NPS-AM-10-157

ACQUISITION RESEARCH SPONSORED REPORT SERIES

Analysis, Design, and Prototyping of Accounting Software for Navy Signal Intelligence Collection Systems Return on Investment Reporting

6 January 2011

By

Capt. James M. Torres, USA, and

Lt. Charles C. Spivey III, USN

Advisors: Dr. Thomas J. Housel, Professor, and

Dr. Man-Tak Shing, Professor

Graduate School of Operational and Information Sciences

Naval Postgraduate School

Prepared for: Naval Postgraduate School, Monterey, California 93943

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Chair of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request Defense Acquisition Research or to become a research sponsor, please contact:

NPS Acquisition Research Program
Attn: James B. Greene, RADM, USN, (Ret)
Acquisition Chair
Graduate School of Business and Public Policy
Naval Postgraduate School
555 Dyer Road, Room 332
Monterey, CA 93943-5103
Tel: (831) 656-2092
Fax: (831) 656-2253
e-mail: jbgreene@nps.edu

Copies of the Acquisition Sponsored Research Reports may be printed from our website www.acquisitionresearch.org

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

ABSTRACT

Prior research studies have demonstrated a practical methodology for quantifying the return on investment (ROI) of Navy Cryptologic Carry-on Program (CCOP) signals intelligence (SIGINT) collection systems and a practical pathway for implementing a performance accounting system that generates these estimates. This research is a continuation of previous work on the requirements and design of an accounting software to provide return on investment (ROI) estimates for CCOP SIGINT collection systems. We follow the Unified Process, an iterative, incremental software development process and apply use case analysis to obtain requirements of the accounting software. We then develop a high-level architecture design for a software meeting the requirements, and provide a proof-of-concept prototype to demonstrate the ROI analysis functions in Microsoft Excel.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, we would like to thank our loving wives, Marguerite Torres and Maggie Spivey. Marguerite and Maggie made sacrifices during major life events such as pregnancy and wedding planning throughout the thesis process. Second, we would like to thank little William Torres for his ability to sleep through the night at an early age and for providing levity when needed. Finally, we would like to thank Dr. Tom Housel and Dr. Man-Tak Shing for their patience and guidance as thesis advisors during our research and writing process

THIS PAGE INTENTIONALLY LEFT BLANK

.

NPS-AM-10-157

ACQUISITION RESEARCH SPONSORED REPORT SERIES

Analysis, Design, and Prototyping of Accounting Software for Navy Signal Intelligence Collection Systems Return on Investment Reporting

6 January 2011

By

Capt. James M. Torres, USA, and

Lt. Charles C. Spivey III, USN

Advisors: Dr. Thomas J. Housel, Professor, and

Dr. Man-Tak Shing, Professor

Graduate School of Operational and Information Sciences

Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION ..... - 1 -
A. PROBLEM STATEMENT ..... - 1 -
B. BACKGROUND ..... - 2 -
C. RESEARCH QUESTION ..... - 3 -
D. OBJECTIVE ..... - 3 -
E. SIGNIFICANCE OF RESEARCH ..... - 4 -
F. ORGANIZATION ..... - 6 -

II. LITERATURE REVIEW ON SOFTWARE IMPLEMENTATION ..... - 9 -
B. EVOLUTION OF KVA METHODOLOGY AND SOFTWARE SOLUTION ..... - 9 -
1. FIRST STUDY: ROI & KVA PROCESS FOR CCOP ..... - 9 -
2. SECOND STUDY: OPERATIONAL IMPLEMENTATION METHODOLOGY ..... - 11 -
3. THIRD STUDY: OPTIONS TO IMPLEMENT ..... - 12 -
C. INVESTIGATING NEW DATA SET ..... - 13 -
D. SOFTWARE IMPLEMENTATION PROBLEM ..... - 14 -
E. FIVE CASES OF IT IMPLEMENTATION FAILURE ..... - 14 -
1. Denver International Airport Baggage Handling System ..... - 14 -
2. FBI's Virtual Case File ..... - 15 -
3. Bank of America's MasterNet ..... - 19 -
4. Therac-25 Medical Accelerator ..... - 20 -
5. Patriot Missile Failure ..... - 22 -
F. SOFTWARE IMPLEMENTATION ISSUES FOR MILITARY AND CLASSIFIED INTELLIGENCE SYSTEMS ..... - 23 -
G. TOP FIVE SOFTWARE IMPLEMENTATION ISSUES ..... - 24 -
1. Sloppy Development Practices Resulting in Software Errors ..... - 24 -
2. Badly Defined System Requirements ..... - 25 -
3. Poor Communication Among Customers, Developers, and Users ..... - 26 -
4. Unmanaged Risks ..... - 27 -
5. Unrealistic or Unarticulated Project Goals ..... - 28 -
H. SUGGESTED METHODS FOR SUCCESSFUL SOFTWARE IMPLEMENTATION ..... - 28 -

III. METHODOLOGY ..... - 31 -
A. UNIFIED PROCESS ..... - 31 -
1. Inception ..... - 32 -
2. Elaboration ..... - 33 -
3. Construction ..... - 33 -
4. transition ..... - 33 -
5. our use of the process ..... - 34 -
B. REQUIREMENTS ANALYSIS ..... - 34 -
1. Use Case Analysis ..... - 34 -

2. Major findings ..... - 35 -
3. risk analysis ..... - 41 -
4. cots software ..... - 41 -

IV. SOFTWARE DESIGN AND IMPLEMENTATION ..... - 43 -
A. DESIGN OF IDEAL SOFTWARE SOLUTION ..... - 43 -
B. DESIGN AND IMPLEMENTATION OF SOLUTION PROTOTYPE - 47 -

V. SOFTWARE PROTOTYPE/FINDINGS ..... - 49 -
A. ROADBLOCKS FOR IMPLEMENTATION ..... - 49 -
B. ANOTHER NEW DATA SOURCE ..... - 49 -
C. DATA COLLECTION METHODOLOGY FOR PROTOTYPE TESTING ..... - 50 -
D. DATA ANALYSIS/ROI CALCULATIONS ..... - 50 -
E. COTS SOFTWARE SOLUTION IMPLEMENTATION ..... - 51 -
F. DETAILED DESCRIPTION OF SOFTWARE SOLUTION ..... - 52 -

VI. CONCLUSION/RECOMMENDATIONS ..... - 57 -
A. SOFTWARE IMPLEMENTATION METHODOLOGY ..... - 57 -
B. BENEFITS OF ROI ANALYSIS ..... - 58 -
C. FOLLOW-ON RESEARCH ..... - 58 -
D. GENERALIZABILITY OF KVA METHOD AND ROI ANALYSIS. - 59 -

APPENDIX A. VISION DOCUMENT ..... - 61 -
A. VISION DOCUMENT ..... - 61 -
1. INTRODUCTION ..... - 61 -
2. POSITIONING ..... - 61 -
3. STAKEHOLDER DESCRIPTIONS ..... - 61 -
4. Product Overview ..... - 61 -
5. SUMMARY OF SYSTEM FEATURES ..... - 62 -

APPENDIX B. SYSTEM SOFTWARE REQUIREMENTS ..... - 63 -
A. INTRODUCTION ..... - 63 -
1. PURPOSE ..... - 63 -
2. SCOPE ..... - 63 -
3. Objectives and Success Criteria ..... - 63 -
4. Definitions, Acronyms, and Abbreviations ..... - 64 -
5. References ..... - 64 -
B. PROPOSED SYSTEM ..... - 64 -
1. Overview ..... - 64 -
2. Requirements ..... - 64 -
3. Constraints ..... - 67 -
C. SYSTEM MODELS ..... - 67 -
1. Use Case Diagram ..... - 67 -
2. Expanded Use Case Scenarios ..... - 68 -
3. System Sequence Diagrams ..... - 71 -
4. Domain Model ..... - 73 -
5. System Operation Contracts ..... - 74 -

APPENDIX C. SOFTWARE DESIGN SPECIFICATION ..... - 79 -
A. INTRODUCTION ..... - 79 -
1. Purpose ..... - 79 -
2. Objectives and Success Criteria ..... - 79 -
3. References ..... - 79 -
B. ARCHITECTURAL DESIGN ..... - 80 -
1. Scope ..... - 80 -
2. Design Goals ..... - 80 -
3. Logical Architecture ..... - 80 -
4. Rationales ..... - 81 -
C. OBJECT DESIGN ..... - 82 -
1. Scope ..... - 82 -
2. Interaction Diagrams ..... - 82 -
3. Design Class Diagram ..... - 82 -
D. DATA DICTIONARY ..... - 84 -
LIST OF REFERENCES ..... - 85 -

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Problem Space ..... - 1 -
Figure 2. Unified Process ..... - 32 -
Figure 3. Use Case Diagram ..... - 37 -
Figure 4. Domain Model for the System ..... - 40 -
Figure 5. Logical Architecture ..... - 45 -
Figure 6. Design Class Diagram ..... - 46 -
Figure 7. Data Entry Tab of Solution Prototype ..... - 53 -
Figure 8. ROI Analysis Tab of Solution Prototype ..... - 54 -
Figure 9. Graphical Display 1 Tab of Solution Prototype ..... - 55 -
Figure 10. Graphical Display 2 Tab of Solution Prototype ..... - 56 -
Figure 11. Use Case Diagram ..... - 68 -
Figure 12. System Sequence Diagram for UC-1 ..... - 72 -
Figure 13. System Sequence Diagram for UC-2 ..... - 73 -
Figure 14. Domain Model for the System ..... - 74 -
Figure 15. Logical Architecture of the System ..... - 81 -
Figure 16. System Design Class Diagram ..... - 83 -

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Requirements of the System with Use Case Cross References ..... - 38 -
Table 2. Requirements of the System ..... - 65 -
Table 3. Data Dictionary ..... - 84 -

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACS	Automated Case Support
CCOP	Cryptologic Carry-on Program
COTS	Commercial Off-the-shelf
CRC	Cryptologic Resource Coordinator
DCD	Design Class Diagram
DoD	Department of Defense
GAO	Government Accountability Office
GUI	Graphical User Interface
ICP	Intelligence Collection Process
ISR	Intelligence, Surveillance and Reconnaissance
IT	Information Technology
JFC	Joint Forces Command
JWICS	Joint Worldwide Intelligence Communications System
KL	Kleiglight
KVA	Knowledge-value Added
MOP	Measure of Performance
NTR	Naval Transformation Roadmap
OMB	Office of Management and Budget
OO	Object Oriented
OPNAV	Chief of Naval Operations
ROI	Return On Investment
SCI	Secret Compartmentalized Information
SCIF	Sensitive Compartmented Information Facility

SDS	System Design Specification
SIGINT	Signals Intelligence
SPAWAR	Space and Naval Warfare Systems Command
SSR	System Software Requirements
TTL	Time to Learn
UML	Unified Modeling Language
VCF	Virtual Case File

I. INTRODUCTION

A. PROBLEM STATEMENT

This research will address the specific problems associated with implementing performance accounting software within classified military networks used for intelligence collection. The software will provide near-real-time routine return on investment (ROI) analysis that will aid decision-makers in the budgeting process for the Navy's Cryptologic Carry-On Program (CCOP) Signals Intelligence (SIGINT) systems. Numerous software solutions fail because of poor development and implementation practices. This research will focus on determining the best way to successfully implement a software program within a military organization. This thesis provides a software design and implementation plan that can be used to process near-real-time performance data and provide ROI analysis of Navy CCOP systems. It also provides general guidance on avoiding software implementation problems. Figure 1 provides a visual representation of our problem space in the form of a Venn diagram.

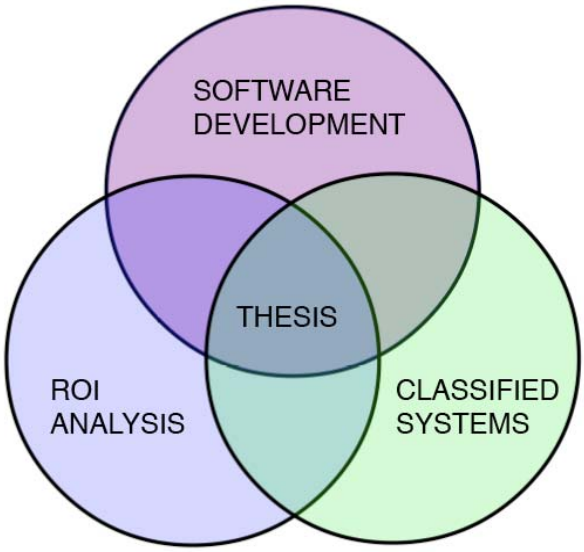

Figure 1. Problem Space

B. BACKGROUND

Dr. Tom Housel (Naval Postgraduate School) and Dr. Valery Kanevsky (Agilent Labs) created the Knowledge-value Added (KVA) approach. It is based on the assumption that humans and technology increase value to organizations by taking inputs and changing them into outputs through core processes (Housel & Bell, 2001, pp. 92–93). The KVA approach evaluates and assigns a value to the knowledge embedded within the core processes.

The KVA method was studied and specifically applied to Navy CCOP systems in three previous thesis studies. The first thesis, written by LCDR Rios and titled *Return on Investment Analysis of Information Warfare Systems*, developed a method for applying KVA analysis with ROI estimates to Navy CCOP systems. The second thesis project, *Using Knowledge Value Added (KVA) for Evaluating Cryptologic IT Capabilities: Trial Implementation,* written by LT Lambeth and LT Clapp, applied Rios's method to a real-world data set from Navy CCOP system operations during an 18-month deployment onboard a naval vessel. The third thesis project, *Collecting Retrieving and Analyzing Knowledge Value Added (KVA) Data from U.S. Navy Vessels AFLOAT*, by LT Homer, developed and evaluated the feasibility of three data-collection processes to be used in obtaining real-world CCOP KVA data from any United States Navy ship carrying CCOP systems.

During the course of his study, Rios determined that GaussSoft KVA, a commercial off-the-shelf (COTS) performance accounting software, was the only existing software that provided the necessary functionality. Rios stated, "Although several accounting software packages have included KVA analytical capabilities, the NPS research team has identified GaussSoft KVA software as the most comprehensive software platform for conducting the level of analysis required by DoD program managers" (2005, p. 15). Lambeth and Clapp and Homer continued to use GaussSoft for KVA analysis in their follow-on research. Therefore, we will maintain the assumption that GaussSoft performance accounting software is the only one suited to provide routine KVA-based ROI estimates. Unfortunately, GaussSoft KVA is not accredited for use on

classified networks. Our software is designed to run on classified networks and captures only the functionality of GaussSoft KVA required for near-real-time ROI analysis of Navy CCOP systems.

The previous studies provided ROI analysis for Navy CCOP systems using the KVA methodology. As a result of these studies, the researchers developed a practical methodology for quantifying the ROI of Navy CCOP systems using the KVA framework and a pathway for implementing the performance accounting system that generates these estimates. Homer (2009), however, uncovered a new data stream that would allow a near-real-time performance analysis of CCOP systems. We utilized the knowledge gained from the previous theses to develop software that incorporates the newly discovered near-real-time stream of performance data necessary for KVA analysis.

C. RESEARCH QUESTION

The primary research question of this project is the following: Can performance accounting software that provides near-real-time ROI analysis be successfully implemented for evaluation of Navy CCOP systems? While addressing the primary research question, we addressed several follow-on questions, such as:

- What are some common issues that result in software project failures?
- What are effective methods for avoiding these common issues?
- How can these methods be applied to the development of a software solution for ROI analysis of Navy CCOP systems?
- Can performance accounting software that provides ROI reporting be applied to other US Navy CCOP systems?
- Are previously established assumptions and procedures for KVA analysis of CCOP systems also valid for the automated reporting database used to report on CCOP ROI performance?
- What is the relative usefulness of automated CCOP reporting data compared to the value of the human-in-the-loop reports that were evaluated in previous CCOPs thesis research?

D. OBJECTIVE

The objective of this research project is to develop and implement a performance accounting software suite that program managers can leverage to measure the ROI

performance of Navy SIGINT collection systems. Previous research applying the KVA methodology to CCOP information technology (IT) has proven that the methodology provides valid ROI analysis of the Navy's CCOP systems and processes. Furthermore, prior work has attempted to create a plan for deploying software to collect, process, and analyze real-world CCOP performance data to provide routine ROI estimates. Previous researchers were unable to implement software to conduct KVA analysis of CCOP performance data because the data was located in different networks, databases, and formats. We used case studies of previous software development failures to create a software development plan that sought to avoid common mistakes associated with IT implementation failures. We designed and implemented a software solution for conducting near-real-time KVA analysis on the newly discovered data set of CCOP performance data. The goal of this research was to conduct a trial implementation of a performance accounting software that would support collection and ROI data for the performance of Navy SIGINT collection systems.

E. SIGNIFICANCE OF RESEARCH

This research project will directly support the Navy's Intelligence, Surveillance, and Reconnaissance (ISR) program with defendable Measures of Performance (MOPs) and ROI analysis. ISR is utilized by the military to collect, process, and analyze information within a battlespace and then disseminate that information to the warfighter or to national agencies conducting operations in that battlespace. ISR encompasses many areas such as optical photography, infrared emissions, interception of visual and voice electromagnetic signals, and interception of radar signals. IT systems are widely used in the ISR field because of their ability to automate the collection, processing, and analysis of large quantities of information in a short period of time.

The 2003 *Naval Transformation Roadmap (NTR)* provides strategic direction for Navy ISR programs and is based on the Joint Forces Command (JFC) transformation roadmap as well as on other joint initiatives. The *NTR* emphasizes that Navy ISR programs of the future will focus on becoming integrated within a joint ISR construct (Department of the Navy, Office of the Chief of Naval Operations, 2003). Joint ISR

system integration also results in joint ISR competition for funding. In 2007, the United States Government Accountability Office (GAO), in a report titled *Intelligence, Surveillance, and Reconnaissance: Preliminary Observations on DoD's Approach to Managing Requirements for New Systems, Existing Assets, and Systems Development*, stated,

> Without better visibility and performance evaluation, DoD does not have all the information it needs to validate the demand for ISR assets, to optimize the capability offered by these assets, to achieve a joint approach to employing its ISR assets, and to acquire new systems that best support warfighting needs. (GAO, 2007)

Therefore, it is imperative that program managers are armed with accurate and defendable MOPs for ISR systems when competing for DoD ISR funding.

ROI analysis provides decision-makers with data to evaluate system or organizational performance. Department of Defense Directive 8115.01 "implements policy and assigns responsibilities for the management of DoD IT investments as portfolios within the DoD Enterprise" (DoD, 2005, p. 1). An IT Investment Portfolio, as defined in DoD Instruction 8115.02, "includes outcome performance measures (mission, functional or administrative measures) and an expected return on investment" (DoD, 2006, p. 12). Software capable of providing ROI analysis in near real-time will provide a distinct advantage to IT portfolio managers.

