2008-12

Application of Real Options theory to software engineering for strategic decision making in software related capital investments

Olagbemiro, Albert O.

Monterey, California. Naval Postgraduate School, 2008.

https://hdl.handle.net/10945/10317

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

APPLICATION OF REAL OPTIONS THEORY TO SOFTWARE ENGINEERING FOR STRATEGIC DECISION MAKING IN SOFTWARE RELATED CAPITAL INVESTMENTS

by

Albert O. Olagbemiro

December 2008

Dissertation Co-Supervisors: Man-Tak Shing
Johnathan Mun

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2008	**3. REPORT TYPE AND DATES COVERED** Dissertation	
4. TITLE AND SUBTITLE: Application of Real Options Theory to Software Engineering for Strategic Decision Making in Software Related Capital Investments		**5. FUNDING NUMBERS** **Project #:F08-023, JON: RGB58**	
6. AUTHOR(S) Albert O. Olagbemiro			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Acquisition Research Program OUSD 08 Graduate School of Business and Public Policy Naval Postgraduate School Monterey, CA 93943		**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

As it stands today, software is the major expense in software-intensive weapons systems. Therefore software is, and should be treated as a capital investment and an approach emphasizing a strategic investment methodology in its acquisition is necessary. The strategic flexibility in software engineering decisions can be valued as a portfolio of options or real assets, much akin to options on financial securities which have real economic value under uncertainty. This approach would emphasize the linking of program management decisions to current and future unknown situations within the stipulated parameters of cost, schedule and functionality thus giving the managers a set of choices or options. This dissertation describes a strategic decision-making process that is based on the general concepts of initiating the software acquisition process with a situation assessments phase, identifying un-resolvable high-level uncertainties, generating the appropriate strategic actions and deriving the benefits or value created either explicitly or in the form of Real Options. We present a framework based on Real Options theory to allow decision makers to better balance customer requirements as dictated by operational needs within financial viability and schedule constraints through the identification, valuation and optimization of strategic decision pathways created in the form of Real Options. We apply the framework to the software component (Future Combat Systems Network) of the U.S. Army Future Combat System (FCS). Our study found that when properly formulated, a Real Options approach could be used as an effective risk management tool to guide decision-making at the software acquisition level further complementing the risk-driven spiral development approach currently being utilized in the U.S. Department of Defense (DoD) evolutionary acquisition model.

14. SUBJECT TERMS Real Options, Strategic Investments, Software Acquisitions, Risk Management, Software Engineering			**15. NUMBER OF PAGES** 199
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

APPLICATION OF REAL OPTIONS THEORY TO SOFTWARE ENGINEERING FOR STRATEGIC DECISION MAKING IN SOFTWARE RELATED CAPITAL INVESTMENTS

Albert O. Olagbemiro
Captain, United States Air Force (Reserve)
B.S., University of Maryland Baltimore County, 1998
M.S., Johns Hopkins University, 2000
M.B.A., Johns Hopkins University, 2002

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2008

Author: _____
Albert O. Olagbemiro

Approved by:

_____	_____
Man-Tak Shing, Associate Professor of Computer Science, Dissertation Co-Supervisor and Committee Chairman	Johnathan Mun, Research Professor of Information Sciences Dissertation Co-Supervisor
_____	_____
Mikhail Auguston Associate Professor of Computer Science	Tarek Abdel- Hamid Professor of Information Sciences

_____
Bret Michael, Professor of Computer Science and Electrical and Computer Engineering

Approved by: _____
Peter Denning, Chair, Department of Computer Science

Approved by: _____
Douglas Moses, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As it stands today, software is the major expense in software-intensive weapons systems. Therefore software is, and should be treated as a capital investment and an approach emphasizing a strategic investment methodology in its acquisition is necessary. The strategic flexibility in software engineering decisions can be valued as a portfolio of options or real assets, much akin to options on financial securities which have real economic value under uncertainty. This approach would emphasize the linking of program management decisions to current and future unknown situations within the stipulated parameters of cost, schedule and functionality thus giving the managers a set of choices or options.

This dissertation describes a strategic decision-making process that is based on the general concepts of initiating the software acquisition process with a situation assessments phase, identifying un-resolvable high-level uncertainties, generating the appropriate strategic actions and deriving the benefits or value created either explicitly or in the form of Real Options. We present a framework based on Real Options theory to allow decision makers to better balance customer requirements as dictated by operational needs within financial viability and schedule constraints through the identification, valuation and optimization of strategic decision pathways created in the form of Real Options.

We apply the framework to the software component (Future Combat Systems Network) of U.S. Army Future Combat System (FCS). Our study found that when properly formulated, a Real Options approach could be used as an effective risk management tool to guide decision-making at the software acquisition level further complementing the risk-driven spiral development approach currently being utilized in the U.S. Department of Defense (DoD) evolutionary acquisition model.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The United States Air Force afforded me an incredible opportunity to earn a Ph.D. in Software Engineering from the Naval Postgraduate School, the flagship institution of the United States Navy for which I am truly indebted and forever grateful. However I could not have achieved this milestone in my life and my career without the tremendous support I received from the first-rate professors, students, and friends who guided me through this journey, most importantly the incredible support I received from my family.

I wish to thank my dissertation committee members Professor Man-Tak Shing, Professor Bret Michael, Professor Mikhail Auguston, Dr. Johnathan Mun and Professor Tarek Abdel-Hamid, for their time, insightful guidance and remarkable patience in teaching and mentoring me in my research. I offer special thanks and gratitude to Prof. Shing and Dr. Mun for all of the extra hours they devoted to my research. To Prof. Shing, your constant guidance and thought-prodding questions was instrumental to my staying on the right track during the course of my study. To Dr. Mun, your constant guidance and long discussions we had on Real Options were critical to my success and without your guidance I would have been lost. To Prof. Auguston and Prof. Abdel-Hamid, your support and different perspectives helped shape my studies. I also offer special thanks to Prof. Shing and Prof. Michael for all their efforts to ensure I received the necessary funding to cover my studies.

To LTC Tom Cook (PhD), whom I first met at Missile Defense Agency and later went on to earn his PhD at the Naval Postgraduate School, it was always great to work with you both as a superior officer at Missile Defense Agency and as a student at the Naval Postgraduate School. Your guidance and mentoring has been invaluable throughout my studies at the Naval Postgraduate School

I would also like to acknowledge and thank Professor Richard Riehle and Professor Osmundson for the knowledge they impacted on me through their courses. This knowledge definitely provided me with the foundations to conduct me research.

I would also like to thank my superiors at the 459th Air Refueling Wing, Andrews Air Force Base, Maryland, notable Lt Col Dale Bateman for his constant, mentoring and guidance. Your constant questions on my progress served to keep me on track while balancing operational requirements at the Squadron. Thanks to Capt Justin O'Brien, a dear colleague whom I have served with in both peacetime and wartime deployments. Your interest in my progress and constant pulse checks was extremely commendable.

I thank my Mother and Father, both distinguished academics in their own rights for instilling in me the value of education very early on life. Your constant prayers and guidance and encouragement have kept me going and I am who I am today due to values and beliefs you instilled in me and I am confident that this achievement serves as an indication of your efforts and makes you proud of me. I must also thank my brothers Bamiji and Kolawole, as well as my sisters, Yetunde and Victoria for their support and encouragement during my time in school.

Most importantly, I would like to thank my most precious wife Neema and adorable son Alexanderpaul for their patience, support and steadfast love as I burnt the midnight oil in pursuit of this endeavor. Your love, encouragement and words of wisdom and sacrifices made me overcome the hurdles and challenges I faced and words alone cannot explain how much I appreciate you and your support and I am truly indebted to you.

To my son Alexanderpaul, the little boy in my life, your smiles and hugs always make me remember why I began this journey and I dedicate this dissertation to you with the hopes that you would someday continue what has now become a family legacy of the pursuit of intellectual curiosity and academic excellence.

I. INTRODUCTION

A. MOTIVATION

Software engineering is defined as an engineering discipline concerned with all aspects of software production. Given this definition, software in itself is a complex intangible artifact, acquired and produced via a set of decision-making activities integrated with a software engineering process with the goal being to deliver a product that meets the customers' needs and requirements within the stipulated cost and schedule constraints. The role of software as a "technology platform" in U.S. Department of Defense (DoD) weapons systems has steadily increased over the last 40 years (Figure 1) jumping from providing a mere 8% of weapons systems functionality in 1960 to providing 80% of weapons systems functionality in 2000.

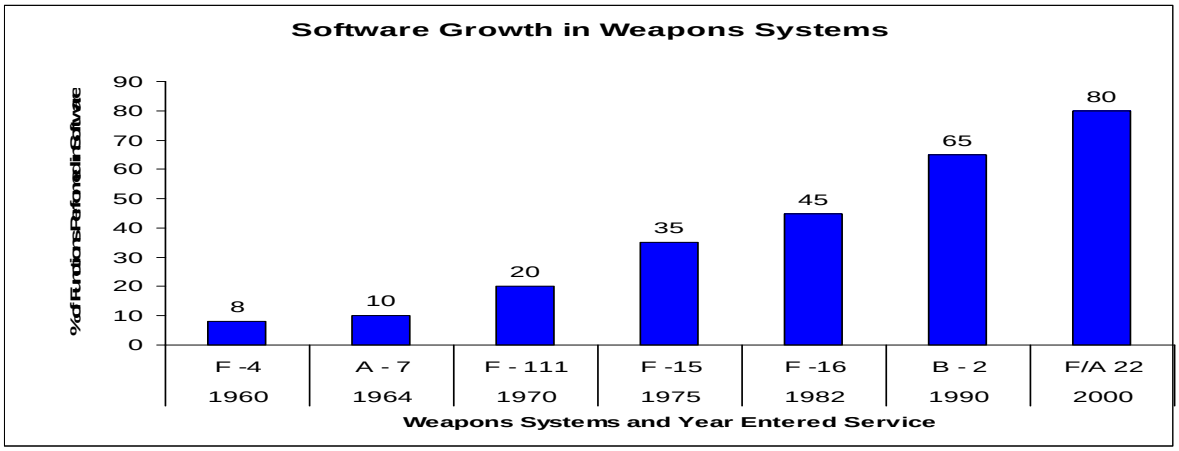


Figure 1. Software Growth in Weapons Systems (From: [10]).

Thus, considering the immense presence and ever-increasing role which software plays not only in weapons systems, but in our every day lives, industry and government, software can be viewed as an inexhaustible technological resource, vital and necessary for the good functioning and evolution of modern day society [4].

The software acquisition lifecycle, which encapsulates the activities related to its procurement, development, implementation and subsequent maintenance, continues to present challenges to software executives and program managers due to increasingly complex organizational requirements and the ever increasing role which software plays in U.S. Department of Defense (DoD) weapons systems. Various factors and considerations, most of which are complex in nature compound the software acquisition process, factors which present themselves in the form of "uncertainties", and which have the potential of introducing risks if the uncertainties are not adequately addressed and or resolved. Consequently, an integrated decision-making approach that explicitly addresses the risks associated with the pertinent software acquisition activities of procurement methods, software development approaches, implementation strategy and future evolution of the software needs to be considered to better guide the software acquisition decision-making process.

B. STATEMENT OF THE PROBLEM

In the U.S. DoD, technology acquisitions in the form of software intensive weapons systems serves as the cornerstone of the transformation strategy currently adopted by the U.S. Military in its efforts to modernize its fleet of weapons systems for future conflicts. However, the benefits of these force "enablers" continue to be plagued by massive cost and schedule overruns. The resulting impact has often led to a reduction in the scope of desired functionality as depicted in Table 1 below which leaves the war-fighters needs unfulfilled.

Program	Initial Investment	Initial Quantity	Latest Investment	Latest Quantity	% Unit Cost Increase	% Quantity Decrease
Joint Strike Fighter	$189.8 billion	2,866 aircraft	$206.3 billion	2,459 aircraft	26.7	14.2
Future Combat Systems	$92 billion	18 System	$163.7 billion	14 systems	54.4	22.3
F-22A Raptor	$81.1 billion	648 aircraft	$65.4 billion	181 aircraft	188.7	72.1

Table 1. **Program Management Failures of Top Three Major Weapons Systems.** [1]

[1] Numbers were complied from various GAO reports and were current as of 2007.

This dilemma is highlighted in a 2007 interview of the Army's Program Executive Officer (PEO) for Ammunition, in which "the ability to acquire and maintain, safe, reliable supportable and modifiable software systems which met user requirements in an environment of rapid technological advances" was identified as their biggest challenge in software acquisitions [5]. Furthermore, the U.S. Government Accountability Office (GAO) responsible for reviewing weapon systems investments found consistent problems of cost increases, schedule delays, and performance shortfalls exacerbated by factors such as pressure on program managers to promise more than they could deliver. These concerns infer a resounding theme which continues to be resonated within the software acquisition community: ***Meeting customer requirements within cost and schedule constraints***. This has led to calls for reform in the DoD acquisition strategy and the incorporation of broad improvement strategies which embraces the best practices of software-intensive systems acquisition from the commercial sector.

Balancing the satisfaction of a customer's ever-changing requirements within the realms of meeting both current and future uncertain operational needs against the costs and schedule constraints poses a cumbersome challenge to the software executive, thereby making software-investments a very risky venture; risky in the sense that software engineering and investment decisions are plagued by uncertainties which more often than not leads to varying degrees of risk ranging from operational shortfalls to cost and schedule overruns.

The inefficiencies of current management techniques as shown in the acquisition failures (Table 1) (failure to meet war-fighter's needs on time and schedule) highlight the needs of new management approaches that proactively plan for, and factor in uncertainty into their acquisition strategy. This is because the acquisition of software, its development and the operational use of the software are all dominated by human action, human judgment and decision making, and inevitably human error [2]. The outcome is, therefore, often uncertain and unpredictable, and leads to unavoidable uncertainties [2]. Thus the highly uncertain nature of software investment decisions makes the management of uncertainties a major issue in software engineering. The work of Ziv and

Richardson, which led to the proposal of the *"Uncertainty Principle in Software Engineering"* (UPSE), laid out the groundwork for identifying the software development uncertainties facing the software project manager and the Real Options approach proposed in this study is an attempt to identify, quantify, value, hedge and manage the uncertainties which introduces both technical and managerial risks in the form of product quality and cost and schedule overrun.

C. RESEARCH OBJECTIVES

This research takes a holistic approach towards addressing both the technical and managerial risks which plague software acquisitions through the application and employment of techniques based on financial options theory (Real Options). As it stands today, software is the major expense in software intensive systems. Therefore software is, and should be treated as a capital investment and an approach emphasizing a strategic investment methodology in its acquisition is necessary. This approach would emphasize the linking of program management decisions to current and future unknown situations within the stipulated parameters of cost, schedule and functionality thus giving the managers a set of *choices* or *options*.

These choices are called "Real Options" and originate from research done to price financial option contracts in the field of financial derivatives. Its underlying premise is based on the recognition that strategic flexibility in software engineering decisions can be valued as a portfolio of options or real assets, much akin to options on financial securities which have real economic value under uncertainty [9]. It centers on real (non-financial) assets and is valuable because they enable the option holder (software program manager) to take advantage of potential benefits while controlling risk.

It is assumed, that when employed within the context of a software-related capital investment effort, as a strategic decision-making framework a Real Options approach makes the case that strategic action identifies and creates valuable options which could be valued and exercised (if appropriate), starting the cycle of value creation and new options all over again [6]. This strategic decision-making process (Figure 2) is based on the general concepts of initiating the software acquisition process with a situation

assessments phase, generating the appropriate strategic actions and deriving the benefits or value created either explicitly or in the form of Real Options. While risks associated with large-scale software-related capital investments or acquisitions have been effectively managed through the application of stochastic frameworks, we believe that a framework based on Real Options theory would better balance customer requirements as dictated by operational needs within financial viability and schedule constraints through the identification, valuation and optimization of strategic decision pathways created in the form of Real Options.

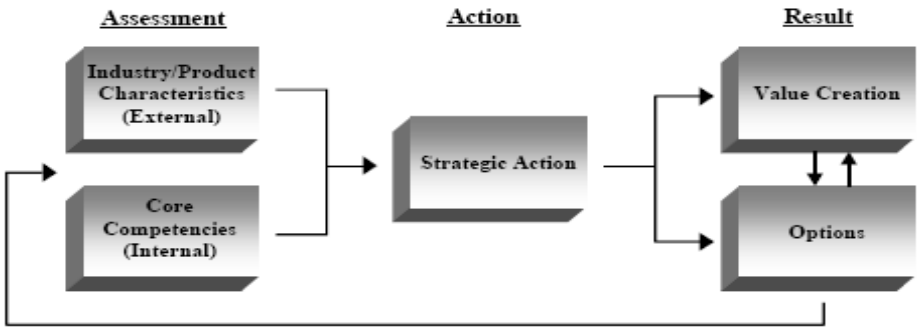


Figure 2. Strategy Formulation and Real Options (From: [6])

The Real Options approach calls for the existence or satisfaction of certain pre-conditions before it can be applied which we believe are directly correlated to the various activities associated with software related capital investments. These pre-conditions as outlined in [14] call for the following:

1. The existence of a basic financial model used to evaluate the costs and benefits of the underlying software asset (e.g. Net Present Value (NPV) as the Real Options approach builds on the existing tried-and-tested approaches of current financial modeling techniques.
2. The existence of uncertainties during the software-related capital investment decision-making process otherwise, the Real Options analysis becomes useless as everything is assumed to be certain and known.
3. The uncertainties surrounding the software-related capital investment decision-making process must introduce risks which directly impact the

decision-making process. Real Options could then be used to hedge the downside risk and take advantage of the upside uncertainties.

4. Management must have the flexibility or option to make mid course corrections when actively managing the project.
5. Management must be smart enough to execute the Real Options when it becomes optimal to do so.

This study seeks to investigate the application of the Real Options approach to both the technical and managerial risks associated with software-related capital investments from an integrated perspective with the goal being to develop a generic yet integrated Real Options-based decision-making framework that could be applied to address risks associated with DoD software acquisitions.

D. HYPOTHESIS

The traditional Real Options methodology, when enhanced and properly formulated around a proposed or existing software-investment, provides a framework for guiding software acquisition decision-making by highlighting the strategic importance of managerial flexibility in balancing the satisfaction of a customer's requirements within the realms of the associated cost and schedule constraints.

