications of the Radio Frequency Identification (RFID) infrastructure, and impact of the program on select groups crossing the border, including U.S. and Canadian children and Native Americans.
Daily coordination with the prime contractor during deployment--This supports the commonly identified critical success factor: Program officials maintained regular communication with the prime contractor.	CBP officials explained that key WHTI officials participated in a 9:00 a.m. daily teleconference with the contractor while WHTI was being deployed to ensure proper coordination and the rapid resolution of problems. CBP officials explained that this daily coordination was necessary given that deployment had a significant impact on port-of-entry operations; namely, each lane was taken offline for 1 to 2 days while the infrastructure was deployed. For example, an official told us that the electric system which provided power to the lanes at a port of entry was not adequate. This official said that the issue was identified and raised during the daily morning conference, someone was assigned to begin working on the problem during that meeting, and the issue was resolved.
Prioritization of planning	CBP officials explained that their initial instinct given the aggressive implementation timeline was to focus on technical solutions, developmental efforts, and deployment. However, those officials stated that the WHTI program began with, and completed, key planning efforts which eventually secured the success of the program. Examples of these planning efforts include policy changes, regulatory requirements, and process reengineering changes.
Well-planned acquisition approach; active contract management	CBP officials explained that the program obtained extensive input from potential contractors on WHTI requirements as a result of those potential contractors' review of the draft statement of work for the WHTI design, procurement, testing, and deployment of the RFID/ License Plate Reader (LPR) infrastructure. Those officials stated that questions from the potential contractors improved the quality of the request for proposals and the resulting contract. Additionally, CBP officials explained that they utilized a fixed-price structure for the above-mentioned country. Those officials said that this structure reduced the government's risk of realizing cost overruns.

Critical Success Factors	Descriptions
	Further, CBP officials stated that the contracting officer for that contract was collocated with program office officials. As a result, CBP officials explained that the contracting officer was fully aware of operational issues and requirements, provided needed guidance, and expedited contract modifications.
Testing prior to deployment **This supports the commonly identified critical success factor: Users participated in testing of system functionality prior to formal user acceptance.**	CBP officials explained that the program's testing prior to deployment was critical to the success of the WHTI program. In particular, those officials stated that the LPR and RFID design and performance were tested at a mock port-of-entry test facility constructed at an old private airport in Virginia. CBP officials said that the lanes with RFID and LPR infrastructure were used to optimize the system so that i(1) would be able to detect multiple RFID cards in one vehicle within that lane, and (2) would not be overly sensitive as to detect RFID cards from other lanes. Additionally, those officials explained that numerous vehicle speeds, models (e.g sedans, sports cars, SUVs, etc.), and associated camera technologies. Further, according to CBP officials, tests were done in all weather and lighting conditions to ensure the cameras could capture acceptable images under all circumstances. Moreover, those officials told us that a group of core end users was brought to this facility to test the forth coming technology. As a result, CBP officials explained that when many of these end users returned to their ports of entry, they became advocates for the WHTI technology.
Funding for public outreach	CBP officials stated that they believe that Congress's recognition of the significant social and cultural changes required of U.S. and Canadian citizens to successfully implement WHTI led Congress to appropriate funding for an effective communications and outreach campaign to increase awareness about new requirements for travel documents. CBP officials stated that this campaign, which relied on professional advertising media (e.g., TV, print, radio, and billboard advertising) provided by a private public relations firm, was something that normally would not be funded for a federal program, but was critical in obtaining buy-in from the local border communities and the traveling public, thus ensuring the success of the program. CBP officials explained that WHTI deployed millions of dollars in technology; however, if travelers did not obtain RFID-enabled travel documents, technology would be underutilized. According to the WHTI program

Critical Success Factors	Descriptions
	manager, because of these outreach efforts, WHTI had a compliance rate of 90 percent on the first day that WHTI documents were required to be presented at the land border.
Just-in-time operational and technical training	CBP officials told us that the end users were trained just prior to, during, and immediately after, deployment. Those officials noted that even after the lanes were accepted by CBP officials at the ports of entry, WHTI program officials stayed withe end users for 5 to 7 days to ensure that the end users were fully prepared to use the system. Those officials told us that by the time WHTI was fully implemented, over 10,000 officers had been trained in new operating procedures, application use, and familiarization with the new lane equipment and travel documents.

Implementation of these success factors may not always ensure successful project deployment because, as the research shows, many different factors contribute to success. However, these examples may help organizations address the many issues and challenges they face. Within these same agencies, several failed projects were also identified.

2.1 Defining Failure

Researching and analyzing the outcomes of software projects, as well as data from the failed software projects and related Government Accountability Office (GAO) reports, enabled the identification of one or more of the following attributes or characteristics of software project failure (Flowers, 1997) (GAO, 2006) (GAO, 2009) (GAO-12-7, 2011) (Tan, 2011) (The Standish Group, 2013) (GA0-14-705T, 2014) (DoD, 2015):

- Catastrophic or deadly outcomes
- Completely abandoned projects or projects never released into production
- Project cancellation

- User dissatisfaction leading to abandoning the system post-deployment
- Quality goals not achieved
- Significant cost overruns that exceed 30 percent or more of the estimated budget
- Significant schedule slippage that exceed 30 percent or more of the scheduled duration

Table 4 outlines the statistics of several industry studies of failed software projects. Three of the four surveys uncovered that over one half of software projects fail. The fourth survey showed a failure rate of nearly half. With an average of 55.25 percent of failed software projects, it is apparent that software projects fail more often than they succeed.

Table 4: Statistics of Software Project Outcomes Based on Industry Studies

Surveyor	% Failed Projects	% Successful Projects
The Standish Group, 2013	61%	39%
Robbins-Gioia, 2001	51%	49%
The Conference Board Survey, 2001	48%	52%
KPMG, 1997	61%	39%
Average	55.25% Failure	44.75% Success

This research identified key failure factors that directly correlate to project failure. Table 5 identifies the failure factors and demonstrates that their effect on project failure is multiplicative and spans the four major phases of the SELC. This indicates that a variety of factors working in concert can create the perfect storm to contribute to software project failure (Tan, 2011) (Ewusi-Mensah, 2003). Table 5, adapted from Ewusi-Mensah, 2003, also lists the criticality impact scores for each factor to demonstrate the impact of each

factor's effect across the four stages of the lifecycle. The scores are denoted as Low Critical = 1, Medium Critical = 2, and High Critical =3.

Table 5: Software Project Failure Factors and SELC Impact

	Software Engineering Lifecycle Stages			
Failure Factors	**Requirements**	**Design**	**Implementation**	**Testing**
Unrealistic Project Goals and Expectations	3	3	3	3
Changing or Unclear Requirements	3	3	3	3
Insufficient Technical knowledge	1	3	3	3
Problematic Technology	1	3	3	3
Lack of Executive Leadership Support	3	2	2	2
Insufficient User Commitment	3	2	1	2
Project Cost Overruns	1	2	3	3
Project Schedule Delays	1	2	3	3
Insufficient Project Management and Control	1	3	3	3

2.2 Factors Present in Software Project Failure

The federal government alone, in the past ten years, has spent over $600 billion on software-intensive IT projects (GAO-13-524, 2013). However, "far too often IT projects, especially large projects, cost hundreds of millions of dollars more than they should, take years longer than necessary to deploy, and deliver technologies that are obsolete by the time they are completed" (CIO.gov, 2015). Table 6 below lists the failure factors identified in this research. The research has determined that schedule and budget metrics are extremely helpful in determining project trouble. Unfortunately, they are lagging indicators (Horne, 2013). A set of qualitative leading indicators that help sense trouble early in the SELC is crucial to identifying troubled projects that would otherwise look healthy. The absence of routine problems is also a huge red flag, because if it seems too good to be true it probably is. Often cost and schedule delays are not reported until the project is in critical

danger (GAO-06-647, 2006) (GAO-13-524, 2013). Therefore, in addition to cost and schedule factors of failure, the research has identified additional failure factors present in the reviews of failed software projects.

The following sections will define and outline these in more depth and discuss their impact on software project failure and success. In this research the author identified a set of 202 software projects, performed analysis to investigate failure factors for the set, and identified the failure factors that were manifest in projects that failed. The research determined that software project failure factors spanned the project lifecycle and were interrelated one with the other. Software project failure generally resulted from many failure factors working together. (Lehtinen, Mantyla, Vanhanen, Itkonen, & Lassenius, 2014).

Why are projects continuing to fail despite the vast amounts of research on the topic? Managers continue to fund failing software project efforts because they lack the empirical data and evidence to prevent them from failing. Table 6 compares the failure factors found throughout the literature research and describes the core set found in the 202 software projects in further detail.

Table 6: Failure Factor Comparisons Found in the Literature

Boehm 1991	Barki, Rivard and Talbot 1993, 2001	Ewusi-Mensah 2003	Standish Report 2014	Government Accountability Office (GAO) 2014
Personnel Shortfall	Newness of the technology	Unrealistic project goals and objectives	Incomplete Requirements	Lack of program official engagement with stakeholders
Unrealistic schedules and budgets	Application size	Inappropriate project-team composition	Lack of User Involvement	Program staff did not have necessary knowledge and skills

Boehm 1991	Barki, Rivard and Talbot 1993, 2001	Ewusi-Mensah 2003	Standish Report 2014	Government Accountability Office (GAO) 2014
Developing the wrong functions and properties	Lack of expertise	Project management and control problems	Lack of Resources	Lack of senior department and agency executive support
Developing the wrong functions and properties	Application complexity	Inadequate technical know-how	Unrealistic Expectations	Lack of end user and stakeholder involvement in requirements development
Gold plating	Organizational environment	Changing requirements	Lack of Executive Support	Lack of end user participation in testing of the system prior to end user acceptance
Continuing stream of requirements changes		Problematic technology base/infrastructure	Changing Requirements & Specifications	Government and contractor staff were inconsistent and unstable
Shortfalls in externally furnished components		Lack of executive support and commitment	Lack of Planning	lack of prioritized requirements
Shortfalls in externally performed tasks		Insufficient user commitment and involvement	Didn't Need It Any Longer	Program official lack of communication with prime contractor
Real-time performance shortfalls		Cost overruns and schedule delays	Lack of IT Management	Program did not receive sufficient funding

2.3.1 Unrealistic Project Goals and Expectations

The major goal and objective of systems engineering, software engineering, and project management is to deliver high-quality software on time and within budget. However, unrealistic project goals and expectations contribute to software project failures both in the literature and in the 202 projects studied to develop the predictive model. This reflects the inability of the project team to specify project goals, expectations, and outcomes or from

lack of agreement and consensus on the project goals, leading to confusion about what the project is expected to achieve. This factor is one of the most significant, as Table 5 demonstrates with high critical impact scores across all phases of the lifecycle.

2.3.2 Changing or Unclear Requirements

Delivery of high-quality software on time and within budget begins with excellent requirements. Changing and unclear requirements posed several challenges leading to failed software projects. Poor requirements management was among the major challenges encountered in this research of failed software projects. Vague and ambiguous requirements were identified as a cause of failure because developers often had to guess the true intent of the requirement. The data from the failed projects studied indicated that scope and feature creep plagued projects because a process to manage the requirements was lacking. The occurrence of scope creep indicates that requirements were not identified during elicitation or that key user stakeholder groups were overlooked or not identified. As with Unrealistic Goals and Expectations, this factor was also very significant, as Table 5 demonstrates with high critical impact scores across all phases of the lifecycle.

2.3.3 Insufficient Technical Knowledge

Insufficient technical knowledge is a leading cause of software project failure. This can manifest as a lack of technical experience on the part of the project team. Inexperience with the technology required to develop the software system results in additional risks because the project team requires significantly more time to learn the technology and ramp up. Lack of experience in the technology leads to a lack of clear understanding of the requirements, design, development, and test planning for the software project. Insufficient technical

knowledge leads to failure because the project team lacks the technical competency to solve the software design problem in a manner that can lead to successful implementation.

2.3.4 Problematic Technology

Cutting-edge technology is often the platform for innovative and successful software projects. However, problematic, immature, or changing technology was also .co-identified in the research as a cause of failure. During the analysis of failed projects, the inability of the technology infrastructure to support the software project was often identified as a factor in project failure. Technology that is outpacing the rate at which the project team can develop the software leads to substantial rework to catch up. The opposite is true where the technology is so new that the project team has a difficult time learning and applying it to the design, development, and test planning.

2.3.5 Lack of Executive Leadership Support

Support from executive leadership is crucial to software project success. Unclear or inadequate support from executive leadership leads to poor project governance arrangements that precipitate unclear lines of responsibility. Failure to establish effective leadership in the business, technical, and organizational domains often results in a lack of project monitoring and control that in turn results in a project running out of control. Lack of support also leads to failure to establish clear decision making ownership, resulting in indecision and project confusion. Management leadership must be aware that the project will encounter setbacks throughout the lifecycle and that, therefore, they must be prepared to support and rally behind the project despite the setbacks. A lack of consistent leadership support and direction often contributed to high staff turnover, which in turn caused project delays and in turn contributed to project failure.

2.3.6 Insufficient User Commitment

Insufficient user commitment – lack of identification, engagement, and feedback from the required users and other stakeholders of the software system – is also a factor in the failed software projects identified in this research and in the literature. User engagement is necessary throughout the software system lifecycle, as user needs, desires, and expectations are captured as requirements from which the system is developed and tested. Lack of commitment results in failed systems because the crucial requirements for how the system will operate and the intended interactions with the users are not effectively captured and thus cannot be accurately validated.

2.3.7 Project Cost Overruns

Project cost overruns are a contributing factor to project failure and abandonment. Cost overruns are defined as software project cost estimates, identified during the acquisition and planning stages, being exceeded by more than 30% during the software development lifecycle. Several policies are in place on government acquisition programs to notify the appropriate leadership and decision makers when costs exceed guidelines and thresholds, including the Nunn-McCurdy Act. Programs in Nunn-McCurdy breach status that have exceeded original cost baselines by 30% are reported to Congress and those breaching by 50% or more face possible termination (AcqNotes, n.d.).

2.3.8 Project Schedule Delays

The impact of schedule delays is detrimental to project success. Throughout this research on software project failures, schedule delays were identified as a leading cause of

termination and abandonment. Project schedule delays breaches are defined in the DoDI 5000.02 Directive (2015) as a critical change in schedule that causes a delay of either one year or 25% or more of the baseline schedule. Adding more resources to an already late software project causes resource strain, with the adverse effect of still lower team performance and continued delays (Brooks, 1995).