This research will benefit Space and Naval Warfare Systems Command (SPAWAR) management by providing near-real-time automated ROI data on the operational performance of CCOP systems. Military budget cuts and increasing demand for technologically advancing SIGINT systems result in difficult choices for acquisition executives and funding managers for SIGINT collection systems. Providing effective and efficient performance measurements for SIGINT collection systems will aid decision-makers as they face these difficult budgetary choices. The results of this research will ultimately aid the U.S. Navy's Chief of Naval Operations (OPNAV) CCOP Program Office (OPNAV N201) and SPAWAR management in making decisions during the POM/budgeting process for Navy ISR systems.

The benefit of this research is not limited to Navy CCOP systems alone because software that applies KVA methodology can benefit many other organizations within the DoD and in the civilian sector. GaussSoft KVA software has been implemented in over 200 civilian companies and the KVA framework has been applied to many DoD processes to provide ROI analysis of IT systems.

F. ORGANIZATION

In this section, we provide an overview of the information that we will present in the remainder of the thesis. In Chapter II, we conduct a literature review that spans our problem space. First, we briefly review the work of Rios, Lambeth and Clapp, and Homer in developing a process for conducting ROI analysis of Navy CCOP systems using the KVA methodology. We highlight the discovery of a database that meets many of the recommendations of the previous researchers and allows for near-real-time ROI analysis. We also explain the benefits of this database over the previously evaluated human-in-the-loop reports. Next, we discuss those issues that are specific to systems in a classified environment and how they affected our research. Finally, we examine five cases of software failure in order to determine the lessons learned from each and how they can be applied to our work. These combined lessons learned are distilled into what we consider to be the top five software implementation issues. We discuss the relevance of each issue and how we attempted to mitigate its effects.

Chapter III is a discussion of our methodology for successfully creating a software solution. We describe the Unified Process for software development and the benefits of using it. We examine the Inception, Elaboration, Construction, and Transition phases of the Unified Process and briefly describe how we used the process during our research. Next, we detail our approach to requirements analysis over the Inception and Elaboration phases. This section highlights the important information contained in our Vision Document, use cases, and System Software Requirements (SSR). Our complete Vision Document and SSR can be found in Appendices A and B, respectively.

In Chapter IV, we discuss our design strategy and implementation plan during the Elaboration phase. We provide an overview of our completed System Design

Specification (SDS) for the ideal software solution. The complete SDS can be found in Appendix C. During the course of the study, we were forced to reevaluate our project goals and create a less ambitious prototype outside of the confines of the Unified Process. Therefore, we discuss the design considerations of this prototype as well.

Chapter V begins with a brief summary of the roadblocks that hindered our implementation of a software solution. Next, we detail the discovery of another new data set, its importance, and our attempts to incorporate it into our research. We then provide an overview of the testing we conducted on our prototype as well as an analysis of the COTS software options. Finally, we discuss our ROI calculations and provide a detailed description of the prototype's functionality.

In Chapter VI, we provide our conclusions on the research as well as recommendations for future work. This is directly followed by three appendixes containing a detailed description of the requirements and design of our ideal software solution.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW ON SOFTWARE IMPLEMENTATION

B. EVOLUTION OF KVA METHODOLOGY AND SOFTWARE SOLUTION

The first step in our literature review is to assume that previous KVA analysis assumptions for Navy CCOP systems are valid. These assumptions provide the algorithms necessary to process CCOP performance data. The next step is to assume that the GaussSoft software functionality is the standard by which we will measure our software implementation. Previous theses have provided the foundation on which these assumptions are made, and this section of our thesis will set the stage for the software solution needed to provide ROI analysis for CCOP systems. This review is not meant to provide an in-depth look at the previous works; all of the previous works are available through the Naval Postgraduate School.

1. FIRST STUDY: ROI & KVA PROCESS FOR CCOP

Rios's thesis provided a proof of concept for applying the KVA methodology to Navy CCOP systems. His research was focused on supplying Navy ISR portfolio managers with a standardized ROI analysis that incorporates surrogate revenues based on imbedded knowledge in process outputs. Rios investigated ROI and the dilemma facing non-profit organizations, such as the DoD, when they attempt to assign a value to their services and products. He highlighted the fact that the DoD is better suited to provide a cost analysis for its personnel and systems because there are no revenue values associated with the outputs generated by DoD human and IT systems. Although ISR systems and operators do not generate a product that is sold with a monetary value, Rios investigated and applied market comparables similar to reports generated by the Intelligence Collection Process (ICP) to formulate an estimated price-per-output. Subject-matter experts in the fields of Navy ISR operations, KVA theory, and Navy CCOP systems were used to apply the KVA methodology to Navy CCOP systems. The KVA valuation framework that was created used the Kleiglight (KL) report as the output of Navy CCOP systems to conduct performance analysis. The KL report is a classified SIGINT report produced by system operators. The ship-borne ICP was defined to include human

operators, CCOP systems, processes, and sub-processes required to generate the KL reports. As a result, Rios's research demonstrated the applicability of applying the KVA methodology to CCOP systems and operators producing KL reports as an output. The study provided an ROI analysis, with KVA valuation, for a realistic sample of CCOP systems and operators on a typical six-month deployment (Rios, 2005). Rios states, "The solution to the valuation of IT systems has been referred to as one of the 'holy grails' of the Information Systems (IS) field" (2005, p. 46). Rios's research laid the foundation on the quest for this "holy grail." In his conclusion, Rios wrote,

> It was the goal of this research to provide the means to extract measures of value and effectiveness to the CCOP Program office through the use of the Housel-Kanevsky Knowledge Value Added (KVA) Methodology. Applying KVA to the USS READINESS Case Study showed that the program managers could build metrics that are meaningful and useful in performing sound financial analysis of each system's performance at the process and subprocess level. KVA analysis also identified a new category and source of raw data which can provide insights into the relationship of cost and value of organizations, processes, and asset investments. This new data allows managers and senior decision makers to discuss the "value" of seemingly intangible assets in a defensible, empirical and replicable manner. Lastly, KVA facilitates the transformation and continuous process improvement of the DoD's global intelligence mission. Through KVA analysis, the operational value of CCOP systems can be measured and managed to ensure a responsible stewardship of the nation's resources and ensure that the soldiers and sailors who use these systems are receiving the right tools with the right capabilities required to perform their duties in defense of the nation. (2005, pp. 46–47)

Rios provided several key recommendations for future ROI analysis of CCOP systems. The first recommendation was that KVA performance data needed to be accessible in near real-time. Second, CCOP performance data, due to the classification of the systems and reports being generated, was located on various classified networks and needed to reside in one domain for processing ease. It was also recommended by the SPAWAR sponsor that CCOP performance data be analyzed over a longer period to provide a more established performance baseline. Lastly, the study researched KVA software and recommended GaussSoft as the KVA valuation software to be used on CCOP performance data (Rios, 2005, pp. 49–50).

2. SECOND STUDY: OPERATIONAL IMPLEMENTATION METHODOLOGY

Lambeth and Clapp's (2007) thesis was the second study in a series of research efforts to provide ROI analysis using the KVA methodology for Navy CCOP systems. Rios's study laid the foundation for applying KVA analysis to CCOP, and Lambeth and Clapp expanded on the research by transitioning from the hypothetical crew and system in Rios's study to real-world data collected from CCOP systems and three different crews onboard *USS GONZALEZ* (DDG 66) during an 18-month deployment. Lambeth and Clapp refined the assumptions and methodologies of Rios by providing costs and Time to Learn (TTL) calculations for two additional CCOP systems. Their study reaffirmed that the assumptions for processes associated with the ICP and market comparable data were still valid. KL reports remained as an output, and they added another operator-created SIGINT report—SIGINT Technical Report Using Models (STRUM)—as an output as well. These outputs were assigned a weighted value based on priority and complexity. Lambeth and Clapp parsed the KVA performance data by hand from KL reports, and they found this task to be tedious and cumbersome.

Their findings led to several conclusions about CCOP system performance and provided ROI analysis of CCOP systems. Performing the KVA analysis also led to insight about how crew proficiency, system tasking, and location positively or negatively affected the ROI of CCOP systems. In the end, Lambeth and Clapp further validated the use of KVA analysis for CCOP systems with a trial implementation aboard DDG 66 (Lambeth & Clapp, 2007).

Recommendations for their research included the creation of a community-wide database with explanations for the KVA assumptions that were made for individual systems. This would include time to learn (TTL) calculations for ISR systems and human cost estimators for operators of ISR systems. While the study did provide KVA analysis, recommendations were made that a near-real-time implementation of KVA analysis be made to support not only CCOP program managers but also Cryptologic Resource Coordinators (CRC) within strike groups. If KVA analysis was completed near real-time, then a CRC could use the KVA analysis as a dashboard to view CCOP system

performance, and the CRC could also seek out reasons for lack of CCOP performance, such as lack of crew proficiency. Qualitative analysis such as system position and tasking was recommended in addition to KVA analysis to provide more robust insight. Lastly, Lambeth and Clapp proposed that GaussSoft software was still the only software suited for use as a KVA analysis tool. It was noted that GaussSoft was not accredited for use in Secret Compartmentalized Information (SCI) spaces, and Lambeth and Clapp recommended that the process for GaussSoft accreditation be started (Lambeth & Clapp, 2007, pp. 41–43).

3. THIRD STUDY: OPTIONS TO IMPLEMENT

The third study, by Homer (2009), used the previous work of Rios and Lambeth and Clapp to provide an implementation plan for producing KVA analysis of all CCOP systems on board Navy ships. Homer analyzed the CCOP performance data required to produce KVA analysis and determined that not all data needed was within the KL report. Specifically, he identified critical data such as CCOP systems used to generate a KL report and total work time needed to complete the KL report. Homer also identified non-critical data points, such as location and date-time group, to provide qualitative analysis. Homer proposed a form to capture the necessary data and provided three implementation options to provide KVA analysis on all afloat CCOP systems. The first option was to use the data capture form and provide a stand-alone laptop for CCOP operators to input the necessary performance data when generating KL reports on deployment. The second option was to create a new message to be transmitted along with the KL report that captured the required performance data associated with the KL report. The third option was to change the KL report message format in order to capture all of the CCOP performance data needed for KVA analysis. Homer recommended that the first option be implemented because of the time and money advantages associated with it, despite the fact that near-real-time KVA analysis would not be achieved because the stand-alone laptop would travel with CCOP systems and then return to SPAWAR after deployment for KVA analysis (Homer, 2009).

At the conclusion of Homer's work, while presenting his findings, SPAWAR executives and CCOP system experts identified a new set of performance data that

previous NPS researchers were not aware of. Homer recommended that this new data set be investigated for performance data required to conduct KVA analysis. Additionally, he suggested that KL reporting might not be the complete measuring stick for CCOP system performance because it varied by crew, location, and tasking (Homer, 2009, pp. 37–39).

C. INVESTIGATING NEW DATA SET

The new data set discovered at the end of Homer's research is the reason that this study has veered away from the implementation plan suggested by Homer. Had previous researchers been aware of the data set, they would likely have sought to implement a KVA software solution more expeditiously as well. Previous works have shown that KL reporting is subject to crews, location, and tasking and does not reflect the complete value-utilization of CCOP systems. SPAWAR system experts have identified that the new data set is a direct output of CCOP systems. This data set fulfills many of the recommendations made by previous researchers by providing a single, combined source of CCOP data, over a long timeframe, that can be accessed in near real-time. The new data set is an automated output from CCOP systems based on the electromagnetic wave signals that are collected as input, and the software solution will need to transform these data into valid inputs for KVA analysis.

KL reporting appeared not to be a complete representation of CCOP performance. To demonstrate why this is the case, the analogy of a "cop on the beat" will be used. If a police officer was hired to patrol an area for 20 years and if he made very few arrests in those 20 years, is the ROI of the officer negatively affected? The officer has been walking the beat as he was hired to do. Arrests are not the only measure of performance for the officer because his presence and observations while on patrol are also an output that needs to be accounted for as a deterrent to crime, among other reasons. Similarly, the CCOP systems are designed to provide SIGINT whether a signal of interest is identified or not. Just because KL reports are not being generated does not mean that the systems are not collecting and processing electromagnetic waves. The new data set is an output of the signals that are collected and processed automatically by the CCOP systems, while the KL reporting can be equated to the police officer making an arrest. KL reports are outputs of the CCOP systems based on crew inclination, tasking, and

location. Therefore, the new data set is a more objective representation of CCOP system performance.

D. SOFTWARE IMPLEMENTATION PROBLEM

While the failure rates for software implementation have decreased over the years, the relatively low success rate for large-scale software systems remains a serious problem that resulted in an estimated waste of $55 billion in 2004. In creating its annual CHAOS reports, the Standish Group studied over 40,000 projects during a 10-year period (1994–2004). During that time, project success rates increased from 16% to 34% of all projects. Project failure rates declined from 31% to 15% of all projects. The Standish Group defined failures as those systems that do not function as intended or are never used at all. In the most recent of these surveys, 51% of projects were considered 'challenged.' These projects were over time, over budget, and/or lacking critical features and requirements. The average cost overrun of all projects in 2004 was 43%, down from 180% in 1994. Although there may have been further successes in the intervening six years, software development and implementation remains a highly failure-prone industry, resulting in tens of billions of dollars in yearly losses. In order to increase our chance of success and to provide general guidance, we examined five cases of software failure to determine common issues that lead to failure.

E. FIVE CASES OF IT IMPLEMENTATION FAILURE

1. DENVER INTERNATIONAL AIRPORT BAGGAGE HANDLING SYSTEM

The Denver airport attempted to implement a $230-million, high-tech, computerized baggage-handling system that was finally cancelled in 2005 after a decade of work. The system was designed to use computers and thousands of remote-controlled carts operating on a mostly underground 21-mile-long track. The carts would carry luggage from the check-in counters to sorting areas and then to flights waiting at airport gates. Each piece of luggage had a bar-coded tag attached that could be scanned by the system to ensure proper luggage delivery. The system was designed and built by BAE Automated Systems, Inc. Bruce Webster, principal of Webster & Associates LLC, a

Washington-based company that consults on troubled IT projects, described what happened:

> There are a few lessons that large companies just don't seem to learn. The first lesson is that the best way to build a large, complex system is to evolve it from a small system that works. No one bothered to get a small system up and running in the first place—they went for the big bang. Once the system gets to a certain point there is an attitude that the project is too big to fail, that we "have to make it work now." There is an unwillingness in upper management to believe that things are as bad as they are. (as cited in Weiss, 2005, p. 1)

Mark Keil, a professor of computer information systems at Georgia State University and a researcher on failed IT projects, said that the project should have been cancelled in 1994 when the system failed to work as designed. There were so many problems with the system that it even delayed the opening of the airport by approximately 16 months at a cost of $340 million. By the time the airport opened, it had resorted to manual baggage handling for all inbound flights at an additional cost of $70 million. Work on the system finally stopped in 2005 after more than a decade of trying to get it operational (Weiss, 2005).

The lesson learned from this case is that any software project should start with a small working prototype that is incrementally tested and built upon. This will ensure that there is always a working system and that the problems that must be addressed at any one time are relatively small. We initially created a rapid prototype that captured the absolute minimum functionality required of our system. Since it is entirely possible that this program will become quite large, it will be important to continue to make small incremental changes.

2. FBI'S VIRTUAL CASE FILE

In September 2000, Congress approved $379.8 million for the FBI Information Technology Upgrade Project, which was eventually split into three parts and became known as *Trilogy*. The information presentation component would provide all 56 FBI field offices with new Dell Pentium PCs running Microsoft Office as well as with new scanners, printers, and servers. The transportation network component was to provide

secure local-area and wide-area networks over which the new hardware could be fully utilized. The user application component would become the Virtual Case File (VCF) (Goldstein, 2005).

The initial purpose of the software portion of Trilogy was to make the five most heavily used investigative applications accessible via a web interface in order to rebuild the FBI's intranet and to identify a way to replace the FBI's 40-plus investigative applications, including the obsolete Automated Case Support (ACS) system. In May and June 2001, the FBI awarded Trilogy contracts to two government contractors: DynCorp for the hardware and network infrastructure and SAIC for software. The original delivery date for all three components was the middle of 2004. More important, instead of paying a fixed price for the components, the FBI used cost-plus-award fee contracts, which meant the Bureau would be responsible for any unforeseen or additional costs. After the 9/11 attacks, the inability of FBI agents to share basic information using their obsolete systems became a front-page scandal. At this point, it was decided that the current plan for the software portion would not make agents more effective (Goldstein, 2005).

The Bureau eventually decided that it needed an entirely new database, graphical user interface (GUI), and applications that would allow agents and intelligence analysts to share investigation information. The new system would host millions of records containing information on everything from witnesses, suspects, and informants to evidence such as documents, photos, and recordings. The new system was dubbed the Virtual Case File (VCF). The Bureau wanted to provide the software to agents as fast as possible. Unfortunately, the FBI did not have an enterprise architecture blueprint to guide hardware and software investment decisions. The importance of an enterprise architecture cannot be understated:

> This blueprint describes at a high level an organization's mission and operations, how it organizes and uses technology to accomplish its tasks, and how the IT system is structured and designed to achieve those objectives. Besides describing how an organization operates currently, the enterprise architecture also states how it wants to operate in the future, and includes a road map—a transition plan—for getting there. (Goldstein, 2005, p. 4)

Without an enterprise architecture as a guide, a team of FBI agents had to feel their way in the dark to determine how the organization currently operated and how that should be translated into the VCF system (Goldstein, 2005).

In December 2001, the FBI asked SAIC to stop building the web front end and to devise a new application, database, and GUI to completely replace ACS. In January 2002, the FBI requested an additional $70 million to accelerate Trilogy. The request was approved for $78 million. DynCorp committed to delivering its components by July 2002, and SIAC agreed to deliver the initial version of the VCF in December 2003. SAIC and the FBI had committed to creating a completely new system in 22 months that would replace ACS all at once in a flash cutover. In other words, agents would log off ACS at the end of one week and log on to VFC the next Monday. Once this occurred, there was no going back to ACS, and there was no backup plan if the flash cutover did not work. Fortunately for the FBI, a flash cutover attempt never took place (Goldstein, 2005).