E. RESEARCH METHODOLOGY

In an assessment of previous work conducted, it was discovered that while the Real Options approach had been largely studied independently from both technical and managerial perspectives, insufficient work had been done from a full spectrum "point of view" particularly, in the software engineering domain, i.e. the application of Real Options to both technical and managerial risks simultaneously in the context of a capital intensive software investment effort. Consequently, the approach proposed in this research explicitly identifies both the technical and managerial challenges facing software acquisition efforts and the formulation of a framework to address both concerns simultaneously rather than independently. The underlying premise for employing an integrated approach towards managing both technical concerns and managerial concerns

was based on the belief that both concerns should not be treated independently due to high correlation between technical and managerial issues. The following shows the research method undertaken:

1. **Tactics for Producing the Proposed New Method:** This study, determines and establishes compliance with Real Options pre-conditions 1 through 3 and explicitly infers that management has both the flexibility to develop Real Options to make mid course corrections and the wisdom to execute the Real Options when it becomes optimal to do so during the course of a software acquisition effort, thereby meeting the requirements established in pre-condition 4 & 5.

   The approach involved the development of a systematic methodology of uncertainty identification and risk quantification using the volatility of the returns on the software investment as a proxy to measure the risk facing a given software-related capital investment effort.

   The crucial issue of measuring and estimating the risks posed by uncertainty (pre-condition 3) to the software acquisition is addressed by employing a volatility estimation/refinement technique, based on Dempster-Shafer Theory (DST) or (Evidence Theory) which relies on "beliefs" and "plausibility" probability assignments.

2. **Methods to Substantiate New Method:** To substantiate the results, a Real Options framework is developed explicitly showcasing the refinement of volatility estimation through the application of Dempster-Shafer Theory on Evidence. We attempted to validate the proposed approach using the software component (Future Combat Systems Network) of the troubled multi-billion dollar U.S. Army Future Combat Systems acquisition program as an example. Where feasible variables were derived and/or computed and in situations where data was unattainable, reasonable estimates along with the appropriate justification were utilized.

F. SIGNIFICANCE AND POTENTIAL IMPACT

Software is about change. "Design for change", "Invest with change in mind" are thus promoted as a value-maximizing strategy provided one could anticipate changes [8]. In most cases a software product is outdated even before it is delivered to the customer. More often than not, this is mostly due to changing customer requirements in response to changing business conditions. While arguments might be made by proponents and opponents that *Moore's Law*[2] holds or does not hold in this situation, the reality is change is inevitable in order to stay competitive in today's environment. Therefore software change or evolution is inevitable. However, the time and mode of change itself is what is uncertain as it might be too early to predict, hence the need for a flexible risk-driven approach towards software-related capital investments both before and during the investment process.

This phenomenon of continuous change has created the dilemma of huge cost and schedule overruns. In a yearly assessment and summary of major weapons acquisition programs published by the Government Accountability Office (GAO) it was reported that many, if not most, acquisition programs are experiencing cost overruns and schedule delays in their software development segments. For example in fiscal year 2006, the U.S. DoD spent as much as 30% ($12 billion) of its estimated budget of $40 billion for research, development, testing and evaluation on reworking software [7], a significant percentage when compared to the private sector software development. GAO also pointed out that, in the past five years, although "DoD has doubled its planned investments in new weapons systems from $700 billion to $1.4 trillion, this huge increase has not been accompanied by more stability, better outcomes or more buying power for the acquisition dollar" [7]. In an environment where program success has meant paying for massive cost-escalation, slipping plans years beyond their planned dates, making major cuts in the software functionality, the approach proposed in this study addresses these issues by taking a proactive approach to risk management by planning for an paying for risk up front. This is not to say that risk management strategies are not being adopted today, but

the failure of management to take a proactive approach towards risk management by employing what are believed to be "tactical" strategies, which unfortunately have not been proven to be successful

G. SUMMARY OF MAJOR CONTRIBUTIONS

This study contributes to the body of knowledge in the areas of software acquisition risk management by emphasizing the value of employing a *proactive* risk management approach. While there are several acquisition or decision-making frameworks currently utilized for decision-making in DoD software-related capital investments, at the time of this study we were not able to identify any situation where the Real Options approach had been explicitly applied to a ***single*** software acquisition program to guide the investment decision-making process; rather, we were able to identify instances where it had been proposed to employ Real Options within the context of IT portfolio management to manage a suite of IT investments.

Specifically this study emphasizes the value of employing an agile approach in software-related capital investments/acquisitions by exploiting and borrowing on the concepts of the value of flexibility as currently adopted in Agile Development approaches to software development and incorporating this flexibility at the software acquisition level. We also emphasize the value of information realized by exposing the unknowns (uncertainties) much earlier in the software acquisition process and taking advantage of the embedded flexibility of adapt to the uncertainties.

In this study we explicitly propose the use of Dempster-Shafer Theory on Evidence as a volatility estimation refinements mechanism, since volatility is a key input parameter needed for Real Options analysis. While volatility is just one of the parameters needed for Real Options analysis, it is the most difficult of all the parameters to estimate. We attempt to overcome the complexity of volatility estimation by proposing the use of Dempster-Shafer Theory on Evidence, a technique first proposed for application in the domain of *sensor fusion.* It is a mathematical theory of evidence, based on *belief*

2 Moore's Law describes the driving force of technological and social change by positing that advances in technology increases exponentially, by doubling approximately every two years.

functions and *plausible reasoning*, which is used to combine separate pieces of information (evidence) to calculate the probability of an event. We posit that it could be used to address both aleotoric and epistemic uncertainties inherent in software-related capital investments by "*fusing*" and reducing uncertainties to the maximum extent possible as they become revealed thereby facilitating a more accurate estimate of the risks propagated by uncertainty and allowing us to develop the appropriate option in response based on a more accurate volatility measure.

Lastly, this strategic program management approach provides a means of overcoming the limitations associated with the spiral development process currently utilized in the Evolutionary Acquisition (EA) approach adopted in the DoD 5000 series acquisition directives by providing the much needed upfront risk management planning at the strategic level to complement the risk management approaches employed in the spiral development process.

H. LIMITATIONS

The Real Options approach requires us to compute two different values with some degree of confidence. These values are the cost of the software acquisition effort and the probability that circumstances would develop such that we would like to exercise the option. However due to the lack of detailed data on the Future Combat Systems program at the level of granularity which we would have desired, we made several assumptions and provided justification as applicable. To determine the probability estimate we utilized modeling techniques along the lines of the three key assumptions made below and refined our estimate using Dempster-Shafer Theory on Evidence. This study is the first attempt at investigating the feasibility of utilizing Dempster-Shafer Theory on Evidence within the context of a software-related capital investment for volatility estimation, and while we were able to demonstrate its use, we did make some assumptions in the computation of our belief functions due to the lack of detailed data at our desired level of granularity in the estimates provided by the Cost Analysis Improvement Group and the Institute for Defense Analysis.

While we successfully applied the Dempster-Shafer Theory within the constraints of our assumption in our study, it is not without its limitations and critics. Critics of the Dempster-Shafer Theory notably Judea Pearl, have argued that

> it is misleading to interpret belief functions as representing either "probabilities of an event," or "the confidence one has in the probabilities assigned to various outcomes," or "degrees of belief (or confidence, or trust) in a proposition," or "degree of ignorance in a situation" [23].

To overcome this criticism we represent belief functions as probabilities that a given proposition is *provable* from a set of other propositions [23].

I. ASSUMPTIONS

As mentioned in the limitations section of this study, detailed data was not available at the desired level of granularity. While we were able to obtain data for the overall Future Combat Systems program, which included combined costs of both hardware and software components, we were not able to isolate software related costs, i.e. costs of the Future Combat systems Network (FCS-N) (software component of the FCS program), from the overall cost data. Consequently, for the purposes of establishing a financial model of the Future Combat Systems program, we made three key assumptions.

1. For cost purposes, it was assumed that the cost of the Future Combat Systems Network is equal to the costs of the overall Future Combat Systems program. What is accomplished by making this assumption is rather than picking an arbitrary value as the cost of the Future Combat Systems Network, we treat the overall Future Combat Systems program as consisting of software alone.
2. We assume the estimated future benefits (Asset Value) of the Future Combat Systems Network, while unknown, translated into a positive value under traditional Net Present Value techniques (i.e. benefits outweigh costs).

3. We assume the independent assessments provided by the Cost Analysis Improvement Group and the Institute of Defense Analysis includes belief assignments based on masses of evidence during the application of Dempster-Shafer Theory. That is, the executive or decision maker is provided not only with a raw set of the risk factors of requirements creep, integration risk, performance risk, but with additional measures: belief in the estimation of the risk factors and certainty of the estimation during the decision making process.

J. DISSERTATION ORGANIZATION

This dissertation is organized around seven chapters, containing results from our Risk Simulator provided by Real Options Valuation Inc., and a list of references and bibliography.

Chapter II discusses the relevant background material and highlight weaknesses and gaps in the current state of knowledge by discussing key areas of concern and limitations of current investment valuation models. We conclude this chapter by providing the reader with some insights into real options theory.

Chapter III presents detailed insights in the area of uncertainty, a key component necessitating the need for the real options approach proposed. Specifically we categorize the uncertainties facing program mangers into technical and managerial uncertainties and further expand on the details associated with each category of uncertainty.

Chapter IV addresses the issue of estimating the volatility of the returns of a software investment based on the prevalent risk factors introduced by uncertainty. We also depict how the proposed Dempster-Shafer Theory based on the Dempster's rule of combination is used to combine evidence from different sources.

Chapter V culminates in the formulation of the proposed Real Options framework that could be applied to software-related capital investments.

Chapter VI validates the proposed framework by exploring the conceptual application of our proposed framework to the U.S. Army's Future Combat System Network (software component) acquisition program.

Chapter VII presents a summary of our findings, contributions and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ASSESSMENT OF PREVIOUS WORK

> The proper measurement of the success of software investments requires more than compliance with models of technical perfection. Technical excellence of programming code is highly desirable but clearly insufficient. Nowadays the most profitable and popular software is notorious for its bugs and glitches. Economic utility and independent measures of customer satisfaction must be the ultimate arbiters of all judgment about the utility of the software.
>
> (Strassmann, 1997 p.147)

A. SOFTWARE INVESTMENTS BACKGROUND

The software investment process is a complex process plagued by a myriad of activities which could be broadly categorized into technical and managerial issues, both of which are inextricably linked. Understanding the risks associated with software investment activities is therefore an important consideration for a project manager and must be captured in an investment strategy designed to accomplish the goals of the investment effort and ultimately executed in an acquisition plan. Therefore given its complexity, a software project manager must be well equipped and postured to deal with a large synergy involving players ranging from the project manager him/her self to the end users [11]. This is accomplished through the development and use of a systematic decision-making approach that would bridge the gaps between technical and managerial activities in the form of a framework which would enhance decision-making associated with choosing the appropriate courses of action in a complex, uncertain, or conflict-ridden situation.

Technical activities may range from requirements specification to implementation while managerial activities may range from planning to risk management with the early and unambiguous definition of software requirements being key to the investment process. Regardless of the activity, both of these categories of activities presents

challenges in the form of uncertainties necessitating the need for a methodology to identify uncertainty, estimate and ultimately manage risk to further guide the investment process.

From a technical perspective, the software architecture is viewed as the earliest design artifact, which realizes the requirements of the software system [24] with the main focus of the software architecture being to provide a "high level" abstract design plan with just enough detail to manage complexity, to provide the overarching framework and guidance for detailed design. Thus software architecture can be viewed as:

> the manifestation of the earliest design decisions, which comprise the architectural structure (i.e., components and interfaces), the architectural topology (i.e., the architectural style), the architectural infrastructure (e.g., the middleware), the relationship among them, and their relationship to the other software artifacts (e.g., low-level design, testing etc.) [24]

However, in virtually all software engineering activities, the software architecture is usually driven by customer requirements, with several iterations occurring between requirements and architectural design until a preliminary design is finalized, from which a baseline for detailed design can be established. Thus we can reasonable claim that customer requirements, as identified by an operational need reflect the earliest manifestation of the software investment effort which precedes the software architecture.

B. INVESTMENT STRATEGIES

Traditionally speaking there are two main approaches which guide a software investment strategy; custom development of the software and acquisitions based on the use of COTS products, otherwise known as the "build vs. buy" phenomenon. Each of these present their pros and cons and need to be carefully evaluated before choosing one approach over the other. This is highlighted in Figure 3 below in which differences are depicted between both approaches in the form of "time to realization of benefits" [20]. However it must be noted that the study conducted in [20] lacked a clear delineation between software and the associated infrastructure (hardware etc) within the investment framework. Specifically the research in [20] highlights that in the COTS acquisition approach, the organization has the option of spending an amount of money (K) to acquire

the IT asset. At any point of time (t) during an interval T, K is known with certainty; however, future changes in K are uncertain. After the asset is acquired, the organization starts receiving a set of cash flows (C) representing the differential benefits derived from acquiring the IT asset.

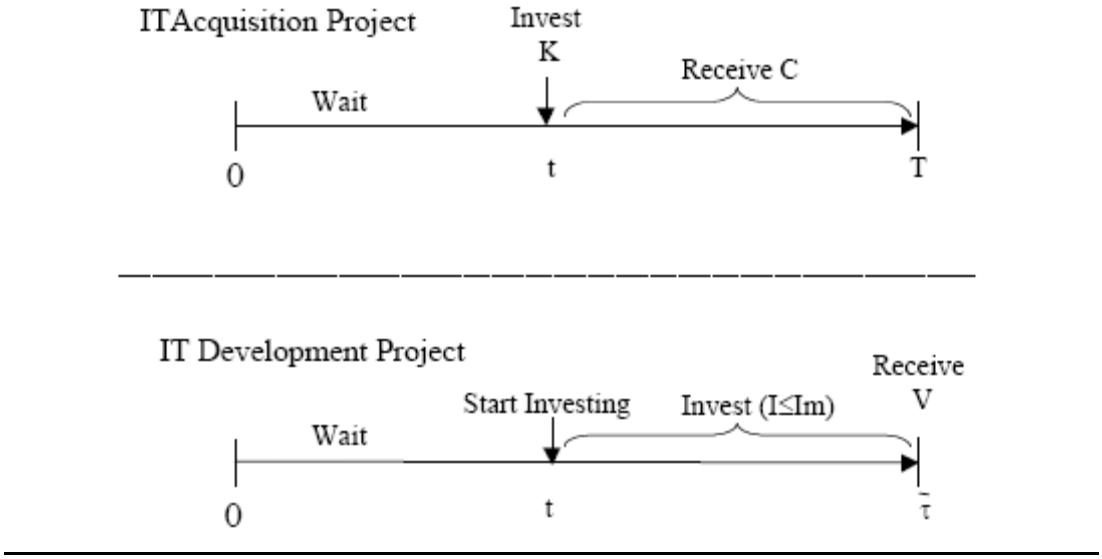


Figure 3. Two Types of IT Investment Projects (From: [20]).

Given that both the cost and the benefits are uncertain, it might be better to wait before making the investment. Furthermore, if the cost of a particular IT asset decays over time, there is an additional incentive for waiting before acquiring the asset. However, benefits also decrease with time because waiting will reduce the length of period in which the organization will be able to receive the benefits associated with the investment. Therefore, both elements have to be taken into consideration for making an optimal decision [20].

In the *development* approach, the asset is not acquired instantaneously; rather, it is the result of a development project having an uncertain duration (τ) in which the firm keeps investing at a rate that is less than or equal to a maximum investment rate (Im). Only until the project is completed and the remaining cost (K) is zero, the firm receives the underlying asset (V) [20].

While the software investment approach is somewhat analogous to the IT investment approach depicted above, this paradigm however, is not completely representative of all software engineering investment efforts, as software development usually takes a considerable amount of time, and the benefits might not be obtained until the project is completely finished. It is however possible to start enjoying benefits of the software investment, albeit partially when modern software processes such as incremental development which support custom software development or COTS development within an evolutionary context are utilized.

C. WEAKNESSES AND GAPS IN STATE OF THE KNOWLEDGE

In general one of the key challenges facing software and its acquisition are requirements issues. This dilemma could include scenarios of inadequate or insufficient thought or effort going into the upfront investment decision to outright ignoring the future evolution of the software investment effort under consideration. The theoretical foundations of strategic planning offer a focus on "long-term" thinking, and when we think "long-term" in software engineering, evolution readily comes to mind. An evolutionary frame of mind should therefore form the underlying premise under which all software-related capital investment are made with emphasis being on the keyword "capital", where future operational requirements would serve as the driver of evolution.

While options theory has been largely studied and applied to decision-making in the area of Information Technology (IT) portfolio management and to the various software artifacts, e.g. software architecture [24], little work if any has been done to study its application to all the decision-making activities (both technical and managerial) within a single framework to manage a single software investment as opposed to a portfolio of software investments. A holistic approach towards managing technical and the associated managerial issues is therefore needed to keep up with the advances in the technical aspects of software engineering and to also better manage risk. This dilemma is highlighted in several reports published by the U.S. Government Accounting Office

(GAO), and in a white paper on effective technology investment strategies [13], the author calls for the need for a more rigorous scientific process for developing a company's technology strategy and technology investment, in lieu of the trial-and-error process.

Thus given these gaps in the current state of the knowledge, software executives and program managers need better decision making tools and methodologies to guide their software investment decision-making activities. To accomplish this, the DoD proposed the Evolutionary Approach to software acquisition which we discuss below along with the technical and managerial perspectives.

1. Evolutionary Perspective

In May 2003, the U.S. Department of Defense (DoD) promulgated revised 5000 series acquisition directives and instructions that mandated an evolutionary approach known as Evolutionary Acquisitions (EA) (Figure 4) which relied on the spiral development process.

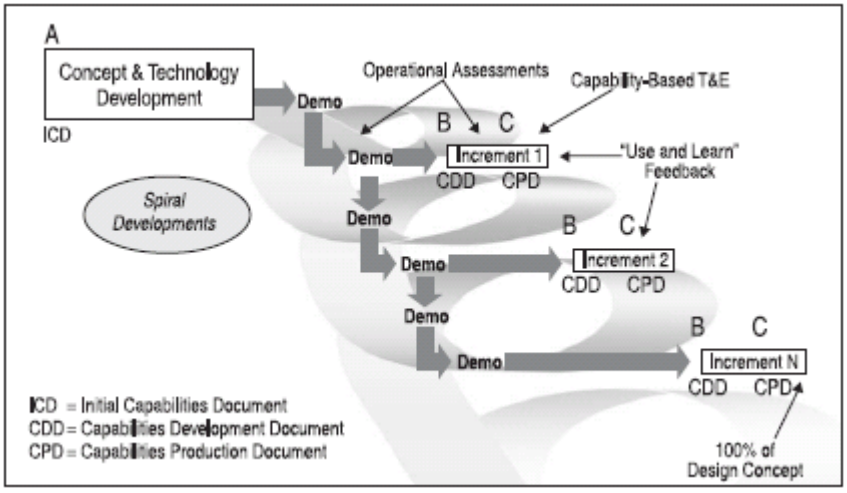


Figure 4. Evolutionary Acquisition and Spiral Development (From: [12]).