2.3.9 Insufficient Project Management and Control

Project management and control implement success measures and criteria throughout the software project lifecycle. Lack of sufficient project management and control, the research uncovered, stemmed from a failure to plan the project and its outcomes. The research indicated that a lack of project management and control leads to poor decision making because there are no checks and balances to steer the software project in the right direction.

2.3.10 Project Failure in Testing Phase

Testing is a critical activity in the development of successful software projects because, as noted earlier, software is intangible and cannot be quantified as a physical product can be. Undercutting or eliminating adequate testing can prove fatal to a software project. Without clear testing procedures, it is impossible to determine whether the project has met systems, business, and user requirements of the software project before moving into the production environment. Software project failure in testing signifies that defects have not been uncovered and reported immediately and that tests are not conducted against proper requirements. A buggy system deployed into production can lead to user abandonment, cancellation, or, worse, lethal failure.

2.3 Software Project Failure Case Studies

Systems and software products can be found in almost every major industry, including the defense, aerospace, automotive, medical, telecommunications, industrial, and semiconductor fabrication industries (Walls, 2006) (ISO/IEC/IEEE, 12207, 2008). The data used in this paper were derived from failed software projects found throughout those same industries, with selected examples depicted in the snapshot shown in Table 7. The project names were converted to project identification numbers when structuring the data within the database in preparation for building the software predictive models. FP in the table stands for failed project and the numbers identify each specific project. These software project failure examples spanned a historical timeline dating from the 1990s until 2013.

Table 7: Snapshot of Failed Software Projects ID for R Coding

Project ID	Result of Failure	Estimated Cost	Estimated Start Date
FP1	Not Delivered	$185,000	
FP2	Abandoned	1 billion	2005
FP11	Abandoned	3 billion	2004
FP13	Abandoned	260 million	2001
FP26	Abandoned		2013
FP29	Cancelled	$54.4 million	2004
FP30	Abandoned after deployment	$400 million	2004
FP31	Abandoned	$527 million	2004
FP34	Cancelled	$170 million	2002
FP35	Cancelled	$33 million	2002
FP38	Cancelled	$130 million	2001
FP39	Abandoned	$25 million	2000
FP40	Cancelled	$12 million	1999
FP41	Cancelled	$11.2 million	1999
FP43	Cancelled	$4 billion	1997
FP44	Cancelled	$40 million	1997
FP46	Rocket explodes	$350 million	1996
FP47	Cancelled	$25.5 million	1995
FP48	Cancelled	$2.6 billion	1994
FP49	Cancelled	$44 million	1994

Project ID	Result of Failure	Estimated Cost	Estimated Start Date
FP51	Cancelled	$600 million	1993
FP52	Abandoned	$130 million	1993
FP53	Cancelled	$11.25 million	1990
FP54	Abandoned	$15 million	1993
FP56	Cancelled	$165 million	1992
FP105	Cancelled	$560 million	2008

In addition to the snapshot in Table 7, several specific examples of software project failure should be described. One example is the $327.6 million Mars Climate Orbiter (MCO) developed by NASA's Jet Propulsion Laboratory in 1998. Communication with the Orbiter was lost minutes after it approached Mars as a result of a software defect that caused a navigation error. The failure occurred because different groups within the engineering team were using different units of measurement. One team working on the thrusters measured in English units of pounds-force-seconds, while the others used metric Newton-seconds. Researchers from the navigation team identified the defect. Their concerns were dismissed, however, and the result was that the thrusters operated at a level 4.45 times more powerful than expected by the operators. According to the report developed by the NASA MCO Mishap Investigation Board (MIB), if this root-cause software defect had been addressed it could have been corrected prior to the launch. Unfortunately, trajectory correction maneuver number 5 was not performed and the result of that software error is now a spacecraft that is lost forever (NASA, 1999).

Another example of a single catastrophic software project failure is the Patriot Missile System Bug of February 21, 1991, which occurred during Operation Desert Shield. The Patriot Missile System was deployed as a defense against enemy aircraft and missiles. The tracking software for the Patriot Missile System predicts the enemy target in a time series by using the velocity of its target and the present time. The development team was aware

of an existing defect in the targeting software that caused the internal clock to slowly drift away from accurate time and had developed "the workaround" to reboot the system periodically. The longer the system was left running the greater the clock's time drift. However, the system was left running for 100 hours without "the workaround" reboot, causing a time drift and lag of 0.34 of a second. On that day, an Iraqi missile was launched toward the U.S. airfield in Dhahran, Saudi Arabia. The Iraqi missile was detected by the Patriot Missile System; however, when the system tried to calculate the next location of the Iraqi missile, the system was erroneously looking at an area over half a kilometer away from the missile's true location. Not seeing the target in the location, the Patriot Missile System assumed there was no enemy missile and cancelled the interception. The enemy missile carried on to its destination over the Dhahran airfield where it claimed the lives of 28 soldiers and injured 98 (GAO-IMTEC-92-26, 1992) (Naur & Randell, 1969).

Another example of a failed software project is the Department of Defense (DoD) Expeditionary Combat Support System (ECSS). The project was canceled in December 2012 after spending over $1 billion and failing to deploy within the five-year schedule estimate. ECSS was an Enterprise Resource Planning (ERP) program being developed for the Air Force to manage logistics. It was to provide relevant information to include personnel data, availability of a parts in inventory, in transit location and other types of data. A root cause report on the program determined that the greatest factor to failure was lack of sufficient technical expertise (Institute for Defense Analysis, 2011) (GAO-15-675T, 2015). The report identified the following additional factors:

- Lack of sustained involvement of senior leadership with authority and accountability for the direction and execution of the ERP

- Lack of leadership willingness and ability to make impactful decisions relative to proceeding with the implementation
- Lack of integrated governance to include representation of and participation by impacted stakeholders.
- Personnel lacked the requisite skill set and experience to define and execute an ERP implementation
- Over schedule and over budget

ECSS experienced many of the failure factors identified in the research and was cancelled prior deployment.

The DoD Defense Enterprise Accounting and Management System (DEAMS), designed to provide reliable, accurate, and timely financial data, has experienced significant cost increases and schedule delays. The cost increase is an estimated 45 percent above the original cost of $1.1 billion. DoD DEAM user assessments identified data discrepancy issues and the failure to generate audit financial reports requiring manual workarounds (GAO-15-675T, 2015).

Why are so many software projects failing? The GAO reports indicated that these and other failed software projects suffered from a lack of disciplined and effective project management, including project planning, requirements definition, and program oversight and governance. This research identifies factors that contribute to software project abandonment, cancellation, non-delivery, and ultimate failure to develop a predictive model that can predict the outcomes of future software development projects. This research has confirmed these failure factors with rigorous content analysis of 202 software projects data as a basis for developing the predictive model.

2.4 Other Tools Developed to Counter Project Failure

This literature review has identified several approaches and methodologies to prevent software project failure, including research by Bronte-Steward (2009), who developed a three-phase risk estimating technique that is implemented during the early planning phases of a project lifecycle before much cost and work have been expended (Bronte-Stewart, 2009). The author researched the cost and frequency of project failure by looking into the findings of government reports and industry surveys identifying many of the same failure factors found in this research and illustrated in Table 8.

Table 8: Typical Causes of Failure/Success in IT Projects

Research reports by a Government Select Committee (2005), OASIG (1996), National Audit Office (NOA)/Office of Government Commerce (OGC) (2005), Standish Group CHAOS (1995), and Schmidt et al. (2001) (Source: Bronte-Steward [2009])

Select Committee	OASIG	NAO / OGC	CHAOS (Top Requirements:)	Schmidt et al
IT projects are driven by business (not technical) decisions	Many IT investments are seen only as technology led and aimed at cost cutting	Evaluation of proposals driven by initial price rather than long term value, especially securing delivery of business benefits	Clear business objectives and Realistic expectations	Misunderstanding user requirements
Insufficient involvement from users	Users do not influence development enough	Lack of effective engagement with stakeholders	User involvement	Lack of user involvement or commitment
Clear objectives should be set from the start	Need to set and review strategic objectives for change	Lack of clear link between the project and the organisation's key strategic priorities, including agreed measures of success	Clear and firm statement of requirements	Unclear and changing scope and objectives
Lack of commitment from senior management	Management agenda is often too limited or narrow	Lack of clear senior management ownership and leadership	Executive management support	Lack of top management commitment
Large projects may be overambitious	Inadequate attention is given to human and organisational issues	Too little attention to breaking development and implementation into manageable steps	Minimised scope and smaller project milestones	Number of organisational units involved
Skilled project managers are essential to keep to time and budget and appropriate deliverables	Senior managers do not understand the link between technical and organisational change	Lack of skills and proven approach to project management and risk management	Experienced project managers	Lack of required knowledge and effective project management skills
Success depends on good risk analysis and sound methodologies	Some project management techniques and IT approaches are too technical		Proper planning and formal methodology	Lack of effective project management methodology
Contingency plans should be in place	Must work to detailed implementation plans		Reliable estimates	Not managing change properly
User and operator training must be planned and designed	Failure to organise changes in work and roles properly	Inadequate resources and skills to deliver the total delivery portfolio	Competent, skilled and focussed staff	Inappropriate staffing and ill defined responsibilities
There should be a post-implementation review			Standard software infrastructure	Introduction of new technology
Need professionalism in the definition, negotiation and management of IT contracts		Lack of understanding of , and contact with, the supply industry at senior levels in the organisation	Ownership	

Bronte-Stewart converted the failure factors or issues into a list of risks or threat analysis factors to propose a set of linked tables of grouped categories of the risk criteria and Likert scale and weighting factor (Bronte-Stewart, 2009). The author recommended that the risk estimation tables be used as part of the feasibility study at the onset of IT projects before committing too much labor and cost. "The process can help give an analyst an idea of views of the level and significance of various risks that may threaten the success

of an IT project (Bronte-Stewart, 2009).” Although this risk list is very useful during project initiation, there were no predictive capabilities or properties in this research.

Ewusi-Mensah (2003) developed a framework of abandonment or failure factors that highlights the risks and uncertainties present in the systems development phases of a software project. His study of software project failures using case studies from Confirm Travel Industry Reservation Program, FoxMeyer’s Delta, the IRS’s Tax Systems Modernization, CODIS, and the Denver International Airport’s Baggage Handling System aided in the selection of those factors that highlight the risks and uncertainties present in the systems development phases of a software project (Ewusi-Mensah, 2003).

Takagi et al. (2008) developed a questionnaire designed from five project perspectives: requirements, estimates, planning, team organization, and project management activities. The questionnaire investigated risks and problems in software development within the Social Systems Solutions Business Company (SSBC) at OMRON Corporation. The responses from the questionnaire aided in developing a scheme for characterizing the level of confusion reflected by the software projects. Using the resultant metrics data, Takagi et al. classified software projects into “confused’’ and “not confused,” analyzing the relationship between responses to the questionnaire and the degree of confusion of the projects. They used logistic regression analysis to construct a model to identify the unique characteristics of the confused software projects (Takagi, Mizuno, & Kikuno, 2005). They accurately identified the confused software projects, providing insight into the risks and issues within projects. However, Takagi et al. do not predict software project failure.

Reyes et al. (2011) developed a method to identify the risk factors most influential in predicting project outcome by using the probability of success relative to cost to calculate

the efficiency of the outcome. The authors developed a software project success/risk analysis model to identify, analyze, and control potential risks during software development (Reyes, Cerpa, Candia-Vejar, & Bardeen, 2011). Reyes et al. conducted their research using a survey of 88 questions organized into seven risk categories. They ran a genetic algorithm on six Bayesian prediction models created with past software project data. The authors defined a unique software project example case for each model based on projects reported in the survey that were used to build the prediction model. In each execution of the genetic algorithm on the six models, the algorithm encountered a satisfactory solution to the problem identified by the chromosome with the highest fitness value in the population. This chromosome information determines which risks factors must be controlled and managed in order to improve project success outcomes at an appropriate cost and yielding highest gains (Reyes, Cerpa, Candia-Vejar, & Bardeen, 2011). The authors developed a software project success and risk analysis model to identify, analyze, and control potential risks during software development. Again, however, prediction properties were not noted in their research.

Several authors propose a Bayesian Network to predict and control software project schedule variance and thus project success. The model is used to assist project managers to predict the probability of software schedule variance and to guide software developers in making improvement actions. The authors develop a sensitivity analysis to identify the most important factor contributing to software schedule variance (Wang, Wu, & Ma, 2010). The authors use a Bayesian Network because of the uncertainty of the software schedule, the ability to combine information such as historical project data and expert judgment, the ability to transfer software schedule factors into knowledge through

Bayesian learning, and the ability to conduct ‘what-if’ analysis to assist project managers in making improvement actions by exploring the most important factors in software schedules. The authors explored three related notions by calculating the project schedule variance during software development. They used that information to re-plan the project, re-allocate resources to complete the project on time, and identify the variables that contribute the most to software schedule variance. The research was specific to schedule variance and did not contain predictive properties.

Cerpa et al. (2010) built a logistics regression model by analyzing software project data collected through surveys provided to software developers in order to identify the variables most affecting software project outcome. They constructed prediction models using key success factors to address issues of cost, effort, and schedule estimation using only projects classified as failures from data from 164 distinct software projects. The authors used ROC analysis to compare and evaluate the predictive effectiveness of models over raw variables and factors obtained by principal component analysis (PCA) (Cerpa, Bardeen, Kitchenham, & Verner, 2010). The authors identified several variables that were indicative of software project failure using only failed project data.