From this point forward, the project was continually delayed by various factors. Some of the major factors included no formal project schedules or milestones; incompatible functional pieces of code; turnover of key IT management personnel; an extremely bloated, low-level requirements document; unnecessary creation of software from scratch; massive numbers of change requests, including new functionality and requirement changes; and a lack of hardware on which to test the developed system. In December 2002, Congress approved another $123.2 million for Trilogy, the total cost of which had now reached $581 million. On December 13, 2003, SAIC delivered the VCF, and the FBI declared it Dead on Arrival (DOA). The FBI found 17 functional deficiencies it wanted fixed before the system was deployed. An arbitrator's findings, released on March 12, 2004, found that of the 59 issues derived from the original 17

deficiencies, 19 were the FBI's fault (requirement changes) and the other 40 were SAIC's errors. In June the FBI contracted an independent reviewer, Aerospace Corp., to review the December 2003 delivery of the VFC. During a hearing on February 3, 2005, Senator Judd Gregg (R-NH) disclosed the following from the report:

> The [VCF] architecture was developed without adequate assessment of alternatives and conformance to various architectural standards, and in a way that precluded the incorporation of significant commercial off-the-shelf software. Furthermore, high-level documents, including the concept of operations, systems architecture, and system requirements were neither complete nor consistent, and did not map to user needs. Finally, the requirements and design documentation were incomplete, imprecise, requirements and design tracings have gaps, and the software cannot be maintained without difficulty. And it is therefore unfit for use. (Goldstein, 2005, p. 10)

The FBI officially ended the VCF portion of Trilogy in April 2005, with a loss of at least $105 million on unusable code. The next month, the FBI announced it would buy commercial off-the-shelf (COTS) software at an undisclosed cost to be deployed in phases over the next four years. The project, called *Sentinel*, was expected to cost $425 million dollars and should have been operational in 2009. Currently, Sentinel is not expected to be operational until early 2011. The project is over budget and could possibly get caught in IT cost-cutting by the Office of Management and Budget (OMB). Until Sentinel is in place, the FBI will continue to rely on basically the same combination of paper records and obsolete software that the VCF was meant to replace (Stokes, n.d.).

The lesson learned from this case is the importance of creating an enterprise architecture for an organization and detailed requirements documents for individual software projects. Additionally, the creation of a high-quality requirements document is dependant on adequate communication between customers, developers, and users. Another issue is the complexity involved when working with multiple contractors and multiple stakeholders. Our requirements document captures the needed high-level functionality without dictating how the individual functions should be implemented. Additionally, we used an iterative process to build our requirements incrementally along with the design and construction of our prototype.

3. BANK OF AMERICA'S MASTERNET

Bank of America began a project to manage all of its trust accounts in 1982 called *MasterNet*, and in 1988, the project was cancelled when Bank of America sold all of its remaining trust accounts. Although Bank of America did successfully implement several IT systems during the same time frame, MasterNet was not one of them. The most glaring factor in the failure of MasterNet was that Bank of America neglected to keep pace with technology during the 1970s. Yet, Bank of America executives sought to jump into the 1990s with the technology created during the implementation of MasterNet. The complexity of trust accounts combined with the complexity of the MasterNet project posed many difficult issues for implementation. MasterNet project managers attempted to implement all functionality inputs from everyone in Bank of America that had an interest in managing trust accounts. As a result, 3.5 million lines of code were developed. Project managers also strayed away from existing technology within Bank of America in favor of new technologies, in spite of the fact that few people working on the project were familiar with them. Because of software expansion and increasing complexity, the hardware requirements were met with poorly implemented, low-quality components. The MasterNet project continued to expand in software and hardware complexity until its failure (Szilagyi, n.d.).

Despite all of the issues surrounding the project, Bank of America executives continually promoted the success of MasterNet to the public and placed unrealistic deadlines on MasterNet program managers. As a result, MasterNet was not properly tested before it was implemented, and system failures resulted in millions of dollars in losses being paid to the owners of Bank of America trust accounts. Bank of America lost or sold $38 billion dollars worth of trust investments as a result of MasterNet's failure. Five billion dollars was allocated for technology initiatives inside Bank of America, and millions of these dollars were used for MasterNet (Szilagyi, n.d.).

Although executive management committed significant financial resources to MasterNet, Bank of America executives were focused on several other financial setbacks during the 1980s. Organizational faults between lines of communication from executive management and program managers contributed to lack of understanding on the

complexity of MasterNet. MasterNet was a failed implementation of an information system because of several errors, and it is an important case study in identifying contributing factors that lead to software-implementation failure as well as project management failure as a whole (Szilagyi, n.d.).

The lesson learned from this case is the importance of setting realistic project goals. In this case, unrealistic deadlines resulted in a system that was not properly tested prior to implementation. When it becomes clear that a deadline will not be met, the project must be reevaluated and a new, realistic deadline should be created. We constantly reevaluated our deadlines and were forced to simplify our prototype due to an inability to access needed resources in a timely matter.

4. THERAC-25 MEDICAL ACCELERATOR

The Therac-25 is a computerized radiation therapy machine or a medical linear accelerator (*linac*). Linacs accelerate electrons to create high-energy beams that can destroy relatively shallow tumors with minimal impact on the surrounding healthy tissue. In order to treat deeper tissues, the electron beam must be converted into x-ray photons. Between June 1985 and January 1987, there were six known accidents involving massive overdoses by the Therac-25 that resulted in either death or serious injury. While there were numerous issues with incident reporting and apparent negligence by the manufacturer in informing users about possible dangers, we will focus on the problems with the device itself because they pertain to our study of software implementation issues (Leveson & Turner, 1993).

Two major problems with the Therac-25 were responsible for the massive overdoses. The first issue was a combination of software bugs and poor program design that provided operators with cryptic error messages and allowed them to repeatedly retry failed treatments as standard operating procedure. There were at least two specific repeatable situations in which the software allowed a massive overdose to be administered. Both involved a race condition that caused the machine to be activated in an incorrect state. A race condition is a programming flaw that causes an output to be dependent on the sequence and/or timing of other events. These two specific situations

were each attributed to two of the overdose cases, accounting for four of the six total incidents. The other two cases were never attributed to any specific scenario, but researchers assumed that they were also caused by other race conditions and/or unsafe coding practices (Leveson & Turner, 1993).

The second major problem with the Therac-25 was the lack of any hardware safety features. This meant that the machine relied entirely upon software to ensure safe operation. The software for the Therac-25 as well as for the Therac-20 was built on software originally created for the Therac-6. The Therac-20 was basically an older, larger version of the Therac-25 with similar capabilities that included robust hardware interlocks to ensure safe operation. The Therac-25 had borrowed software routines from the Therac-20 for its electron mode. When users of the Therac-20 became aware of the issues with the Therac-25, they tried to determine if the Therac-20 suffered from the same software issues. After some experimentation, it was determined that certain situations attributed to the same software error resulted in blown fuses on the Therac-20. In the case of the Therac-20, the software error was nothing more than a nuisance because the protective circuits for monitoring the electron-beam scanning would not allow the beam to turn on at an improper setting (Leveson & Turner, 1993).

The lesson learned from this case is that focusing on particular software errors is not the way to make a system safe. Almost all complex software systems can be made to behave in an unexpected fashion in specific scenarios. The primary mistakes in this scenario involved poor software-engineering practices, resulting in faulty code and the creation of a machine that relied on software for safe operation. The specific coding mistakes are not nearly as important as the overall unsafe design of the software. Although it is not always practical to include hardware safety features, they should be included when possible, especially in systems that can cause death or serious injury. Although it is not possible or necessary for our system to incorporate hardware safety features, we used a software development methodology that should result in fairly error-free code. Our choice of an iterative and incremental methodology ensured that thorough testing and evaluation could be conducted during each iteration.

5. PATRIOT MISSILE FAILURE

This final case is an example of a relatively simple programming error within a military weapon system that ultimately resulted in the deaths of 28 Service members. During the Persian Gulf War, on February 25, 1991, a Patriot missile battery based in Dhahran, Saudi Arabia, failed to intercept an incoming Scud missile. The missile struck an American Army barracks, killing 28 Service members and wounding an additional 99. A report by the General Accounting Office (GAO), entitled *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia,* detailed the cause of the failure (Arnold, 1996).

The general problem was an inaccurate calculation (caused by computer arithmetic errors) of the time since boot (last system startup). Specifically, the system's internal clock kept time in tenths of seconds. This time was multiplied by 1/10 to produce the time in seconds. The problem occurred because the calculation was performed using a 24-bit fixed-point register that resulted in a chopped value of 1/10, inducing a small error. When the small chopping error was multiplied by the large number giving time in tenths of a second, a significant error resulted. At the time of the incident, the Patriot battery had been active for approximately 100 hours, resulting in a time error of about 0.34 seconds. Since a Scud missile travels at about 1,676 meters per second, the missile would have traveled about 560 meters in the error time frame. This distance was great enough that the missile was outside the range gate that the Patriot system tracked. In other words, because of the time error, the missile was not where the system expected it to be and, therefore, was not successfully engaged (Arnold, 1996).

There are some additional factors that should be considered when analyzing this incident. The effect of the inaccuracy on the range-gate calculation was directly proportional to the target's velocity and the length of time the system had been running. The system specification called for aircraft speeds and 14-hour continuous operation. The system was not designed to be used against Mach-6 missiles or to be operated for 100 continuous hours. It is clear that both parties were at fault: the military for relying upon a system in a situation that it was not designed for, and the manufacturer for creating software that would only work correctly at the limits of the specification. Had

the military been made aware of these limitations, it could have required periodic rebooting of the system to maintain an accurate-enough time since boot. The best solution would have been to modify the software, which was actually so simple that patched software arrived via air one day after the error was identified. Had the manufacturer originally taken the negligible effort to create more robust code, this incident would probably not have occurred (Dershowitz, n.d.).

There are several lessons learned from this case. The first is the importance of ensuring that the specifications for a system accurately reflect how it will be used. This, again, highlights the importance of creating a detailed requirements document that fully captures a system's functionality. The second lesson is that software should not be created that meets only the bare minimum of a requirement. Developers should anticipate that a product might be used in a situation surpassing its specifications, and they should create code that will mitigate this risk. This is particularly useful for software running mission critical systems that can be made more robust with negligible additional cost. Our system is not mission critical and is unlikely to be used in a situation that exceeds its specification. The Patriot missle system did, however, fail due to a simple arithmetic error. The third lesson learned, one that relates directly to our system, is the importance of testing system calculations. Our system is prone to arithmetic errors due to the calculations required to produce ROI values. We tested our software thoroughly to ensure that arithmetic rounding would not skew our results.

F. SOFTWARE IMPLEMENTATION ISSUES FOR MILITARY AND CLASSIFIED INTELLIGENCE SYSTEMS

While military systems suffer from all of the same issues as commercial systems, there are some additional considerations. The major issue that affects all military systems is the need for systems to undergo certification and accreditation before being authorized for use on military networks. The process is extremely long, complicated, and manpower-intensive. Nevertheless, it is a necessary process that ensures only adequately secure software is installed on the network. The primary issue that must be considered is the additional time that the process requires. Systems take an average of 18 months to

complete the process. This delay must be factored into all software deployment timelines for military networks.

An additional issue pertains to military systems deployed in a classified environment. Clearance requirements can make it more difficult to receive support for systems installed in these environments. Ensuring that contractors with adequate clearance will be available to provide direct support is essential for successful employment of classified systems. This issue directly affected our efforts because we were unable to personally work on the computer systems at SPAWAR. The end result was a simplification of our prototype, and a contractor was assigned to create a portion of the software solution.

G. TOP FIVE SOFTWARE IMPLEMENTATION ISSUES

1. SLOPPY DEVELOPMENT PRACTICES RESULTING IN SOFTWARE ERRORS

Sloppy development practices that cause software errors seem to be a reoccurring issue that plagues software systems. They were certainly the primary problem with both the Denver airport's baggage-handling system and with the Therac-25 medical accelerator. The baggage-handling system had so many errors that it never actually operated correctly and resulted in a total loss of at least $640 million. Errors in the Therac-25 software resulted in the deaths of five individuals and serious injury to at least one other. Although not discussed in detail in the cases, software errors were also attributed to the destruction of the Mariner-1 rocket in 1962, the collapse of the AT&T network in 1990, the explosion of the Ariane-5 rocket in 1996, and the loss of the Mars Climate Orbiter in 1998. All of the errors were relatively simple: a transcription error in a single formula, a single line of buggy code, an unhandled exception, and a units mismatch, respectively. All of these errors could have been discovered and corrected with adequate testing and evaluation (Martin, 2008; Barker, 2007; Lloyd, 1999; Gleick, 1996).

We used the Unified Process, an iterative and incremental development process, to ensure that we could effectively test our prototype. Because the program progresses

through further iterations, the additions had to be kept small to ensure that each version could be thoroughly tested. Additionally, we ensured that unexpected values would result in meaningful errors rather than in causing unexpected behavior.

2. BADLY DEFINED SYSTEM REQUIREMENTS

The FBI's VCF is a perfect example of how a lack of well-defined requirements can delay and eventually destroy a software project. Three major requirements issues affected the development of the VCF. The first was a lack of an enterprise architecture. Such a blueprint is extremely helpful whenever an organization is creating the requirements for a major software project but becomes essential when upgrading the IT infrastructure of an entire organization. An enterprise architecture would have helped stop the creation of the second problem: a bloated, low-level requirements document. Generally, a requirements document should consist of a list of features that describe at a high level what functions the program should perform. The developers then decide how each of the functions should be implemented in the program.

In other words, the requirements documents should dictate the *whats* but not the *hows*. Unfortunately, according to Matthew Patton, who worked as a security engineer for SAIC for three months, the VCF documents contained a lot of the latter. In an interview with Patton, he stated, "They were trying to design the system layout and then the whole application logic before they had actually even figured out what they wanted the system to do" (as cited in Goldstein, 2005, p. 7). Basically, rather than specifying which overall functions were required, the FBI created a huge requirements document that defined in great detail how certain pages should look and operate. Additionally, once the requirements document was accepted, the FBI created a third problem by introducing additional requirement changes throughout the development process. There should be very minimal requirement changes once both the customer and developer have accepted the document. A change made late in a program's development could require significantly more work to implement than if the same change had been made in the beginning (Goldstein, 2005).

The Patriot missile failure also had a clear issue with requirements. If the military intended to continuously operate Patriot batteries to guard against Mach-6 Scud missiles, then it should have included those values in its specification. Because its specification was for defending against much slower-moving aircraft and only 14 hours of operation, the manufacturer delivered a system that would only work reliably up to those maximum speed and time values.

Through analysis of the previous studies and discussions with stakeholders and users, we were able to create a detailed requirements document that adequately captures the requested functionality without dictating how the program should be implemented. When we present the prototype to the stakeholders for evaluation, we will further refine the requirements as needed.

3. POOR COMMUNICATION AMONG CUSTOMERS, DEVELOPERS, AND USERS

Without adequate interaction among customers, users, and developers, a software project is likely to fail. The mostly likely outcome will be the delivery of a system that the customer does not want. This issue is closely tied to the development of a high-quality requirements document. Communication must continue even after the requirements document has been accepted by all parties. Although the requirements document defines the high-level functions required by the customer, continuous communication is required to further refine the requirements and other specific aspects of a system. Communication with users is essential for designing a system that is easy to use and that meets the users' needs. It is entirely possible to create a system that fulfills all of the customer's requirements but is never utilized because users find it too complicated or unwieldy. During the development of the VCF, the FBI and SAIC had regular meetings between developers and various groups of users. Unfortunately, these meetings were not built upon the solid foundation of a high-quality requirements document and did little to fix the situation (Goldstein, 2005).

As previously mentioned, we built upon already captured communication with users and customers and conducted additional meeting as necessary to reach a consensus on the high-level functions required of the system. During one of these meeting, we

discovered the existence of another new data set that may play an important role in future research.

4. UNMANAGED RISKS

A fundamental part of software development is the management of risks. There are a couple of ways to approach this issue; one is risk taking by the customer in terms of financial losses. Given the likelihood of failure, a large software project is always a risky venture. Gradually changing an organization's IT infrastructure greatly reduces the amount of risk undertaken at any one time. The FBI's decision to completely replace its case management system in a flash cutover could not have been much riskier (Goldstein, 2005).

Risk management can also be approached in terms of software development. In this case, our discussion focuses on unhandled exceptions. An unhandled exception is an error that generally causes a program to crash or exhibit unexpected behavior because programmers did not provide a means of dealing with the error. The destruction of the Ariane 5 rocket was caused by an unhandled exception. The guidance system for the rocket shut down when it tried to convert the sideways velocity of the rocket from a 64-bit format to a 16-bit format and received an overflow error. A simple error-handling routine would have completely avoided this disaster (Gleick, 1996).

Although they were not caused by unhandled exceptions, the radiation overdoses by the Therac-25 are a good example of an unmanaged software risk. In this case, engineers were sure that it was impossible for the machine to overdose a patient. They had complete faith that the system's software would provide adequate protection. From a system-engineering point of view, they completely failed to manage risk by not including hardware safety features.

Our primary focus regarding risk management was the proper handling of exceptions. The primary way in which we accomplished this was to make the program as user-friendly as possible by providing meaningful error messages when data is entered incorrectly or entirely omitted.

5. UNREALISTIC OR UNARTICULATED PROJECT GOALS

Unrealistic or unarticulated project goals are clearly a widespread problem for large software projects, as evidenced by at least 51% of the projects studied by the Standish Group in 2004 not meeting their project goals. Although the FBI did a somewhat decent job of articulating their goals for Trilogy, the FBI had unrealistic goals for both cost and completion time for all three portions of the project. This problem stems from developers underestimating the amount of time a project will take. In the case of Trilogy, both DynCorp and SIAC agreed to an accelerated timeline, but neither was able to meet the original timeline with an acceptable product. Although developers should be primarily responsible for setting realistic goals, customers need to have experienced personnel who can evaluate a developer's claims. As previously noted, the FBI's Sentinel program also had unrealistic goals, and it is currently delayed and over budget.

Although we clearly articulated our project goals, they ended up being unrealistic due to unforeseen circumstances. Our original goal was the creation of a prototype of a single software solution that would provide near-real-time ROI analysis of CCOP systems. When we discovered that we would be unable to personally interact with the classified database, we had to scale back our goal to producing a prototype that only made calculations and displayed the results. The portion of the program that will interact with the database is currently being produced by a contractor at SPAWAR.

H. SUGGESTED METHODS FOR SUCCESSFUL SOFTWARE IMPLEMENTATION

In order to avoid the previously discussed software development issues, the first step in any software development should be the creation of a detailed requirements document. This document forms the basis for the entire project and, in its final form, lists all the major functions that must be developed. A working prototype should be the very first item created during software engineering. During each additional iteration, the requirements should be expanded and a new piece of code added and tested. This will ensure that there is always a functional program with no need to piece disparate parts together at a later time. Customer and user feedback should be solicited at the conclusion

of each iteration. Additionally, goals and milestones should be constantly evaluated and changed if it becomes clear that they cannot be met. Finally, programmers should seek to include robust error handling that will be capable of handling unexpected data entry and other errors.