The strategy employed within the EA construct was to deliver capabilities in incremental fashion recognizing the up-front need for future capability improvements closely integrate managerial and technical decision-making. However, the overarching

limitation of this approach could be narrowed down to the chosen development process - the spiral development process. The spiral development process assumes the end-state requirements are known at the inception of the development process [12]. In a study conducted by RAND, it was discovered that EA programs required considerable additional up-front management planning and engineering workload and the budget sources to support them [12]. The spiral development process is a risk-driven development approach which consists of four main phases namely: determining objectives/alternatives, risk analysis, development and planning. The phases are iteratively followed one after the other building progressively on the first iteration until a complete software product is built. Of the four phases, the risk analysis phase is the most important because the project's success is highly dependent on the ability to identify and resolve risk. Risks are continuously discovered and high-priority risks drive the development process. However, besides the obvious limitation of the spiral approach being a somewhat costly approach, we believe that risk management should be a factor that is addressed much earlier in the software engineering process – at the acquisition level, during the investment decision making activities prior to the commitment to acquire and or develop a software system. The appropriate risk mitigation/reduction strategies should be employed much earlier in the software investment/acquisition process.

Furthermore the RAND study [12] also determined that EA programs using the spiral development process needed to focus on capability mission objectives rather than traditional technical requirements and in the analysis of five case studies all of the case studies pointed to the conclusion that the capabilities and requirements definition and management processes continued to remain as major challenges in all EA programs.

2. Technical Perspective

Relative to other engineering fields, software engineering is still very much in its infancy. Computer science, management science and some other fields have greatly influenced the theoretical foundations for software engineering, however the standardization and the lack of a clean theory to make software design decisions

continues to compound the problem of software investments. While the guiding principles of software design which center on abstraction, information hiding, localization, modularity, uniformity, conformability, completeness serve to ensure that a useful software product is developed, by establishing a foundation for which the overall goals of correctness, flexibility, robustness, efficiency and reusability in software design are achieved, the challenge remains, one of correlation of these idiosyncratic principles from the perspective of options theory [25]. The work of Kevin Sullivan [25], also proposes an options-based interpretation in software design activities by connecting options theory to software design in two steps: 1) viewing software design as a value creating opportunity to make irreversible capital investments in software assets of uncertain value with the architecture, documents, program generators, and information hiding interfaces being examples of assets and 2) rationalizing decisions about whether and when to make such investments by appealing to options theory and viewing investment opportunities as call options [25]. The uncertainty in the value of investing in a software asset makes this problem a prime candidate for the application of a decision making theory such as options theory to help guide the overall investment process and consequently software program managers are faced with the problem of overcoming some rigid dictums in software engineering since there are no reliable proxies to guide optimal investment strategies.

Thus software artifacts are and should be treated as software assets due to the particular value they add to the overall investment process. Assets can be software components, objects, software requirement analysis and design models, domain architecture, database schema, code documentation, manuals, standards, test scenarios, and plans and more often than not were either created by prefabrication i.e. the assets were developed with posteriori reuse thought process (designed for reuse). An effective investment strategy should not only address the possible uncertain future evolution of the software but also factor in an effective reuse strategy that could possible support the future evolution of the software. This of course introduces the problem of architectural stability and the ability of the architecture to accommodate the changes, a problem which

Bahsoon [24] attempted to address in his study through the proposal of an "Arch Options" model.

Evolution of software systems and the associated ability of the architecture to support evolution was the key theme of [24] in which a "Arch Options" model was proposed to address architectural stability in the face of changing requirements in an evolutionary context. The "Arch Option" approach provided guidelines on eliciting the likely changes in requirements and relating architectural decisions to value.

The "Arch Option" framework also accounted for the economic ramifications of the change on the structural (e.g., maintainability) and behavioral (e.g., throughput) qualities of an architecture and on relevant business goals (e.g., new market products). In [24], Bahsoon acknowledges that the valuation of the architectural potential to the change is a multi-perspective problem and attempts to tackle the problem by proposing a valuation "point of view" framework for quantifying the options from different perspectives. However their work stops short of identifying ways to manage the valuation under this framework, such as identifying the dimensions, which are critical for understanding architectural stability, prioritizing and weighting the valuation of these dimensions, managing conflicts, and reconciling the options results. This is necessary to provide a sound comprehensive valuation, which takes into account the various valuation points of views. Consequently our proposed approach would embrace a "holistic" approach towards uncertainty identification that would be utilized much earlier on in the software investment process to manage and reduce uncertainties and we then employ Dempster-Shafer Theory of Evidence based on subjective probabilities as a mechanism to further refine our risk estimates associated with the uncertainties. (See Chapter 4 for a detail discussion of the Dempster-Shafer Theory of Evidence.)

3. Managerial Perspective

Economic realities demand that organizations leverage their existing and future software investments to create a competitive advantage, be it in an operational capacity or support capacity to the war-fighter. More often than not, this involves the delicate art of balancing operational needs which might be compounded with uncertainty with costs and

schedule limitations. To accomplish this feat, decision making tools are needed and are either created on an ad-hoc basis or well established decision-making aids utilized. We examine two of the more popular techniques below

a. Decision Tree Analysis

Decision tree's are popular decision support tool used as a visual and analytical means for calculating conditional probabilities to help identify the strategy most likely to produce the optimal solution goal.

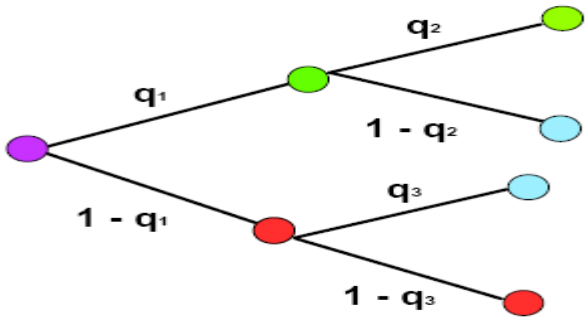


Figure 5. Sample Decision Tree (From: [19]).

It is a chronological representation of the decision process [16] and utilizes a network of nodes. It provides a prescriptive approach to decision-making by allowing the decision maker to select exactly one alternative from a set of possible decision alternatives in situations of uncertainty regarding the future. Nodes indicate decision points, chance events, or branch terminals with the root node representing the first set of decision alternatives. q represents the relative outcome probability, or uncertainty, associated with each chance event.

b. Utility Theory

Utility theory is an attempt to infer subjective value, or utility, from choices and it can be used in both decision making under risk (where the probabilities are explicitly given) and in decision making under uncertainty (where the probabilities are not explicitly given) [22]. With its philosophy based on the doctrine of utilitianism, its underlying assumption is that the decision maker always chooses the alternative for

which the expected value of the utility (EXPECTED utility) is maximum [18]. In other words, expected utility could more precisely be called "probability-weighted utility theory". It involves the construction of a utility function based on assignment rules in which a utility is assigned to each of the possible (and mutually exclusive) consequences of every alternative, depending on the individual preferences of the decision maker. While there are several methods for constructing utility functions, the best-known method is based on indifference judgments of the decision maker about specially constructed alternatives. Objective utility is the dominating approach in risk analysis, and the common way to measure risk is to multiply "the probability of a risk with its severity, to call that the expectation value, and to use this expectation value to compare risks" [21].

It is essential to observe that Decision-tree analysis (DTA) moves the traditional Net Present Value analysis which we would be discussing in the next section, one step forward by allowing the possibility of alternative states of nature. Furthermore, expected utility maximization is only meaningful in comparisons between options in one and the same decision, with some of the clearest violations of this basic requirement found in risk analysis where expected utility calculations are often used for comparisons between risk factors that are not options in one and the same decision (decision theory).

D. INVESTMENT VALUATION METHODS

Estimating the value of a software investment effort is a particularly challenging task because there are many factors that affect the payoffs and costs of the investment effort. While there are several techniques for valuation, valuation approaches could be categorized into three mainstream approaches namely, the market approach, income approach and the cost approach [14]. The market approach looks at comparable assets in the market place and their corresponding prices and assumes that market forces will tend to move the market price to an equilibrium level, the income approach looks at future potential profit of the asset and attempts to quantify, forecast and discount these net free cash flows to a present value, and the cost approach which examines the costs that would be incurred if the asset under consideration were to replaced or created from scratch [14].

In the field of finance, valuation is the process of estimating the market value of a

financial asset or liability with the Capital Asset Pricing Model (CAPM) being the most popular technique.

From the perspective of a software investment, valuation can be defined as the process of estimating the present and future market value of software investment (asset). However all these valuation approaches share one thing in common; they rely on the Discounted Cash Flow (DCF) technique which continues to be the most popular investment valuation techniques. We further expand on this technique in the next section.

1. Discounted Cash Flow (DCF) Model

The DCF method is an approach used to valuation, whereby projected future cash flows are "discounted" at an interest rate or rate of return that reflects the perceived riskiness of the cash flows. The DCF model has two complementary measures; Internal Rate of Return (IRR) and Net Present Value (NPV), with the Net Present Value being the single most widely tool used for large investments made by corporations.

Net Present Value (NPV) measures the excess or shortfall of cash flows, in present value (PV) terms, basically a cost benefit analysis methodology and is simply computed by subtracting costs from benefits, where benefits equal the sum of the present value of future cash flows after taxes, discounted at some market risk-adjusted costs of capital and costs equal the present value of investment costs discounted at the risk free rate or reinvestment rate [14]. It is an indicator of the value an investment adds to a firm. Decisions making under this concept are based on the sign of the cash flows. In financial theory, if there is a choice between two mutually exclusive alternatives, the one yielding the higher NPV should be selected [17]. The general formula for computing NPV is given as follows:

$$\text{NPV} = \sum_{t=1}^{t} \frac{C_t}{(1+r)^t} - C_0$$

where

t - the time of the cash flow

N - the total time of the project

r - the discount rate (the rate of return that could be earned on an investment in the financial markets)

C_t - the net cash flow (the amount of cash) at time t (for educational purposes, C_0 is commonly placed to role as the initial investment).

Table 2 below showcases the decision making criterion associated with NPV.

If...	It means...	Then...
NPV > 0	The investment would add value to the firm	The project may be accepted
NPV < 0	The investment would subtract value from the firm	The project should be rejected
NPV = 0	The investment would neither gain nor lose value for the firm and does not mean that the investment is expected to break even.	We should be indifferent in the decision whether to accept or reject the project. This project adds no monetary value. Decision should be based on other criteria, e.g. strategic positioning or other factors not explicitly included in the calculation.

Table 2. **Net Present Value Decision Making Criteria (From: [17])**

Despite the wide use of the NPV technique, it has one flaw – its inability to explicitly account for managerial flexibility. Thus we explore the RO approach, which takes into consideration a key concept: present day management flexibility to make strategic decisions that have a long-term strategic impact, a concept that is noticeably absent in other valuation methods such as Discounted Cash Flow (DCF) and Net Present Value (NPV) [26].

E. REAL OPTIONS METHODOLOGY

The Real Options approach offers a means of capturing the flexibility of management to address uncertainties as they are revealed. With its history and theory embedded in financial theories, the key valuation concept of (financial) options theory is that an option can be priced based on the construction of a portfolio of a specific number of shares of an underlying asset, and that one can borrow against the shares at a riskless rate to replicate the return of the option in a risk-neutral world [19]. An option gives its

holder the right, *without the obligatio*n, to acquire or dispose of a risky asset at a set *strike price* within a specified time period [27]. If the market conditions are favorable before the option expires, the holder exercises this right, thus making a profit. Otherwise, the holder lets the option expire. This asymmetric nature of options gives them real economic value [27].

When extended to real assets, a Real Option could be defined as a systematic approach and integrated solution using financial theory, economic analysis, management science, decision sciences, and econometric modeling to valuing real physical assets, as opposed to financial assets, in a dynamic and uncertain business environment where business decisions are flexible in the context of strategic capital investment decision-making, valuing investment opportunities and project capital expenditures [26]. In essence a real option is a flexible arrangement that acknowledges the ability of management to make decisions to counter against some unforeseen situations, in this case being uncertainties in software engineering. However creating the options, acquiring the options and managing the options over time and to realize the full potential value are some of the challenges that need to be addressed.

The Real Options approach is able to overcome the limitations of traditional valuation techniques by utilizing the best features of traditional approaches and extending their capabilities under the auspices of managerial flexibility. In a real options view, uncertainty is the randomness of outcomes from a software investment decision. Real options are implicit or explicit capabilities created for real assets [28] that provide the software manager with time-deferred and flexible choices (options) regarding future risks or changes of the software and could explicitly address the issue of software investment choices for future capabilities. It assumes that managers develop a level of foresight sufficient to invest in resources/techniques and processes with 'options' characteristics that provide implicit or explicit claims on future opportunities and generate flexibilities for future investments or changes [28]. Through these capabilities, the software manager may choose to adjust, reduce, increase, or abandon the investment in the future, thereby stabilizing returns from these assets. In other words, it analyzes how software managers can lay claim to future rent-generating capabilities through investment in options. The

real options approach helps to structure the project as a sequence of managerial decisions over time, clarifies the role of uncertainty in project evaluation and allows us to apply models that have been developed for valuing stock options to project investments [20] (Bodie and Merton 1999), and our study seeks to develop a RO framework to explicitly estimate risk using Dempster-Shafer Theory of Evidence and Dempster's rule of combination to guide the decision making process. Using the options logic necessarily entails a rigorous analysis of the software investment, their uncertainties, and costs of creating the options, all of which contribute towards a greater understanding of the strategic role for the software. We shall begin the formulation of our framework by visiting and addressing the issue of uncertainty in the next chapter.

III. ADDRESSING UNCERTAINTY

> As we know, there are known known's; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns—the ones we don't know we don't know.
>
> (Donald H. Rumsfield, Secretary of Defense, 2002)

A. INTRODUCTION

Uncertainties permeate virtually every phase of the software investment process from procurement decision-making, requirements specification, software development and implementation, to the eventual evolution of the software. This sentiment is best echoed by Ziv's uncertainty principle of software engineering in which he posits that uncertainty is inherent and inevitable in software development processes and products. These uncertainties usually present themselves in various forms ranging from changing/incomplete requirements, insufficient knowledge of the problem domain to decisions related to the future growth or evolution of the software. As identified earlier on in Chapter I, the presence of uncertainty is one of the pre-conditions necessary for the application of Real Options (RO). In this chapter, we attempt to capture the uncertainties that appear in the software investment process and how they impact decisions associated with software investments.

First, it is important to understand what the term "uncertainty" means. The word 'uncertainty' as derived from Webster's dictionary defines uncertainty as a situation where:

> The current knowledge available may range from falling short of certainty to an almost complete lack of conviction or knowledge especially about the outcome or result.

When translated literally, the key phrase "may range from falling short of certainty to an almost…" hints at some element of probability and insinuates the presence of some degree of risk, which is either introduced or mitigated within the context of

uncertainty. Software engineering, regardless of process, programming language, or domain, involves significant decision making on the part of the software manager due to the myriad of activities involved in the software engineering process and the uncertainties that surround these activities. The driving force of the uncertainty of future software evolution to meet either future organizational needs or operational challenges also increases the degree of uncertainty associated with the software investment decision making process. As a result, uncertainty introduces and drives risk. We must however emphasize that uncertainty should not be confused with risk as there is an important distinction between the two. Risk is something one bears and is the outcome of uncertainty, as uncertainty is either resolved through the passage of action or left unattended due to inaction [14]. The risks associated with the acquisition of the software need to be identified and analyzed very early on in the decision-making process, and an approach to mitigate the high-priority risks must be incorporated into a software acquisition plan. Therefore, the responsibility lies with the software manager to identify, manage and eliminate the sources of uncertainty by developing a risk management plan so as to identify risks at the earliest possible time, adjust the acquisition strategy to manage high-priority risks, and implement a risk management process to manage risks throughout the acquisition life cycle [40].

B. CATEGORIZATION OF UNCERTAINTIES

In addressing the issue of uncertainty, we focus on uncertainty from three points of view: the uncertainties associated with the software investment decision-making process, uncertainties with the software engineering process and uncertainties associated with the software product itself.

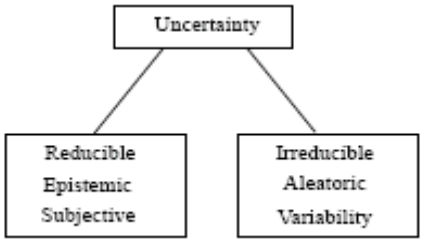


Figure 6. Taxonomy of Uncertainty (From: [29]). Classification of uncertainty into epistemic (reducible) and aleatoric (irreducible).

According to the scientific body of knowledge on uncertainty in aeronautics and astronautics, uncertainties exhibit themselves in either of two forms as depicted in Figure 6 on the previous page. Epistemic uncertainties are considered to be reducible uncertainties while aleatoric uncertainties are considered to be irreducible uncertainties. While epistemic uncertainty deals with our lack of knowledge, lack of information and our own and others' subjectivity concerning an issue, aleatoric uncertainties, on the other hand, deals with the randomness (or predictability) of an event due to variability of input or model parameters when the characterization of the variability is available [29]. In other words, an aleatoric uncertainty is an inherent variation associated with the physical system or the environment. Both epistemic and aleotoric uncertainties are interwoven and form the general framework which uncertainties fall into and also form the framework from which we would be addressing uncertainty in this study.