Lehtinen et al. (2014) conducted an analysis of software project failures with four software product companies to uncover the causes of failure and their relationships to one another (Lehtinen, Mantyla, Vanhanen, Itkonen, & Lassenius, 2014). The authors sought to understand which causes interconnected with which specific process areas and which were perceived as the most promising for process improvement initiatives to control the root causes of failure. Their research used root cause analysis (RCA) to identify the causal relationships among the causes of failure by taking the problem as input and providing a

set of perceived causes and causal relationships as output. The study identified the causes of problems, where they occurred, why they occurred, and their interconnectedness in order to improve the software project outcomes by creating mechanisms for prevention of failure. Lehtinen et al. (2014) developed causal models of software project failure for four case companies by analyzing project failure in each case company. They used focus groups with key representatives from senior management who had the authority to make process changes in their companies. Each case used evidence within the focus group to identify the target problem causing the project failure. Case Defects Company was plagued with a high number of defects detected at the end of the project that caused schedule overruns. The failure selected for deeper analysis was inadequate software testing that resulted in delayed releases due to the large number of defects uncovered at the end of the development project. Case Quality was impacted by uncontrollable side effects of the existing product that were difficult to account for during implementation and to detect during software testing and were symptoms of other causes. Lack of cooperation caused insufficient requirements that lowered development work efficiency and impaired software testing because of missing information (Lehtinen, Mantyla, Vanhanen, Itkonen, & Lassenius, 2014). Case Complicated resulted from complex version dependencies, insufficient software testing and difficult release and deployment configurations. The authors noted a large number of causes related to the difficulty present in release and deployment, and insufficient task output of software testing (Lehtinen, Mantyla, Vanhanen, Itkonen, & Lassenius, 2014). Case Complicated depicts the failure partially caused by a lack of cooperation, lack of task priorities, and an inflexible development process; it shows a large number of detected, proposed, and selected causes related to these failures (Lehtinen, Mantyla, Vanhanen,

Itkonen, & Lassenius, 2014). Lehtinen et al. noted that despite the fact that the failure cases selected for analysis were very different, three common causal relationships bridge the process areas: Weak Task Backlog, Lack of Corporation, and Lack of Software Testing Resources. These stemmed from incorrect decisions on task priorities, vague requirements specifications, and vague task descriptions, which in turn made it difficult for developers to know what to develop. Missing verification criteria and vague defect reports posed a problem for software testing because the testers did not know what to verify. The vague reports complicated the work of managers when it came time to prioritize the tasks (Lehtinen, Mantyla, Vanhanen, Itkonen, & Lassenius, 2014). Their research used RCA to identify the causal relationships of failure by taking the problem as input and providing a set of perceived causes and causal relationships as output. The study identified the causes of problems, where they occurred, why they occurred, and their interconnectedness in order to improve the software project outcomes by creating mechanisms for prevention of failure. However, the small data set of four companies may pose a limitation.

2.5 Industry Uses of Predictive Analytics

Predictive analytics encompasses a range of methods to anticipate outcomes. There are four main reasons for a business to use analytics: to predict, to identify opportunity, to calculate demand, and to identify and prevent fraud and risk (Wessler, 2014). Roy (2013) developed a predictive model for disease prevention to predict patient readmission to the hospital and patient death after discharge. Accurate prediction of those outcomes would allow health care professionals to develop strategies and measures to reduce these serious and costly risks. The patient outcome tool calculates the probability of the outcome for a set of patients and derives a risk score for each patient.

IBM leverages business analytics to help organizations tap into their data in order to, among other benefits, predict trends, identify business opportunities, build customer loyalty, improve financial performance, optimize operations, and prevent fraud. The focus of IBM's use of predictive analytics is to accelerate value creation for organizations struggling to capture the insights embedded in their data in order to "drive smarter decisions and positively influence business outcomes" (Balboni, Finch, Reese, & Shockley, 2013).

Deloitte's Predictive Project Analytics Services (2012) is a commercial product marketed to enhance traditional project management tools; Deloitte dubs it the "next generation project management." Deloitte's tools assess key project management practices to determine the extent of their use and to help identify areas for improvement. The tool appears to be promising in that it identifies gaps early in a project's lifecycle, so that recommendations can be made to improve project management practices and processes. However, an important aspect of using predictive analytics is successful deployment to stakeholders. From the research, it does not appear the tool has gained wide acceptance. As of the date of the writing of this paper, the Deloitte tool has been used to reduce failure on only one project. A large healthcare organization whose clinical project was experiencing significant delays and budget overruns consulted with Deloitte to get the project back on track.

2.6 Information Technology Project Oversight Programs

Oversight programs and initiatives have been implemented to monitor IT project execution in order to identify project risks. OMB developed the IT Dashboard in 2009 to enable federal agencies, industry, and the general public to view detailed information on

federal IT investments. The Dashboard supports management decision making and assesses the effectiveness of federal IT programs; Congress and the Administration use it to make policy and budget decisions (OMB, 2015).

TechStat is another initiative launched by OMB, in 2010, as a result of the dire state of IT projects across the federal government. TechStat is an evidence-based accountability review of an IT project investment that enables the federal government to intervene to turn around or terminate a project that is failing or not producing the intended results for the American population (CIO.gov, 2015). Figure 2 below outlines the implementation phases of the TechStat initiative. The overall goal of the TechStat roll-out was to turn-around or terminate "one third of all underperforming IT investments by June 2012" (CIO Council, 2011). As part of that goal it was stated that Chief Financial Officers (CFO), Chief Information Officers (CIO), and Chief Acquisition Officers (CAO) collaborate to ensure that IT portfolio analysis is integrated into the yearly budget process for the agency. TechStat reviews have uncovered significant opportunities for improvement of IT investments as well as the TechStat process. The TechStat subcommittee will continue to mature the process by analyzing outcomes to identify trends in management and technology approaches that impact success of the programs. The analyses will serve as test cases to assist the Best Practice Committee in developing lessons learned and common challenges to help drive better performance. Since its official launch, many agencies have used TechStat to make decisions that significantly changed the course of the projects. The TechStat outcomes so far have included accelerated delivery and improved governance; they have reduced project scope, eliminated duplication and redundancy, and halted and completely terminated projects. Since its implementation, TechStat reviews have resulted

in over $3 billion in cost reduction as well as an average deliverables acceleration from 24 months to 8 months (CIO Council, 2011).

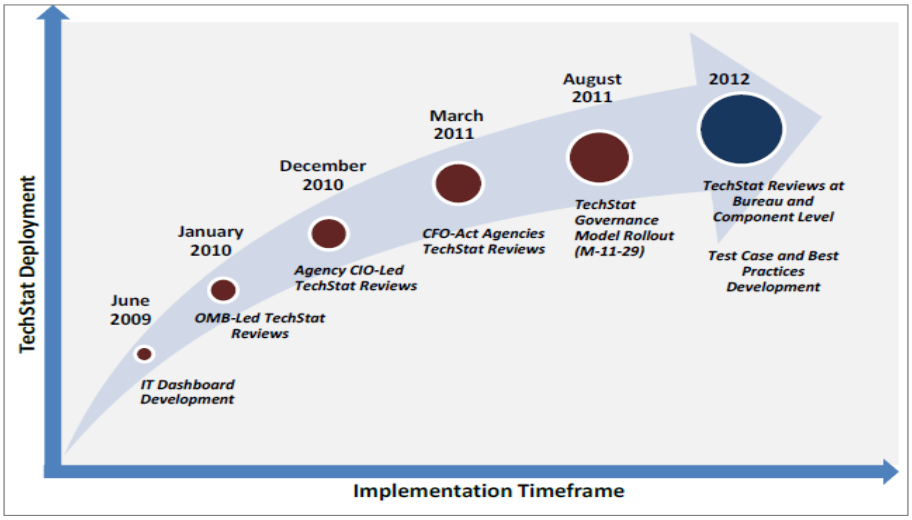


Figure 2: TechStat Implementation Framework

(Source: CIO Council, 2011)

PortfolioStat is an evidence-based review of the IT portfolio to identify potential duplications as well as project investments that do not meet or are not well aligned with the mission or business functions and other key considerations within the agency's IT portfolio (CIO Council, 2015). PortfolioStat was launched by OMB in 2013 to assist agencies in developing a plan to eliminate waste by reducing duplicative systems. While TechStat examines IT performance for a specific project, PortfolioStat examines the agency's entire IT portfolio in order to identify IT investments and eliminate duplication and redundancies in order to maximize the return on investment across the portfolio (OMB, 2012).

The Clinger-Cohen Act of 1996, also known as The Information Technology Management Reform Act of 1996, developed a core set of responsibilities for the federal CIOs. The Act was created to assist agencies in establishing effective processes and

procedures for selecting, managing, and evaluating the results of all major investments in information systems. The CIO must "monitor the performance of IT programs to evaluate the performance of those programs on the basis of the applicable performance measurements and advise the head of the agency regarding whether to continue, modify, or terminate the program or project" (AcqNotes, n.d.). The Act provided guidance on monitoring and measuring the success of new system development to determine when and how to "pull the plug" on underperforming systems. The Clinger-Cohen Core Competencies was published to include information resource management knowledge requirements and learning objectives for the development of the federal government IT workforce. The Core Competencies embody a set of 12 competency areas identified by the CIO Council as fundamental to the effective management of federal technology resources. They are depicted in Figure 3 (CIO Council, 2012).

Figure 3: 2006 Clinger-Cohen Core Competencies

(Source: US Department of the Interior, 2015)

The 12 core competencies areas include several subordinate competencies and their associated learning objectives, as displayed in Table 9. Together, the Act and the core competencies assist in developing a workforce of practitioners that will guide in the successful implementation and deployment of IT software systems.

Table 9: Clinger-Cohen Core Competencies and Learning Objectives

(Source: CIO Council, 2012)

Clinger-Cohen Core Competencies	Learning Objectives
1.0: Policy and Organization	General Discussion: The CIO has one of the most cross-cutting positions in government and must be able to work effectively with a wide range of people across multiple organizations. Additionally, the CIO must be comfortable in a fast-changing environment that includes evolving technologies, legislation, policy, and politics.
2.0: Leadership and Human Capital Management	General Discussion: Management concepts are important but CIOs must move beyond management to leadership. This includes oversight over the individuals within their organization, and working to attract, retain, and develop their personnel.
3.0: Process and Change Management	General Discussion: The paramount role of the CIO is as Chief Visionary of the organization's information and technology—critical enablers for achieving mission and improving efficiency. Change management encompasses far more than a single leader's perspective. The CIO works in strong partnership with the CXOs and other key stakeholders as part of the change management process. Open, effective communications are essential to ensure organizational buy-in.
4.0: Information Resources Strategy and Planning	General Discussion: IT must be a value-adding dimension of the business plan. Information Resources Management (IRM) strategic planning must begin with the business strategic planning process and integrate with the organization's business functions and plans since business planning and IRM planning are parallel and coupled processes. IRM planning should also address cross-governmental and inter-agency planning issues as well as external drivers.
5.0: IT Performance Assessment: Models and Methods	General Discussion: The CIO has the challenge of meeting both customer and organizational needs established in the agency's business plan. In order to ensure those needs are being met, the CIO must understand the importance of the qualitative and quantitative baseline assessment measures and their use in the performance assessment cycle.

Clinger-Cohen Core Competencies	Learning Objectives
6.0: IT Project and Program Management	General Discussion: The relationship between project management and program management is interdependent, not discrete, and progressively cumulative. A project is a specific investment having defined goals, objectives, requirements, lifecycle cost, a beginning and an end that delivers a specific product, service or result. A program is typically a group of related work efforts, including projects, managed in a coordinated way. Programs usually include elements of ongoing work. For program management processes to be mature, project management processes must be mature. IT Program Managers should be skilled in both IT Project and IT Program Management Competencies
7.0: Capital Planning and Investment Control (CPIC)	General Discussion: It is essential that CIOs understand the importance of Capital Planning and Investment Analysis. Capital planning is needed to provide a framework for running government with the same disciplines as private business. In addition to passage of the Clinger-Cohen Act (now codified in Title 40), there is an array of other legislation and fiscal guidance which are significant to effective Capital Planning and Investment Control.
8.0: Acquisition	General Discussion: Acquisition links technology investment to the business outcomes and results, as defined by the end consumer. Acquisition needs to move from what been a singular focus on process to one that considers both process and objectives. Acquisition anticipates what is needed before it is officially stated, and develops requirements that include the end users and must be linked to business outcomes. The CIO must understand the new dynamic, and understand lifecycle management. He/she must move from a risk-averse process to one of risk management, and create an innovative acquisition environment throughout the organization. The CIO should monitor changes in acquisition models and methods. Acquisition includes four stages–(1) Defining the business objective; (2) Requirements definition and approval; (3) Sourcing and (4) Post-Award management—which are each critical to a successful IT acquisition.
9.0: Information and Knowledge Management	General Discussion: Under Title 40, Subtitle III, Chapter 113, Section 11315, Agency CIOs have information resources management (IRM) identified as their primary responsibility. Per Circular A-130, IRM encompasses both information itself and the related resources, such as personnel, equipment, funds, and information technology. As part of their information management responsibilities, the CIO must also deal with Privacy issues; Freedom of Information Act (FOIA) requirements; Open Government mandates; and accessibility issues, as well as the preservation of records to comply with business, operating, regulatory

Clinger-Cohen Core Competencies	Learning Objectives
	and legal requirements. In addition, the CIO may support knowledge management activities to preserve and share subject matter expertise.
10.0: Cybersecurity/Information Assurance (IA)	General Discussion: The Federal Information Security Management Act (FISMA) – codified in Chapter 35 of Title 44, U.S. Code - charges each Federal CIO with the responsibilities to develop and maintain an agency-wide cybersecurity/information assurance(IA) program, including security policies, procedures and control techniques to both protect and defend information, systems and networks. CIOs must be able to assess the risks associated with vulnerable systems and information; determine the levels of security protection required; institute cost-effective methods to reduce risk to acceptable levels; and continuously monitor the capabilities of those techniques and controls. In addition, they must oversee the training programs to ensure that both the protectors and users of information and systems have the knowledge necessary to adequately protect organizational assets. The Office of Management and Budget (OMB) promulgates procedures for FISMA compliance and has levied additional requirements for cybersecurity/IA programs through OMB Circular A-130. Additionally, there are legislative and regulatory requirements that mandate specific care for certain types of information including (but not limited to) sensitive but unclassified information, corporate fiduciary information, personally identifiable information, and personal health information.
11.0: Enterprise Architecture	General Discussion: An enterprise architecture (EA) establishes the agency-wide roadmap(s) to meet mission and strategic goals through the optimal performance of core business processes and supporting information resources (e.g. systems, applications, databases, websites, and networks). Enterprise architecture roadmaps are essential for transforming the existing business processes and IT solutions to an optimal business capability target that provides maximum mission value. EA includes agile plans for transitioning from the current business and technology operating environment to the target environment.
12.0: Technology Management and Assessment	General Discussion: Since the inception of the Clinger-Cohen Act, the CIO's role as technology manager has become increasingly complex. The ability to ensure effective development and deployment of technology requires a broad awareness of current and emerging technology capabilities, standards, policies and law. CIOs must also be able to identify and evaluate the strategic benefits of technology applications within the business environment.