The previous CCOP studies have made use of continuous customer and user feedback in order to determine the best means for providing ROI estimates of CCOP system performance. Planning and risk analysis were conducted for three different implementation options. Subsequent discussions with systems experts revealed the database of records as a more accurate source of data for ROI calculations. This resulted in a change to the overall goals and milestones. The planning phase had to be started again in a the current thesis while building upon previous customer and user feedback.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

A. UNIFIED PROCESS

In order to provide a framework that captures our recommendations for successful software implementation, we decided to use the Unified Process for software development. The Unified Process is a popular iterative and incremental process. In this approach, software development progresses incrementally over four phases. These phases are Inception, Elaboration, Construction, and Transition, and they represent the level of completeness of the project. Each phase is organized into a series of small projects called iterations. Generally, each iteration concludes with the creation of a tested, integrated, and executable portion of the final system (Larman, 2005).

As can be seen in Figure 2, each iteration includes its own business modeling, requirements, analysis and design, implementation, test, and deployment activities. It is important to note that this is only an example of possible activities and that they will be different for any given project. Our project incorporates only the requirements, design, implementation, and testing activities. Although each iteration will usually contain some effort, in all of the activities the relative amount will change over the course of the project.

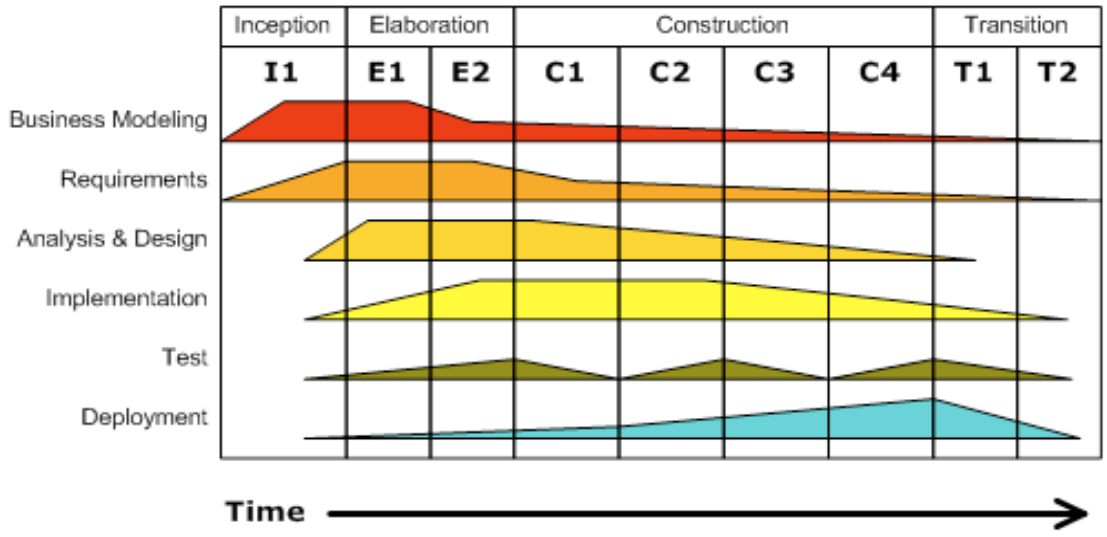

Figure 2. Unified Process

(Dutchguilder, 2007)

The following is a brief summary of each phase of the Unified Process as explained by Larman (2005).

1. INCEPTION

Inception is the initial short step that establishes a common vision and basic scope for a project. It includes analysis of approximately 10% of the use cases; an analysis of high-level, non-functional requirements; and preparation of the development environment so that programming can begin in the Elaboration phase.

The Inception phase should answer the following kinds of questions:

- What is the vision and business case for the project?
- Is it feasible?
- Should the software be purchased or built?
- What is a rough range of cost?
- Should we proceed with the project?

The purpose of the Inception phase is not to define all the requirements but to determine if it is worth a serious investigation in the Elaboration phase. Inception should usually last no longer than a week.

2. ELABORATION

Elaboration is the initial series of iterations during which the team does serious investigation, programs and tests the core architecture, discovers and clarifies most requirements, and mitigates the major risks.

Elaboration often consists of two or more iterations recommended to last between two and six weeks. Each iteration is timeboxed with a fixed end date. Elaboration is not a detailed design phase and does not involve the creation of throw-away prototypes. The code and design completed during this time are production-quality portions of the final system. This is usually referred to as the executable architecture or architectural baseline.

The following are key ideas and best practices pertaining to Elaboration:

- Create iterations that are short, timeboxed, and risk-driven.
- Start programming early.
- Design, implement, and test the risky parts of the architecture.
- Test early, often, and realistically.
- Constantly adapt based on feedback from users, developers, and testers.
- Write out most of the use cases and requirements in detail.

3. CONSTRUCTION

Construction is the largest phase in the project and involves the iterative implementation of the remaining lower risk elements as well as preparation for deployment. Each timeboxed iteration results in an executable release of the software.

4. TRANSITION

Transition is the final phase of the project and involves beta testing and deployment to users. The Transition phase may consist of several iterations that incorporate user feedback to create further refinements to the project.

5. OUR USE OF THE PROCESS

Our software development process completed the Inception phase and began the Elaboration phase of the Unified Process. We were unable to create any production-quality portions of code during the Elaboration phase, but we did succeed in creating a throw-away prototype that was extremely useful for customer evaluation. It is atypical to create a throw-away prototype during the Elaboration phase, but this was our only option for providing a usable solution for near-real-time ROI analysis during our research. The prototype should therefore be considered as an addendum to our software development using the Unified Process.

B. REQUIREMENTS ANALYSIS

The results of our requirements analysis were a Vision Document (Appendix A), use case diagram, two brief use cases that captured the high-level functions of the program, and an SSR (Appendix B). This process was based on use case analysis and spanned both the Inception and Elaboration phases of the Unified Process. We also conducted risk analysis to identify those issues that would prevent us from moving forward with the project and examined the feasibility of incorporating COTS software.

1. USE CASE ANALYSIS

Use cases are text stories of an actor using a system to meet a goal. They are used to discover and record functional requirements. An actor is something with behavior such as a person, software, or organization. Use cases are important because they emphasize the goals and perspective of the user. They are generally a more effective means of determining requirements than asking for a list of system features (Larman, 2005).

The first step of our Inception phase was the creation of a Vision Document and two brief use cases. A brief use case is a terse one-paragraph summary of the main success scenario. By *scenario* we mean a specific sequence of actions and interactions between actors and the system. These brief use cases formed the basis for the features

identified in our Vision Document, and we illustrated their relationship in a use case diagram.

During the Elaboration phase, both use cases were expanded into fully dressed use cases. Fully dressed use cases are highly detailed and structured. They address all expected scenarios and contain supporting sections, such as preconditions and success guarantees. The fully dressed use cases were extremely helpful in identifying the major functional requirements of the system. Both of the fully dressed uses cases, and the requirements they helped discover, are captured in the SSR (Larman, 2005).

2. MAJOR FINDINGS

In our Vision Document, we outlined the positioning of our system. We noted that the capability to provide CCOP system managers with near-real-time analysis of system performance does not currently exist. Additionally, providing ROI data for these systems will provide a means of objectively comparing systems to support future funding allocation decisions.

We also identified the major stakeholders of the system:

- Non-user stakeholders:
  - CCOP system managers,
  - CCOP operators, and
  - Naval ship captains.
- User stakeholders:
  - Military Service members and
  - Contractor personnel.

These users will have varying levels of computer and network knowledge. Therefore, the system will need to be user-friendly to accommodate various levels of expertise.

Next, we would have to provide an overview of the product and the three primary functions it would need to perform. The first is extraction of the data needed for KVA analysis from the newly discovered database. The second is conducting KVA analysis and calculations on the extracted data to produce an ROI for each individual CCOP

system. The final function is a graphical display of the ROI values in an easy-to-understand format. We noted that a fourth and secondary function would be required to enter and store additional data needed for the ROI calculations.

In order to capture these functions, we created brief use cases. We decided that the three major functions could be captured in one use case, *ROI Analysis*, and the secondary function in another, *Data Entry*. These use cases capture the two major interactions a user would have with the system. At this stage we considered a user to be the sole actor that would interact with the system.

We then created both of the use cases in a brief format:

UC-1

Name: ROI Analysis

Actors: Users

Description: Provides users with the ability to select a time period, system(s), and ship(s) for ROI analysis. Once selections are complete, the system will pull necessary data from the database and perform ROI calculations using previously provided cost, life cycle, and complexity values for the selected system(s). The resulting ROI values will be graphically displayed in an easy-to-understand format. Additional display and print options will be provided.

UC-2

Name: Data Entry

Actors: Users

Description: Provides users with the ability to enter cost, life cycle, and complexity data for a given CCOP system. This data will be stored and used for ROI calculations as required.

The relationship between the user the and use cases can be seen in the use case diagram (Figure 3).

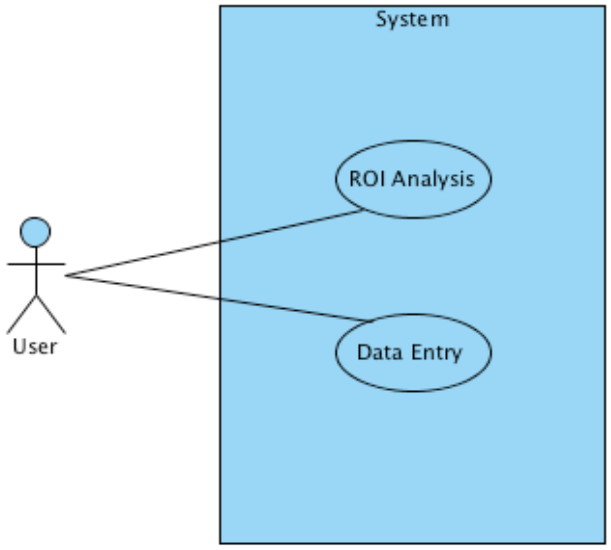

Figure 3. Use Case Diagram

Finally, we summarized the major system features. Each feature is associated with a use case:

- (UC-1) Automated Data Collection:
  - User Specified Time Period (Day/Month/Year Range)
  - User Specified System(s)
  - User Specified Ship(s)
- (UC-1) ROI Calculations
- (UC-1) Graphical Display of ROI Values (various formats)
- (UC-2) Storage of Cost, Life Cycle, and Complexity Data for Individual Systems

During the Inception phase, we began work on the SSR by listing the non-functional requirements of the system. The fully dressed use cases were created during the Elaboration phase and used to help identify the functional requirements of the system. The requirements have been categorized according to the FURPS+ model: *Functional*, *Usability*, *Reliability*, *Performance*, and *Supportability*. A system requirement ID number and relevant use case cross reference is provided for each requirement, as shown in Table 1.

Table 1. Requirements of the System with Use Case Cross References

System Rqmt ID	Requirement	Use Case Cross Reference
Functional		
001	Provide GUI that allows point–and-click selections	UC-1 & UC-2
002	Provide the capability to make selections using pull-down menus	UC-1 & UC-2
003	Provide the ability for the user to select a time frame range at the day/month/year level of granularity	UC-1
004	Provide the ability for the user to select one or more systems for ROI analysis	UC-1
005	Provide the ability for the user to select one or more ships for ROI analysis	UC-1
006	Check user entries for completeness and correctness	UC-1 & UC-2
007	Highlight incomplete or incorrect user entries	UC-1 & UC-2
008	Notify the user when data entry has not been completed for a selected system(s)	UC-1
009	Perform a database search based on user-entered parameters	UC-1
010	Notify the user when database does not contain data needed for ROI calculations	UC-1
011	Perform ROI calculations	UC-1
012	Provide the capability to graphically display ROI values	UC-1
013	Provide the capability for the user to select a graphical display format	UC-1
014	Provide the capability for user to print the graphical display	UC-1
015	Provide the capability to return to the main menu when complete	UC-1 & UC-2

System Rqmt ID	Requirement	Use Case Cross Reference
016	Provide the capability for the user to enter initial cost, recurring cost, life cycle, and complexity for individual systems	UC-2
017	Display and store user-entered data on individual systems (costs, life cycle, complexity)	UC-2
018	Notify the user when costs, life cycle, and/or complexity data is entered incorrectly	UC-2
019	Provide the capability for the user to enter data on multiple systems without returning to the main menu	UC-2
Usability		
020	Display text with a default size no smaller than 12 pt.	__
021	Make display colors modifiable by the operator	__
022	Provide a help dialog that meets industry standards for help functionality	__
023	Provide access to electronic documentation	__
Reliability		
024	Upon complete failure, the system should recover to the last-known good state	__
Performance		
025	Be compatible with a variety of hardware configurations to allow for user needs	__
026	Provide response to user actions within 1 second	__
Supportability		
027	Have patch capability to allow for future modifications for the software to adapt to new operating systems or operating system upgrades	__
028	Be easily migrated to a new operating system after an operating system upgrade	__
029	Maximize use of common object-oriented language	__

Our constraints followed logically from the following requirements:

- Hardware components of the system will consist of pre-existing desktop or laptop computers running a DoD standard version of Microsoft Windows.
- Software components of the system will utilize OO technology. The system will be compatible and adaptable to an OO environment.
- The system will be developed using the Unified Process.

Our SSR also contains a domain model (Figure 4) of the problem. It is a conceptual model that illustrates the relationships between the various entities that are important to a software solution to the problem.

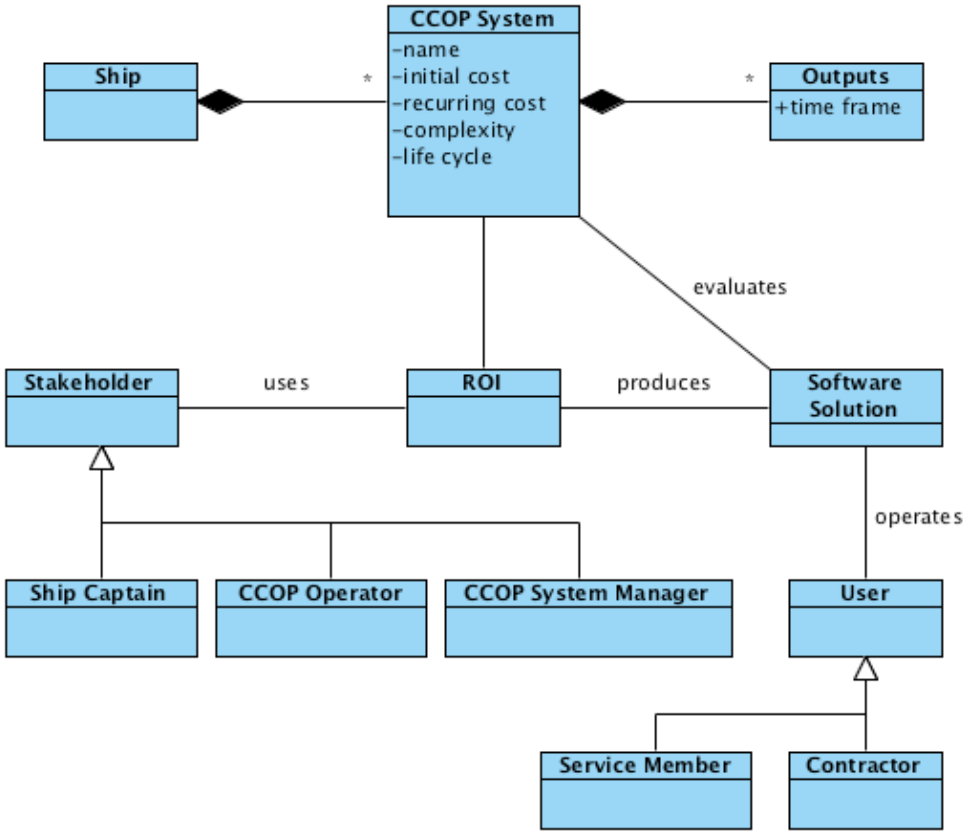

Figure 4. Domain Model for the System

The following is an explanation of the entities and relationships captured by the domain model: Ships contain CCOP systems; each of these systems has a name, initial cost, recurring cost, complexity, and life cycle. Additionally, these systems have outputs. Each output has a time frame during which it was generated. Our software solution

evaluates a CCOP system based on these attributes and produces an ROI. Two kinds of users operate our Software Solution: Service members and contractors. The ROI is specific to a CCOP system and is used by stakeholders. These stakeholders include ship captains, CCOP operators, and CCOP system managers.

3. RISK ANALYSIS

We identified not having access to the database as the biggest risk to our project. Without access to the database itself, we would not be able to implement the data-gathering function of the software. The plan at this time was for SPAWAR to provide us with remote access to a system containing a copy of the database through the Joint Worldwide Intelligence Communications System (JWICS) to the NPS Sensitive Compartmented Information Facility (SCIF).

4. COTS SOFTWARE

Ideally, all four of the functions we identified would be included within one software program. Although the GaussSoft software includes three of the four functions (it cannot pull data from the database), the software is not accredited for use on classified military networks. Microsoft Excel, on the other hand, is accredited and offers enough functionality to cover those same three features. We decided to explore the possibility of using a database in conjunction with Excel to provide a working prototype of the complete system. Additionally, we decided to consider the use of already-purchased COTS software (GaussSoft) whenever possible. Even though it cannot be used in a classified environment, we could demonstrate to the stakeholders at Space and Naval Warfare Systems Command (SPAWAR) how it could be used if they decided to move forward with certification and accreditation.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SOFTWARE DESIGN AND IMPLEMENTATION

The end result of our requirements analysis was our SSR. Our SSR contains two additional artifacts that helped form the basis for our software design. The first is system sequence diagrams (SSDs) for each of the fully dressed use cases. The purpose of an SSD is to illustrate a use case in a visual format. It shows in detail how the system is designed to handle certain actions. The second artifact is an operation contract for each of the operations identified in the SSDs. An operation contract identifies the system state changes that occur when an operation executes. We identified 15 operations and created a contract for each of them. The most important portion of an operation contract is the post conditions. Post conditions are not actions to be performed during the operation but represent the state of objects after the operation is complete (Larman, 2005). Contract C02: selectROI is provided as an example:

Contract C02: selectROI

Operation:	selectROI()
Cross Reference:	Use Cases: ROI Analysis
Preconditions:	- Main menu options displayed
Postconditions:	- Dialog box displayed with time frame, system(s), and ship(s) fields

The SSDs and remaining operation contracts are available in the SSR in Appendix B.

A. DESIGN OF IDEAL SOFTWARE SOLUTION

At the end of the Inception phase, we decided that our software design and implementation could be split along the four system functions and would begin during the first iteration of the Elaboration phase. The Elaboration phase would consist of two iterations. The first iteration would focus on initial design of the database search. This iteration would conclude with a hard-coded, functional database search. The second iteration would involve finishing the initial design of the overall system in the SDS and implementing a user-defined database search. This would effectively address the high

risk functionality of the system. We planned to implement the remaining three functions during multiple iterations of the Construction phase.

During our first iteration of the Elaboration phase, it became apparent that we would not be granted remote access to the database at SPAWAR during the course of our research. However, we discovered that the database was already accessible through a web interface with various predefined SQL queries. Although none of the existing queries met our needs, we incorporated this method of interfacing with the database into our logical architecture. The logical architecture is the large-scale organization of the software classes that make up a program into packages, subsystems, and layers.