C. INVESTMENT DECISION MAKING UNCERTAINTIES

Several factors drive uncertainties in the software investment decision making process. To determine these uncertainties, we first identify the pertinent activities leading to and associated with a software investment effort and establish an overarching framework. The three major activities as identified in Figure 7 below are the software

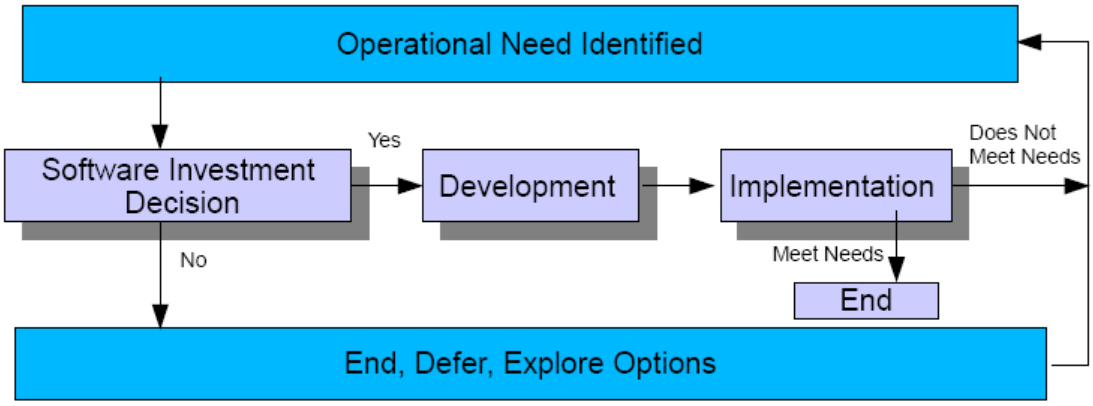


Figure 7. Software Investment Process.

investment decision-making activity during which a decision is made to procure a software system, the associated development activities of the software and implementation related activities.

The overwhelming uncertainty surrounding the software investment decision-making activity is determining what the scope of the acquisition effort would be, the costs and determining the most appropriate acquisition strategy. Possible acquisition strategies range from purchasing the desired capability outright from commercial vendors and then customizing it to suit the customer's needs, building a custom solution from scratch, or employing a "hybrid" approach that includes a combination of both custom development and purchasing of the components from commercial vendors. These are examples of Real Options strategic paths, that is, options to undertake different courses of action (analysis of alternatives). Under normal conditions, the investment decision-making activity phase is initiated upon the conclusion of an "operational needs" assessment to justify the needs for investing in the capability by the using organization (Army, Navy, Air Force, and so forth).

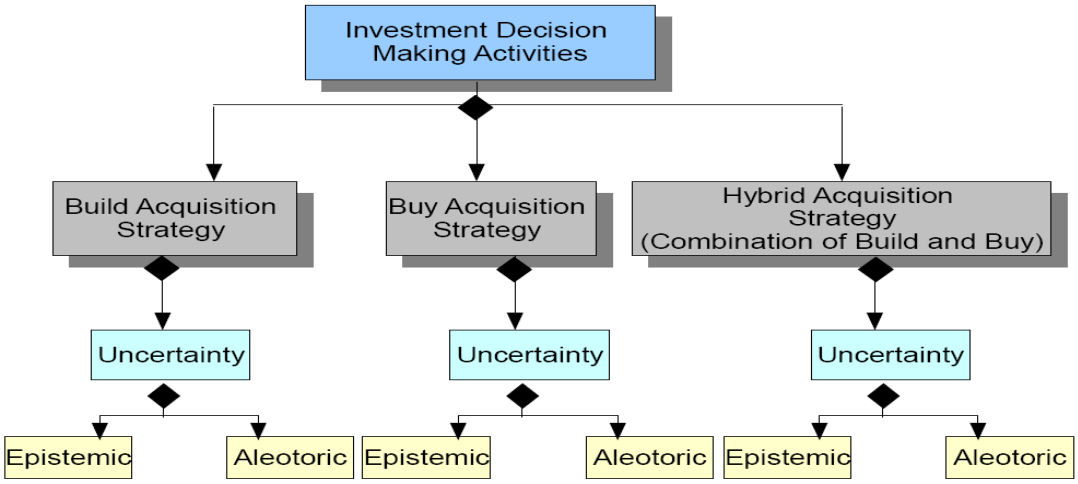


Figure 8. Key Software Investment Decision Making Activities. Investment decision making activities include choosing between a "Build", "Buy" or "Hybrid" acquisition strategy each of which has uncertainties associated with them which could either be epistemic or aleotoric in nature.

In the case that the operational need is conceptual in nature, trade studies should be conducted to either prove or demonstrate the concept before an investment decision is made and before funds are committed to the effort. The typical "high level decisions" associated with the investment decision making activity are depicted in Figure 8 above.

The investment decision making activity drives the development phase, which centers on the technical activities associated with developing the solution to meet the operational need once an investment decision is made and an acquisition strategy selected. This leads to the implementation activities phase whose end result centers on the delivery and fielding of the software as well as supporting the maintenance and possible evolution of the software. In this study, we limit our focus to studying the uncertainties associated only with the investment decision making process due to our belief that uncertainties associated with the investment decision-making activity drives the overall success of the acquisition program, hence the need to address the uncertainties early on in the investment process. We now proceed with developing an in-depth methodology for identifying and addressing uncertainty early on in the software investment process through an uncertainty elicitation phase.

D. UNCERTAINTY ELICITATION

Just as a formal requirements elicitation phase exists in traditional software engineering processes, we propose the introduction of a formal and distinct "uncertainty elicitation" phase as part of the software investment decision-making process. This phase would not include members of a typical requirements team, but would work in tandem with them, to play the "devils advocate" by identifying and documenting uncertainties as they are revealed in the forces driving the need for the software, the acquisition strategy and its ultimate implementation. Implementing an uncertainty elicitation phase would facilitate the identification of uncertainties very early in the acquisition process, and steps could be taken to either refine the requirements to address the uncertainties or strategic options identified to mitigate the risks posed by the uncertainties. In Figure 9, we expand on the "Software Investment Decision" component of Figure 7 and propose a

methodology that could be used to capture and address uncertainties at the inception of a software-related capital investment through the introduction of an explicit step to allow for uncertainty elicitation.

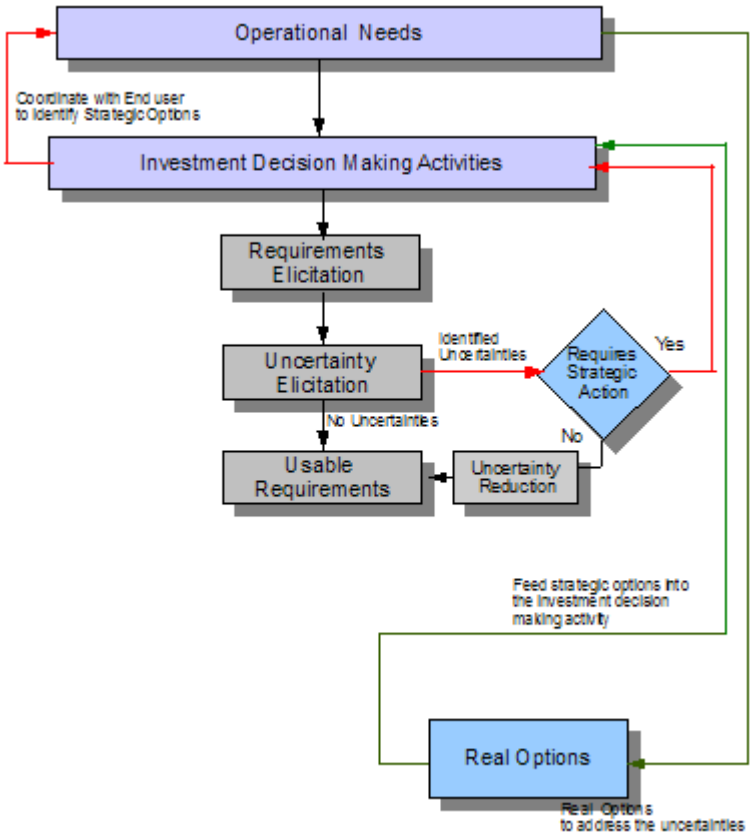


Figure 9. Uncertainty Elicitation. The "usable requirements" are elicited at a very high level and could help guide the investment decision making or selection of an appropriate acquisition strategy.

Once all these activities have been accomplished and a decision is made to proceed with the software investment effort, we now proceed to the development activities phase and attempt to develop an in-depth methodology for uncertainty identification in the remainder of this chapter. Development activities center on all the tasks and activities that are associated with developing the software product. Uncertainty appears in various phases during the development process and we attempt to identify and address them.

E. SOFTWARE ENGINEERING UNCERTAINTIES

As software engineering uncertainties are present at all stages of the software life cycle, these uncertainties which are treated implicitly should be handled explicitly by developing a proactive strategy to either mitigate or hedge against the uncertainty. The challenge remains on how to identify and capture uncertainties explicitly, reason in the face of uncertainties (analysis, trade-offs) and ultimately manage the uncertainties. Accordingly, Lehman's proposal of the basic uncertainty principle of computer application [41], which stipulates that the outcome in the real world of software system operation is inherently uncertain with the precise area of uncertainty not knowable, clearly highlights the inherent difficulties of addressing uncertainty. The extent to which a degree of satisfaction is reached that the operational system meets its intended requirements is ultimately determined by the software professionals' judgment, action, and inaction. Thus, neither developers nor users can fully know system properties, therefore solving these problems in the face of these uncertainties to produce solutions that satisfy, in spite of not knowing what it is that they satisfy, is the heart and essence of software engineering [41].

These uncertainties have given rise to risk and decision analysis methodologies and paradigms, such as modeling and simulation to model the unknown and delaying certain decisions for as long as possible, that are targeted at reducing epistemic uncertainties by increasing one's knowledge of the problem at hand. Since the epistemic uncertainties facilitate the introduction of aleatoric uncertainties in the end product, an approach is needed to vastly reduce epistemic uncertainties in order to reduce the corresponding aleatoric uncertainties which are propagated in the software engineering effort.

In an attempt to provide clarity, we revisit the definition of software engineering, which we loosely define as the "the set of activities leading to the development of a software product." We categorize these activities into two interwoven but distinct phases to further highlight and delineate the uncertainties which arise in each of the two phases: The first phase is the "program management" phase and the second phase is the

"technical" phase. The activities in the first phase are heavily slanted toward managerial activities which involve significant decision making on the part of the software program manager regarding the technical activities surrounding the software development effort. We highlight the development activities in Figure 10.

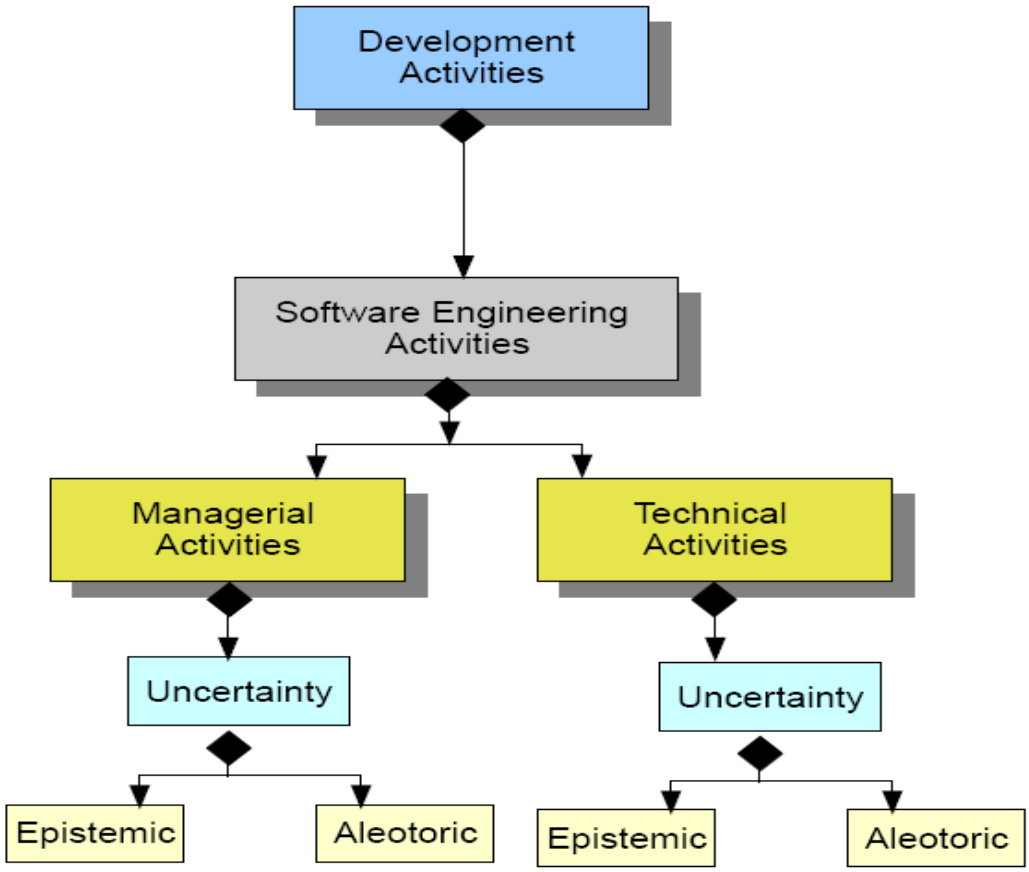


Figure 10. Development Activities & Software Engineering Activities.

F. MANAGERIAL PERSPECTIVE

The managerial phase serves to provide program management guidance to ensure the software product is developed within cost and scheduled constraints. From a managerial perspective, the five major constraints of people, time, functionality, budget, and resources (excluding people) [39] form the basis of the uncertainties which plague the software program manager. These five constraints could be summed up into two major uncertainties, cost estimation and scheduling uncertainties (Figure 11).

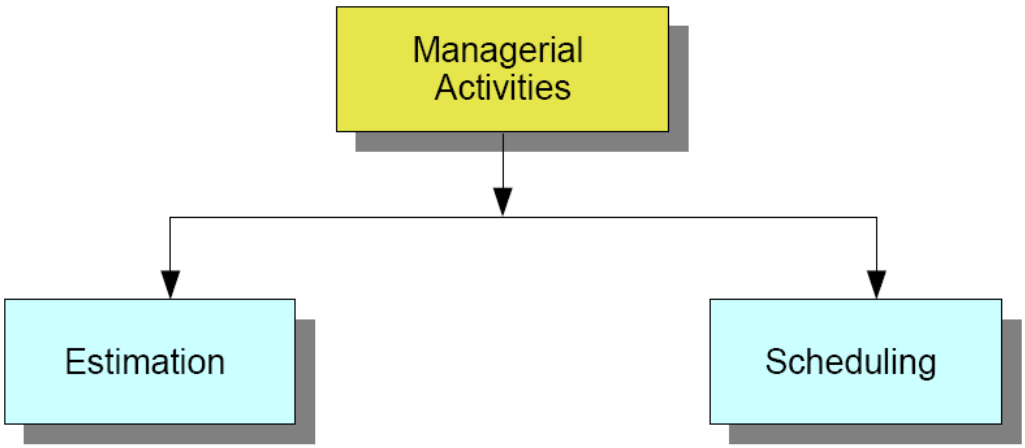


Figure 11. Managerial Activities.

The variables of cost, time, and functionality (scope) are intrinsically linked in the managerial activities centered around cost and schedule estimation. When time and cost are the essence of a project, then functionality becomes defined and is no longer a variable while on the other hand if functionality is defined, then cost and time are defined. Therefore if any one of these points becomes fixed, the other two points become controlled [42]. Maintaining a perfect balance between these three variables would lead to more accurate estimate and reduce the likelihood of cost overruns.

1. Cost Estimation

The accurate prediction of software development costs is a critical issue during the acquisition decision-making process. Cost estimation uncertainties arise due to the intricacies involved in determining the complexity, amount of work (size) and associated costs. Statistically speaking, escalations in the dominant cost of 'labor' have led to spiraling project costs as project scope and schedule escalates resulting in outright cancellations and budget overruns. Productivity of the team can also be difficult to predict. For example, in previous research, the case of employee productivity has been likened to that of returns on financial stock because productivity may vary by orders of magnitude, a situation which is analogous to the theory of financial instruments, where

empirical evidence suggests that the root cause of uncertainty is variance in the return rates, not in the base stock values [34].

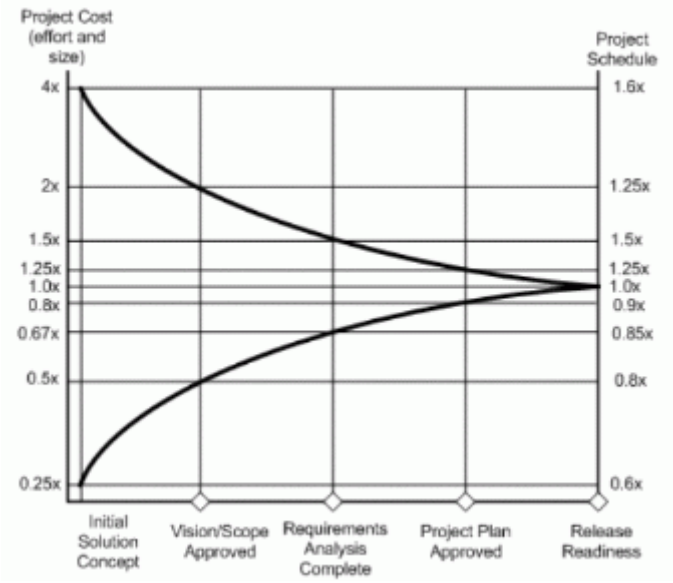


Figure 12. Cone of Uncertainty (From: [37]). The horizontal axis contains common project milestones such as Initial Concept, Approved Product Definition, Requirements Complete while the vertical axis contains the degree of error that has been found in estimates created by skilled estimators at various points in the project.

While there are several software cost estimation methods available such as algorithmic methods, top-down method, and bottom-up method, to name a few, no one method is necessarily better or worse than the other and more often than not the strengths and weaknesses of the estimation methodologies are often complimentary to each other.

Therefore to mitigate these uncertainties, an accurate estimation of a variable phenomenon must include the variability in the phenomenon itself. In other words, costs should be determined for uncertainties within an estimate, hence our proposed Real Options approach. The variability in these factors contributes to the variability of project estimates and as these sources of variability are further investigated and pinned down, the variability in the project diminishes, thereby diminishing the variability in the project estimates [37].

This phenomenon otherwise known as the "Cone of Uncertainty" and is best illustrated as depicted in the Figure 12 above. Since uncertainty-reducing decisions are usually not yet applied at the beginning of a project, the cone is always wider at the beginning of a project. The cone begins to narrow as uncertainty-reducing decisions are implemented. The optimal desired scenario for the project manager is to make the cone as narrow as quickly by applying uncertainty-reducing strategies which produced the desired outcome. This would require a disciplined and comprehensive approach of assessing and the cost estimation uncertainties and determining the project risk factors. In the case that certain costs are unknown, a justifiable cost should be included in the form of a contingency amount.