Despite these initiatives to assist in managing IT investments, the CIO Executive Board estimates that troubled projects waste an estimated $8 million in direct costs out of an allocated $100 million project budget (Horne, 2013). Part of the reason is that most companies use a red/yellow/green dashboard to track project health. However, these can be misleading, as a 2013 GAO investigation of OMB's IT Dashboard review and reporting process identified. GAO determined that the tool is not as accurate as it should be because the auditors could not corroborate the reliability of the reported outcomes and cost savings due to lack of evidence to validate the data. Additionally, the IT Dashboard was found to have deficiencies in the cost and schedule ratings, as they were not always accurate (GAO-13-524, 2013). The Dashboard is a visual representation of performance ratings on individual IT investments using cost schedule and risk evaluation by the CIO. The CIO risk evaluation is assessed against a set of evaluation factors: risk management, requirements management, contractor oversight, historical performance, and human capital. A rating of 1 to 5 is assigned to each risk based on the CIO's best judgement. OMB translates the numerical assignment to a color in the Dashboard, where green signifies low or moderately low risk, yellow signifies medium risk, and red signifies moderately high or high risk as in Table 10. GAO reported that CIOs at several agencies rated many of their IT investments as low risk and did not appropriately reflect the significant cost, schedule, and performance issues identified by the GAO.

Table 10: OMB's IT Dashboard

Table 1: IT Dashboard CIO Rating Colors, Based on a Five-Point Scale for CIO Ratings

Rating (by agency CIO)	Color code
5 – Low risk	Green
4 – Moderately low risk	Green
3 – Medium risk	Yellow
2 – Moderately high risk	Red
1 – High risk	Red

The Clinger-Cohen Act has defined the role of the CIO as integral to the management of IT in leading reforms, controlling system development risks, managing IT spending, and achieving measurable improvement in performance. However, in 2011, GAO reported that federal CIOs were not being consistently responsible for all the areas assigned by the Act. The report found that CIOs were less frequently responsible for information management duties such as records management and security and privacy requirements (GAO-11-634, 2011). Therefore, CIOs could not adequately track the health of the IT investments because the agencies were reviewing only about one third of the at-risk IT investments. Additionally, due to the subjective nature of the risk assignment process, the reported results were inconsistent or not valid in determining the true risk characteristic of the system. The GAO report noted that more must be done to ensure that at-risk investments are undergoing review, sound processes are in place, and reported results are valid. With federal government IT spending projections totaling $82 billion this year alone, GAO indicated that it is of critical national importance to do all that is feasible to ensure that IT investments are successful.

As recently as December 2014, Congress led the enactment of the Federal IT Acquisitions Reform Act (FITARA) to address the issues related to government-wide management of IT investments (OMB, 2015). "Sixteen years following the signing of the seminal Clinger Cohen legislation … and ten years after the E-Government Act passed that established a Federal CIO, program failure rates and cost overruns still plague between 72 and 80% of large government IT programs. … Some have estimated the cost to taxpayer to be as high as $20 billion wasted each year" (Meritalk, 2015). The law requires that OMB make available a public list of all major IT investments and include cost, schedule, and performance data. OMB must then report to Congress any investments deemed high risk for four consecutive quarters. FITARA increases the responsibility, accountability, and authority of the agencies' CIOs to have a significant role in the program, budget, management, governance, and oversight decision processes for IT project investments (GAO-15-675T, 2015). Congress is hoping that successful implementation of FITARA will improve IT investment acquisition practices to eliminate duplicate systems and to develop IT acquisition experts to assist in buying cheaper, faster, and smarter products and services. The aim of FITARA is for "CIOs to check in on IT contracts frequently to avoid failure" (Ravindranath, 2015).

2.7 Literature Review Summary

The research has identified several predictive models and other tools to predict project success or failure. Many of those tools were created with data from one set of project outcomes. Deloitte, for example, developed its tool using data from 2000 successfully completed projects only (White, 2012). Lehtinen et al. (2014) conducted an analysis on only four cases of software project failures.

The research identified several pitfalls to avoid in using predictive analytics. One of of those pitfalls is that organizations cannot simply build a model once and apply it to everything (Fitzgerald, 2014). A new model may be required for every question asked. Why are software projects failing at such staggering rates and how can outcomes be improved? Why are software projects continuing to fail despite the vast amounts of research on the topic? Why are managers and decision makers continuing to fund failing software project efforts? In order for the authors of this paper to understand this phenomenon, the author developed a new model that included data from both sets of project outcomes: successes and failures. The authors also identified and included factors present in the four software engineering lifecycle stages of Requirements, Design, Development, and Test to determine if the factors identified in both the failed and the successful software project data used to develop the model actually contributed to software project failure.

Fitzgerald (2014) pointed out another trap to avoid in developing predictive models – creating models that do not scale and are too complex and expensive to be reused easily. The model for software predictions in this research paper was developed with the open source, free R software tool so that it can be easily transferred to and used broadly within any size software development organization. Small startups that are more vulnerable to software project failures because they have limited IT budgets and resources can use the model developed in this research to make better informed, evidence-based decisions. Larger, complex government entities can use the streamlined model to implement legislation requiring their contractors to use an analytics tool during contract procurement to outline their skills as well as their shortfalls in their past software development program initiatives. The model can be integrated into current IT oversight programs such as the IT

Dashboard in order to assist CIOs in assessing project risk quantitatively, based on true and calculated empirical data as opposed to personal judgement. For example, in the IT Dashboard, historical performance ratings can be determined directly from the outcomes of the predictive model as the method of determining successful and failed projects.

3.0 Research Methodology

The objective of this research is to develop a practical solution to a general systems engineering, software engineering, and project management problem: quantifying project failure and project risk prior to initiating development of a software system.

3.1 Data Collection

The predictive model developed from data of software project failures and successes is based on a framework by Ewusi-Mensah, 2003 that identifies the significant influencing failure factors: Unrealistic Project Goals and Expectations, Changing or Unclear Requirements, Insufficient Technical Knowledge, Problematic Technology, Lack of Executive Leadership Support, Insufficient User Commitment, Project Cost Overruns, Project Schedule Delays, and Insufficient Project Management and Control (Patanakul & Omar, 2010) (Stevens, 2011). The failure factors spanned the four major phases of the SELC: Requirements, Design, Implementation, and Testing.

In developing the model, project failure and success data were collected for 202 software projects using publicly available case studies, news articles, surveys, industry websites, and congressional reports. The researcher also gathered data from reports generated by the Government Accountability Office (GAO), Court litigation cases, and other government artifacts (Frankfort-Nichmias & Nachmias, 2000). Through content analysis (Tsao, Hsu, & Tsai, 2012) the researcher defined key factors of project failure and success and developed a database to capture information pertinent to each individual project. Multiple sources were reviewed for each software project to assure identification of the most informative and accurate information. The researcher performed content coding

to extract the most consistent information and to create groups of failure factors for each project. To identify common failure factors, comparisons of each project to the others with cross-case analysis of each failure factor's impact on project failure was performed. This analysis was crucial to developing the predictive model as it helped to understand and overcome the challenge of structuring the model and transforming the data into a format readable by the R software (Alfons, 2012)

The goal was to develop a methodology to support and enable sound decision making about any software project, its product, and the associated risks, and to use the predictive model to create a "manage by fact environment" where decision-makers listen to their organizations' observed and empirical data (Baldridge, 2012). In order to go forward and develop successful software systems, the organization's capability to develop a software project and its previously delivered software systems must be measured and historical data must be collected during the planning phase to accurately predict a probable future outcome.

3.2 Building the Model

The researcher began developing the predictive model by structuring the software project outcome data. Because the outcomes could be only one of two options—Failure or Success—logistic regression was the method selected to build the model to explain the relationship between the independent variables (Failure Factors) and the outcome or dependent variable (Failure). In the process of structuring the data, the researcher identified missing failure factors within many of the 202 software projects. This was due to the lack of information on those specific factors as it relates to each of the projects. To account for this missing data, the researcher attempted to handle the missing data by transforming

blank cells to the value NA and then assigning NA = 0 in the R program. NA is the missing value code used by R. The researcher then filled in or imputed the missing value based on further analysis of the available data for each of the 202 software project data points. Further future research will look into more robust and sophisticated methods to handle the missing data, such as EM algorithms or multiple imputation.

Because logistic regresion is one of the most commonly used methods of prediction, the researcher chose to use it to predict individual project binomial outcomes where the dependent variable (Failure) is a dichotomous variable that returns either Failure = 1 or Not Failure/(Success) = 0. It models a relationship between the independent variables and a function of the dependent variable. This allows for nonlinear effects on the dependent variable; however, the model itself is linear in its predictors for failure or success. (Lattin, Carroll, & Green, 2003) (Everitt & Hothorn, 2006).

The underlying statistical concept using logistic regression is the generalized linear model (glm) function (R-Development, 2011) (Kuhn, 2008). The glm is a flexible generalization of regression that allows the model to be 'linked' to the response variable via the logit function to enable a wide range of disparate problems, such as the ten Failure Factors, to come together into a powerful yet flexible framework. The logit of a probability is the log of the odds of the response with value of one. For this research the researcher wanted to know the odds of failure and Equation 1 below is the logistic function used in the development of the models (Everitt & Hothorn, 2006):

$$Logit(\pi) = \log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q. \qquad \text{(Equation 1)}$$

Therefore the logit of a probability is the log of the odds of response or outcome with the

value of one as written in Equation 2. This will be demonstrated with the software project probability outcomes of the predictive model.

$$\pi(x_1, x_2, \dots, x_q) = \frac{\exp(\beta_0+\beta_1 x_1+\cdots+\beta_q x_q)}{1+\exp(\beta_0+\beta_1+x_1+\cdots\beta_q x_q)}. \quad \text{(Equation 2)}$$

The researcher used Equations 3 and 4 to fit the predictive model. Equation 4, derived from Equation 3, is a variation of the Logit and Probability functions in R. Because project failure stems from multiple factors identified in Table 11, they are all deemed significant in building the model, as the research will demonstrate.

$$Logit(\pi) = \log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 FFA + \beta_2 FFB + \cdots + \beta_q TestF$$

$where\ \pi = probability\ of\ a\ failure.$ (Equation 3)

Model = glm(Failure~FFA+FFB+FFC+FFD+FFE+FFF+FFG+FFH+FFI+TestF, traindata, family ="binomial") (Equation 4)

Table 11: Failure Factor ID Coding

Failure Factor ID	Failure Factors
FFA	Unrealistic Project Goals and Expectations
FFB	Changing or Unclear Requirements
FFC	Insufficient Technical knowledge
FFD	Problematic Technology
FFE	Lack of Executive Leadership Support
FFF	Insufficient User Commitment
FFG	Project Cost Overruns
FFH	Project Schedule Delays
FFI	Insufficient Project Management and Control
TestF	Project Failure in Testing Phase

In developing the model, the researcher randomly split the original n=202 project success and failure data set into three separate data sets. The first training set was used to functionalize or fit the model. This set comprised n=182 projects containing both successes and failures. The second test set comprised n=10 projects randomly selected containing both successes and failures. The third hold-out validation set comprised n=5 projects also containing both successes and failures. The validation set was developed to assist in performing exploratory analysis to identify the best model fit. The research included all the failure factors identified in Table 11 in the model. FFA through FFI and TestF denote the independent variables or Failure Factors. Equation 4 is used to calculate the probability of a failure for each line item within the original data set. The probability score is a value between 0 and 1. The researcher set a cutoff value of 0.5, with increased likelihood of failure for any results above 0.5 based on analysis of the original data set. Future analysis will be performed to select an optimal cutoff probability using the distribution of predicted probabilities to inform the decision.

4.0 Research Results

4.1 Analysis of Predictive Model A

Predictive Model A was run using all the failure factors as defined in Equation 3 of the original data set. The purpose of this model was to determine whether or not the data from the original data set could in fact accurately predict software project failure. The resulting p-values from Figure 4 indicate that FFA, FFB, FFC, FFE, FFF, FFH and TestF are significant predictors of a failure in the presence of the other predictors. All the failure factors with the exception of FFC, FFD and FFG have significance level p-values less than the industry standard of .05. The low p-values indicate that the project failure attributing to these factors was unlikely to occur simply by chance (Higgins & Green, 2011). Though FFC, FFD, and FFG are not statistically significant in Model A, further modeling approaches such as regularization and cross-validation find these particular factors to be significant later in the analysis of Predictive Model B.

```
glm(formula = Failure ~ FFA + FFB + FFC + FFD + FFE + FFF + FFG +
    FFH + FFI + TestF, family = "binomial", data = traindata)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.77443  -0.06671   0.03867   0.11212   2.85520

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.4133     1.7115  -3.163  0.00156 **
FFA           2.8028     1.0260   2.732  0.00630 **
FFB           3.7279     1.1439   3.259  0.00112 **
FFC           0.6196     1.0059   0.616  0.53791
FFD           1.5375     1.0374   1.482  0.13834
FFE           2.6292     1.1466   2.293  0.02184 *
FFF          -3.8406     1.5269  -2.515  0.01189 *
FFG           0.5103     1.0123   0.504  0.61418
FFH           4.9002     1.1569   4.236 2.28e-05 ***
FFI          -0.6935     1.0813  -0.641  0.52130
TestF         2.9599     1.3222   2.239  0.02518 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 213.996  on 181  degrees of freedom
Residual deviance:  41.579  on 171  degrees of freedom
AIC: 63.579
```

Figure 4: P-value Significance and AIC Scores for Model A

Figure 5 identifies the Failure Factors resulting in actual project outcomes for each of the projects identified in the n = 10 Test Set. Project ID SP7 in row 22 depicts the actual Failure = 0 for a successful project (SP) and the corresponding predicted value 0.04829964 probability of failure. SP7 probability of failure is less than the probability score cut-off of 0.5 and thus accurately predicts a successful project. Project ID SP51 in row 111 also accurately depicts success with a probability score of 0.25483709 and actual Failure = 0. Alternatively, Project ID FP23, FP52, FP59, FP81, FP104, FP119 and FP129 accurately depicts Failure = 1 for a failed project (FP) based on their predicted probability scores greater than 0.5. Only row 125 Project ID FP69 was incorrectly labeled as success where Failure = 0; when in fact it is a failed project with a probability of 0.91079014 of failure.

row.names	Project.ID	FFA	FFB	FFC	FFD	FFE	FFF	FFG	FFH	FFI	TestF	Failure
22	SP7	1	0	0	0	0	1	1	0	0	1	0
37	FP23	1	0	1	1	0	0	0	0	0	0	1
94	FP52	0	1	0	0	1	0	1	1	1	0	1
101	FP59	1	0	0	1	0	0	0	1	1	0	1
111	SP51	1	0	0	1	0	0	0	0	0	0	0
125	FP69	1	0	1	1	0	0	1	0	1	1	0
141	FP81	1	0	0	0	0	0	1	1	1	1	1
164	FP104	1	0	0	0	0	0	1	1	1	1	1
179	FP119	0	1	0	1	1	0	1	1	1	0	1
189	FP129	0	1	0	1	1	0	1	1	1	0	1

```
HTML> predtest
       22        37        94       101       111       125       141       164       179       189
0.04829964 0.38855638 0.99653232 0.95826183 0.25483709 0.91079014 0.99373460 0.99373460 0.99925268 0.99925268
```

Figure 5: Actual Outcome vs Prediction for Model A Test Data

The Receiver Operating Characteristic (ROC) Curve in Figure 6 measures the ability of Predictive Model A, run with the training set and the test set, to correctly classify those projects that failed versus those that were successful. It plots the true positive rate (sensitivity) against the false positive rate (1-specificity) of the model against the given data set. The closer the test follows the left-hand border and the top portion of the ROC curve, the more accurate the test (Tape, 2014). Figure 7 demonstrates that the training set produces a more accurate test than the test set. The test set has never been seen before by the model so it does not perform as well. This is confirmed by Figures 8 and 9.