Our logical architecture utilizes a web-based, graphical user interface (GUI) in our user interface (IU) layer. This layer relies on the Domain layer with packages for Actions, External Systems, and Output. The Actions package contains the classes required based on an analysis of the operation contracts. The remaining packages contain the major elements of the system such as database access, displays, and printers. The External Systems package is dependent upon the Database class contained within our final layer, Services. Figure 5 is a visual depiction of our logical architecture.

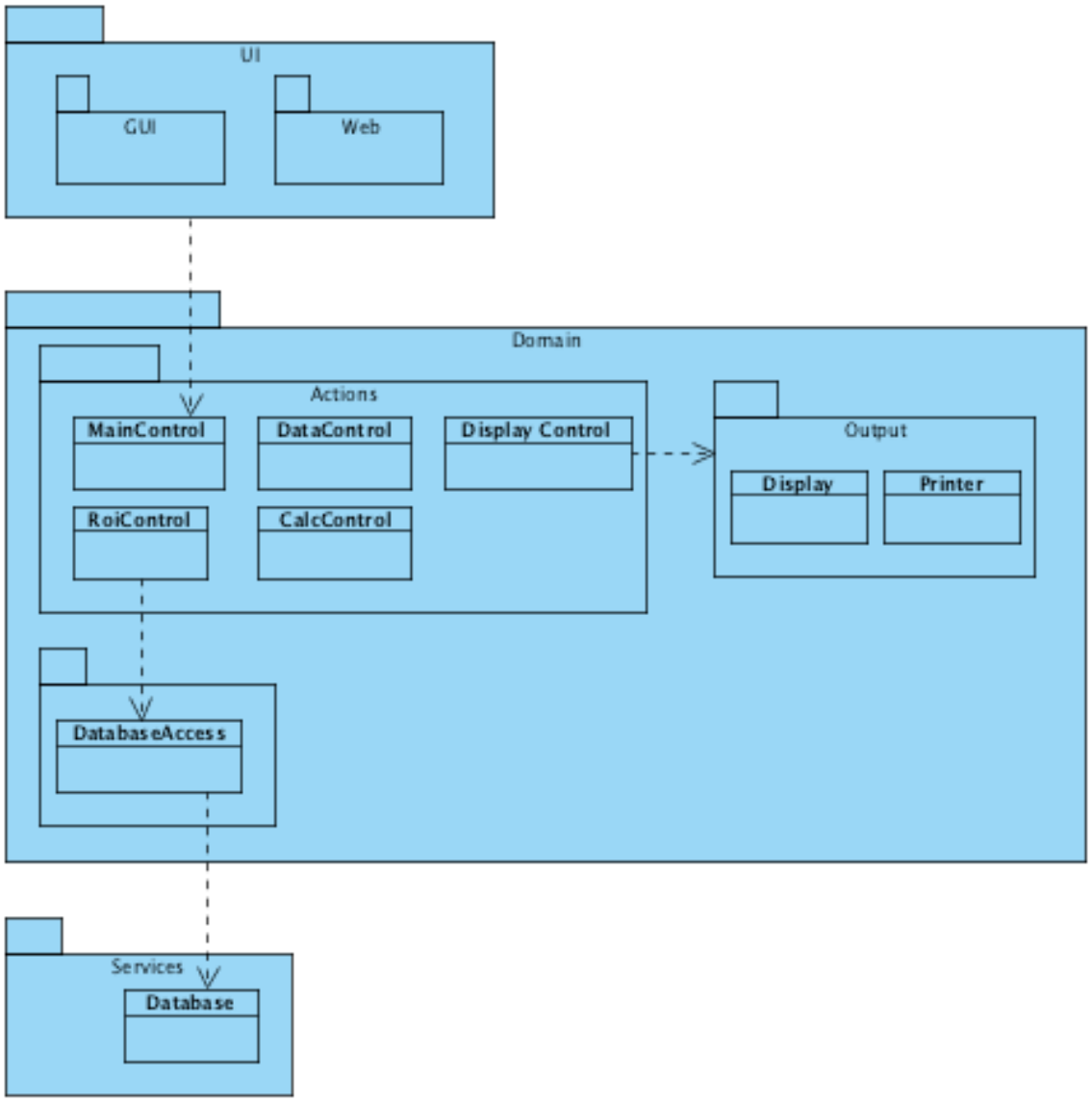

Figure 5. Logical Architecture

Even though we were unable to implement and test any code at this point in the Elaboration phase, we decided to continue with the design. We provided SPAWAR with the values we required in a predefined search of the database. A contractor was assigned to implement the search. We then moved into the second iteration of the Elaboration phase and created a Design Class Diagram (DCD) for the system. This diagram illustrates classes, interfaces, and their associations from a design perspective. Our diagram captures all of the operations identified in the SSDs and further clarified in the operation contracts. Additionally, it further defines the classes in the Actions package of

the Domain Layer of the logical architecture. Because the DCD defines the operations and attributes of each class, it serves as an excellent guide during implementation. Figure 6 is the DCD for our system.

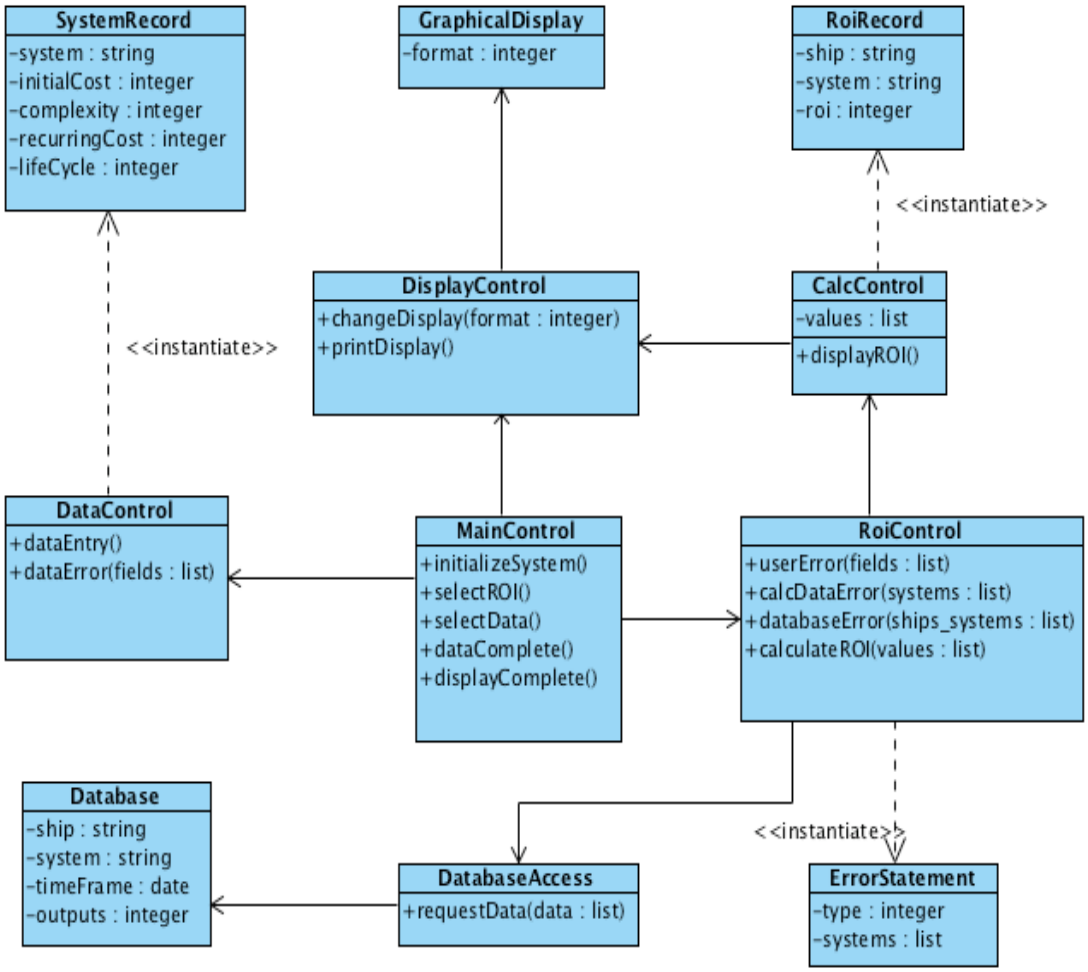

Figure 6. Design Class Diagram

The following is a verbal description of the operations associated with the Data Entry use case. The user selects *Select Data* on the main menu of the graphical user interface, which calls the selectData procedure of the MainControl class. The MainControl class then calls the DataContol class using the dataEntry procedure. DataControl then interacts with the user to create a SystemRecord containing the user-entered information. Finally, the user selects *Data Complete* which calls the dataComplete procedure and returns to the MainControl class.

At this point in the process, it did not appear that the predefined query to the database would be completed with enough time left for us to implement a working prototype according to our original design. We reevaluated our goals and decided that it was more important to have a working prototype that could be used for ROI analysis than to create a single software solution. Since analysis of COTS options was already a goal of our research, we explored the possibilities afforded by Microsoft Excel. Because this prototype would not contain any production-quality code, it would be a throw-away prototype and would be an addendum to the artifacts planned for our software development using the Unified Process.

B. DESIGN AND IMPLEMENTATION OF SOLUTION PROTOTYPE

The purpose of this design was to determine how to utilize Excel in order to create a prototype that would capture the major functions of our software solution. Additionally, the prototype would need to be able to provide near-real-time ROI analysis of Navy CCOP systems when combined with the completed predefined query. Although we did not have specific details regarding the format of the predefined query, we knew that they could be exported to Excel. At minimum, we expect the two programs can work together via cut-and-paste of the Excel-exported search results into our prototype.

We decided to logically separate the different use cases by utilizing the multiple tabs functionality of Excel. In this manner, the tabs would also simulate our main menu options. In total, we created four tabs: ROI Analysis, Data Entry, Graphical Display 1, and Graphical Display 2. We designed the interfaces of each of the tabs so that users could execute the use cases in the same manner they would for a single software solution. Upon selecting the Data Entry tab, users are presented with fields for System, Initial Cost, Life Cycle, Monthly Recurring Cost, and Complexity. Unfortunately, we were unable to implement an error message for incomplete entries. Upon selecting the *ROI Analysis* tab, users are presented with fields for Ship, System, Outputs, and Time Frame. This represents the data values that would be returned from a successful database search. If the fields are entered correctly, an ROI value is calculated and displayed. In this instance, we were able to successfully provide somewhat meaningful error messages for

incomplete fields or for when data entry was not previously completed for a given system. The graphical display tabs provide two graphical display format options.

Overall, we found Excel to be a viable COTS solution that could provide the needed functionally until a permanent solution could be implemented. The most significant drawback is that a user will need to be experienced with Excel in order to adequately make use of the prototype. It is probably most useful as a proof of concept and to generate stakeholder feedback. Because we used the Unified Process as our software development framework, it should be noted that this solution would be considered a throw-away prototype. This is because this prototype could never be incrementally improved upon to reach our overall goal of a single software solution. Throw-away prototypes are not a part of the Elaboration phase of the Unified Process.

V. SOFTWARE PROTOTYPE/FINDINGS

A. ROADBLOCKS FOR IMPLEMENTATION

Because of issues with access to classified information, we were unable to gain remote access to the database. However, we were given access to a website providing predefined SQL searches of the database, which return data in an Excel-exportable format. Through various discussions and a return trip to SPAWAR, we determined that a predefined SQL search based on our specifications would fulfill our requirements and allow for near-real-time ROI analysis. Further issues with classifications and authority to work on systems prevented us from personally implementing the SQL search on the host SPAWAR system. A contractor at SPAWAR is currently in the process of implementing the search. When completed and combined with the calculation and display portions of our prototype, the search will provide near-real-time ROI analysis.

B. ANOTHER NEW DATA SOURCE

During the course of our research, an additional data source was uncovered that consists of health and welfare status reports for the CCOP systems on individual ships. These status reports contain the total number of intercepts per system within a given time frame. Additionally, these reports provide an even more accurate value than the database of records because when records are transferred to the database, some are lost due to network issues. Analysis of the records in the database would therefore provide a lower ROI than is actually being created by the systems.

Comparison of the ROIs calculated using the two data sources would provide a good indicator for how much output is lost due to network issues. Although, the status report data is more accurate, it is not in a format that can be easily used for near-real-time ROI analysis. For this reason, the status report data could not be directly adapted for a software solution.

C. DATA COLLECTION METHODOLOGY FOR PROTOTYPE TESTING

There were two sources of data that we wanted to collect and run through our prototype. The first source was the health and welfare status reports of CCOP systems on individual vessels. We selected eight different vessels that cover the four different versions of CCOP A that were all recently operating in the same oceanic region. The data source contains status reports for the last 90 days of operation. We selected a 30-day period of data for each of the eight vessels. Each status report covers a single day worth of data, resulting in 240 status reports.

For the next iteration of this research, we would need to be able to determine the total number of outputs for the 30-day period, we would need to manually examine the individual reports for each ship. This would allow us to provide an ROI over the selected 30-day period for the CCOP system on each of the eight different ships.

The second source of data is the database of individual records. This data will eventually be accessible through a predefined SQL query that searches over a date range and returns the ship, system, and total number of outputs.

Additional data are needed to make meaningful ROI calculations. These include initial and recurring costs as well as estimated life cycles and complexity for each of the four versions of CCOP A. The system experts at SPAWAR will provide the values for these calculations.

Although we specified the data needed for a proof of concept, none of the specified data were available during the course of our thesis. We used simulated data to test our prototype and ensure our calculations were correct.

D. DATA ANALYSIS/ROI CALCULATIONS

Our plan was to analyze the data by calculating an ROI for each set of data for each ship over a 30-day period. ROI values were calculated using Excel with the following formula:

ROI = ((total monthly outputs * complexity factor) – monthly cost)/monthly cost.

For the status reports data set, the total monthly output is the sum of the daily total intercepts from 30 reports. We intended to calculate this for each of the eight ships. For the record database, the total monthly output is the value returned when the predefined SQL query is executed for a ship/system combination. The complexity factor is a dollar value that captures the KVA of a given system. For our purposes, we have based the complexity factor on the total number of days to learn the various functions performed by the system. By *days to learn* we are literally referring to the number of days it would take a human to learn to do the same functions as well as the theory required to make the systems function as designed. In this way, the complexity factor is a measure of the knowledge embedded within the system. Aside from providing a logical means of weighing individual outputs, the complexity factor provides justifiable measurement for comparing the complexity and ROI values of different systems because the resulting estimates are auditable for accuracy. The monthly cost is the sum of the initial purchase price divided by the life cycle in months and the average recurring monthly cost. The result is a unitless ROI value that captures the KVA of the system.

E. COTS SOFTWARE SOLUTION IMPLEMENTATION

Our software prototype for ROI analysis of Navy CCOP systems focuses on the record database because the status reports are not currently in a format that can be easily accessed without significant user interaction. Creation of an automated tool to collect the data in the various status reports into a single data source is beyond the scope of this thesis.

After the creation of a Vision Document, use cases, a System Software Requirements document and after additional meetings at SPAWAR, we determined that the only available means for extracting the data from the database was through a predefined SQL database search. Our specification required that the search input be time frame (day/month/year) with an option for specifying specific ships and systems. The search would return ship, system, and total number of outputs for the specified time frame. Additionally, the search would be accessible through a web browser and the results would be exportable into Excel. When our prototype is combined with the predefined SQL query, it will allow for near-real-time ROI analysis of CCOP systems.

Once the search was completed, we needed a way to make calculations and display our ROI results. For the purposes of our prototype, we attempted to use two COTS products: Microsoft Excel and GaussSoft Radial Viewer. Because the data from the search would already be exportable into Excel, it was relatively simple to add the additional data needed for calculations, perform the calculations, and display the results. In order to evaluate Radial Viewer, we provided our completed Excel spreadsheet to GaussSoft so they could mirror our calculations. Based on some preliminary feedback, it appears that GaussSoft would be an effective solution. Additionally, through a comparison of our prototype to GaussSoft, we determined that our prototype captured the functionality needed for near-real-time ROI analysis. Because GaussSoft is not currently accredited for use in a classified environment, we decided to use our Excel spreadsheet as our initial prototype. The prototype will be extremely useful in demonstrating our solution and determining those areas in which we should concentrate further work. Because of time and access constraints, we were unable to create production quality code resulting in a single software solution. By creating a prototype that captures the high-level functionality of our ideal system, we have a working solution for near-real-time calculation of ROI for Navy CCOP systems.

F. DETAILED DESCRIPTION OF SOFTWARE SOLUTION

This portion of the thesis provides a how-to manual for using the prototype of the software solution for near-real-time ROI analysis. The first step is Data Entry. This first step is accomplished on the Data Entry tab of the Excel spreadsheet, shown in Figure 7.

System	Initial Cost (Dollars)	Life Cycle (Months)	Monthly Recurring Cost (Dollars)	Complexity (Dollars)	Monthly Cost (Dollars)
CCOP A Ver1	$1,000,000	120	$2,500	$1,000	$10,833.33
CCOP A Ver2	$1,250,000	120	$2,500	$1,000	$12,916.67
CCOP A Ver3	$1,500,000	120	$2,500	$1,000	$15,000.00
CCOP A Ver4	$1,750,000	120	$2,500	$1,000	$17,083.33

Figure 7. Data Entry Tab of Solution Prototype

The Data Entry tab contains fields for costs, life cycle, and complexity for individual systems. The headings in red denote the required fields for each system. *System* is the name of the system and must be identical to the name that will be entered during ROI analysis. *Initial Cost* is the initial, one-time purchase price of the system. *Life Cycle* is the total amount of time that the system is expected to be used before being retired. *Recurring Cost* is the average monthly cost of keeping the system operational to include maintenance and updates. *Complexity* is the dollar value assigned to an individual output based on the knowledge embedded in the system and based on KVA analysis of TTL. The *Monthly Cost*, in yellow, is automatically calculated for later use in ROI calculations and should not be modified. Once these values are entered for all systems that will be analyzed, the user can select the *ROI Analysis* tab of the spreadsheet. This tab is shown in Figure 8.