2. Scheduling

The scheduling activity is inextricably tied to the project's technical baseline and is essential to developing a cost estimate for the overall effort. It usually starts with the development of a work breakdown structure (WBS) which represents a hierarchical set of independent tasks with the precedence relationship that exists among tasks defined. Input from the cost estimation phase is usually required in the WBS and the final product is presented in the form of precedence networks and Gantt charts in which the critical path is identified.

Techniques such as the PERT model (Program Evaluation and Review Technique) were developed to address uncertainty in the estimation of project parameters. However, the main problem with PERT is that it gives accurate results only if there is a single dominant path through a precedence network [38]. In order to deal with an uncertain environment it is necessary to consider setting and employing a certainty threshold before a project is started, and ensure that it is monitored throughout the execution lifetime of the software project and rescheduling is performed if the certainty threshold is breached [35]. In situations where a single path is not dominant, PERT usually provides overly optimistic results hence the need for a Monte Carlo risk simulation approach, which would repeatedly sets values for each random variable by sampling from each variable's statistical distribution [38]. The variables can be task

duration, cost, start and finish time which are used to compute the quantities such as the critical path, slack time etc. However the Monte Carlo simulations for software development also does not provide accurate estimates of project parameters (duration, finish time, cost, etc.) due to the greater uncertainties related to requirements, tools, resources, budget, etc. compared to many other industries [38]. This has led to the proposal of different approaches to project scheduling with uncertainties such as the application of the theory of constraints (TOC) [38] to project management which employs the concept of project buffers to which we frame as "Options" to accurately support project scheduling activities.

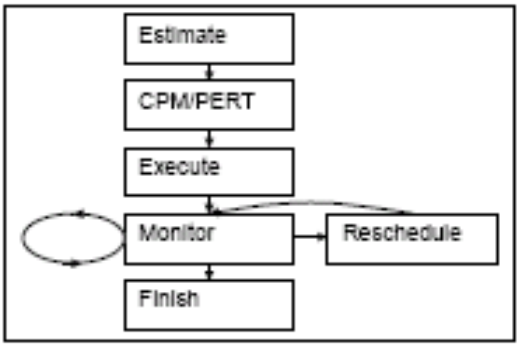


Figure 13. Scheduling Process. (From: [35])

The project buffers or "Options" would be based on the identification, analysis and quantification of the risks posed by the uncertainties and in the event that sufficient information is not available, a "worst-case" analysis may be utilized. The amount of allowance provided by the buffer would be based on assessing the degree risk posed by the uncertainty associated with all remaining project activities.

G. TECHNICAL PERSPECTIVE

The second phase in the software capital investment process is heavily slanted towards "technical" activities and is concerned with all the technical activities needed to develop the software product itself. The uncertainties which are typical of this phase include uncertainties associated with requirements analysis, transition from system

requirements to design and code, uncertainties in software re-engineering and software reuse, and uncertainties in operating the software product itself. For example, taking the case of software quality attributes such as performance and maintainability, which cannot be directly measured until after the fact (after the software has been fielded), the activities associated with enhancing these quality attributes cannot be estimated or measured during the development process. Such concerns therefore consequently introduce aleatoric uncertainties in the software product. Technical uncertainties can be categorized into the four major activities of specification, design/implementation, validation, and evolution activities as identified in [1], all of which are carried out regardless of the acquisition strategy selected, i.e., build vs. buy, as even commercial off-the-shelf (COTS) products need to be tailored and customized using these activities to meet the customers needs.

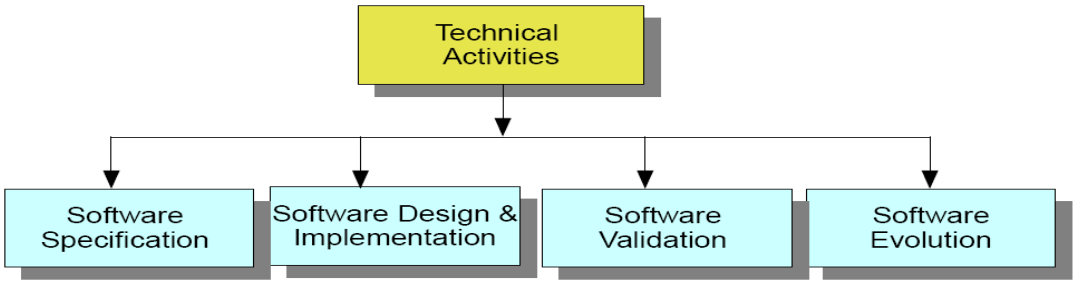


Figure 14. Technical Activities.

1. Software Specification

The Software Specification phase is typically the first phase in any acquisition process once a software need has been identified. Requirements are elicited through various mechanisms and a software requirements specification is generated. The key underlying uncertainty facing the software specification activity is the probability of being faced with incomplete, unknown or changing requirements. More often than not, these changes are proposed once the software is already in operational use, as users begin to see better ways to perform existing functionality or because there is a mismatch in performance expectations. This scenario highlights the realities of the challenges facing software specification and exemplifies "Humphrey's Requirements Uncertainty

Principle" which posits that for any new software system, the requirements will not be completely known until after the users have used it. The prudent manager would ensure that to the extent possible, the requirements developed in this phase are testable amongst other things.

2. Software Design and Implementation

The software design and implementation phase usually follows the software specification phase. The tasks which are inherent to this activity are architectural design, detailed design and program design as depicted in the Figure 15. This phase involves the initial development of high-level software architecture which is continuously modified until the requirements are completely refined. A detailed design is generated from the high level architecture and the activities of coding, integration, and testing are carried out during this phase. Uncertainties in this phase are usually a propagation of the uncertainties in the software specification phase and manifest themselves in the software design. While recent empirical studies of software development under realistic conditions has lead to the introduction of several agile development process such as eXtreme Programming (XP) and SCRUM which promote development iterations, open collaboration with the customers, and adaptability throughout the life-cycle of the project in response to both Humphrey's and Ziv's uncertainty principles, to help address software engineering uncertainties, these approaches in our opinion do not adequately address the challenges posed in this phase as we believe agile approaches by virtue of its definition is somewhat more of a "reactive" approach to software development than proactive approach because of the elimination of what is perceived to be wasteful, yet necessary activities associated with traditional software development processes. Also while beneficial we are of the opinion that caution must be exercised when employing agile methods, especially in organizations that do not embrace certain standards of discipline that are associated with an engineering discipline (i.e. documentation, configuration management etc.) in order to preserve the integrity of the relatively young profession of software engineering as a traditional engineering discipline. Furthermore while modern concepts such as posteriori reuse (design for planned reuse) are

increasingly becoming widely advocated standards in which software artifacts are created with reuse in mind with the goal of improving productivity, quality, time-to-market and long-term cost, this concept is still evolving in the software engineering community.

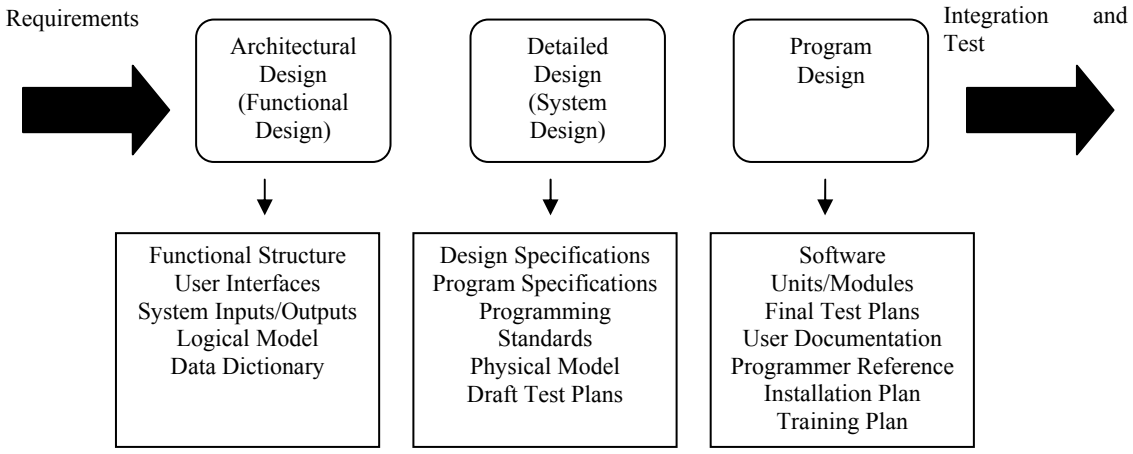


Figure 15. Software Design and Implementation Process (From: [36]).

Uncertainties in the software design process can be presented along the dimensions of two major points of view; the developer's point of view and the customer's point of view. From a developer's point of view, design decisions should be evaluated as early as possible in the software development life cycle, and key requirements, both functional and non-functional requirements, must be fully addressed during the architectural design phase. It is not acceptable to proceed through detailed design and implementation phases only to discover that the decisions made early in the process, at the architectural level, did not turn out to promote the desired functionality to be achieved [30].

From a customer's point of view, there are two options towards achieving their solution; building a component or buying the needed component. The developers' concerns identified above would apply in both scenarios (build vs. buy) even in the case that a solution needs to be bought, because the customer is faced with the challenge of procuring a software component that it is capable of meeting its requirements straight out

of the box (which is rare) or with some customization, particularly with respect to their long-term needs and the system's response to change.

While meeting current requirements is definitely a focus of both points of view, future evolution serves as the driving mechanism on which decision would be based, hence the need for a decision-making framework to mitigate theses uncertainties. However, before the Real Options approach can be employed, the factors which would guide the decision-making process or evaluation process need to be assessed. These factors are realized by identifying and quantifying the uncertainties that appear in software design.

Software design situations are characterized by complexity and uncertainty with complexity representing the amount of relevant information that is available in a given situation and uncertainty representing the availability and reliability of the information that is relevant in a given situation [43]. Consequently, effective approaches to software design would require a closer look at complexity and uncertainty as important characteristics of a situation and embrace a set of simple concepts for relating situational characteristics to different approaches to reduce both complexity and uncertainties in software design. While techniques such as abstraction and decomposition are commonly employed to deal with the complexity associated with software design in specifications, approached based on prototyping have emerged as means to cope with software design uncertainty [44].

Previous research into software design approaches led to the proposition of the Principle of Limited Reduction [44] which concluded that software engineers should rely on mixed approaches to software design in order to effectively cope with both complexity and uncertainty because it is almost impossible to hope to reduce complexity without adversely increasing uncertainty and vice-versa.

3. Software Validation

The software validation activity serves to increase the degree of confidence in a software product by determining if the software product which is being delivered is the right software. It is normally preceded by verification activities aimed at ensuring the

software product is designed to deliver all functionality to the customer and typically involves reviews and meetings to evaluate documents, plans, code, requirements and specifications.

As is the case with the software design and implementation phase, uncertainties which surround this activity are due to the ripple effect associated with software specifications and the inability to guarantee a defect free product during the testing process. While verification serves to verify the correctness of the software product, validation ensures that functionality, as defined in the requirements, is the intended behavior of the product and typically involves actual testing of the software product and takes place after verifications are completed. The validation phase serves as a quality assurance procedure in which processes are designed to ensure that software product meets its intended use and the validation activities are usually undertaken both during, as well as at the end of the software development life cycle to ensure that all requirements have been fulfilled. However due to the potential ripple effect associated with software specification uncertainties, this activity has the potential of being plagued by the "garbage-in, garbage out" syndrome.

4. Software Evolution

In general, the software evolution activity addresses the change associated not only with the software systems, but also with the case tools and the development processes used to support software engineering activities in order for the software system to remain relevant in terms of its use. In Lehman's view, software evolution is "intrinsic" and not primarily due to shortsightedness of developers or users, but rather is due to the fact that software it is a man-made product which continuously evolves and the rate of evolution of a used software system will far exceed the rate of evolution of a biological or physical system, because the fundamental laws of biological and physical systems do not change, while the laws of software, being entirely man-made and malleable, can be and are changed at a moment's notice [41].

In our view, software requirements are the driving mechanism through which uncertainty is introduced into the technical phase of software capital investments, with

uncertainties in current requirements and future requirements playing a key role. In Lehman's work on software evolution, he classifies software systems into three major categories; S, P and E systems based on system correctness levels which are correlated to the three major sources of uncertainty which arise in the software engineering process: Gödel-like, Heisenberg-type and Pragmatic uncertainties. These three types of uncertainties could be interpreted as uncertainties in the problem domain, uncertainties in the solutions domain and uncertainties in human participation respectively. We expand on these uncertainties in the next section.

H. TYPES OF SOFTWARE ENGINEERING UNCERTAINTIES

Gödel-like uncertainties occur when the properties of a program cannot be known from the representation, because the software systems and their specifications are abstract models of the real world. Heisenberg-type uncertainties occur as the system is being developed and grows during use and exhibit themselves in the form of changing requirements either due to unsatisfactory behavior post implementation or because of the emergence of new requirements while pragmatic uncertainties are problems in actually performing the development activities.

In the case of Gödel-like uncertainties, we believe that since a model is only an abstract representation of reality and as such in most cases does not offer the level of detail that is desired, modeling and simulations techniques when properly employed, serves as a cost saving mechanism through which various events/scenarios could be modeled in a controlled environment before actual implementation. Furthermore this could serve as a "proof of concept" in the form of a prototype, although even though it might help reduce some uncertainties, from a cost perspective, the uncertainty of system properties might still persist.

Changing requirements are a normal occurrence in software development efforts. Therefore the challenge facing the software manager trying to deal effectively with Heisenberg-type uncertainties is two fold: how to accurately manage the changing requirements and how to avoid sunk costs as a result of an un-useable product due to incorrect or new requirements.

Uncertainties	Sources of Uncertainty			
	Software Development Process	Employee (Software Engineer)	Firm	Market Conditions
Gödel-like uncertainties Uncertainties resulting from ambiguous system properties	Inadequate modeling techniques and tools. Availability of required technology.	Lack of skilled engineers in the domain to help in eliciting system properties	Firm/Skill profile mismatch with customer requirements	Demand for new skills Uncertain supply of new skills
Heisenberg-type uncertainties Uncertainties resulting from changing requirements	Lack of clear requirements and inability to capture the changing requirements.	Ripple effects caused by changes to a requirements	Lack of understanding of domain	Variations in demand for supply of services
Pragmatic uncertainties Uncertainties resulting from performing development activities, e.g. cost	Incorrect estimation techniques	Erosion of existing skills Inability to learn new skills Employee dissatisfaction, lack of commitment	High financial leverage Variations in profitability	Business Cycles Competitive pressure for cost reduction

Table 3. **Summary of Software Engineering Uncertainties.**

This also introduces the challenge on how to accurately reflect or account for this uncertainty in the cost and schedule thereby contributing or introducing Pragmatic uncertainties ranging from funding problems, market conditions to employee related issues which could all impact the software development effort.

In Table 3, we have attempted to highlight the three categories of the uncertainty and identify sources of uncertainty associated with each category.

We use a hypothetical example to further illustrate these uncertainties in the scenario of developing a software application for the space shuttle. If the space shuttle is designed to travel to the moon and back, what are the uncertainties facing this effort and what kind of options would the software manager want to consider when designing this software?

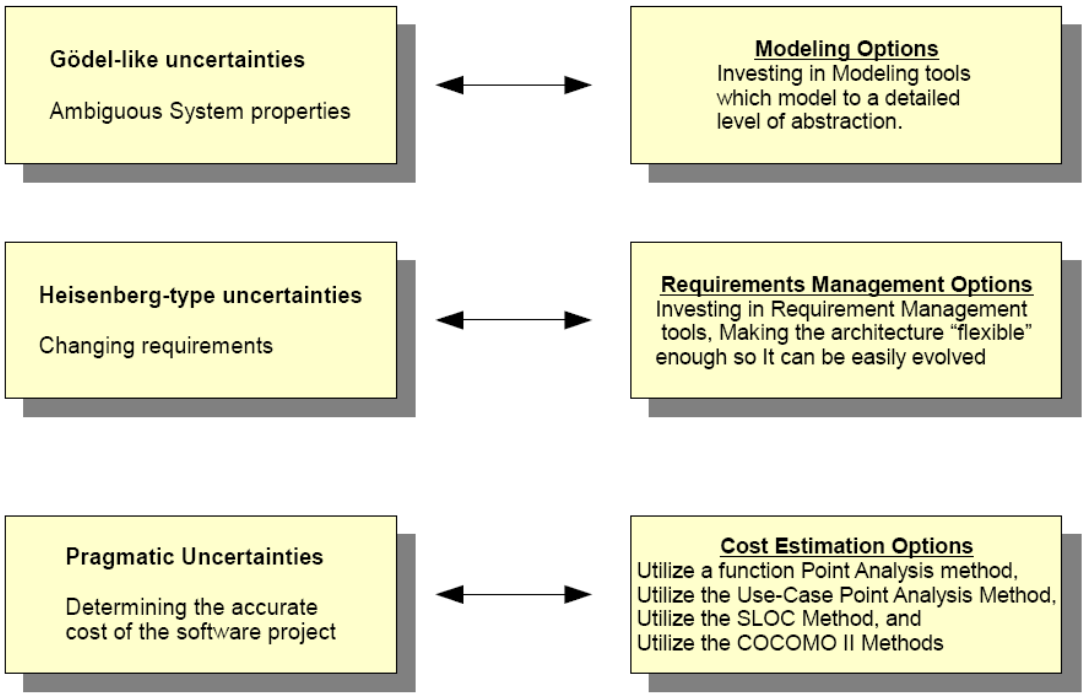


Figure 16. Sample Space Shuttle "Real Options".

A forward-thinking manager would identify future use of the space shuttle software as an uncertainty by thinking evolution, evolution, evolution! In this scenario, s/he would design the space shuttle software with evolution in mind and would attempt to determine the extent of evolution depending on the scenarios that he/she could come up with. For example, the software manager might want to make an investment in a robust architecture that could support a plug and play navigational system in case NASA decides to use the space shuttle to visit Pluto in the future so that only one component is changed (navigational software module). This forward thinking definitely requires some creativity and intuition on the part of the project manager. Figure 16 showcases these uncertainties and sample Real Options that can be used to manage them.