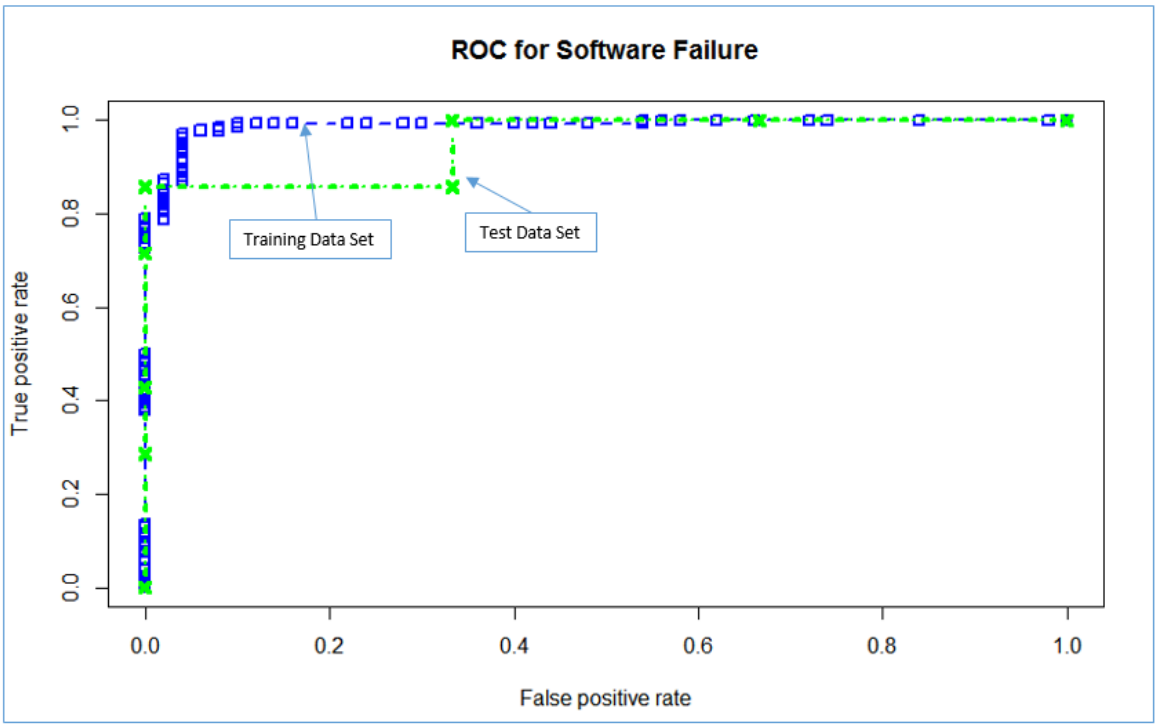


Figure 6: ROC Curve for Training Set and Test Set for Model A

```
HTML> mattrain
     obs
pred   0   1
   0 46   2
   1  4 130
attr(,"class")
[1] "confusion.matrix"
```

```
HTML> mattest
     obs
pred 0 1
   0 2 1
   1 1 6
attr(,"class")
[1] "confusion.matrix"
```

Figure 7: Confusion Matrix for Train and Test of Model A

The Confusion Matrices in Figure 7 demonstrate that the model correctly classifies project success or failure in the training set approximately 96 percent of the time as opposed to only 80 percent for the test set as identified in the proportions correct field of Figures 8 and 9 respectively. Additionally, the training set performs better for the Area Under Curve (AUC) calculations, the omission rate, sensitivity, specificity and Kappa compared to those

same values for the test set. Due to the degradation in performance of Predictive Model A when run with the test set, the researcher created a second Predictive Model B.

threshold	AUC	omission.rate	sensitivity	specificity	prop.correct	Kappa
0.5	0.9524242	0.01515152	0.9848485	0.92	0.967033	0.916232

Figure 8: Accuracy Measures for Training Data Set Model A

threshold	AUC	omission.rate	sensitivity	specificity	prop.correct	Kappa
0.5	0.7619048	0.1428571	0.8571429	0.6666667	0.8	0.5238095

Figure 9: Accuracy Measures for Test Data Set Model A

4.2 Analysis of Predictive Model B

Based on the low accuracy results of the test set for Model A, the researcher created Predictive Model B and used the hold-out validation set n=5 not used in the training of Model A to fine-tune the performance of the model. The researcher used the LiblineaR function in building Predictive Model B. LiblineaR allows estimation of predictive models for Regularized logistic regression to select the best factors leading to failure (Fan, Chang, Hsieh, Wang, & Lin., 2008). LiblineaR produces seven types of generalized linear models by combining several types of regularization schemes and loss functions:

- 0 – L2-regularized logistic regression
- 1 – L2-regularized L2-loss support vector classification (dual)
- 2 – L2-regularized L2-loss support vector classification (primal)
- 3 – L2-regularized L1-loss support vector classification (dual)
- 4 – multi-class support vector classification by Crammer and Singer
- 5 – L1-regularized L2-loss support vector classification
- 6 – L1-regularized logistic regression
- 7 – L2-regularized logistic regression (dual)

Regularizations prevent overfitting by eliminating the random error or noise and preserving instead the underlying relationship between the dependent and independent variables, especially because this research contained a small number of training examples. Overfitting occurs when a model has too many parameters relative to the number of observations. A model that has been overfitted may often exhibit poor predictive performance due to exaggeration of minor nuances in the data. Using regularization will allow Model B to scale to larger sample data sets in the future, with significant computational savings. Companies like Facebook, Google, Yahoo, and Microsoft use regularization--one can only imagine the vast amounts of data analyzed within each of these organization.

A drawback to regularization is that it may introduce a constraint cost penalty proportional to the size of every coefficient. The cost constraint is the trade-off between correct data classification and regularization. If this cost constraint is large, the model fit will tend to keep all failure factor parameters used. If a smaller cost constraint penalty is used, the end result may be the elimination of too many failure factors and, as a consequence, bias or poor fit in the model will result. The authors sought a solution to minimize the total cost from constraint violations. To account for the cost constraint violation risk in arbitrarily selecting one of the L1 or L2 regularization schemes, the authors went one step further to prevent bias by using the 10-fold cross-validation technique to identify the best regularized model with the best cost parameter, and the best accuracy prediction. In developing the model and running the program through different regularization cost parameters, best Type = [6] which corresponds to L1-regularized

logistic regression, best Cost = [100] for the cost constraint and best Accuracy = [0.956044] for the highest accuracy of all the model options were identified.

Predictive Model B was therefore developed to determine how well the results of the analysis will generalize to the independent validation set. Though it is very important to remove failure factors in the model that were not significant because they unnecessarily take up degrees of freedom without adding to predictive ability, Predictive Model B with L1 regularized logistic regression and cross-validation combined proved that all the failure factors were contributors to software project failure and were integral in accurately predicting software project outcomes.

The researchers tested Model B performance using the n = 5 validation data set and based on the results of the accuracy measures, confusion matrix, the ROC Curve, and AUC, Predictive Model B performed better than Model A, as demonstrated by Figures 10 through 14.

```
glm(formula = Failure ~ FFA + FFB + FFC + FFD + FFE + FFF + FFG +
    FFH + FFI + TestF, family = "binomial", data = traindata)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.77443  -0.06671   0.03867   0.11212   2.85520

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.4133     1.7115  -3.163  0.00156 **
FFA           2.8028     1.0260   2.732  0.00630 **
FFB           3.7279     1.1439   3.259  0.00112 **
FFC           0.6196     1.0059   0.616  0.53791
FFD           1.5375     1.0374   1.482  0.13834
FFE           2.6292     1.1466   2.293  0.02184 *
FFF          -3.8406     1.5269  -2.515  0.01189 *
FFG           0.5103     1.0123   0.504  0.61418
FFH           4.9002     1.1569   4.236 2.28e-05 ***
FFI          -0.6935     1.0813  -0.641  0.52130
TestF         2.9599     1.3222   2.239  0.02518 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 213.996  on 181  degrees of freedom
Residual deviance:  41.579  on 171  degrees of freedom
AIC: 63.579

Number of Fisher Scoring iterations: 8
```

Figure 10: Significance P-values and AIC for Model B

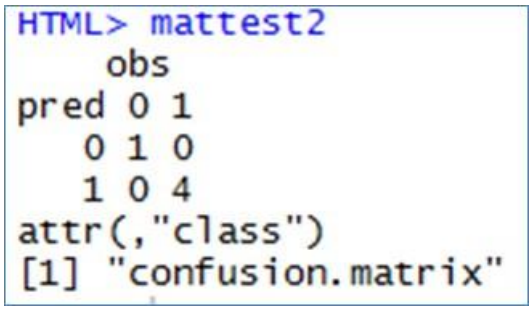


Figure 11: Confusion Matrix for Model B

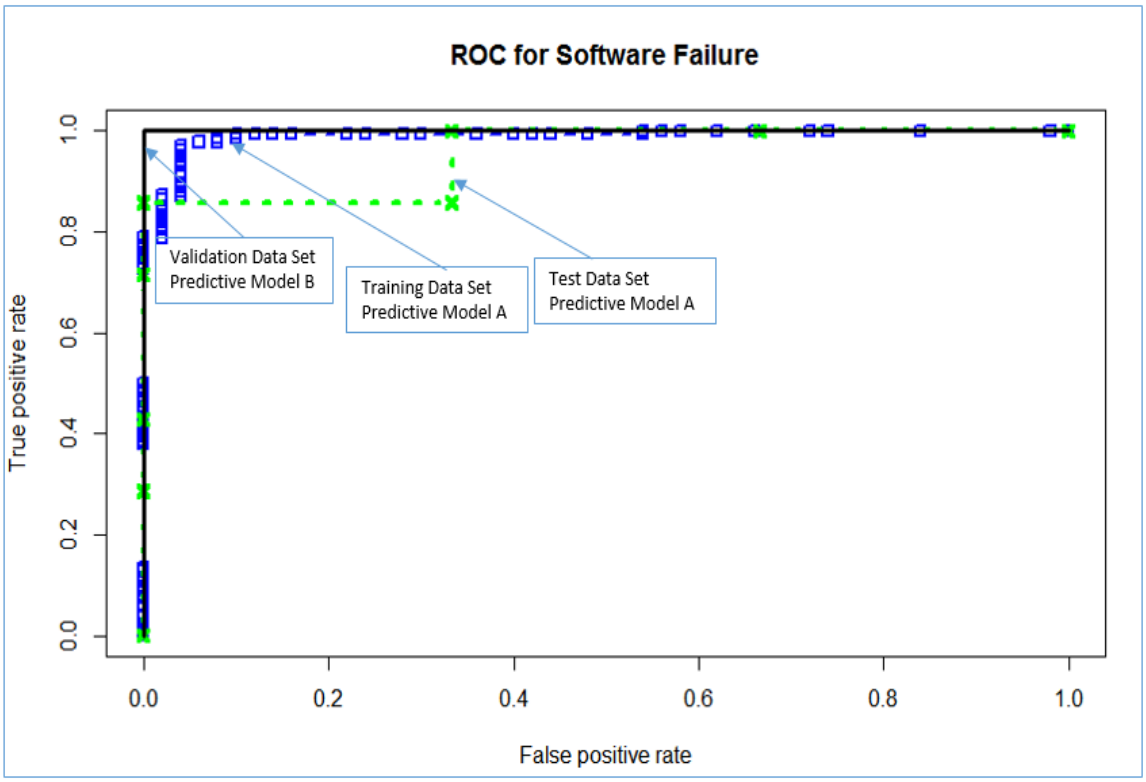


Figure 12: ROC Solid Line Represents Model B

	threshold	AUC	omission.rate	sensitivity	specificity	prop.correct	Kappa
1	0.5	1	0	1	1	1	1

Figure 13: Accuracy Measures for Model B

Actual Outcomes versus the Predictions for Predictive Model B are displayed in Figure 14 to confirm the model's good performance. As demonstrated by Figures 10 through 14, Predictive Model B correctly predicts each project outcome.

row.names	ProjectID	FFA	FFB	FFC	FFD	FFE	FFF	FFG	FFH	FFI	TestF	Failure
8	FP8	1	0	1	1	0	0	0	1	0	0	1
32	FP18	0	1	0	0	1	0	1	1	1	1	1
51	SP21	1	0	1	1	0	1	1	0	0	0	0
59	FP1	1	1	0	0	0	0	1	0	1	0	1
68	FP10	0	1	0	1	1	0	1	1	1	0	1

```
HTML> predReg
          8          32          51          59          68
0.98841950 0.99981969 0.02223404 0.71793497 0.99925268
```

Figure 14: Actual Outcomes vs Predictions for Predictive Model B

Neither regularization nor 10-fold cross validation are novel concepts in predictive analytics and logistic regression, however, the synergistic combination of the two methods identified a sound predictive model that can be used to predict software project failure in industry. This software product developed with the R free open-source tool is based on data of failed and successful software-intensive projects found in the federal government and commercial industries. The data was collected from all domains of industry, all sizes of organizations, and varying software project costs. As such the predictive model can be leveraged as an analytic tool in any of these domains to improve software project outcomes by making sound evidence-based decisions on the current development health of the software system and overall products developed within the organization in the past.