Ship	System	Outputs	Time Frame (Days)	ROI	Complexity (Dollars)	Monthly Cost (Dollars)
Ship A	CCOP A Ver1	65,000	30	6086.49	$1,000	$10,833
Ship B	CCOP A Ver1	80,000	30	7491.29	$1,000	$10,833
Ship C	CCOP A Ver2	103,000	30	8089.47	$1,000	$12,917
Ship D	CCOP A Ver2	116,000	30	9110.60	$1,000	$12,917
Ship E	CCOP A Ver3	132,000	30	8927.32	$1,000	$15,000
Ship F	CCOP A Ver3	163,000	30	11024.12	$1,000	$15,000
Ship G	CCOP A Ver4	178,000	30	10570.45	$1,000	$17,083
Ship H	CCOP A Ver4	213,000	30	12649.11	$1,000	$17,083
				#N/A	#N/A	#N/A
Example 3	CCOP A Ver4	100,000		#DIV/0!	$1,000	$17,083
Example 2	CCOP A Ver4		30	-1.01	$1,000	$17,083
Example 1	CCOP B Ver4			#N/A	#N/A	#N/A
				#N/A	#N/A	#N/A

Figure 8. ROI Analysis Tab of Solution Prototype

The ROI Analysis tab contains fields for system(s), ship(s), total number of outputs, and length of the time frame. The headings in red denote the required fields for each entry. *Ship* is the name of the ship on which the system is installed. *System* is the name of the system and is checked against the system names entered in the Data Entry tab. If a system name does not match a name in the Data Entry tab, #N/A (Example 1) will be displayed in the ROI column. *Outputs* is the total number of outputs created by the system over the given time frame. If no value is entered for Outputs, a negative number (Example 2) will be displayed in the ROI column. *Time Frame* is the length of the time frame (in days) over which outputs are counted. If no value is entered for Time Frame, #DIV/0! (Example 3) will be displayed in the ROI column. *ROI* (in blue) is the calculated ROI value for an individual system and should not be modified by the user. *Complexity* and *Monthly Cost* are pulled from the Data Entry tab for calculation purposes and should not be modified by the user.

The other eight data values are examples of what properly entered data would look like in order to produce the calculated ROI values. The current format of this tab is subject to change upon completion of the predefined SQL search to allow for cut-and-paste once the data is exported into Excel. This will remove the need for manual entry of individual data fields.

Once the user has completed all of the fields on the ROI Analysis tab, he or she can select the Graphical Display 1 tab for a visual representation of the calculated ROI values. Figure 9 is an example of the Graphical Display 1 tab displaying the eight example data sets previously discussed.

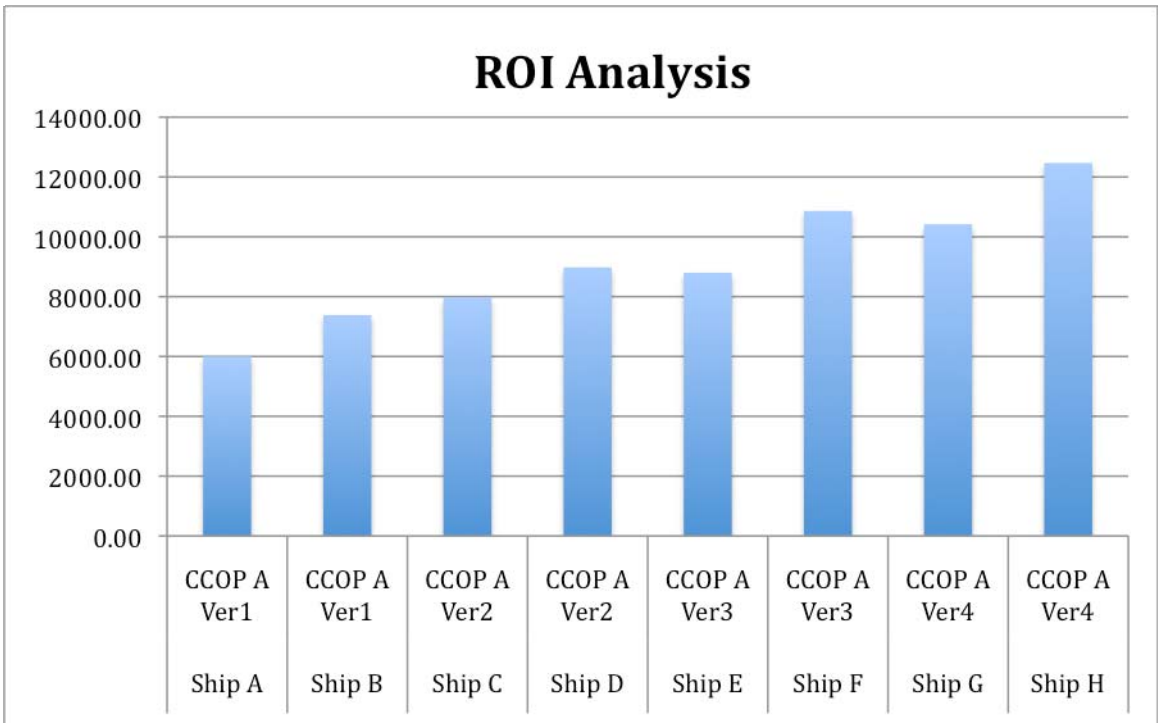

Figure 9. Graphical Display 1 Tab of Solution Prototype

The display format shown in Figure 9 presents the calculated ROI for each data set as an individual column that allows for easy comparison between individual ships and systems. The prototype contains one additional view format, shown in Figure 10.

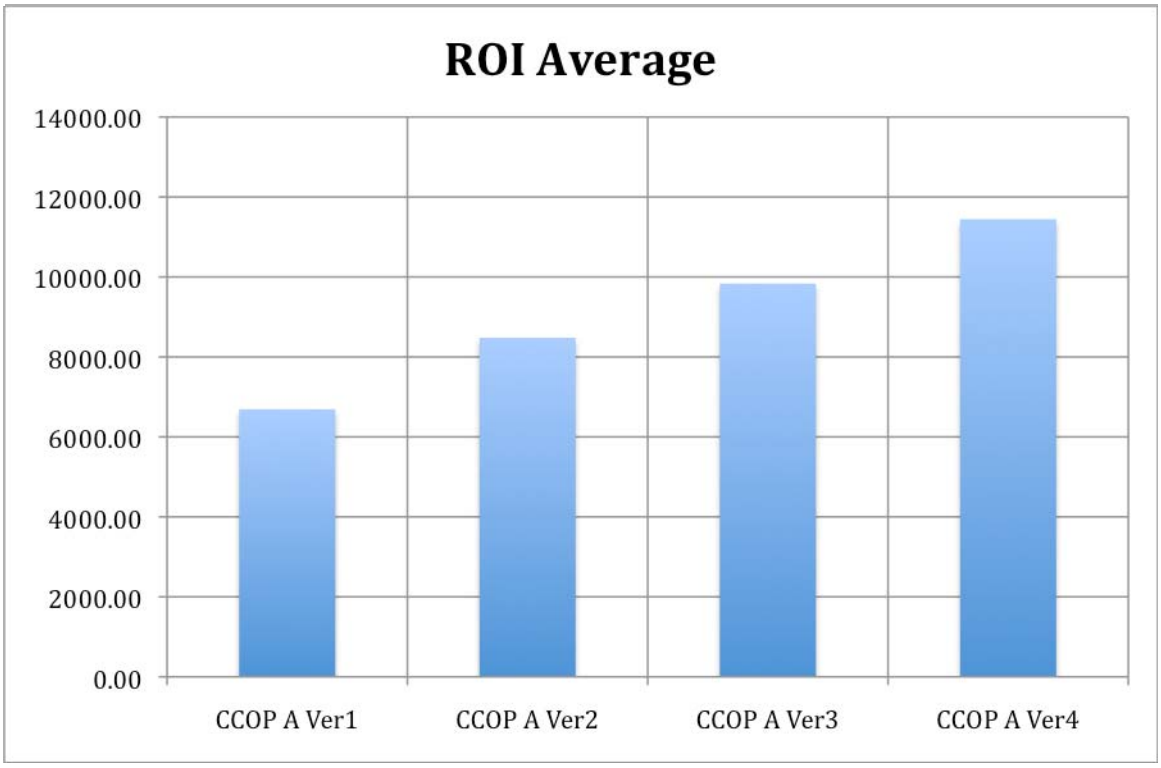

Figure 10. Graphical Display 2 Tab of Solution Prototype

The display format shown in Figure 10 presents the average ROI over all ships carrying a given system, allowing for easy comparison of different systems. From this hypothetical example, we can clearly conclude that subsequent versions of CCOP A have incrementally increasing ROIs. Because of the limitations of Excel, the data ranges for these figures must be manually changed whenever the number of data entries changes.

VI. CONCLUSION/RECOMMENDATIONS

A. SOFTWARE IMPLEMENTATION METHODOLOGY

Our software development approach successfully avoided the common software implementation issues we identified. By building upon previous interactions between developers, customers, and users, we further refined the needs of these groups during our Inception phase. As part of the requirements analysis, we created a Vision Document and use cases that laid the foundation for our System Software Requirements (SSR). The SSR itself detailed the requirements of the system and the manner in which it would operate through expanded use cases, system sequence diagrams, and system operation contracts. We conducted a risk assessment and identified those issues that could prevent us from meeting our goals. When it became apparent that a predefined database query would not be completed within our time frame, we modified our goals to focus on the calculation and display portions of a software solution. Finally, we created a working prototype to provide ROI analysis of Navy CCOP systems. The Unified Process proved to be a highly useful framework for software development.

Software implementation is not an easy or trivial process, but this research demonstrated an effective plan to implement KVA software on classified networks. We have presented the path to implementation, and Navy CCOP program managers should consider making KVA software implementation a long-term solution to provide ROI on Navy SIGINT collection systems. Need-to-know access and restricted access to databases and networks are obstacles that this research project navigated around by reevaluating our goals as necessary. Additionally, software certification and accreditation will be an issue for permanent implementation of KVA software. Utilizing software developers and system experts already operating within the CCOP program could minimize these obstacles.

Our analysis of five software implementation failures identified five major issues as well as a methodology for avoiding them. This methodology can be applied to any software implementation project. Although we recommended the Unified Process, any process that uses an iterative incremental approach to software development should help

to avoid the common pitfalls. The most important factor we identified was the incremental development of an accurate and detailed requirements document. Developers, customers, and users must approve this document before software implementation begins. The other major recommendation is the creation of a functional prototype. Additional iterations should gradually build upon the prototype so that a functional and thoroughly tested system exists at each stage of development.

B. BENEFITS OF ROI ANALYSIS

Department of Defense funding has decreased due to the spending required by two wars and an economic recession. Despite budget decreases, the demand for ISR is as high as it has ever been. Implementing software with defendable and valid KVA assumptions provides the Navy CCOP program managers with ammunition to fight for acquisition and continued program funding where it is justified. Implementing software that provides near-real-time ROI analysis is a step in the right direction to assist IT portfolio managers with difficult budgeting decisions.

Additional benefits to having a near-real-time ROI analysis is that OPNAV and CCOP program managers will not be the only customers that would gain from this information. Fleet commanders, strike group commanders, unit commanders, and CCOP system operators could be provided with remote ROI analysis to use as a measuring stick for CCOP system performance under their control. It is recommended that CCOP program managers support implementation of KVA software to provide this near-real-time ROI analysis. Also, all CCOP systems should directly input performance data into the KVA software to provide an ROI comparison of all systems. KVA assumptions established in previous studies should be revisited at implementation and again on a periodic basis. Qualitative data such as location and mission tasking should also be incorporated into the KVA software solution.

C. FOLLOW-ON RESEARCH

Follow-on research should continue to document findings and store them at the appropriate classification levels in order to prevent gaps in knowledge from previous research. Additional research should also address measures of effectiveness or outcome-

based performance. For instance, maybe the system only created 10 outputs and appeared to result in a low ROI, but perhaps 3 of the 10 times it created an output that led to the capture of a high-value target. In this scenario, low system performance might be an acceptable risk in continuing funding because of the outcomes resulting from system operation.

The next step in this research is to combine the software solution prototype with the completed predefined database query. This will allow for near-real-time ROI analysis of CCOP systems. At this point, the prototype should be demonstrated to customers and users for their evaluation and feedback. This would complete the Elaboration phase of the Unified Process started during this study. Based on customer and user feedback, the Construction phase would then be started, which would focus on combining the predefined query and production-quality calculation/display code into a single software solution. Certification and Accreditation of the GaussSoft program should also be started so that if can be evaluated using the same classified data sets.

A comparison of the status report data versus the predefined query data should also be conducted. Specifically, searches should be executed for the same time frames and ships that we identified in the status reports data set. This would allow for a comparison of ROI values from the two data sets and provide estimation of how much data is lost due to network errors. A significant difference between ROI values might mean that the status reports should replace the predefined query as a source of data for ROI calculations.

D. GENERALIZABILITY OF KVA METHOD AND ROI ANALYSIS

The KVA method and ROI analysis are not limited to providing performance data for Navy CCOP systems. These methodologies should be expanded across the fleet and the entirety of the DoD to provide ROIs for all military systems. The ability to compare system performance is invaluable for making informed budgetary decisions. This will be especially important in coming years when the military will most likely be downsized due to needed cuts in government expenditures.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. VISION DOCUMENT

A. VISION DOCUMENT

1. INTRODUCTION

This thesis envisions a software solution for the calculation of near-real-time ROI values for Navy CCOP systems using KVA analysis.

2. POSITIONING

The capability to provide CCOP system managers with a real-time analysis of system performance does not currently exist. Providing an ROI for these systems will provide a means of objectively comparing systems to support future funding allocation decisions.

3. STAKEHOLDER DESCRIPTIONS

The non-user stakeholders of the system are CCOP system managers, operators, and ship captains.

The users of the system are military and contractor personnel. These users have varying levels of computer and network knowledge. The system will need to be user-friendly to accommodate various levels of expertise.

4. PRODUCT OVERVIEW

The system will have three primary functions. The first will be to gather user-requested data from a database of records. The user will be able to specify a time period, system(s), and ship(s). The second function will be calculation of ROI values using database values and previously entered cost, life cycle, and complexity values for selected system(s). The system will store user-entered values for cost, life cycle, and complexity for individual systems. The final function will be a graphical display of the calculated ROI values in a default format. Additional display formats will also be accessible. The user will have the option of printing the graphical display.

5. SUMMARY OF SYSTEM FEATURES

- Automated Data Collection:
  - User Specified Time Period (Day/Month/Year Range)
  - User Specified System(s)
  - User Specified Ship(s)
- Storage of Cost, Life Cycle, and Complexity Data for Individual Systems
- ROI Calculations
- Graphical Display of ROI Values (various formats)

APPENDIX B. SYSTEM SOFTWARE REQUIREMENTS

A. INTRODUCTION

1. PURPOSE

This requirements document outlines the software requirements for the system. These requirements have been derived from the system use cases. The intended readers of this document are the software and systems engineers of the system and the stakeholders.

2. SCOPE

The System Software Requirements (SSR) applies to the initial vision of the system. This vision can be found in the Vision Document. The scope includes all requirements that may be implemented in the initial and follow-on versions of the system. The subset of requirements to be applied to the initial implementation is based on the use cases and the system operation contracts. Once approved, this SSR will define the baseline system requirements.

3. OBJECTIVES AND SUCCESS CRITERIA

The system will be designed to achieve the following objectives:

- Allow user selection of time frame, system(s), and ship(s) for ROI analysis.
- Automatically collect necessary data from the database based on user selections.
- Allow user entry of cost, life cycle, and complexity data for individual systems.
- Store user-entered cost, life cycle, and complexity data for use in ROI calculations as required.
- Accurately calculate ROI values for a given user selection.
- Graphically display ROI values in an easy-to-understand format.
- Provide additional display formats and the ability to print the graphic display.

4. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Please see the list of acronyms and abbreviations at the beginning of this thesis.

5. REFERENCES

The system requirements are drawn from the following sources:

a. Spivey Torres; Vision Document

b. Spivey, Torres; Use Cases

c. Spivey, Torres; Use Case Diagram

d. Spivey, Torres; System Sequence Diagrams

e. Spivey, Torres; Domain Model

f. Spivey, Torres; System Operation Contracts

B. PROPOSED SYSTEM

1. OVERVIEW

The system provides near-real-time ROI analysis of Navy CCOP systems. Users can tailor ROI analysis based on time frame, system(s), and ship(s).

2. REQUIREMENTS

Table 2 lists the requirements of the system. The requirements have been categorized according to the FURPS+ model: Functional, Usability, Reliability, Performance, Supportability, and + for ancillary and sub-factors. In table 2 - Requirements of the System, the first column is an identification number for the requirement, the second column lists the requirement itself, and the third column is a list of those operations contracts that rely upon the requirement.

Table 2. Requirements of the System

System Rqmt ID	Requirement	Operations Contract Cross Reference
Functional		
001	Provide GUI that allows point-and-click selections	C01-C13
002	Provide the capability to make selections using pull-down menus	C03, C04, C05, C11, and C12
003	Provide the ability for the user to select a time frame range at the day/month/year level of granularity	C02
004	Provide the ability for the user to select one or more systems for ROI analysis	C02
005	Provide the ability for the user to select one or more ships for ROI analysis	C02
006	Check user entries for completeness and correctness	C03
007	Highlight incomplete or incorrect user entries	C03
008	Notify the user when data entry has not been completed for a selected system(s)	C04
009	Perform a database search based on user-entered parameters	C05
010	Notify the user when the database does not contain data needed for ROI calculations	C06
011	Perform ROI calculations	C07
012	Provide the capability to graphically display ROI values	C08
013	Provide the capability for the user to select a	C09

System Rqmt ID	Requirement	Operations Contract Cross Reference
	graphical display format	
014	Provide the capability for the user to print the graphical display	C10
015	Provide the capability to return to the main menu when complete	C11 and C15
016	Provide the capability for the user to enter initial cost, recurring cost, life cycle, and complexity for individual systems	C12
017	Display and store user-entered data on individual systems (costs, life cycle, complexity)	C13
018	Notify the user when costs, life cycle, and/or complexity data is entered incorrectly	C14
019	Provide the capability for the user to enter data on multiple systems without returning to the main menu	C15
Usability		
020	Display text with a default size no smaller than 12 pt.	__
021	Make display colors modifiable by the operator	__
022	Provide a help dialog that meetisindustry standards for help functionality	__
023	Provide access to electronic documentation	__
Reliability		
024	Upon complete failure, the system should recover to the last-known good state	__
Performance		
025	Be compatible with a variety of hardware	__

System Rqmt ID	Requirement	Operations Contract Cross Reference
	configurations to allow for user needs	
026	Provide response to user actions within 1 second	__
Supportability		
027	Have patch capability to allow for future modifications for the software to adapt to new operating systems or operating system upgrades	__
028	Be easily migrated to a new operating system after an operating system upgrade	__
029	Maximize use of common object-oriented language	__

3. CONSTRAINTS

- Hardware components of the system will consist of pre-existing desktop or laptop computers running the DoD standard version of Microsoft Windows.
- Software components of the system will utilize object-oriented technology. Therefore, the system will be compatible with and adaptable to an object-oriented environment.
- The system will be developed using the Unified Process.

C. SYSTEM MODELS

1. USE CASE DIAGRAM

Figure 11 shows the use case diagram for the system. Two use cases have been defined: UC-1–ROI Analysis and UC-2–Data Entry

.