The most likely scenarios would then be prioritized and categorized into uncertainties and options developed and analyzed appropriately to counter the uncertainties. Thus, the essence of employing a Real Options approach based to the

potential evolution of the software requires the enhancement of traditional requirements engineering processes to allow for the creative forecasting of future requirements and development of the appropriate options to allow for future evolution even if their might be technology maturation concerns.

I. MANAGING SOFTWARE ENGINEERING UNCERTAINTIES

In an attempt to manage software engineering uncertainties, we capture the uncertainties from both a managerial and technical perspective in our Uncertainty Elicitation Model and visually depict the contributing factors in Figures 17 and 18 in what we call the "2 **T**'s" of software engineering uncertainty. (Given that our depiction resembles the capital letter "**T**".)

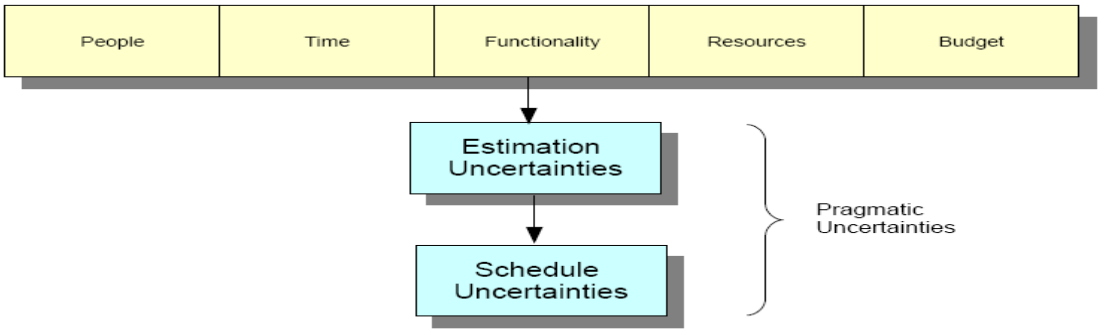


Figure 17. Software Engineering Management Uncertainties. Managerial Uncertainties of people, time, functionality, budget, and resources contribute to both estimation and schedule uncertainties which are considered to be pragmatic uncertainties.

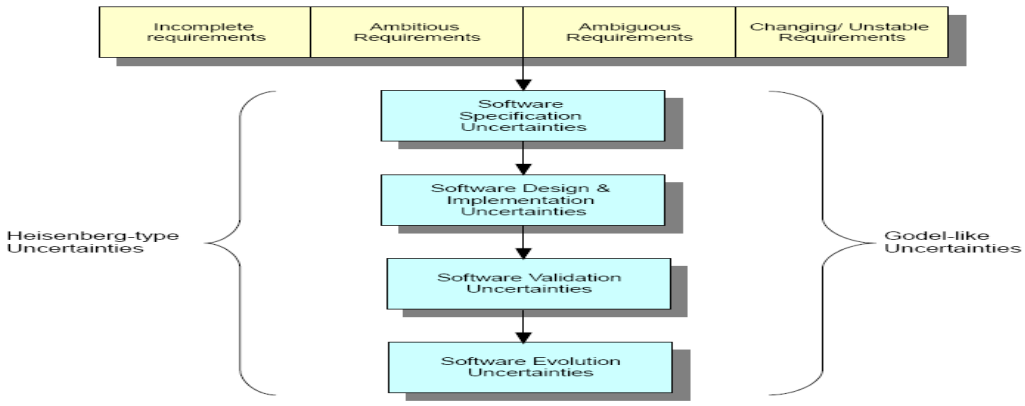


Figure 18. Software Engineering Technical Uncertainties. Technical uncertainties of incomplete requirements, ambitious, ambiguous, changing/unstable requirements contribute to software specification uncertainties, which leads to software design and implementation, software validation and software evolution uncertainties all of which can be categorized as exhibiting both Heisenberg- type and Gödel-like uncertainties.

The "2**T**" approach would serve as the basis on which uncertainties are elicited during the elicitation phase as proposed in our Uncertainty Elicitation Model. An expanded view of the model is depicted in Figure 19.

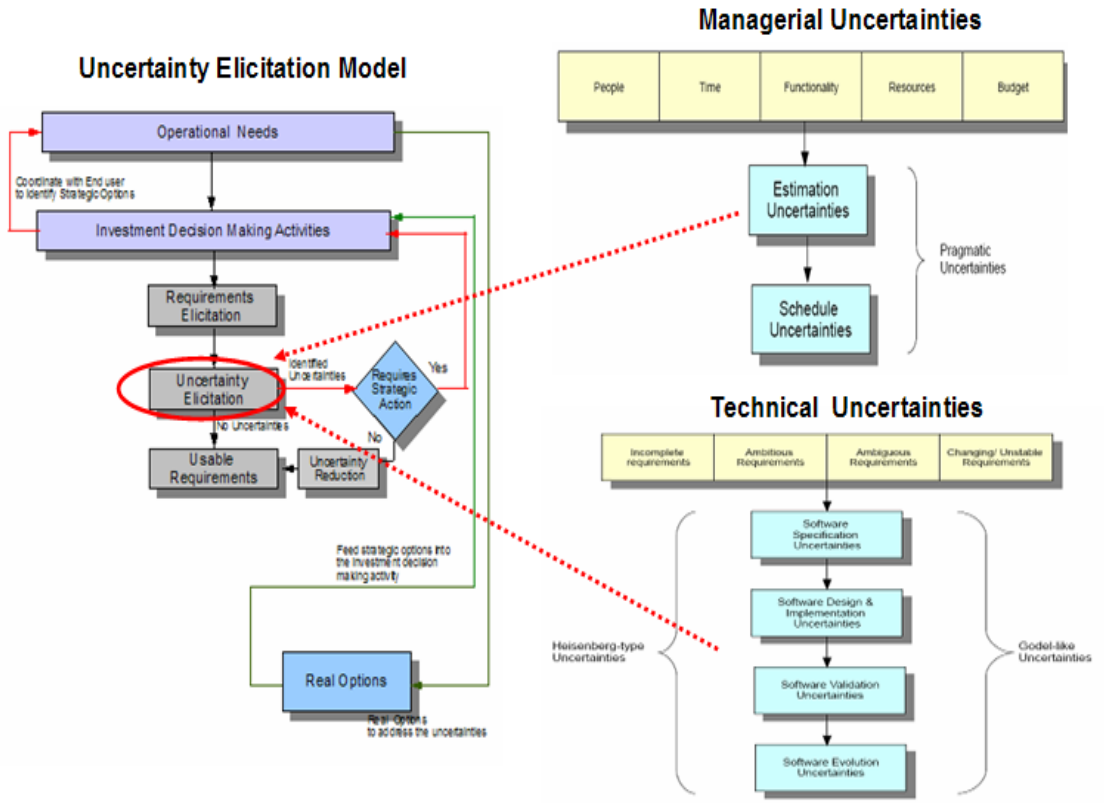


Figure 19. Expanded View of Uncertainty Elicitation Model.

Given all the uncertainties that appear in software engineering, one must try to reduce and/or eliminate the uncertainties to enable a more accurate end result in the software investment process. Hence, we need a quantification mechanism to quantify these uncertainties as risks with the goal being to assign an appropriate mathematical model to a real-world situation with respect to objective and subjective uncertainty [32]. While there are several mathematical theories that describe uncertainty and provide its measures such as probability, possibility and evidence theory, probability theory and possibility theory are best suited for describing aleatoric uncertainty and epistemic uncertainty respectively. Evidence theory, on the other hand, is well suited to handle both types of uncertainty because it does not require the separation of the two types of uncertainties due to its unique ability to represent the degree of belief (confidence) that may be attributed to a given proposition on the basis of given evidence, and its ability to combine evidence from different sources (Dempster's rule) [31]. Consequently, as the next step in our analysis we propose characterizing both the management and technical

uncertainties as depicted in Figure 20 by modeling the uncertainties within a single model developed based on the logic of evidence theory so as to further delineate between aleotoric and epistemic uncertainties, determine the uncertainties that could be reduced, understand the dependence between events and act accordingly by developing Real Options to mitigate the risks perpetrated by the uncertainties.

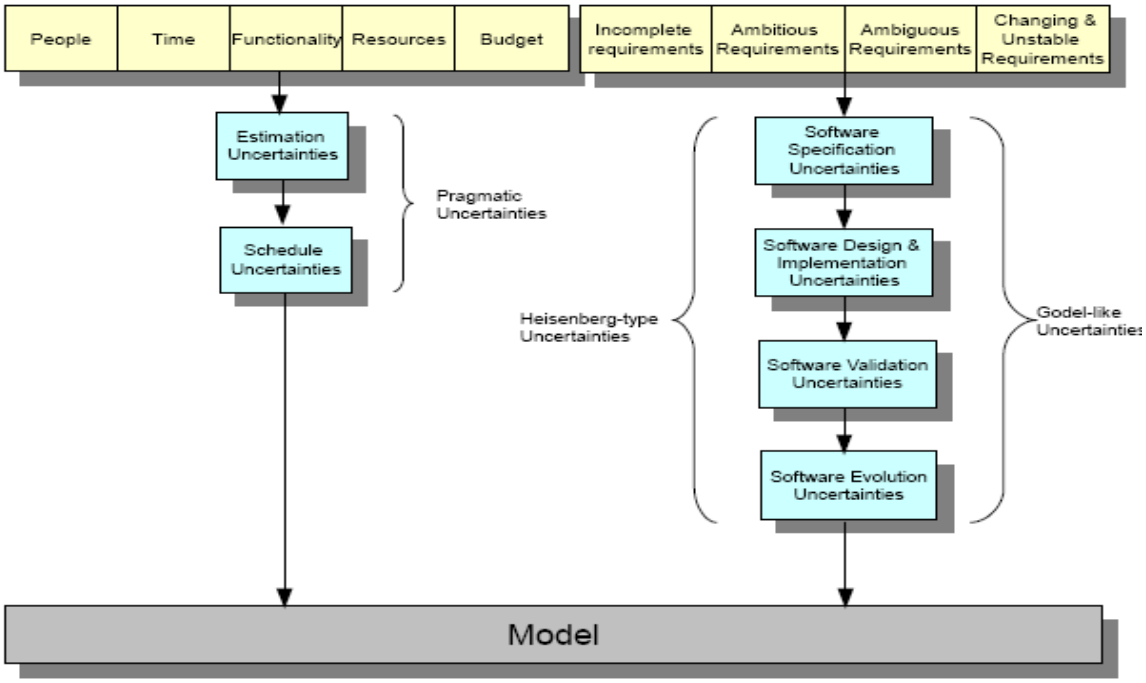


Figure 20. Modeling Software Engineering Uncertainties.
Both the Managerial and Technical uncertainties are fed into a risk model and epistemic and aleotoric uncertainties characterized from the inputs

Up to this point, we have succeeded in proposing a methodology for eliciting both the managerial and technical uncertainties surrounding a software-related capital investment. Once we have captured all the possible uncertainties, we determine the risk which these uncertainties present and characterize the risk as a measure of volatility on the software investment effort so that the appropriate options can be developed to hedge against the risk.

From a Real Options perspective, uncertainty is the randomness of outcomes from a software investment decision. Real Options are implicit or explicit capabilities created for "real assets" that provide the software manager with time-deferred and flexible

choices (options) regarding future changes of the software [28]. Real Options theory explicitly addresses the concerns of software investment choices for future capabilities. It assumes that managers develop a level of foresight sufficient to invest in resources/techniques and processes with 'options' characteristics that provide implicit or explicit claims on future opportunities and generate flexibilities for future investments or changes [28]. Through these capabilities, the software manager may choose to adjust, reduce, increase, or abandon the investment in the future, thereby stabilizing returns from these assets.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ESTIMATING VOLATILITY

A. INTRODUCTION

In the previous chapter we addressed the issue of uncertainty and follow up in this chapter by introducing a methodology for estimating volatility as we continue to develop our Real Options (RO) framework. As previously discussed in this study, uncertainty implies risk, and consequently, uncertainty must be duly quantified as a risk factor to gauge the magnitude of its impact on the underlying asset. The process of translating or equating software engineering uncertainties into a quantifiable property begins with the quantification of the identified uncertainties, computing the impact of uncertainties and ultimately developing a risk analysis framework in which the associated risks are identified, predicted and modeled using simulation and the results analyzed. This approach is representative of the current approach to risk management. However in our study, we seek to proceed beyond the conventional paradigm of risk analysis by explicitly utilizing Dempster-Shafer Theory of Evidence (DST) to reduce both aleotoric and epistemic uncertainties by using subjective probability estimates to refine our objective estimate thereby further acknowledging the flexibility of software manager to make decisions to incorporate Real Options to mitigate and hedge against risk.

We specifically choose to use the DST approach because it accounts for ignorance, a key strength and value added capability to our analysis since software engineering is unique in that there are several different software development approaches and processes without any standardization across the industry. Therefore it is our belief that DST when applied, could address the issue of ignorance or "unknown unknowns" in our probability estimates which otherwise reflect "known unknowns" such as in situations where it is not immediately clear from the historical data as to the type of software development approach that was utilized in sample historical projects (e.g., Rational Unified Process, Evolutionary Approach, Component Based Software

Engineering). Our subjective probability estimate based on DST would help establish a confidence interval on the key objective probability estimates to consequently reduce uncertainty.

B. RISK OVERVIEW

From a Real Options perspective, estimating volatility as a risk measure represents the most challenging task associated with the application of Real Options. Software risk can be broadly defined as a measure of the likelihood and loss of an unsatisfactory outcome affecting the software project, process and product. Risk can be captured in different dimensions and it could be estimated in terms of the volatility of the returns on the underlying asset or the degree of uncertainty about the future value of the opportunity. In general the risks associated with software-related capital investments can be analyzed in one of two possible ways: an objective approach, which is quantitative in nature in which risks are analyzed by calculating the probability of their occurrence and impact on the investment effort under consideration and a subjective approach which is qualitative in nature and is based on an experts opinion, intuition, assessment based on degrees of possibility about the risks rather than the reliance of historical experiences.

In general, there are nine major theories in decision making that guide risk management. These theories are Bayes theorem, Chaos theory, Creativity theory, Decision theory, Game theory, Portfolio theory, Probability theory, Uncertainty theory and Utility theory. Bayes theorem addresses the dynamic nature of risk by providing a method to alter judgment as events unfold, Chaos theory posits that chaos and uncertainties are market opportunities which must exploited, Creativity theory allows us to use our knowledge and imagination to develop ideas that are either original or novel, Decision theory uses probabilities to determine outcomes in complex problems, Game theory uses heuristics to determine which alternatives to explore in large search spaces, Portfolio theory utilizes the concept of diversification to reduce risk, Probability theory uses probability estimates to determine the degrees of certainty and forecast an outcome,

Uncertainty theory uses probability to model unknown, uncertain or subjective decision problems, and finally Utility theory, which models preferences towards risk by selecting the alternatives that maximizes the expected utility function [45].

Evidence theory on the other hand, otherwise known as Dempster-Shafer Theory of Evidence, which we utilize in our methodology, originated from Bayes Theorem of probability analysis. It has its strengths in its ability to represent and combine different types of evidence obtained from multiple sources. However, it requires the satisfaction of two constraints, the first being the independence of the sources of information and secondly the handling of conflicting evidence. We will expand on our methodology using DST later on in this chapter, but first we revisit the issues of risk estimation.

1. Software Investments Risks Estimation

Based on the uncertainties we identified using our "2**T**" approach proposed in our Uncertainty Elicitation Model (Figure 19) in the previous chapter, we now attempt to determine and quantify the risks posed by these uncertainties. We examine the risk of requirements changes (technical activity), cost and schedule overruns (managerial activity) based on the uncertainties identified as well as the impact these risks pose to the relative future value of the software investment effort. We use the term "relative future value" because the future value of government investments is not easily quantifiable. We further expand on this in the next chapter. These risks are then modeled using a Monte Carlo simulation to determine the volatility of the software investment effort.

2. Requirements Risk Estimation

Based on our research thus far we can safely assert that both Gödel-like uncertainties and Heisenberg-type uncertainties associated with software related capital investments are key drivers of pragmatic uncertainties. We put emphasis on the word "key" because there are other external contributing factors that also contribute to pragmatic uncertainties. However, Heisenberg-type uncertainties, specifically requirements issues serve as the key initiator of the uncertainties associated with software related capital investments and requirements volatility and inadequate requirements have

a significant impact on either the success or failure of the software investment effort. Typical changes might involve changes in functional, level of service (interoperability, performance), design constraints, quality attributes, and interface requirements amongst others. Furthermore, technological breakthroughs or unanticipated advancements in the software industry while development is underway or future evolution of the software product could also contribute to requirements changes.

Ever changing requirements continues to impact software investment efforts, and more often than not, it forces managers to choose between requirements, i.e., which requirements to accept and which requirements to reject with the full understanding that ignoring changes in requirements has the consequence of the delivered product failing to meet the customers needs while accepting changes in requirements has the potential of impacting costs and schedule. Furthermore, changes in requirements while a software investment effort is underway also poses the risk of introducing unwanted, unanticipated or unknown impact on existing requirements, not to mention the associated costs and scheduled delays depending on the phase of the investment or software development process experiencing significant requirements changes. While the standard practice has been to "freeze" requirements prior to the commencement of any development activities, more often than not, this does not work and is also not representational of the DoD doctrine to support the flexible development and rapid delivery of products to meet the war-fighters needs in an ever changing environment in response to operational needs.

Therefore in order to accurately estimate requirements volatility and its impact on the future value of a software-related capital investment, not only must the risk of requirements changes be quantified, it must also be specifically predicted and quantified based on the phase in the software development process in which the changes are more likely to occur. Hence the need of an approach that would explicitly acknowledge not only the probability of occurrence based on previous objective estimates, but also the possibility of occurrence based on subject expert opinions (Delphi Method) that acknowledges either the degree of belief or ignorance in the objective probability estimates which we attempt to address using the DST approach.

3. Schedule Risk Elicitation

In general software engineering scheduling risks could be addressed either from a development standpoint or delivery standpoint, although the development schedule ultimately drives the delivery schedule. While the decision variables which typical drive the scheduling activity could be either managerial or technical in nature, we posit that technical issues present the most risk because managerial risks could easily be controlled. Specifically the decision variable of requirements changes in the form of requirements increases or reductions has the unintended consequence of impacting the software delivery schedule by either pushing the delivery date out by months and in some case years past the anticipated delivery date or shortening the delivery schedule in response to either operational needs, cost constraints, or as a result of risk mitigation. This dilemma is further highlighted in Figure 21 below which showcases the difference between the anticipated delivery date of a product and the actual delivery date of a software product as the size of the software product increased due to requirements changes.