5.0 Research Objectives

The objective of this research was to develop a predictive tool capable of being integrated into current Systems Engineering Lifecycle processes. As such the researchers have identified areas in industry the model can be integrated.

5.1 Model Implementation in Industry

5.1.1 Model B Use on New Projects Not Yet Started

The authors propose that Predictive Model B be implemented in the systems engineering and software engineering processes at project inception and planning when decisions are being made about which projects to embark upon in the project portfolio. Managers would begin by collecting data of past projects developed within the project portfolio for the organization to determine how well past projects performed. If an organization is newly formed and developing their very first software product, the authors recommend collecting publicly available project performance data from competitors in the same market; similarly to how the data was collected for this research.

Once data has been collected from retrospective past software project analysis it would be entered into the Model B to provide a probability of failure score for the particular software project. The failure score categories were selected by analyzing the actual versus predicted failures to determine whether the failure threshold of 0.5 selected during the build stage of the models could be confirmed. Based on the results of Model A and the

exploratory analysis conducted in the development of Model B, the following failure probability break-out scores were identified and are described as follows:

- **Scores of 0 to 40%** indicate a healthy and mature organization with processes in place to mitigate risk and ensure, goals, objectives and requirements are clearly defined during planning and inception and are monitored throughout the software development stages. This organization develops robust and successful software systems that are deployed on time, on budget and satisfy user requirements. There are no perfect software projects, therefore probability scores within individual factors tracking near failure should be documented as risks and monitored and controlled throughout the project lifecycle.
- **Scores 40.01 to 50%** indicate a medium probability of failure. The potential risks to the program as identified by the model should be documented and risk remediation strategies must be developed and utilized throughout the lifecycle if the decision makers decided to proceed with the project. Process improvement efforts would be implemented within the phase or phases that posed the most risk to the health of the software system.
- **Scores 50.01 to 70%** indicate a high probability of failure. The risks to the program as identified by the model should be documented and risk remediation strategies must be developed and utilized throughout the lifecycle. Strict monitoring and control as well as regular audits of the program should be performed throughout the lifecycle. Process improvement efforts must be implemented within the phase or phases that posed the most risk to the health of the software system and tracked throughout the lifecycle.

- **Scores 70.01 to 100%** indicate a severe probability of failure of projects within that portfolio. Decision makers should proceed with great caution as the stakes are high and future failure is imminent. The risks to the program as identified by the model should be documented and risk remediation strategies must be developed and utilized throughout the lifecycle. Strict monitoring and control as well as regular audits of the program should be performed throughout the lifecycle. Process improvement efforts must be implemented within the phase or phases that posed the most risk to the health of the software system and tracked throughout the lifecycle. These are programs the research would recommend not proceeding with and replace with others that would be more beneficial to the overall health of the organization and its software project portfolio.

5.1.2 Model B Use on Software Projects in Flight

Though it is recommended the model be used before project inception, the model is an excellent tool to measure the health of a software project during any stage of the software development life cycle to determine the present and future health of the software project. The model would be a tool to identify the current state of the project and to determine the improvements to undertake to ensure success. A single project data point at the current stage in the development life cycle would provide the input to the model. The results would be interpreted as follows:

- **Scores of 0 to 40%** indicate a healthy software project on track to successful completion with high quality, on time and on budget. However, there are no perfect software projects, therefore probability scores within individual factors tracking

near failure should be documented as risks and monitored and controlled throughout the project lifecycle.

- **Scores 40.01 to 50%** indicate a medium probability of failure. The potential risks to the program as identified by the model should be documented and risk remediation strategies must be developed and utilized throughout the remaining lifecycle. Process improvement efforts would be implemented within the phase or phases that posed the most risk to the health of the software system with continuous monitoring and audits through continued use of the model.

- **Scores 50.01 to 70%** indicate a high probability of failure. The risks to the program as identified by the model should be documented and risk remediation strategies must be developed and utilized throughout the remaining lifecycle stages. Strict monitoring and control as well as regular audits of the program should be performed throughout the lifecycle by continued use of the model. Process improvement efforts must be implemented within the phase or phases that posed the most risk to the health of the software system and tracked throughout the lifecycle. If the project is identified with a high probability of failure in the development or testing phases, decision makers should weigh the options to potentially cancel the project and reallocate the resources to other more viable software projects identified by the model.

- **Scores 70.01 to 100%** indicate a severely troubled software project that is doomed for failure if risk mitigation strategies are not applied as soon as possible. Decision makers should proceed with great caution as the stakes are high and future failure is imminent. Strict monitoring and control as well as regular audits of the program

should be performed throughout the remaining lifecycle stages by continued use of the model. Process improvement efforts must be implemented within the phase or phases that posed the most risk to the health of the software system and tracked throughout the remaining lifecycle stages. These are software projects the research would recommend not proceeding with and implement others that would be more beneficial to the overall health of the organization and its software project portfolio.

5.2 Predictive Model Deployment

Ineffective Executive governance coupled with ineffective project management allows software projects to continue development at staggering rates of failure. Insight and valuable information of troubled projects is not being communicated up stream in a way that decision makers can understand and take action. The literature states that organizations do not have adequate processes in place for dealing with troubled software projects and a majority wait until the project has missed lagging indicators of time and budget targets before taking action despite many executives' knowledge that early intervention on troubled projects would facilitate better management of limited resources.

A review of some of the tools in the research identified mechanisms to predict and estimate the likelihood of software project failure and specific project management process to implement to mitigate failure. However, the research did not identify concrete examples to make predictive tools and their use a mainstream requirement to improve software project outcomes.

5.2.1 The Model as a Legislative Tool

As a legislative tool, the researcher proposes levying a requirement on all federal government projects to conduct this analysis during project planning and throughout the lifecycle and submit results to the Government Accountability Board or other government authority for review prior to moving to the next stage of the software lifecycle. DoDI Directive 5000.02 (2015) states, "The Defense Acquisition System exists to manage the Nation's investments in technologies, programs, and product support necessary to achieve the National Security Strategy and support the United States Armed Forces. In that context, our objective is to acquire quality products that satisfy user needs with measurable improvements to mission capability at a fair and reasonable price." The software intensive aspects of Major Defense Acquisition programs (MDAP) and Major Automated Information Systems (MAIS) programs carry the greatest risk consequences in terms of management level, reporting requirements and documentation and analysis to support program decisions. The model will assist the Major Decision Authority (MDA) and supporting staff organizations at program decision reviews using evidence-based data derived from the model to facilitate the examination of the system to allow the MDA to decide whether a program is ready to proceed to the next milestone.

The research identified several segments of the directive in which the model and its subsequent probability results can be recommended for use. In depiction of the generic acquisition milestone and decision points in Figure 15, the directive states that in practice all decisions must be made prior to Request for Proposal (RFP) release, however for DoD, the Development RFP Release Decision Point is where plans for the programs must be carefully scrutinized to ensure all risks are identified, understood and are under control. It is the critical decision point in acquisitions and determines if "the program will either

successfully lead to a fielded capability or fail, based on the soundness of the capability requirements, the affordability of the program, and the executability of the acquisitions strategy" (DoD, 2015). The strategic incorporation of the model results will be beneficial in meeting the following objectives within the directive, as depicted in Figures 15 and 16:

1. To ensure the program plan is sound, its objectives are clearly understand and the program will be successfully executable and remain within budget prior to releasing the RFP in section 6 of DoDI 5000.02 Directive.

2. To ensure contractors have a track record of delivering quality software systems that meet the user needs, within cost and schedule parameters by including language and justification for them to provide the results of the model to show past performance in their responses as a qualification criteria.

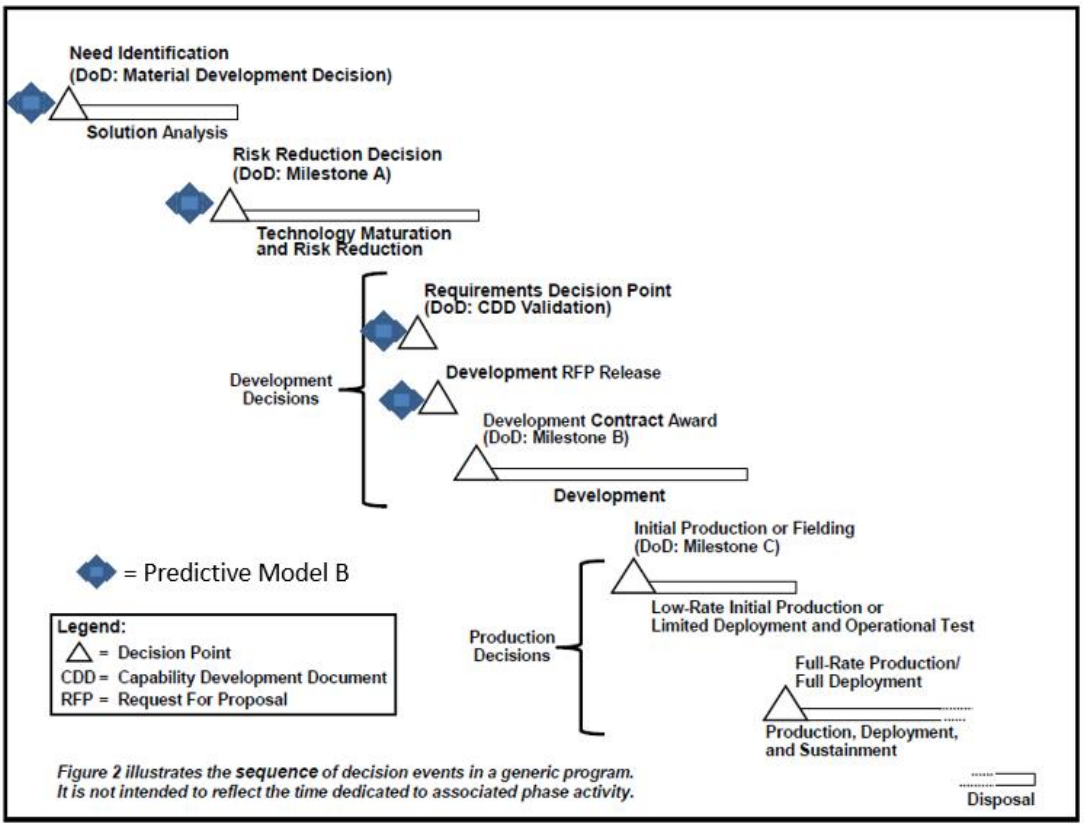


Figure 15: Generic Acquisition Phases and Decision Points in Predictive Model B

Incorporation of Predictive Model B in DoDI Directive 5000.02 Operation of the Defense Acquisition System in determining if the program is viable and whether contractors are fully capable of delivering a successful system within scope, cost, quality, and schedule.

3. To perform periodic reviews of the program in progress and identify opportunities for adjustment, improvement or redirection by the Configuration Steering Boards in paragraph 5d(5)(b). Performance data will be identified by the model to inform the RFP Release Decision point and throughout the project life cycle.

4. To assess the performance of incrementally deployed Software Intensive Programs. The development process in Figure 16 describes systems that are deployed in increments of 1-2 year cycles. The use of the model will determine whether or not each increment is a fully functional new capability. The model will assess each past

deployed increment to predict the likelihood of success for the succeeding increment and to identify and mitigate any risks that might impede success.

5. To regularly check the health of high risk projects and report their health to Congress and designated decision makers. A GAO Report (2006) describing High Risk IT projects provided the recommendation that the Director of the Office of Management and Budget (OMB) establish a process for government agencies to update high risk projects on a regular basis. The model would be an invaluable tool in that health assessment.

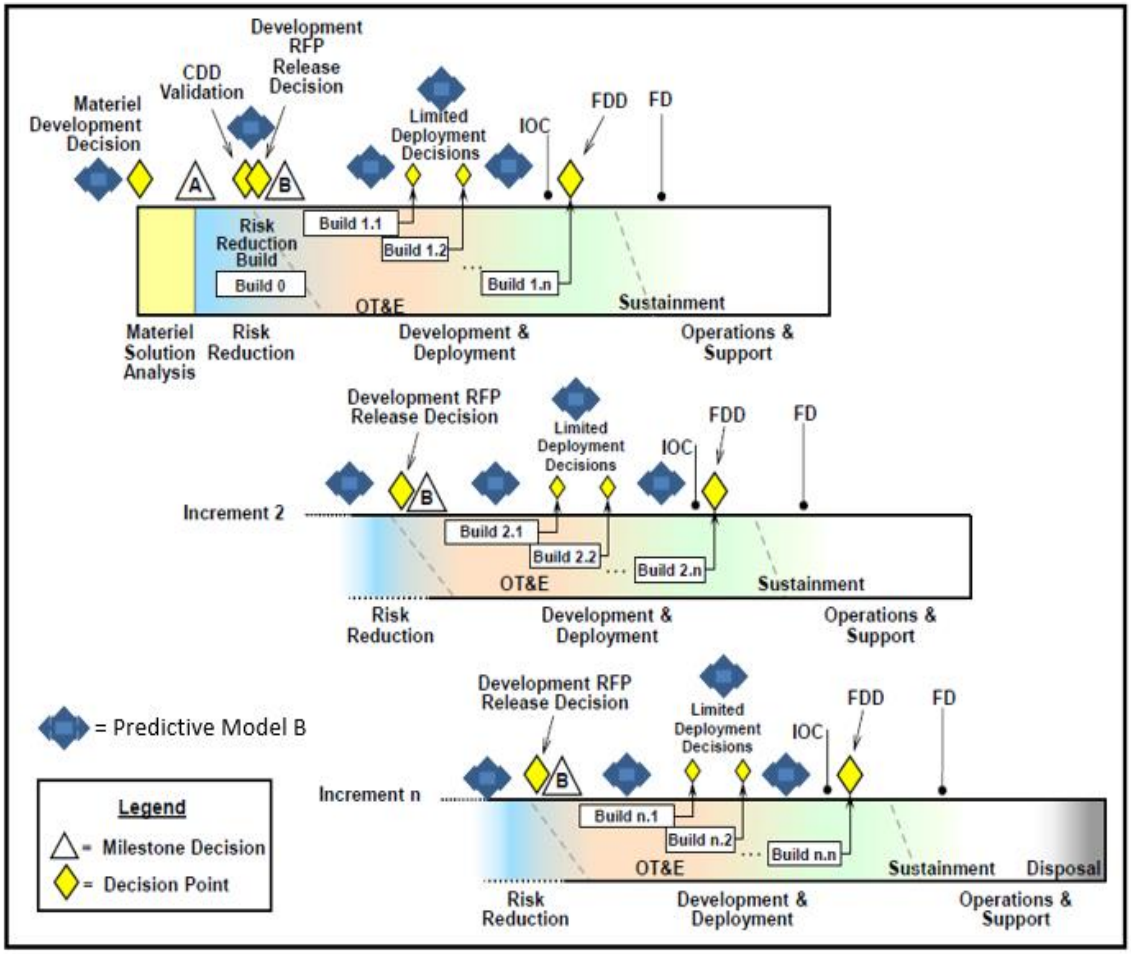


Figure 16: Incrementally Deployed Software-Intensive Program Milestone Decisions and Decision Points

Incorporation of Predictive Model B in DoDI Directive 5000.02 Operation of the Defense Acquisition System to inform these critical software project decisions in determining project cancellation or continuation.