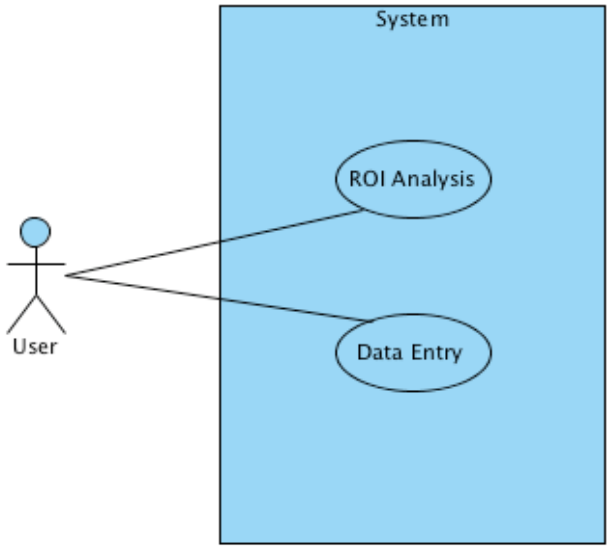

Figure 11. Use Case Diagram

2. EXPANDED USE CASE SCENARIOS

Both of the use cases have been expanded to UC-1: ROI Analysis and UC-2: Data Entry. Below are the expanded use case scenarios.

UC-1: ROI Analysis

Scope: System

Level: User goal

Primary Actor: User

Stakeholders and Interests:

- User: Wants an application that is easy to use and quickly calculates and displays ROI values.
- CCOP Program Managers: Need accurate ROI values displayed in an easy-to-understand graphical format. Additional display formats should be provided.

Preconditions:

Cost, life cycle, and complexity data have been entered for selected system(s). The database contains data on system(s) and ship(s) for the time frame selected.

Success Guarantee (or Postconditions):

ROI values are graphically displayed in an easy-to-understand format. Additional display options are provided.

Main Success Scenario (or Basic Flow):

1. User initializes the system.
2. System displays the menu with options.
3. User selects *ROI analysis option*.
4. System displays a dialog box with options for time frame, system(s) and ship(s).
5. User enters time frame, system(s) and ship(s).
6. User selects *OK*.
7. System calculates ROI values based on the selected values.
8. System displays ROI values in a graphical format.
9. User selects a different display option.
10. System displays ROI values in the selected format.
11. User selects *print*.
12. System displays print dialog box.
13. User selects *exit*.
14. System displays the main menu.

Extensions (or Alternative Flows):

7a. User enters insufficient data or data in wrong format.

1. System prompts user to check appropriate data fields.
2. User corrects fields and selects *OK*.

7b. System does not contain cost and life cycle data for selected system(s).

1. System informs user that data is not available for selected system(s).

2. User corrects fields and selects *OK*.

7c. Database does not contain data on selected system(s) and ship(s) during the time frame.

1. System informs user that data is not available for selected time frame.
2. System prompts user to check appropriate data fields.
3. User corrects fields and selects *OK*.

UC-2: Data Entry

Scope: System

Level: User goal

Primary Actor: User

Stakeholders and Interests:

User: Wants to enter cost, life cycle, and complexity data for an individual system to be used in future ROI calculations.

Preconditions:

User has cost, life cycle, and complexity data for individual system(s).

Success Guarantee (or Postconditions):

System stores cost, life cycle, and complexity data for use in future ROI calculations.

Main Success Scenario (or Basic Flow):

1. User initializes the system.
2. System displays the menu with options.
3. User selects *data entry*.
4. System displays a dialog box with options for system, initial cost, recurring cost, life cycle, and complexity.
5. User enters system, initial cost, recurring cost, life cycle, and complexity.
6. User selects *OK*.
7. System displays entered data and requests if user would like to enter data for another system.

8. User selects *no*.

9. System displays the main menu.

Extensions (or Alternate Flows):

7a. User enters insufficient data or data in the wrong format.

1. System prompts user to check appropriate data fields.

2. User corrects data and selects *OK*.

8a. User selects *yes* to enter data for another system.

1. System displays a dialog box with options for system, initial cost, recurring cost, life cycle, and complexity.

2. User enters data and selects *OK*.

3. SYSTEM SEQUENCE DIAGRAMS

Figures 12 and 13 are the system sequence diagrams for the expanded use cases UC-1–ROI Analysis and UC-2–Data Entry, respectively. The purpose of these diagrams is to illustrate the use cases in a visual format. They show in detail how the system is designed to handle certain actions.

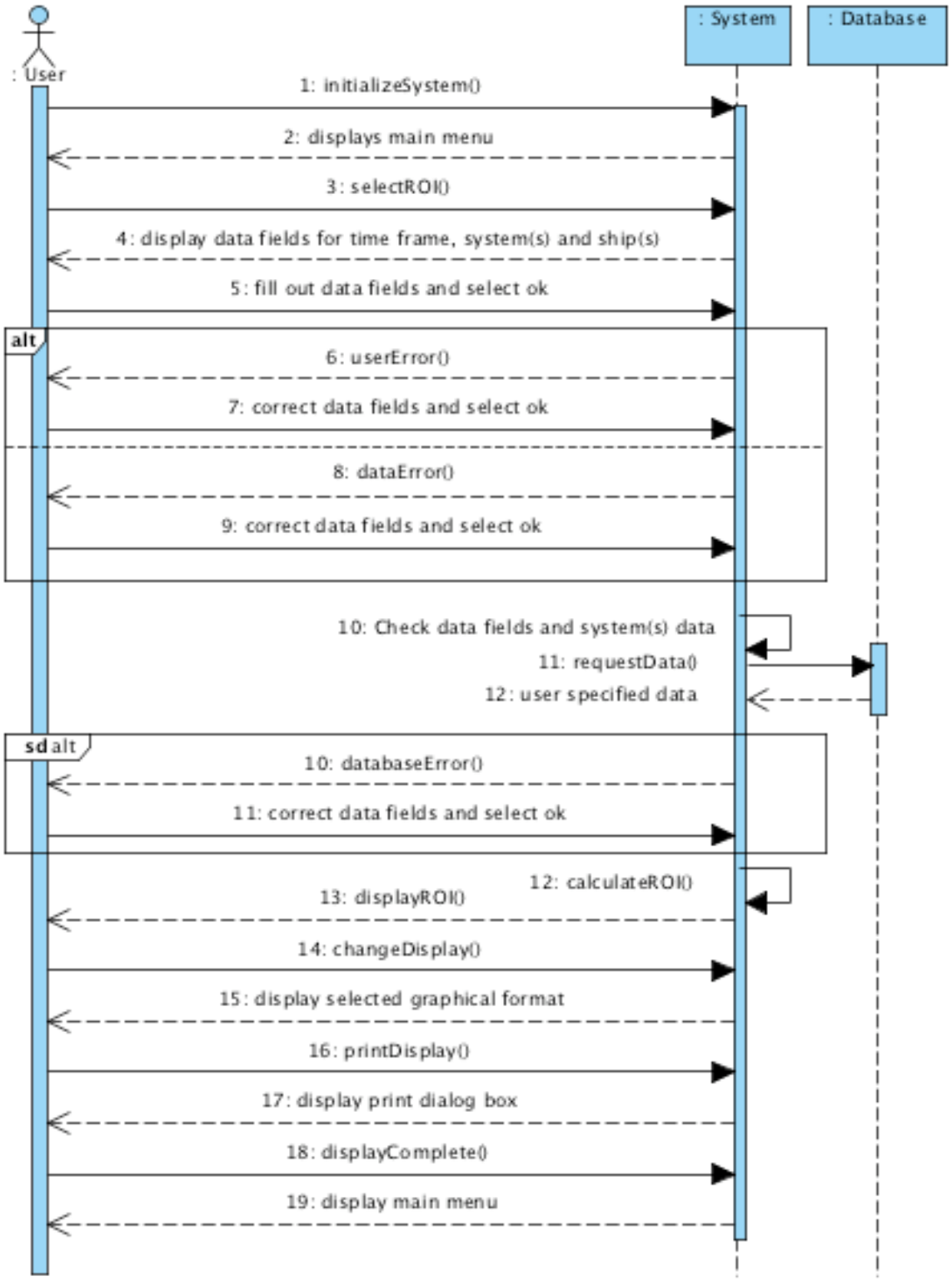

Figure 12. System Sequence Diagram for UC-1

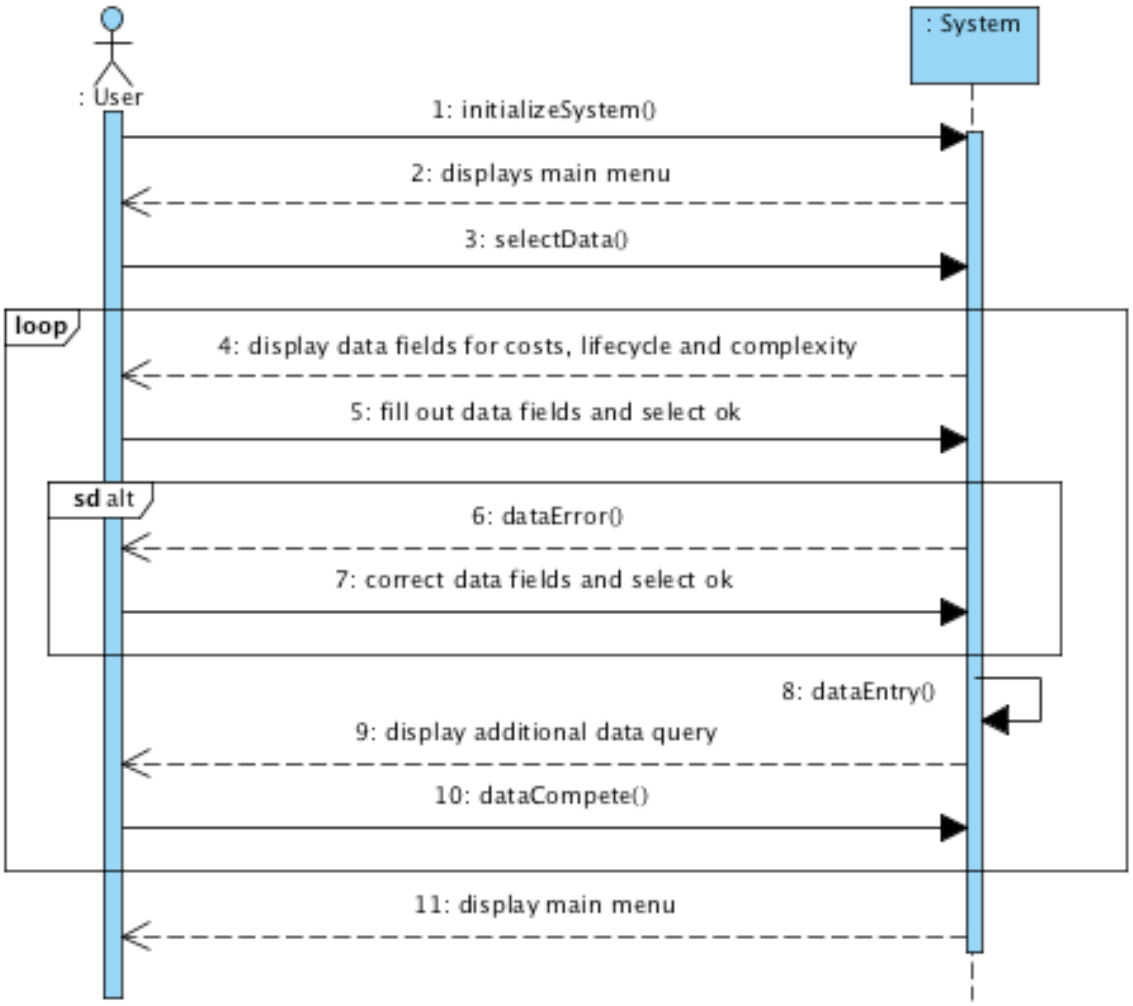

Figure 13. System Sequence Diagram for UC-2

4. DOMAIN MODEL

Figure 14 shows the domain model for the system. It is a conceptual model that illustrates the relationships between the important concepts that make up the system.

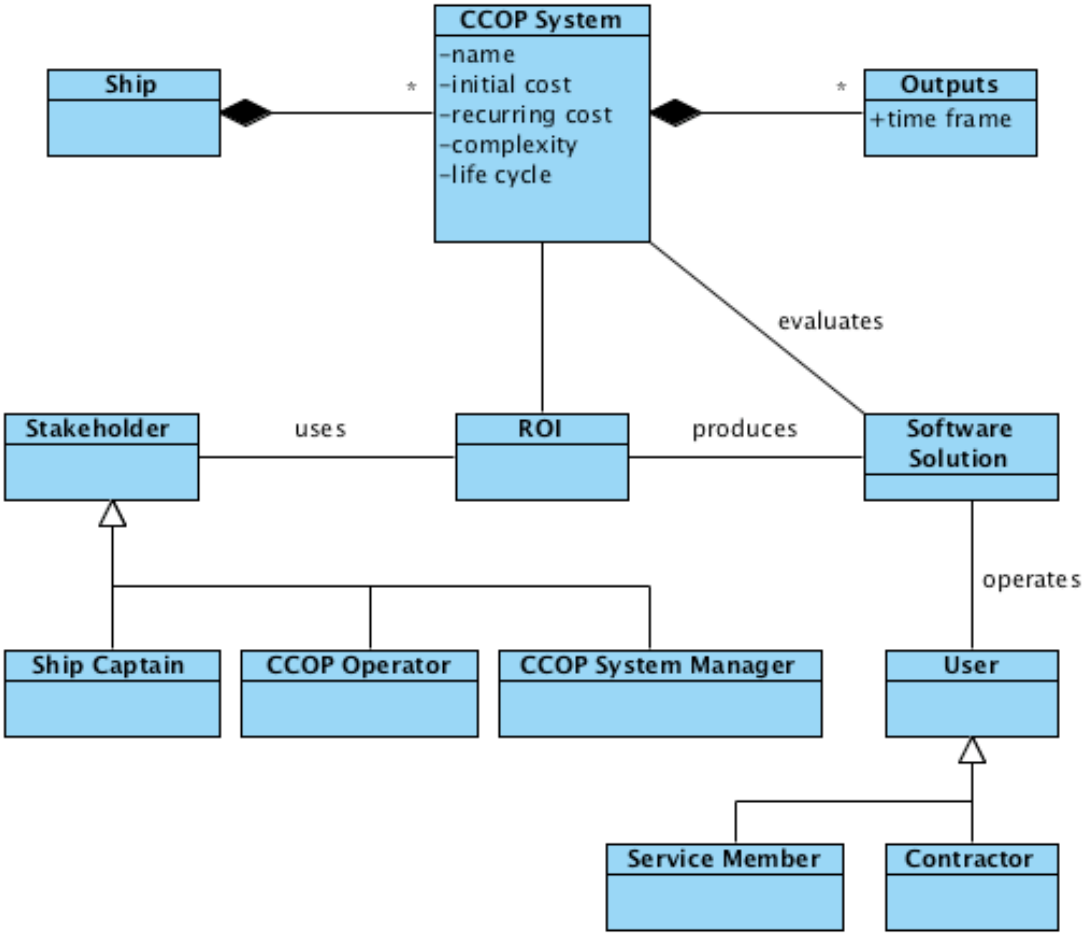

Figure 14. Domain Model for the System

5. SYSTEM OPERATION CONTRACTS

Below are the system operation contracts for the system. These identify the system state changes that occur when an operation executes.

Contract C01: initializeSystem

Operation: initializeSystem()

Cross Reference: Use Cases: ROI Analysis, Data Entry

Preconditions: - Start screen displayed

Postconditions: - Main menu options displayed

Contract C02: selectROI

Operation: selectROI()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - Main menu options displayed

Postconditions: - Dialog box displayed with time frame, system(s) and ship(s) fields

Contract C03: userError

Operation: userError()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - User has entered insufficient/incorrect information and selected *OK*

Postconditions: - Dialog box displayed with time frame, system(s) and ship(s) fields

- Insufficient/incorrect fields are highlighted

Contract C04: calcDataError

Operation: calcDataError()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - User has entered correct information and selected ok

- System does not contain cost, life cycle, and complexity data for selected system(s)

Postconditions: - Dialog box displayed with time frame, system(s) and ship(s) fields

- Statement is displayed informing user that cost and life cycle data is not available for selected system(s)

Contract C05: requestData

Operation: requestData()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - User has entered correct information and selected *OK*

- System contains cost, life cycle, and complexity data for selected system(s)

Postconditions: - User specified data is returned from database

Contract C06: databaseError

Operation: databaseError()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - User specified data is retuned from database

- Data does not contain information on user selected system(s)

Postconditions: - Dialog box displayed with time frame, system(s) and ship(s) fields

- Statement is displayed informing user that database does not contain information on selected system(s)/ship(s) for given time frame.

Contract C07: calculateROI

Operation: calculateROI()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - System has received necessary data from database

Postconditions: - ROI values are calculated and stored

Contract C08: displayROI

Operation: displayROI()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - ROI values calculated and stored

Postconditions: - ROI values are graphically displayed in default format

- Options displayed for other view formats

Contract C09: changeDisplay

Operation: changeDisplay()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - User selects new display format

Postconditions: - ROI values are graphically displayed in selected format

Contract C10: printDisplay

Operation: printDisplay()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - ROI values are graphically displayed in selected format

- User selects print option

Postconditions: - Print dialog box is displayed

Contract C11: displayComplete

Operation: displayComplete()

Cross Reference: Use Cases: ROI Analysis

Preconditions: - ROI values are graphically displayed in selected format

- User selects *exit*

Postconditions: - Main menu options displayed

Contract C12: selectData

Operation: selectData()

Cross Reference: Use Cases: Data Entry

Preconditions: - Main menu options displayed

Postconditions: - Dialog box displayed with initial cost, recurring cost, and life cycle fields.

Contract C13: dataEntry

Operation: dataEntry()

Cross Reference: Use Cases: Data Entry

Preconditions: - User has entered data and selected *OK*

Postconditions: - Entered data is displayed and saved for future calculations.

- User is queried to enter additional data

Contract C14: dataError

Operation: dataError()

Cross Reference: Use Cases: Data Entry

Preconditions: - User has entered insufficient/incorrect data and selected *OK*

Postconditions: - Dialog box displayed with initial cost, recurring cost, and life cycle fields

- Insufficient/incorrect fields are highlighted

Contract C15: dataComplete

Operation: dataComplete()

Cross Reference: Use Cases: Data Entry

Preconditions: - User is queried to enter additional data and selects *no*

Postconditions: - Main menu options displayed

APPENDIX C. SOFTWARE DESIGN SPECIFICATION

A. INTRODUCTION

1. PURPOSE

The purpose of this software design specification is to document the first iteration of the system. Our intent is to produce a working prototype based on our operation contracts and to test it against the system requirements document. The intended readers of this document are the software engineers of the system and the stakeholders.