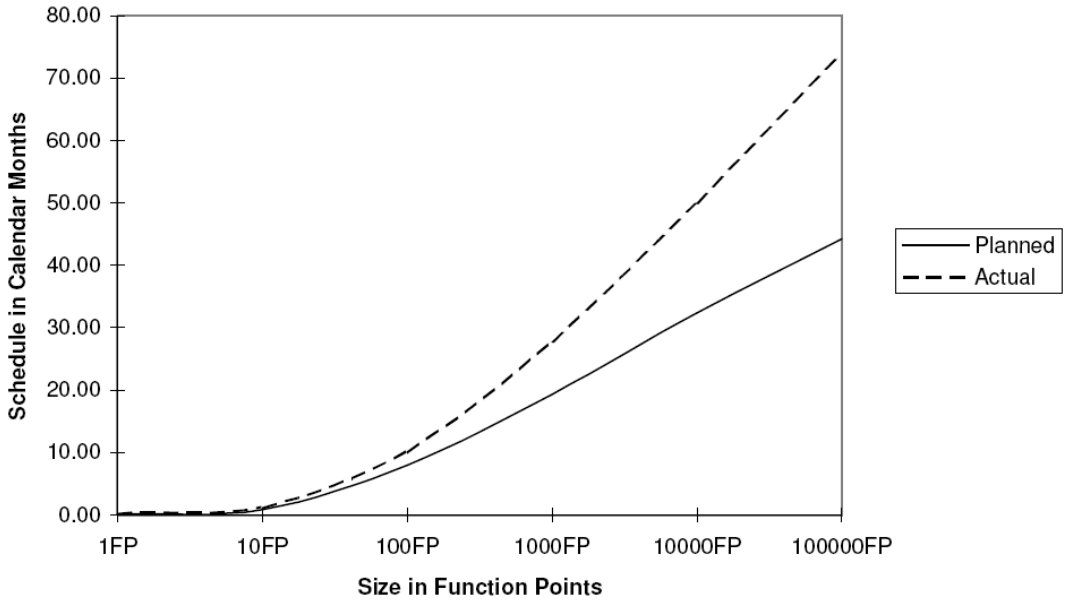


Figure 21. Differences Between Planned and Actual Software Schedules (From: [38]).

Therefore in order to hedge against the uncertainty of requirements changes, better and flexible estimation techniques which account for the changes in requirements

need to be developed, although schedule estimation is beyond the scope of our study. In our study, we assume the schedule estimation techniques are sufficient enough for us to provide initial schedule estimates which we can then refine, by addressing the uncertainty that was factored into the cost estimation using our DST methodology.

4. Cost Estimation Risk Elicitation

Initial estimation of the costs of the software investment effort is a critical component of the software investment process as it provides a basis for justifying the investment effort. This estimate must not only account for all the normal tasks and activities which needs to be accomplished during the whole investment effort, but must take into consideration other factors to include requirements volatility, time for resolving defects time associated with "spin up" during contract re-competes and possible litigation if several contractors are involved as is the case with several DoD acquisition programs to name a few. Therefore it is paramount that all the pertinent decision variables that impact the overall cost of the software investment effort be accurately captured. However, given the three sample decision variables of requirements, schedule and cost changes, the question now becomes how do we capture this information? For this we now resort to the next phase in our methodology which is the evidence gathering phase.

C. EVIDENCE GATHERING

In order to estimate the volatility of the returns associated with our current software investment effort, we need to gather evidence to help derive our estimates. Historically, gathering of evidence using previously completed software-related capital investments as a proxy is a difficult task for the following reasons:

1. The current software investment effort under consideration might be the first of its kind with no known comparables
2. Information is rarely or actively collected and managed in a disciplined fashion
3. Even when information is collected, accessibility by third parties is usually difficult due to the proprietary nature of the information.

Thus more often than not the software executive is faced with the identifying alternate sources of information to either assert or dispute their initial volatility estimates. In our study, we propose the use of two sources of information, the first being historical data (objective approach) and the second being expert opinions obtained using the Delphi method (subjective approach). We choose to use both methods because we believe intuition and judgment (subjective approach) should supplement quantitative analysis (objective approach). More often than not, past success and failures serve as key indicators of the future. Thus historical data can be used to predict and explore "what-if" scenarios on future projects based on the use of forecasting and analytical analysis.

The Delphi Method is a technique first introduced by the RAND Corporation in the 1940's as a methodology for the elicitation of the opinion of an expert or groups of experts to guide decision making by the making predictions about future events. It places emphasis on a iterative, systematic, disciplined and interactive process of individual interviews (usually conducted using questionnaires) and the outcome is based on the Hegelian Principle of achieving consensus through a three step process of thesis, antithesis, and synthesis [48]. In the thesis and antithesis steps, the team of experts present their opinion or views on the given subject, establishing views and opposing views, and consensus is ultimately reached during the synthesis phase as opposing views are brought together to form the new thesis. Widely used as an estimating tool, the Delphi Method has been used to estimate values for factors which appear in software estimation models such as cost estimation [47] and risk estimation. Furthermore, it is one of the approved techniques published by the U.S. Army Cost and Economic Analysis Center in February 2001 for preparing or reviewing economic analyses in support of the decision making process.

In the event that there is no historical data available, we resort to obtaining information using the Delphi Method based on a minimum of two independent estimates provided by two independent experts or two separate groups of independent experts. The first estimate would provide the base case estimate with the remaining two independent estimates serving to asses or refine the initial estimate independently. In the case that we do have historical data but are unable to find projects meeting any or all of the criteria

above, we proceed to “fit” the data to as close as possible to mimic our current software investment effort by employing interpolation techniques to understand and forecast our project based on the trends depicted in the historical data.

1. Data Fitting Techniques

To fit our data, we proceed to determine if the risks depicted in the historical data are positively correlated to the risk presented in our current project. We proceed to conduct a regression analysis by modeling and analyzing the historical data consisting of values of the dependent variable as a function of one or more independent variables and an *error term* which represents unexplained variation in the dependent variable. Each of the parameters is estimated so as to give a "best fit" of the data using the least squares method. The least squares method corresponds to the maximum likelihood criterion if the experimental errors have a normal distribution and the best fit in the least-squares sense is that instance of the model for which the sum of squared residuals has its least value, a residual being the difference between an observed value and the value given by the model [69].

The least squares problems fall into two categories, linear and non-linear representative of linear and non-linear data in our case. The linear least squares problem has a closed form solution, but the non-linear problem does not and is usually solved by iterative refinement; at each iteration the system is approximated by a linear one, so the core calculation is similar in both cases [69].

Confidence limits can be found if the probability distribution of the parameters is known, or an asymptotic approximation is made, or assumed. Likewise statistical tests on the residuals can be made if the probability distribution of the residuals is known or assumed. The probability distribution of any linear combination of the dependent variables can be derived if the probability distribution of experimental errors is known or assumed. However running a regression analysis introduces two problems: autocorrelation and multicollinearity.

Multicollinearity occurs when two or more explanatory variables in the regression model are highly correlated, making it difficult or impossible to isolate their individual

effects on the dependent variable [68]. It is easily solved by obtaining more data or dropping one of the variables. Autocorrelation on the other hand occurs when the error term in one time period is positively correlated with the error term in the previous time period (positive first-order). This leads to downward-biased standard errors (and, thus, to incorrect statistical tests and confidence intervals). The presence of autocorrelation can be tested by calculating the Durbin-Watson statistic below [70].

$$d = \frac{\sum_{t=2}^{n} (e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

Its value always lies between 0 and 4. A value of 2 indicates there appears to be no autocorrelation [70]. If the Durbin–Watson statistic is substantially less than 2, there is evidence of positive serial correlation. As a rough rule of thumb, if Durbin–Watson is less than 1.0, there may be cause for alarm [70]. Small values of *d* indicate successive error terms are, on average, close in value to one another, or positively correlated. Large values of *d* indicate successive error terms are, on average, much different in value to one another, or negatively correlated [70].

Once we have completed the evidence gathering phase, we are now ready to proceed to the analysis phase in which we analyze the information to obtain the volatility of our software investment effort. To accomplish this, we utilize a Monte Carlo Simulation approach.

D. VOLATILITY ESTIMATION: MONTE CARLO SIMULATION

Volatility is used to quantify the risk of the asset based on the standard deviation of the changes of the decision variable within a specific time period divided by its mean to obtain a common sized percentage volatility measure (and the result is annualized). In conventional financial investment theory, the Capital Asset Pricing Model (CAPM) is the most commonly used technique for assessing market risk. The CAPM formula takes into account the asset's sensitivity to non-diversifiable risk (also known as systemic risk or market risk), often represented by the quantity beta (β) in the financial industry, as well

as the expected return of the market and the expected return of a theoretical risk-free asset. From a Real Options perspective, risk is a key driver in the value of a real option, and is positively related to value. In the traditional NPV approach, high volatility signifies high risk, high risk signifies high discount rates, and high discount rates mean lower values. However, a high volatility is linked to high value in Real Options valuation because greater volatility creates a wider range of possible future values of our opportunity as we only will exercise our option if the value of the opportunity exceeds the exercise price [46]. Greater uncertainty on the down side would not hurt as we simply will not exercise. However, greater uncertainty on the up side produces a greater excess of opportunity value over exercise price, and a correspondingly greater option value [46].

A Monte Carlo simulation which is the simulation technique we employ in our study is an advanced computer-based simulation technique which is more or less an extension of traditional simulation techniques of sensitivity and scenario testing in which the critical success drivers or variables that affect the bottom-line variables are simulated thousands of times taking into account interdependencies between the variables to emulate all potential combinations and permutations of outcomes [14]. We therefore model the risks facing our software investment effort using a Monte Carlo approach, and specifically employ the Risk Simulator software provided by Real Options Valuation, Inc. to derive the volatility of the software investment effort.

Estimating volatility from historical data or from the "fitted data" obtained from extrapolating historical data is a fairly simple process. We simply apply the logarithmic relative returns function and then take the sample standard deviation on it, annualize it by multiplying by the square root of periods in a year. However, in the case of the evidence obtained using the Delphi Method, we cannot directly apply this approach because the information elicited is based on subjective probabilities, which cannot be modeled in the traditional sense that classical objective probabilities are modeled. As mentioned earlier on in the chapter, this volatility estimate gives an objective assessment based on the known unknowns and in order to account for the software executive ignorance in the case of "unknown unknowns", especially in innovative software investment efforts, we employ DST to further refine our objective estimates. In other words, the volatility

estimate computed at this point using the historical data might have little or no relationship with the potential volatility of our current software investment effort, because other than telling us that similar projects experienced a certain amount of volatility, it does not tell us the volatility of our software investment effort and the historical estimates might not necessarily reflect intrinsic details that reflect the difference between our current project and the historical projects.

Therefore to refine the historical volatility estimate of the previously completed software investment, we propose the solicitation of additional information using several expert opinions based on the Delphi Method (at least 2 opinions) as our source of information and then use the DST approach to combine the information we received from the experts and establish boundaries of the historical volatility estimates obtained in the form of beliefs and plausibility based on subjective probabilities that take into consideration unique attributes of the current software investment effort. We further explain the mechanics of the DST approach in the next section.

E. VOLATILITY REFINEMENT USING DEMPSTER SHAFER'S THEORY

As mentioned in our introduction, DST is well positioned to address both epistemic and aleotoric uncertainty. While traditionally, probability theory has been used to characterize both types of uncertainty, recent criticisms of the probabilistic characterization of uncertainty claim that traditional probability theory is not capable of capturing epistemic uncertainty [50]. DST is a mathematical theory of evidence originally developed by Dempster in 1967 [49] with roots embedded in Bayesian statistical analysis and later expanded by Shafer in 1976 [49]. While Bayesian inference requires all unknowns to be represented by probability distributions, which awkwardly implies the probability of an event for which we are completely ignorant, DST takes over by introducing belief functions to distinguish ignorance and randomness by assigning probability mass to subsets of parameter space, so that randomness is represented by the probability distribution and uncertainty is represented by large subsets [72]. In other words while Bayesian theory requires probabilities for each uncertainty of interest, the theory of belief functions provides a non-Bayesian way of using mathematical probability

to quantify subjective judgments [51]. It measures degrees of belief [or confidence] for one uncertainty on the probabilities for a related uncertainty.

The premise behind DST is it can be interpreted as a generalization of probability theory where probabilities are assigned to sets as opposed to mutually exclusive singletons. In the case that there is sufficient evidence to permit the assignment of probabilities to single events, the Dempster-Shafer model collapses to the traditional probabilistic formulation where evidence is associated with only one possible event [50]. DST relies on three basic functions: the basic probability assignment function, a primitive of evidence theory which does not refer to probability in the classical sense, and two non-additive continuous measures called *Belief* and *Plausibility* which are both used to combine separate pieces of information (evidence) to calculate the probability of an event, while at the same time defining the upper and lower bounds respectively of an interval that contains the precise probability of a set of interest [23].

Since evidence can be associated with multiple possible events, e.g., sets of events, the evidence in DST can be meaningful at a higher level of abstraction, a key benefit needed at the strategic decision-making level without having to resort to assumptions about the events within the evidential set [50]. Furthermore the DST model can be used to cope with varying levels of precision regarding information with no further assumptions needed to represent the information as demonstrated during a study in addressing uncertainties in systems [50]. We posit that the demonstrated approach also allows for the direct representation of uncertainties associated with software-related capital investments since we can characterize vague inputs as sets or intervals with the resulting output also being a set or an interval.

1. Mechanics of Dempster-Shafer Theory

DST is a theory about two things: 1) Degrees of belief and 2) Weights of evidence, with a key benefit of DST being the ability to represent ignorance in the face of uncertainty especially when there is no information so far. In probability theory, uniform distributions are used to represent ignorance, however the problem with this approach is

that we represent the space of possibilities affected by the probabilities we get [71]. The theory of belief functions is based on two ideas [51]:

1) The idea of obtaining degrees of belief for one question from subjective probabilities of a related question,

2) Dempster's rule for combining such degrees of belief when they are based on independent items of evidence. Degrees of belief obtained in this way differ from probabilities in that they may fail to add to 100%.

Both ideas are consistent with the Real Options pre-conditions as the degrees of belief are established on a frame of discernment meant to address uncertainty. DST starts off by assuming a Universe of Discourse Θ otherwise known as the Frame of Discernment which is a set of mutually exclusive alternatives.

Thus a frame of discernment A of a set of mutually exclusive alternatives or possibilities can be represented as

$$\Theta = \{A_1 \ldots\ldots A_n\} \quad \textit{(1)}$$

where A_1 through A_n represents the set of possibilities or mutually exclusive alternatives. A key stipulation of DST is that it should be only used to combine belief functions that represent independent items of evidence. The independence required is simply probabilistic independence applied to the questions for which we have probabilities, rather than directly to the question of interest. In other words it means that the sources of information (or at least their current properties as sources of information) are selected independently from well-defined populations. For example, given the situation of determining if a specific software requirement would *increase* program costs (Θ_1) or *not increase* program costs (Θ_2), we can represent the number of possibilities as

$$\Theta = \{\Theta_1, \Theta_2\} \quad \textit{(2)}$$

We can then develop a set of propositions based on the three axiomatic propositions of belief functions [71]. These axioms are:

(B1) *Bel* ($\emptyset$) = 0 *(3)*

(B2) $Bel\ (\Theta) = 1$ ……*(4)*

(B3) Given a set of subsets of Θ , $A_1 \ldots\ldots\ldots A_n$

$Bel\ (A_1 \cup \ldots\ldots \cup A_n) \geq \Sigma\, Bel\ (A_i) - \Sigma\, Bel\ (A_i \cap A_j) + \ldots + (-1)^{n+1} Bel\ (A_1 \cap \ldots\ldots \cap A_n)\ldots.$ *(5)*

where axioms (B1) and (B2) are assertions from probability theory asserting necessary things are maximally likely while impossible things are minimally likely, and axiom (B3) asserts the sub-additive nature of DST, which is a key departure from the probability rules. Specifically this is highlighted in inclusion-exclusion rule for probabilities with the "equals" (=) symbol being replaced by the "greater than equals to" ($\geq$) symbol. If there is no evidence to support either hypothesis Θ_1 or Θ_2 (to show that the software requirement would increase or not increase program costs), then a belief function can be developed such that:

$Bel\ (\{\Theta_1\}) = 0 = Bel\ (\{\Theta_2\})$……*(6)*

where "0" corresponds to having no evidence, a key input assumption of DST, which is a deviation from the school of thought in probability theory in that probability theory stipulates the sum of all probabilities must equal 1 as depicted in the equation 7 below.

$P(\{\Theta_1\}) + P(\{\Theta_2\}) = P\ (\Theta) = 1$……*(7)*

Thus under DST the following claim can be asserted:

$Bel\ (\{\Theta_1\}) + Bel\ (\{\Theta_2\}) \leq 1$ ……*(8)*

and

$Bel\ (\Theta) = 1$ and $Bel\ (A) = 0$ for every $A \subset \Theta$ ……*(9)*

where the belief function (Eqn 9) represents ignorance due to the lack of information to support either hypothesis (Θ_1 or Θ_2) regardless of the set of possibilities in Θ and asserts that some possibility in Θ must be true hence both possible propositions are equally likely. Furthermore, since $Bel(\{\Theta_1\})$ and $Bel(\{\Theta_2\})$ are independent (i.e.

historical data and information provided by the panel of experts – the Delphi Method), the following claim can be made:

$Bel(\{\Theta_1\}) = 1/3, Bel(\{\Theta_2\}) = 0$……..……*(10)*

Eqn 10 holds because the evidence supporting one hypothesis is not evidence against its rivals until a fair bit of evidence has been gathered. In other words, it is not a zero-sum game until late in the game [71]. In the case that we have *"Masses"* of probabilities, *belief* in a hypothesis is constituted by the sum of the masses of all sets enclosed by it (i.e. the sum of the *masses* of all subsets of the hypothesis). Masses represent probabilities that are strictly additive, however they are distributed over all propositions rather than the singletons of Θ and they encode degrees of belief [71]. Therefore a belief function based on masses can be represented as a *mass function*, which is basically a function on the subset of possibilities 2^{Θ} satisfying the two axioms below (Eqn 11 & 12), leading to the derivation of a function m: $2^{\Theta} \rightarrow$ [0,1] called a *basic probability assignment*.