Another example of the model's legislative integration is within the Clinger-Cohen Act (CCA), enacted by Congress in 1996 to reform and improve the way Federal agencies acquire and manage IT resource. The act was mandated by senior government officials in the implementation of strategies to control development risks, to better manage IT spending and succeed in achieving measureable improvements in software project performance (US Department of the Interior, 2015). The use of the model as a predictive tool may be considered a required aspect of CCA Compliance at the selected Decision gates in the software engineering life cycle for federal government agencies. CCA Regulatory compliance is an aspect of the DoD 5000.02 Directive where noncompliance is reported to Congress. The use of the Model and the generated probability of failure score for a program would better define the areas of noncompliance and the extent to which the software project has deviated to inform an evidence-based decision to cancel or to proceed with strict risk mitigation strategies.

5.2.2 The Model as a Procurement Decision Tool

The authors recommend the Model be used as procurement decision tool requiring this analysis and its results in the Request for Proposal (RFP) for all contractors bidding on software procurement contracts. The tool can be used as a forcing function requiring contractors to list out all work performed and their outcome. The tool would facilitate transparency and provide a probability of success score for each contractor. This will allow

decision makers the ability to compare past performance with quantitative results to make evidence based decisions on contractor selection.

The Federal Acquisition Regulation (FAR) governs the acquisition process by which the federal government acquires services and goods to regulate government personnel activities in carrying out that process. This tool can be embedded in the FAR (2014) Part 9 Contractor Qualifications. Subpart 9.2(a)(1) Qualification Requirements states the policy in carrying this out, "The head of the agency or designee shall, before establishing a qualification requirement, prepare a written justification". The justification would be that software projects are failing at staggering rates. Despite rigid controls, project management and systems engineering rigor, software project failures still persist. The qualification requirement would consist of using the Predictive Model B to assess the overall health of the contractors' software development programs to maximize competition and to ensure the federal government achieves the highest possible value for software products and services.

5.2.3 The Model as a Systems Engineering, Software Engineering, and Project Management Enabling Tool

Systems engineering is an interdisciplinary approach enabling the realization of successful systems by defining customer needs and required functionality early in the development cycle. systems engineering considers both the business and technical needs of all stakeholders with the goal of providing a quality product that meets user needs, within cost, schedule, scope and quality parameters; all while minimizing undesirable consequences. This can be accomplished through the inclusion of and contributions from experts across relevant disciplines and coordinated by the systems engineer. Systems

engineers are commissioned to explore issues and undesirable consequences to make critical decisions expeditiously (INCOSE, 2010).

According to IEEE, software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software. Software engineering can also be defined as the study of these approaches or the application of engineering to software. Software engineering's primary objective is producing programs that meet specifications and are produced on time and within budget.

Project Management is the application of a collection of techniques, tools, and processes to manage and direct the use of resources in the successful execution of a complex, unique, one-time task with constraints of scope, quality, cost, and time. Each task calls for a unique set of tools and techniques specific to the task environment and its life cycle (PMI, 2015). The underlying theme of the three disciplines is the development of successful projects on time, on budget, within scope and quality parameters. According to the Systems Engineering Handbook, schedule and cost overruns are lessened with increased systems engineering rigor. The handbook went on to note that cost and schedule overrun predictions are very difficult to gauge with low systems engineering effort. However, the staggering rate of software project failure indicates that more needs to be done. The authors propose the incorporation of the model in key decision processes and areas not just of systems engineering, but within software engineering and project management as well.

Why should an organization care about analytics, systems engineering, software engineering, and project management? To borrow a phrase from the SE Handbook, "to better understand, evaluate, control, learn, communicate, improve, predict, and certify the

work performed." Figure 17 identifies the overlaps in the three disciplines and the areas where the predictive tool would be integrated to predict issues and facilitate decision making. For the purposes of this research the researcher proposes incorporating the Predictive Model within the IEC/ISO/IEE 15288 and 12207 guidance as well as the PMI process and knowledge areas. The model as a tool would assist stakeholders within each discipline to collaborate one with the other in achieving the goal of delivering a successful software system. The model would allow full transparency amongst those responsible for a project's success. Project outcome probability scores would be made available during project acquisition and procurement in making source selections where the past performance would drive the identification of key processes and mitigation strategies to implement to ensure success. The model would be used in project planning to identify areas of concern and as an assessment and control process tool to ensure the project is on track. The model would be incorporated into the software implementation process to monitor its development and to make critical decisions at designated milestone gates throughout the lifecycle. The model would be used within the software audit and review process to ensure the user needs are being met within budget, schedule and quality constraints. The model can be leveraged in the problem identification and risk management where key failure factors are documented and mitigation strategies are developed and tracked to ensure a successful outcome.

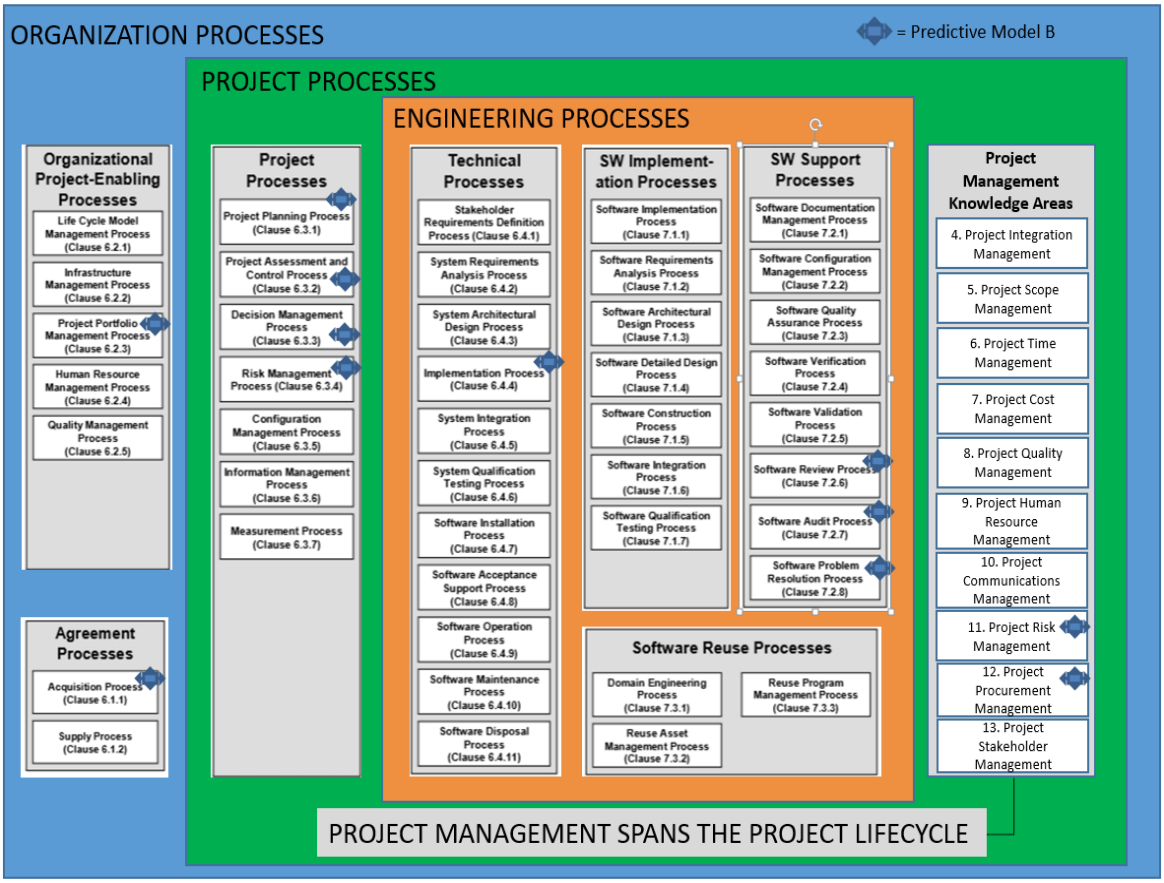


Figure 17: Project Management

ISO/IEC/IEEE 15288 and 12207 Systems and Software Life Cycle Processes Overlay and Predictive Model Incorporation

"[The] growth of complexity is accelerating and we need tools to influence successful projects. How do we use modelling to take on a systems engineering challenge that is quantitatively different from the past?" (Wade, 2014) One answer, and the focus of this research, is the emerging technology of predictive analytics that uses data and information to predict future software project outcomes. Effective implementation using a select group of synergistic system engineering tools, practices and processes requires vision. This research can help to turn the tide on project failure to push software development projects toward success by utilizing good systems engineering practices. The output of the model

can be used to guide the project team to ensure that all stakeholders have the same vision and drive toward successful project outcome.

6.0 Conclusion

6.1 Research Overview

This research will assist in carrying out quantifiable Systems Engineering that aids in making informed decisions. During the Introduction of the SE Vision 2025 at the INCOSE IS in 2014, a panel member asked the key questions that all decision makers and project stakeholders must ask during project planning: "How do you measure value? What are the differences between success and failure?" (Wade, 2014) This same panel member immediately followed those questions with a crucial response: "Organizations that have data do analysis to make decisions in a decisive way" (Wade, 2014). The results of the model and ultimately this research are to assist decision makers and stakeholders in developing mitigation strategies in their systems and software engineering programs to prevent project failure by implementing risk reduction strategies that decrease the occurrence of defects in the system. Complete defect prevention is not realistic, given the nature of software and the increasing complexity of systems. However, a case study evaluating project decisions sums up this research by stating, "Effective decisions are crucial to the success of any software project, but to make better decisions you need a better decision-making process to systematically evaluate portfolio decisions and avoid the bad choices that lead to project failure" (Hoover, Rosso-Llopart, & Taran, 2010).

This research demonstrates that there is no one silver bullet that will solve the problem of software project failure (Brooks, 1995). In fact, a combination of tools, techniques and processes are required to improve software project outcomes. Systems engineers, software engineers, and project managers need information for planning, estimating, and tracking

project work; the Predictive Model can be a support mechanism within software development. Information is needed to project the future, to educate, and to communicate, for identifying and resolving problems, and, finally, for making evidence-based decisions in software programs. The authors have demonstrated a predictive tool that can be deployed across many domains with the one clear objective of delivering successful software projects on time, on budget, and of the highest quality. It is a tool to help stir legislation to adopt policies in software engineering to facilitate and mandate disclosure of past and present software project performance. It is a tool to assist in effective acquisition and procurement to acquire knowledgeable, capable, and success-driven resources with the track record of achieving software project success. It is a tool that is easily assimilated into the systems engineering, software engineering, and project managers' current toolbox to bind the three disciplines under one common goal, agenda, and objective: to deliver successful software.

6.2 Future Work

The research for this paper did have some limitations, including the relatively small data set and selection bias in the research data. However, the information sources were very reliable in that the bulk of the research consisted of failure and success data extracted from GAO reports, court litigation cases, and Government Investigation Board Reports. The researcher would like to expand the predictive model to measure its performance on large project databases such as the Standish Chaos database, which consists of 50,000 projects (The Standish Group, 2013). Additionally the researcher would like to drill down into each of the Software Lifecycle elements of Requirements, Design, Implementation, and Testing

in order to identify the key co-factors contributing to failure at those levels to determine the cost and schedule tradeoffs. These are likely to include:

1. In the Requirements stage: Predict how many times requirements will change and the overall impact on project quality. What is the maximum or minimum number of change requests for requirements that will allow a project to be successfully completed and satisfy all requirements?
2. In the Design stage: How many defects must be identified and removed in the design stage of a software project to predict failed or successful outcomes. How many would move the planned project to success. What is the positive predicted cost and schedule impact of finding defects at this stage as opposed to finding them in later stages?
3. In the Development stage: How many defects per line of code (LOC) will be found in a planned software project based on past project failures. What is the threshold that steers projects to failure?
4. In the Testing stage: Predict how many severity level (1-4) defects will be found in the testing phase of the planned software project and devise ways to mitigate them in the planning and design phase.

Bibliography

AcqNotes. (n.d.). *Contracts & Legal: Clinger-Cohen Act*. Retrieved from AcqNotes: A simple Source of DoD Acquisition Knowledge for the Aerospace Industry: http://acqnotes.com/acqnote/careerfields/clinger-cohen-act

Alfons, A. (2012). *Package 'cvTools': Cross-validation tools for Regression models.* CRAN.

Arandjelovic, P., Bulin, L., & Khan, N. (2015, February). *Why CIOs Should Be Business-Strategy Partners*. Retrieved from McKinsey and Company Insights and Publications: http://www.mckinsey.com/insights/business_technology/why_CIOs_should_be_business-strategy_partners?cid=other-eml-alt-mip-mck-oth-1502

Balboni, F., Finch, G., Reese, C. R., & Shockley, R. (2013). *Analytics: A Blue Print for Value: Converting Big Data and Analytics Insights into Results.*

Baldridge. (2012). *Baldridge Core Values-Management by Fact*. Retrieved from Baldridge.