2. OBJECTIVES AND SUCCESS CRITERIA

The system will be designed to achieve the following objectives:

- Allow user selection of time frame, system(s), and ship(s) for ROI analysis.
- Automatically collect necessary data from the database based on user selections.
- Allow user entry of cost, life cycle, and complexity data for individual systems.
- Store user-entered cost, life cycle, and complexity data for use in ROI calculations as required.
- Accurately calculate ROI values for a given user selection.
- Graphically display ROI values in an easy-to-understand format.
- Provide additional displays formats and the ability to print the graphic display.

3. REFERENCES

The system requirements are drawn from the following sources:

a. Spivey Torres; Vision Document

b. Spivey, Torres; Use Cases

c. Spivey, Torres; Use Case Diagram

d. Spivey, Torres; System Sequence Diagrams

e. Spivey, Torres; Domain Model

f. Spivey, Torres; System Operation Contracts

g. Spivey, Torres; Software Requirements, p. 62

B. ARCHITECTURAL DESIGN

1. SCOPE

The Software Design Specification (SDS) applies to the System Software Requirements (SSR). This can be found in reference *g*, Software Requirements.

The scope includes all requirements that may be implemented in the initial and follow-on versions of the system. The subset of requirements to be applied to the initial implementation is based on references *b*, Use Cases, and *f*, Operations Contracts.

2. DESIGN GOALS

We envision a single software solution for near-real-time ROI analysis of Navy CCOP system using the KVA methodology.

The design goals include the desirable attributes of Version 1 of the system. These attributes are derived from the non-functional requirements. The requirement priority order is: Functional, Usability, Performance, Reliability, and Supportability. Version 1 of the system is a fully functional graphic user interface (GUI) that provides implementation of operation contracts 1-15 in support of Use Cases 1 and 2. GUI priority order is as follows: main menu interface, data entry interface, ROI analysis interface, ROI display interface, and print interface.

3. LOGICAL ARCHITECTURE

Figure 15 depicts the logical architecture of the system.

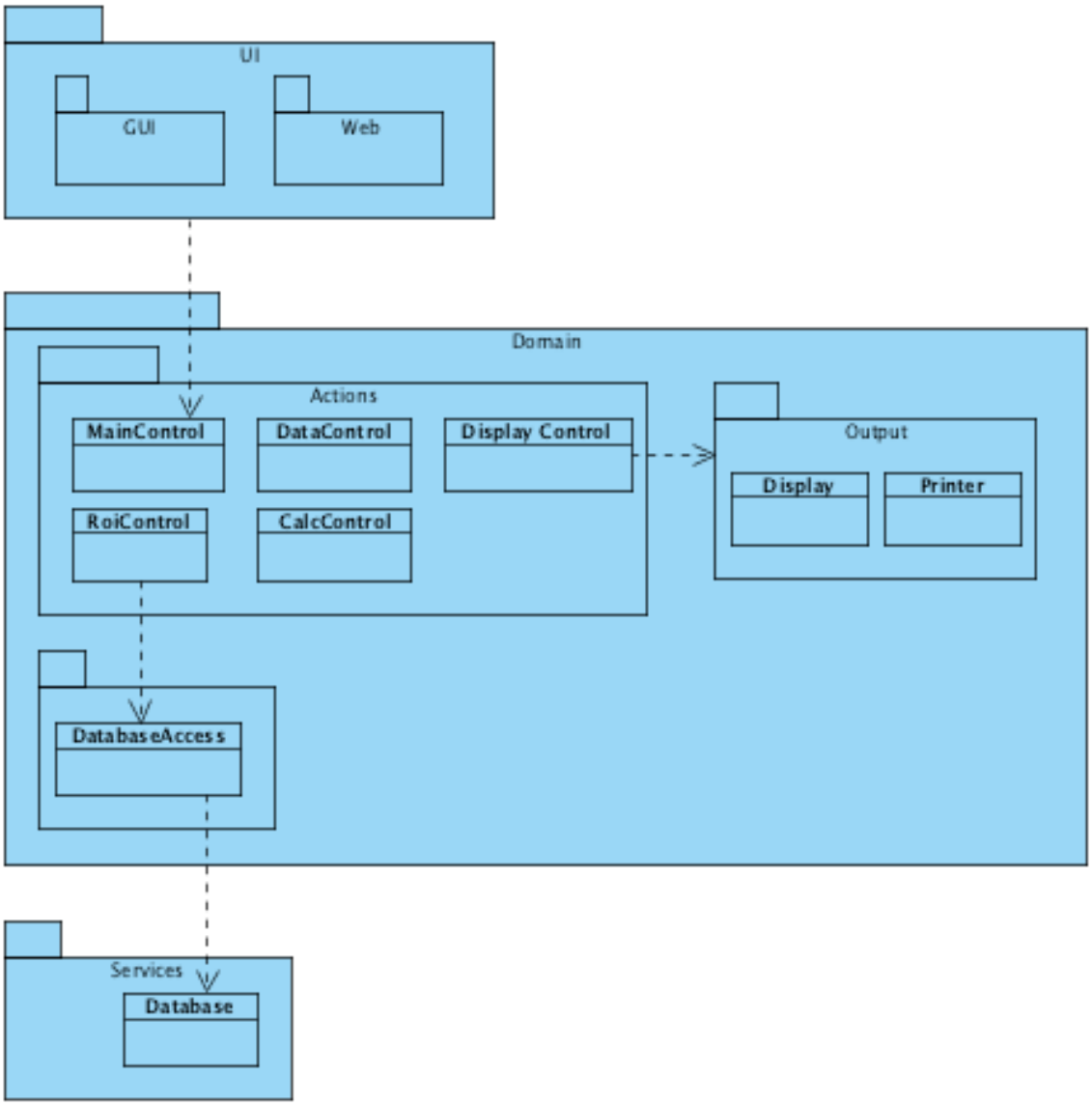

Figure 15. Logical Architecture of the System

4. RATIONALES

Our logical architecture attempts to simplify our project in a cohesive way. We decided against an application layer because we will only have one user interface. Our implementation uses an open architecture or relaxed layer architecture, which allows a layer to use features from many lower layers. This hopefully results in a more efficient and compact code that will allow modification ease. We have decided our logical architecture will be layers modeled as Unified Modeling Language (UML) packages, and

we will include three layers. The UI layer is modeled as a package named *UI*, the Domain layer as *Domain*, and the Services layer as *Services*. Our domain layer attempts to encompass all requirements specified in our use cases, system sequence diagrams (SSDs), and domain model. The Interface and Services layers are independent enough for possible code reuse.

C. OBJECT DESIGN

1. SCOPE

The scope includes all requirements that may be implemented in the initial and follow-on versions of the system. The subset of requirements to be applied to the initial implementation is based on references *b*, Use Cases, and *f*, Operations Contracts.

2. INTERACTION DIAGRAMS

Because all 15 of our operation contracts are relatively simple, none of them was chosen for further design.

3. DESIGN CLASS DIAGRAM

Figure 16 shows a Design Class Diagram for the system. It incorporates fleshed out class information derived from all 15 operation contracts.

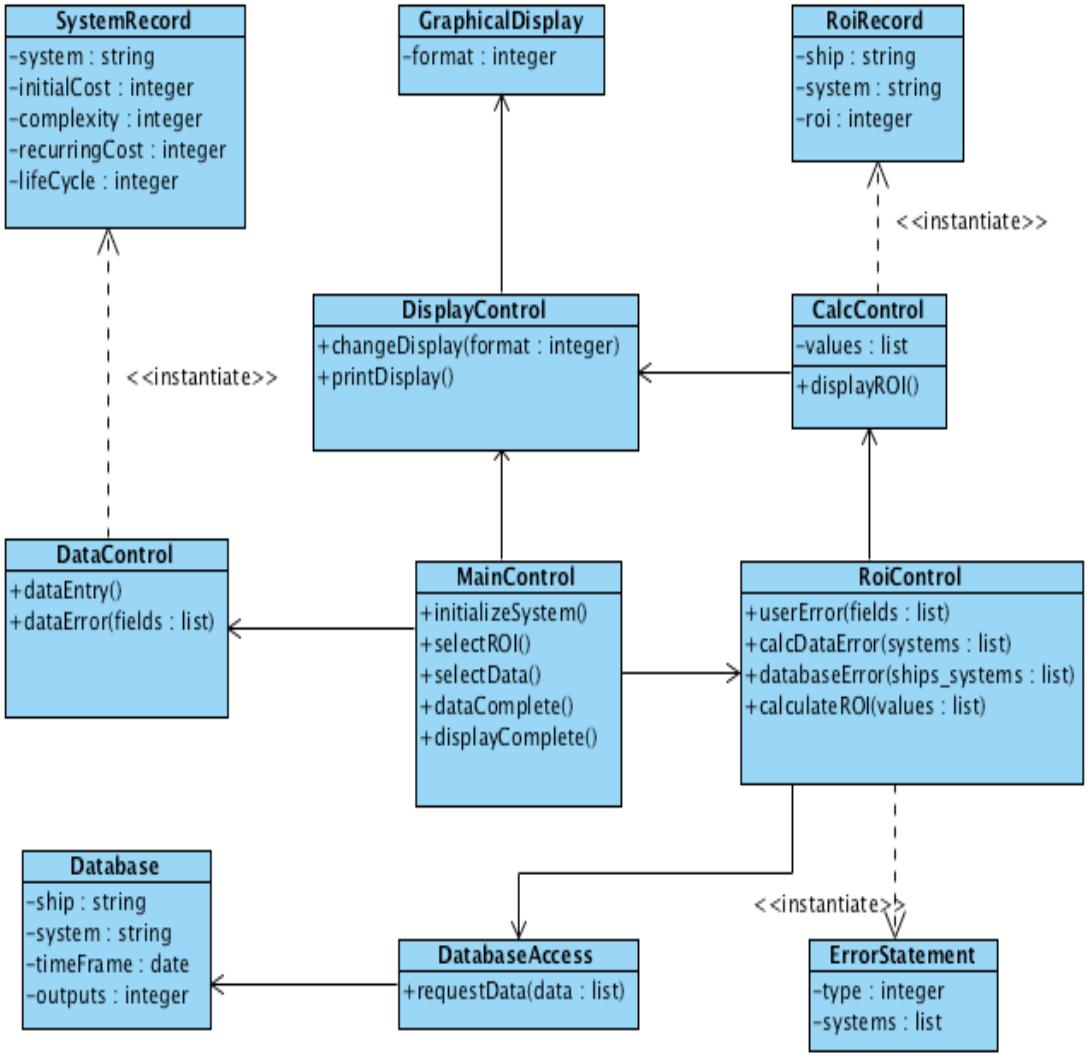

Figure 16. System Design Class Diagram

D. DATA DICTIONARY

Table 3. Data Dictionary

Term	Definition and Information	Format	Default
ship	Ship name	string	
system	System name	string	
timeFrame	Data time group	date	
outputs	Number of outputs	integer	
values	Information needed for ROI calc	list	
ships_systems	Ships/systems not in database	list	
data	Data needed from database	list	
systems	Systems without data entry complete	list	
type	Kind of error statement	integer	
roi	Calculated ROI value	integer	
format	Display format	integer	
initialCost	Initial system cost (dollars)	integer	
recurringCost	Monthly cost of system (dollars)	integer	
lifecycle	Expected system lifespan (months)	integer	
complexity	KVA evaluation of system	integer	

LIST OF REFERENCES

Arnold, D. N. (1996). Two disasters caused by computer arithmetic errors. Retrieved from http://www.ima.umn.edu/~arnold/455.f96/disasters.html

Barker, C. (2007). The top 10 IT disasters of all time [Web log post]. Retrieved from http://www.zdnet.com/news/the-top-10-it-disasters-of-all-time/177729

Department of Defense (DoD). (2005). *Information technology portfolio management* (DoD Directive 8115.01). Washington, DC: Government Printing Office.

Department of Defense (DoD). (2006). *Information technology portfolio management implementation* (DoD Instruction 8115.02). Washington, DC: Government Printing Office.

Department of the Navy, Office of the Chief of Naval Operations. (2003). *Naval transformation roadmap 2003: Assured access & power projection...from the sea*. Washington, DC: Author.

Dershowitz, N. (n.d.). Software horror stories. Retrieved from http://www.cs.tau.ac.il/~nachumd/horror.html

Dutchguilder. (2007). File:Development-iterative.gif. In *Wikipedia: The free encyclopedia*. Retrieved on September 2010, from http://en.wikipedia.org/wiki/File:Development-iterative.gif#filelinks

Gleick, J. (1996). A bug and a crash. Retrieved from http://www.around.com/ariane.html

Goldstein, H. (2005). Who killed the virtual case file? Retrieved from http://spectrum.ieee.org/computing/software/who-killed-the-virtual-case-file/0

Government Accountability Office (GAO). (2007). *Intelligence, surveillance, and reconnaissance: Preliminary observations on DOD's approach to managing requirements for new systems, existing assets, and systems development*. Washington, DC: Author.

Homer, J. B. (2009). *Collecting, retrieving and analyzing knowledge value added (KVA) data from U.S. Navy vessels afloat* (Master's thesis, Naval Postgraduate School). Retrieved from http://www.acquisitionresearch.net/

Housel, T. J., & Bell, A. H. (2001). *Measuring and managing knowledge*. Boston, MA: McGraw-Hill/Irwin.

Lambeth, I. D., & Clapp, H. N. (2007). *Using knowledge value added (KVA) for evaluating*

cryptologic IT capabilities: Trial implementation (Master's thesis, Naval Postgraduate School). Retrieved from http://www.acquisitionresearch.net/

Larman, C. (2005). *Applying UML and Patterns: An introduction to object-oriented analysis and design and iterative development*. Upper Saddle River, NJ: Pearson Education.

Leveson, N., & Turner, C. S. (1993). An investigation of Therac-25 accidents. Retrieved from http://courses.cs.vt.edu/~cs3604/lib/Therac_25/Therac_1.html

Lloyd, R. (1999). Metric mishap caused loss of NASA orbiter. Retrieved from http://www.cnn.com/TECH/space/9909/30/mars.metric.02/

Martin, T. (2008, February 12). 20 famous software disasters [Web log post]. Retrieved from http://www.devtopics.com/20-famous-software-disasters/

Rios, C. G., Jr. (2005). *Return on investment analysis of information warfare systems* (Master's thesis, Naval Postgraduate School). Retrieved from http://www.acquisitionresearch.net/

SoftwareMag.com. (2004, January 15). Standish: Project success rates improved over 10 years. *Software Magazine: The IT Software Journal*. Retrieved from http://www.softwaremag.com/L.cfm?Doc=newsletter/2004-01-15/Standish

Stokes, J. (n.d.). Slashing the federal IT budget: Can someone (please) help the FBI? Retrieved from http://arstechnica.com/tech-policy/news/2010/06/cutting-the-federal-it-budget-could-someone-help-the-fbi.ars

Szilagyi, J. G. (n.d.) Bank of America's MASTERNET system: A case study in risk assessment. Retrieved from http://csse.usc.edu/classes/cs510_2000/notes/masternet.pdf

Weiss, T. R. (2005). United axes troubled baggage system at Denver airport. Retrieved from http://www.computerworld.com/s/article/102405/United_axes_troubled_baggage_system_at_Denver_airport?taxonomyId=73&pageNumber=1

2003 - 2010 SPONSORED RESEARCH TOPICS

Acquisition Management

- Acquiring Combat Capability via Public-Private Partnerships (PPPs)
- BCA: Contractor vs. Organic Growth
- Defense Industry Consolidation
- EU-US Defense Industrial Relationships
- Knowledge Value Added (KVA) + Real Options (RO) Applied to Shipyard Planning Processes
- Managing the Services Supply Chain
- MOSA Contracting Implications
- Portfolio Optimization via KVA + RO
- Private Military Sector
- Software Requirements for OA
- Spiral Development
- Strategy for Defense Acquisition Research
- The Software, Hardware Asset Reuse Enterprise (SHARE) repository

Contract Management

- Commodity Sourcing Strategies
- Contracting Government Procurement Functions
- Contractors in 21st-century Combat Zone
- Joint Contingency Contracting
- Model for Optimizing Contingency Contracting, Planning and Execution
- Navy Contract Writing Guide
- Past Performance in Source Selection
- Strategic Contingency Contracting
- Transforming DoD Contract Closeout
- USAF Energy Savings Performance Contracts
- USAF IT Commodity Council
- USMC Contingency Contracting

Financial Management

- Acquisitions via Leasing: MPS case

- Budget Scoring
- Budgeting for Capabilities-based Planning
- Capital Budgeting for the DoD
- Energy Saving Contracts/DoD Mobile Assets
- Financing DoD Budget via PPPs
- Lessons from Private Sector Capital Budgeting for DoD Acquisition Budgeting Reform
- PPPs and Government Financing
- ROI of Information Warfare Systems
- Special Termination Liability in MDAPs
- Strategic Sourcing
- Transaction Cost Economics (TCE) to Improve Cost Estimates

Human Resources

- Indefinite Reenlistment
- Individual Augmentation
- Learning Management Systems
- Moral Conduct Waivers and First-tem Attrition
- Retention
- The Navy's Selective Reenlistment Bonus (SRB) Management System
- Tuition Assistance

Logistics Management

- Analysis of LAV Depot Maintenance
- Army LOG MOD
- ASDS Product Support Analysis
- Cold-chain Logistics
- Contractors Supporting Military Operations
- Diffusion/Variability on Vendor Performance Evaluation
- Evolutionary Acquisition
- Lean Six Sigma to Reduce Costs and Improve Readiness
- Naval Aviation Maintenance and Process Improvement (2)
- Optimizing CIWS Lifecycle Support (LCS)

- Outsourcing the Pearl Harbor MK-48 Intermediate Maintenance Activity
- Pallet Management System
- PBL (4)
- Privatization-NOSL/NAWCI
- RFID (6)
- Risk Analysis for Performance-based Logistics
- R-TOC AEGIS Microwave Power Tubes
- Sense-and-Respond Logistics Network
- Strategic Sourcing

Program Management

- Building Collaborative Capacity
- Business Process Reengineering (BPR) for LCS Mission Module Acquisition
- Collaborative IT Tools Leveraging Competence
- Contractor vs. Organic Support
- Knowledge, Responsibilities and Decision Rights in MDAPs
- KVA Applied to AEGIS and SSDS
- Managing the Service Supply Chain
- Measuring Uncertainty in Earned Value
- Organizational Modeling and Simulation
- Public-Private Partnership
- Terminating Your Own Program
- Utilizing Collaborative and Three-dimensional Imaging Technology

A complete listing and electronic copies of published research are available on our website: www.acquisitionresearch.org

THIS PAGE INTENTIONALLY LEFT BLANK

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CALIFORNIA 93943

www.acquisitionresearch.org