Boehm, B. (1991). Software Risk Management: Principles and Practices. *IEEE Software*, 32-41.

Bronte-Stewart, M. (2009). Risk Estimation From Technology Project Failure. *Fourth European Conference on Management of Technology*, (pp. 1-19). Glasgow.

Brooks, F. (1995). *The Mythical Man-Month.* Addison-Wesley.

Cerpa, N., Bardeen, M., Kitchenham, B., & Verner, J. (2010). Evaluating Logistic Regression Models to Estimate Software Project Outcomes. *Information and Software Technology*, 934-944.

CIO Council. (2011). *A Year in Review: Outcomes and Lessons Learned from Implementing Agency-Led TechStat Reviews Across the Federal Government.* Washington, DC: Management Best Practices Committee.

CIO Council. (2012). *2012 Clinger-Cohen Core Competencies and Learning Objectives.* Washington, DC.

CIO Council. (2015, September). *PortfolioState*. Retrieved from CIO.gov: https://cio.gov/drivingvalue/portfoliostat/

CIO.gov. (2015, August). *TechStat*. Retrieved from CIO.gov: https://cio.gov/drivingvalue/techstat/

Dey, S., Jacob, K., Lopez, J., & Trivedi, K. (2013). Failure Data Analytics to build Failure Prediction Mechanisms. *2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 102-103). IEEE.

DoD. (2015). *DoDI Directive 5000.02: Operation of the Defense Acquisition System.* Washington, D.C.: USD(AT&L).

Eisner, H. (2005). *Managing Complex Systems: Thinking Outside the Box.* Hoboken: John Wiley & Sons.

Eisner, H. (2011). *Systems Engineering: Building Successful Systems.* Morgan & Claypool.

Everitt, B., & Hothorn, T. (2006). *A Handbook of Statistical Analyses Using R.* Boca Raton: Taylor & Francis Group.

Ewusi-Mensah, K. (2003). *Software Development Failures: Anatomy of Abandoned Projects.* Cambridge: MIT Press.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin., C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 1871-1874.

Fitzgerald, M. (2014, August 14). *The Four Traps of Predictive Analytics*. Retrieved from MIT Sload Managment Review: http://sloanreview.mit.edu/article/the-four-traps-of-predictive-analytics/

Flowers, S. (1997). *Software Failures: Management Failure: Amazing Stories and Cautionary Tales.* London: John Wiley & Sons.

Frankfort-Nichmias, C., & Nachmias, D. (2000). *Research Methods in the Social Sciences.* New York, NY: Worth Publishers.

GA0-14-705T. (2014). *Patient Protection and Affordable Care Act: Preliminary Results of Undercover Testing of Enrollment Controls for Health Care Coverage and Consumer Subsidies Provided Under the Act, GA0-14-705T.* Washington, DC: Government Accountability Office.

GAO. (2006). *Information Technology: Agencies and OMB Should Strengthen Process for Identifying and Overseeing High Risk Projects, GAO-06-647.* Washington, D.C.: Government Accountability Office.

GAO. (2009). *Polar-Orbiting Environmental Satellites, GAO-09-564.* Washington, DC: Government Accountability Office.

GAO-06-647. (2006). *Information Technology: Agencies and OMB Should Strengthen Process for Identifying and Overseeing High Risk Projects.* Washington, D.C.: Government Accountability Office.

GAO-09-564. (2009). *Polar-Orbiting Environmental Satellites.* Washington, DC: Government Accountability Office.

GAO-10-158. (2010). *Secure Border Initiatives: DHS Needs to Address Testing and Performance Limitations That Place Key Technology Program at Risk.* Washington, D.C.: Government Accountability Office.

GAO-11-634. (2011). *Federal Chief Information Officers: Opportunities Exist to Improve Role in Information Technology Management.* Washington, DC: Government Accountability Office.

GAO-12-7. (2011). *Information Technology: Critical Factors Underlying Successful Major Acquisitions, GAO-12-7.* Washington, D.C.: Government Accountability Office.

GAO-12-7. (2011). *Information Technology: Critical Factors Underlying Successful Major Acquisitions, GAO-12-7.* Washington, D.C.: Government Accountability Office.

GAO-13-524. (2013). *Information Technology: Additional Executive Review Sessions Needed to Address Troubled Projects.* Washington, DC: United States Government Accountability Office.

GAO-15-675T. (2015). *Information Technology: Additional Actions and Oversight Urgently Needed to Reduce Waste and Improve Performance in Acquisitions and Operations.* Washington, DC: Government Accountability Office.

GAO-IMTEC-92-26. (1992). *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia.* Washington D.C.: Government Accountability Office.

Hawkins, D. (2004). The Problem of Overfitting. *Chem Information Computer Science*, 1-12.

Higgins, J., & Green, S. (. (2011, March). Cochrane Handbook for Systematic Reviews of Interventions. The Cochrane Collaboration. Retrieved from http://handbook.cochrane.org/chapter_12/12_4_2_p_values_and_statistical_significance.htm

Hilborn, R., & Mangel, M. (1997). *The Ecological Detective: Confronting Models with Data.* Princeton: Princeton University Press.

Hitchins, D. K. (2003). *Advanced Systems Thinking, Engineering, and Management.* Norwood: Artech House.

Hoover, C., Rosso-Llopart, M., & Taran, G. (2010). *Evaluating Project Decisions: Case Studies in Software Engineering.* Upper Saddle River: Addison-Wesley.

Horne, A. (2013, November). *How to Spot a Troubled Project*. Retrieved from CIO Leadership Council: https://www.executiveboard.com/member/cio/blog/13/how-to-spot-a-nascent-healthcare-gov-in-your-project-queue.html?referrerTitle=Search%20-%20CEB%20CIO%20Leadership%20Council

Hubbard, D. (2014). *How to Measure Anything: Finding the Value of "Intangibles" in Business.* Hoboken: John Wiley & Sons.

IEEE/ANSI. (2000). IEEE 1471-2000. New York. Retrieved from Recommended Practice for Architectural Description for Software-Intensive Systems: http://standards.ieee.org/findstds/standard/1471-2000.html

INCOSE. (2010). *Systems Engineering Handbook* (Version 3 ed.). Seattle, WA.

INCOSE. (2014). A World in Motion: Systems Engineering Vision 2025.

Institute for Defense Analysis. (2011). *Expeditionary Combat Support System: Root Cause Analysis.* Alexandria, VA: Institute for Defense Analysis.

ISO/IEC/IEEE. (2008). 12207. *Systems and Software Engineering: Software Life Cycle Processes*.

ISO/IEC/IEEE. (2008). 15288 . *Systems and Software Engineering: Systems Life Cycle Processes*.

Jones, C. (1995). Patterns of Large Software Systems: Failure and Success. *IEEE Computer*, 86-87.

K. C. (1997).

Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 1-26.

Lattin, J., Carroll, J., & Green, P. (2003). *Analyzing Multivariate Data.* Belmont: Brooks/Cole.

Lehtinen, T., Mantyla, M., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived Cause of Software Project Failures - An Analysis of Their Relationships. *Information and Software Technology*, 623-643.

Lunden, I. (2013, July 15). *Forrester: $2.1 Trillion Will Go Into IT Spending in 2013: Apps and the U.S. Lead The Charge*. Retrieved from TechCrunch: techcrunch.com

Ma, N. L., Khataniar, S., & Wu, D. N. (2014). Predictive Analytics and Patient no Shows IEEE Publication. *2014 International Conference on Information Science and Application (ICISA)* (pp. 1-4). IEEE.

Madni, A., & Sievers, M. (2014). Systems Integration: Key Perspectives, Experiences, and Challenges. *Systems Engineering: The Journal of The International Council on Systems Engineering, 17*(1), 37-51.

Meritalk. (2015). *The Federal IT Acquisition Reform Act (FITARA)*. Retrieved from The Federal IT Acquisition Reform Act (FITARA): http://www.meritalk.com/pdfs/FITARA_Overview.pdf

Moore, J. W. (2006). *The Road Map to Software Engineering: A Standards-Based Guide.* Hoboken: John Wiley/IEEE Computer Society Press.

Mori, T., Tamura, S., & Kakui, S. (2013). Incremental Estimation of Project Failure Risk with Naive Bayes Classifier. *IEEE International Symposium on Emperical Software Engineering and Measurement* (pp. 283-286). IEEE.

NASA. (1999). *Mars Climate Orbiter Mishap Investigation Board Phase I Report.* NASA.

NASA. (2010). Systems Engineering Handbook. DIANE Publishing.

Naur, P., & Randell, B. (1969). *Software Engineering: Report on Conference Sponsored by NATO Science Committee.* Garmish: NATO Science Committee.

NIST. (2002). *The Economic Impacts of Inadequate Infrastructure for Software Testing.* National Institute of Standards and Technology.

OMB. (2012). *Memorandum for the Heads of Executive Department Agencies: Implementing PortfolioStat.* Washington, DC.

OMB. (2015, August). *IT Dashboard FY2016 Edition*. Retrieved from IT Dashboard: https://itdashboard.gov/

OMB. (2015, June 10). *Office of Management and Budget: Management and Oversight of Federal Information technology*. Retrieved from OMB Memorandum M-15-14: Management and Oversight of Federal Information Technology: https://management.cio.gov/#omb-memorandum-m-15-14-management-and-oversight-of-federal-information-technology

Oppenheim, B. (2014, July 1). Presentation to Lean Systems Engineering Working Group. INCOSE International Symposium.

Patanakul, P., & Omar, S. S. (2010). Why Mega IS/IT Projects Fail: Major Problems and What We Learned from Them. *2010 Technology Management for Global Economic Growth (PICMET)* (pp. 1-12). Phuket: IEEE.

Pettey, C., & van der Muelen, R. (2013, January 16). *Gartner Newsroom*. Retrieved from Gartner Executive Program Survey of More Than 2,000 CIOs Shows Digital Technologies Are Top Priorities in 2013: http://www.gartner.com/newsroom/id/2304615

PMI. (2015). *What is Project Management?* Retrieved from Project Management Institute: http://www.pmi.org/About-Us/About-Us-What-is-Project-Management.aspx

R.-D. T. (2011). *Package 'stats': The R Stats Package.*

Ravindranath, M. (2015, April 30). *Nextgov*. Retrieved from OMB Unviels FITARA Guidelines for Public Comment: http://www.nextgov.com/cio-briefing/2015/04/omb-releases-fitara-guidelines-public-comment/111520/

Reyes, F., Cerpa, N., Candia-Vejar, A., & Bardeen, M. (2011). The Optimization of Success Probability for Software Projects Using Genetic Algorithms. *Journal of Systems and Sofware*, 775-785.

Richardson, J., & Gwaltey, W. (2007). *Ship It! A Practical Guide to Successful Software Projects.* Raleigh: The Pragmatic Bookshelf.

Ring, J. (2014, June 30). Systems Science is Fundamental: How Systems Science can Help Improve SE Practices. (D. J. Martin, Interviewer) Henderson, NV: INCOSE International Symposium.

Robbins-Gioia. (2001).

Roedler, G., Rhodes, D., Schimmoller, H., & Jones, C. (2010). *Systems Engineering Leading Indicators Guide.* Seatle: INCOSE.

Roy, G. (2013). Prototype Development of Dynamic Predictive Models for Disease Prevention. *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1681-1685.

Ryan, J., Sarkani, S., & Mazzuchi, T. (2014). Leveraging Variability Modeling Techniques for Architecture Trade Studies and Analysis. *Systems Engineering: The Journal of INCOSE, 17*(1), 10-25.

Sakamoto, Y., Ishiguro, M., & Kitagawa, G. (1986). *Akaike Information Criterion Statistics.* D. Reidel Publishing Company.

Selby, R., & Boehm, B. (2007). *Software Engineering: Barry W. Boehm's Contributions to Software Development, Management and Research.* Hoboken: John Wiley & Sons.

Singer, J. (2014, June 30). Systems Science is Fundamental:How Systems Science can Help Improve SE Practices. (D. J. Martin, Interviewer) INCOSE International Symposium.

Stevens, R. (2011). *Engineering Mega-Systems: The Challenge of Systems Engineering in the Information Age.* Boca Raton: Taylor & Francis Group.

SWEBOK. (2004). Guide to the Software Engineering Body of Knowledge (SWEBOK). Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc (IEEE).

Takagi, Y., Mizuno, O., & Kikuno, T. (2005). An Empirical Approach to Characterizing Risky Software Projects Based on Logistic Regression Analysis. *Empirical Software Engineering, 10*(4), 495-515.

Tan, S. (2011). *How to Increase Your IT Project Success Rate.* Gartner Research.

Tape, T. M. (2014, September 30). *Plotting and Intrepretating an ROC Curve*. Retrieved from Interpreting Diagnostic Tests: http://gim.unmc.edu/dxtests/roc2.htm

The Conference Board Survey. (2001).

The Standish Group. (2013). *The Chaos Manifesto.* The Standish Group.

Tsao, Y.-C., Hsu, K., & Tsai, T.-T. (2012). Using Content Analysis to Analyze the Trend of Information Technology Toward the Academic Researchers at The Design Departments of University of Taiwan. *2012 2nd International Conference on Consumer Electronics* (pp. 3691-3694). Yichang: IEEE.

US Department of the Interior. (2015, September 4). *What is Clinger-Cohen*. Retrieved from Indian Affairs: http://www.bia.gov/WhoWeAre/AS-IA/OCIO/ClingerCohen/index.htm

Verner, J., Sampson, J., & Cerpa, N. (2008). What Factors Lead to Software Project Failure? *International Conference on Research Challenges in Information Science (RCIS)*, (pp. 71-80).

Wade, D. J. (2014, July 3). Introduction to the SE Vision 2025. (P. Martin, Interviewer) Henderson, NV: INCOSE International Symposium.

Walls, C. (2006). *Embedded Software.* Amsterdam: Elsevier/Newnes.

Wang, X., Wu, C., & Ma, L. (2010). Sofware Project Schedule Variance Prediction Using Bayesian Network. *2010 IEEE International Conference on Advanced Management Science (ICAMS)* (pp. 26-30). Chengdu: IEEE.

Wessler, M. (2014). Predictive Analytics for Dummies. Hoboken: John Wiley & Sons.

White, N. (2012, May 8). *A Five Stage Approach to Predictive Analytics*. Retrieved from CIO Journal: http://deloitte.wsj.com/cio/2012/05/08/a-five-stage-approach-to-predictive-project-analytics/