(M1) = $m(\emptyset) = 0$……*(11)*

(M2) $\sum_{A \in \Theta} m(A) = 1$ …..…..… *(12)*

where

m(A) is A's *basic probability number* whose value expresses the proportion of all relevant and available evidence that supports the claim that a particular element of Θ (the universal set) belongs to the set A but to no particular subset of A [50]. In other words it represents the strength of some evidence or our exact belief in the proposition represented by A. The value of *m*(A) pertains only to the set A and makes no additional claims about any subsets of A. Any further evidence on the subsets of A would be represented by another basic probability number, e.g., B $\subset$ A. m(B) would the basic probability number for the subset B [50]. The function m: $2^{\Theta} \rightarrow$ [0,1] becomes a *belief function* if it satisfies

Bel (Ø) = 0, *Bel* (Θ) = 1……..…….*(13)*

Since *masses* represent probabilities that are strictly additive, and encode degrees of belief accordingly, the lower bound *belief* for a set A defined as the sum of all the

basic probability assignments of the proper subsets (B) of the set of interest (A) ($B \subseteq A$). In other words, it is the degree of belief in A which is the sum of all the mass allotted to subsets of A and can be represented by the following formula [71].

$$Bel(A) = \sum_{B \subseteq A} m(B) \quad \text{.....(14)}$$

Plausibility on the other hand is 1 minus the sum of the masses of all sets whose intersection with the hypothesis is empty. It is an upper bound on the possibility that the hypothesis could possibly happen, i.e., it "could possibly happen" up to that value, because there is only so much evidence that contradicts that hypothesis [23]. In essence plausibility expresses how much we should believe in A if all currently unknown facts were to support A. Plausibility can be represented as follows:

$$Pl(A) = 1 - Bel(\overline{A}) \quad \text{.....(15)}$$

where

$\overline{A}$ is the complement of A in the classical sense. Consequently given the computation of both belief and plausibility above, the precise probability of an event in the classical sense lies between the upper and lower bounds established by plausibility and belief respectively. In the event that plausibility equals to belief as shown in Eqn 15, all the probabilities are classical probabilities

$$Bel(A) = P(A) = Pl(A) \quad \text{.....(16)}$$

F. DEMPSTER'S RULES FOR COMBINATION OF EVIDENCE

Combining information or evidence from multiple sources (historical data and Delphi method) in the form of belief assignments serves to aggregate the information with respect to its constituent parts. Dempster proposed a standard combination rule which can be represented as [50]:

$$m_{12}(A) = \sum_{B \cap C = A} \frac{m_1(B) m_2(C)}{1 - K} \text{ when } A \neq \emptyset \quad \text{.....(17)}$$

where

$$K = \sum_{B \cap C = \emptyset} m1(B)m2(C)$$

which is computed by summing the products of the belief probability assignments (bpa's) of all sets where the intersection is null represents basic probability mass associated with conflict, and m_{12}(A) is calculated from the aggregation of two bpa's m_1 and m_2.

1. Mechanics of Dempster Rule of Combination

Assuming we have two pieces of evidence, based on say historical data and expert judgment (Delphi Method), we now combine the pieces of evidence from both sources using the Dempster's combination rules by computing the orthogonal sum of both pieces of evidence. First we determine all the pairs of sets whose intersection is A for a given set A such that

$A_1 \cap A_2 = A$. We then add up the basic probability assignments $m_1(A_1)$ and $m_2(A_2)$, giving us

$$\sum_{A_1 \cap A_2 = A} m_1(A_1) m_2(A_2) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{(19)}$$

Then the orthogonal sum of m_1 and m_2 defined by $m = m_1 \oplus m_2$ could then be given as $m(\emptyset) = 0$ and is demonstrated in the matrix below (Table 4) in which we compute the orthogonal sum of three hypothetical risk factors (Risk1, Risk2 and Risk} affecting a software investment program based on two independent expert assessments.

			Independent Expert 1 Subjective estimates on Objective probabilities		
	Risk Factors		{Risk1} m1= 0.80	{Risk1,Risk2} m1= 0.15	{Risk1,Risk2,Risk3} m1 = 0.05
Independent Expert 2 Subjective estimates on 3 Objective probabilities	{Risk1}	m2 =0.70	m1({Risk1})*m2({Risk1})	m1({Risk1,Risk2})*m2({Risk1})	m1({Risk1,Risk2,Risk3})*m2({Risk1})
	{Risk1,Risk 2}	m2 =0.20	m1({Risk1})*m2({Risk1,Risk2})	m1({Risk1,Risk2})*m2({Risk1,Risk2})	m1({Risk1,Risk2,Risk3})*m2({Risk1,Risk2)
	{Risk1,Risk2,Risk3}	m2 =0.10	m1({Risk1})*m2({Risk1,Risk2,Risk3})	m1({Risk1,Risk2})*m2({Risk1,Risk2,Risk3})	m1({Risk1,Risk2,Risk3})*m2({Risk1,Risk2,Risk3})

Table 4. **Orthogonal Sum Of Basic Probability Assignments.**

Based on the matrix (Table 4), we can obtain the resulting three evidence functions.

$m_1 \oplus m_2(\{Risk1\})$
$= m_1(\{Risk1\}) * m_2(\{Risk1)\} + m_1(\{Risk1,Risk2\}) * m_2(\{Risk1\})$
$+ m_1(\{Risk1,Risk2,Risk3\}) * m_2(\{Risk1\}) + m_1(\{Risk1\}) * m_2(\{Risk1, Risk2\}) +$
$m_1(\{Risk1\}) * m_2(\{Risk1, Risk2, Risk3\})$

$m_1 \oplus m_2(\{Risk1,Risk2\})$
$= m_1(\{Risk1,Risk2\}) * m_2(\{Risk1,Risk2\})$
$+ m_1(\{Risk1, Risk2\}) * m_2(\{Risk1, Risk2, Risk3\})$
$+ m_1(\{Risk1, Risk2, Risk 3\}) * m_2(\{Risk1, Risk2\})$

$m_1 \oplus m_2(\{Risk1,Risk2,Risk3\})$
$= m_1(\{Risk1,Risk2,Risk3\}) * m_2(\{Risk1,Risk2,Risk3\})$

We now proceed to put all these concepts together through the use of an example in the following section.

G. VOLATILITY COMPUTATION: EXAMPLE

In order to estimate the volatility of the future returns on a software investment effort based on requirements, scheduling and cost estimation risks using our approach, we develop a simple hypothetical example in which a software executive needs to make an investment in the software platform of a KC-X Air refueling aircraft. In this example, the key assumptions are as follows:

The discounted future benefits or value of the investment program ($305 million) has been justified using traditional Net Present Values (NPV) analysis with a projected cost of ($55 million) and resulting in a NPV of $250 million (we discuss the topic of valuing software investments in the next chapter). Similarly based on a preliminary analysis of the software investment program using the "2**T**" approach proposed in the previous chapter, the executive has identified requirements uncertainties as being a key

contributing factor to risk. We further assume there is historical data available and we are able to come up with initial estimates based on the historical data at hand. The executive also has the option of using only estimates obtained from the Delphi Method (expert opinions). However, the executive decides to use historical data. The project was projected to contain 1000 requirements based on the Use Cases and expected to be delivered in 12 months. Therefore based on these underlying assumptions, we develop the framework of this problem in a model and run a Monte Carlo simulation using the Risk Simulator software provided by Real Options Valuation, Inc.

Specifically, the software executive starts of by developing and fitting his data estimates based on historical data of previous programs of similar size, scope and complexity and creates three scenarios, A, B and C, of the most likely, least likely and likely based on previous experiences depicted in the historical data. Next we select a probability distribution. Since we are dealing with random quantities which are positive in nature and whose values cannot fall below zero we select the lognormal distribution which utilizes the following four mathematical constructs to compute four key measures known as "moments" (returns, risk, skewness and kurtosis) of our logarithm distribution (A moment is used to describe the centroid of a statistical set of data encapsulating the area of concern). These measures are computed using the following formulas as outlined in [14]

$$\text{f(x)} = \frac{1}{x\sqrt{2\pi \ln(\sigma)}} e^{-\frac{[\ln(x) - \ln(\mu)]^2}{2[\ln(\sigma)]2}} \quad \text{for } x > 0; \mu > 0, \sigma > 0$$

$$\text{mean} = \exp\left(\mu + \frac{\sigma^2}{2}\right)$$

$$\text{standard deviation} = \sqrt{\exp(\sigma^2 + 2\mu)[\exp(\sigma^2) - 1]}$$

$$\text{skewness} = \left[\sqrt{\exp(\sigma^2) - 1}\right](2 + \exp(\sigma^2))$$

$$\text{excess kurtosis} = \exp(4\sigma^2) + 2\exp(3\sigma^2) + 3\exp(2\sigma^2) - 6$$

The model is run and the simulation calculates numerous scenarios of the model by repeatedly picking values from the lognormal distribution for the uncertain variables using the values in the model. Based on the model results, the volatility of the software investment program due to risk of requirements increase was computed by the model as 0.0017. This volatility measure was determined by computing the logarithmic returns of our $305 million investment based on the risk of requirement changes over the three scenarios. This volatility resulted in the cost exceeding the projected budget by $1,829,697.42 and consequently led to the reduction of NPV from $250,000,000 to $248,170,302.58.

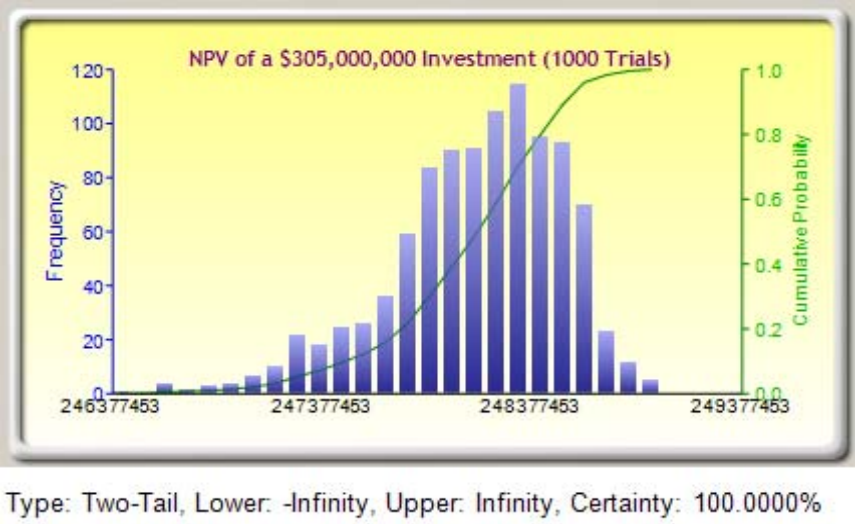
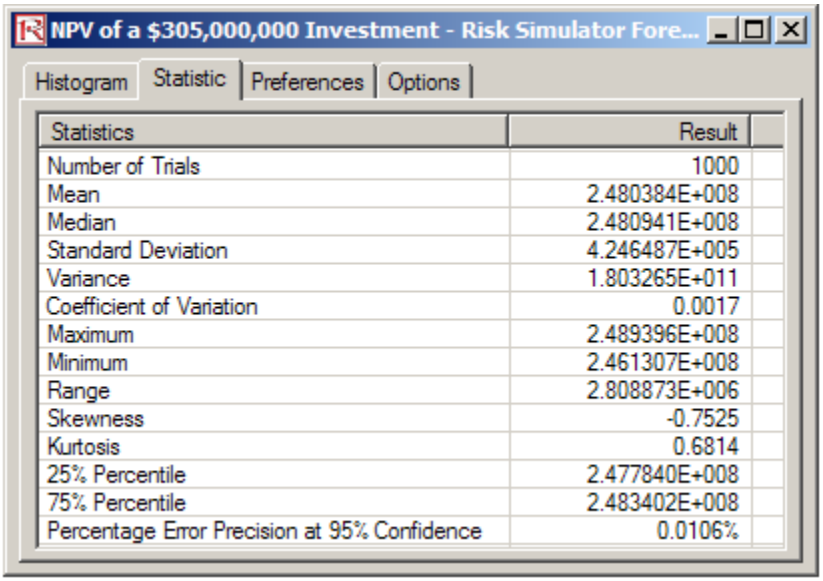


Figure 22. NPV Returns on $305 million Software Investment. The frequency histogram shows the frequency counts of values occurring in the total number of trials simulated. The vertical bars shows the frequency of a particular NPV occurring out of the total number of trials, while the cumulative frequency depicted by the smooth line shows the total probability of all values at and below the maximum NPV occurring in the forecast.

Figure 22 above depicts a graphical distribution of the returns associated with our investment while Figure 23 on the next page depicts the statistics associated with the computations.

In further analysis conducted in the model to determine the likelihood/probability of requirements changes by phase in the software development process, it was

determined that the probability of requirements increases during the specification, development and validation phase of the software development process were 0.32, 0.41 and 0.27 respectively. These associated histograms and statistics are depicted in Figures 24 – 29.



Statistics	Result
Number of Trials	1000
Mean	2.480384E+008
Median	2.480941E+008
Standard Deviation	4.246487E+005
Variance	1.803265E+011
Coefficient of Variation	0.0017
Maximum	2.489396E+008
Minimum	2.461307E+008
Range	2.808873E+006
Skewness	-0.7525
Kurtosis	0.6814
25% Percentile	2.477840E+008
75% Percentile	2.483402E+008
Percentage Error Precision at 95% Confidence	0.0106%

Figure 23. Volatility of the returns on the $305 million Software Investment. The forecast statistics summarizes the distribution of the forecast values in terms of the four moments of a distribution.

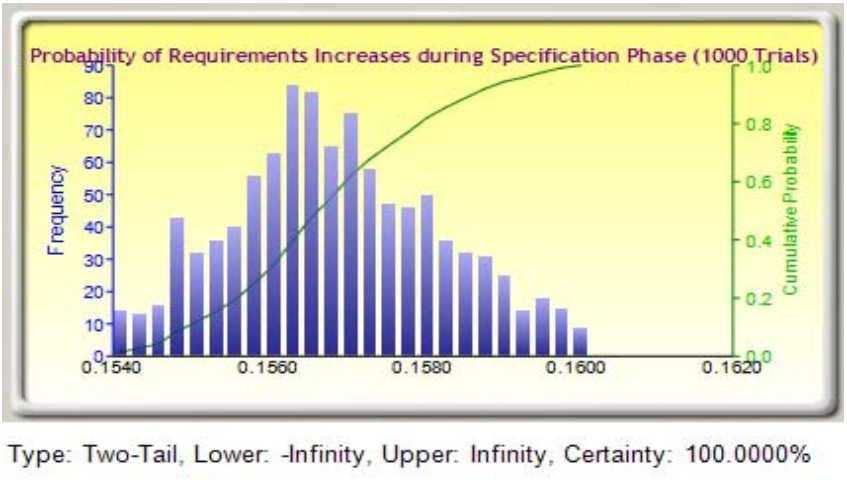
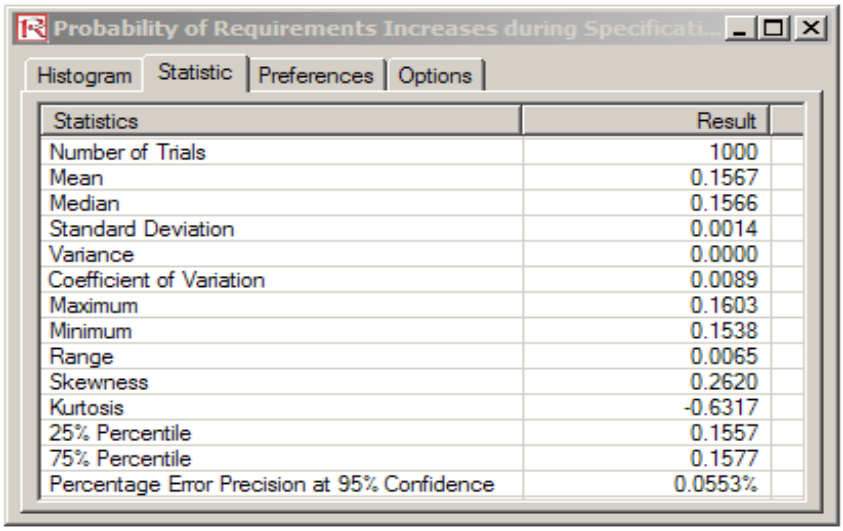


Figure 24. Probability of Requirements Increases during the Specification Phase.

The frequency histogram (Figure 24 on previous page) shows the frequency counts of values occurring in the total number of trials simulated. The vertical bars shows the frequency of a particular probability of requirements increase during the specification phase occurring out of the total number of trials, while the cumulative frequency depicted by the smooth line shows the total probability of all values at and below the maximum probability of requirements increase during the specification occurring in the forecast.



Statistics	Result
Number of Trials	1000
Mean	0.1567
Median	0.1566
Standard Deviation	0.0014
Variance	0.0000
Coefficient of Variation	0.0089
Maximum	0.1603
Minimum	0.1538
Range	0.0065
Skewness	0.2620
Kurtosis	-0.6317
25% Percentile	0.1557
75% Percentile	0.1577
Percentage Error Precision at 95% Confidence	0.0553%

Figure 25. Statistics of Requirements Increases during Specification Phase. The forecast statistics summarizes the distribution of the forecast values of the probability of requirements increases during the specification phase in terms of the four moments of a distribution.

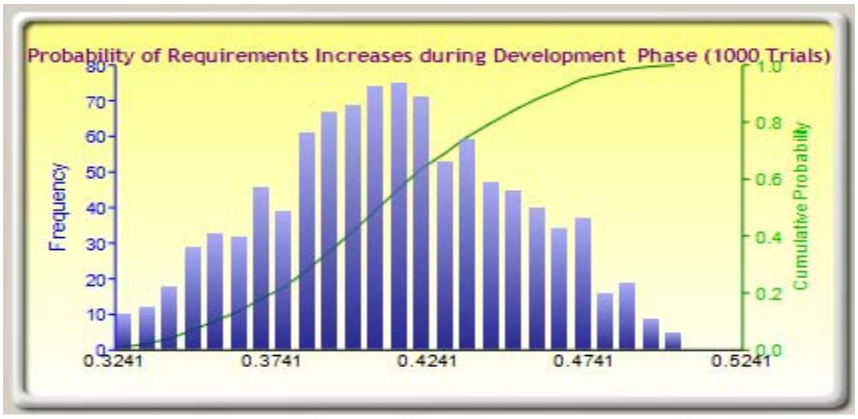
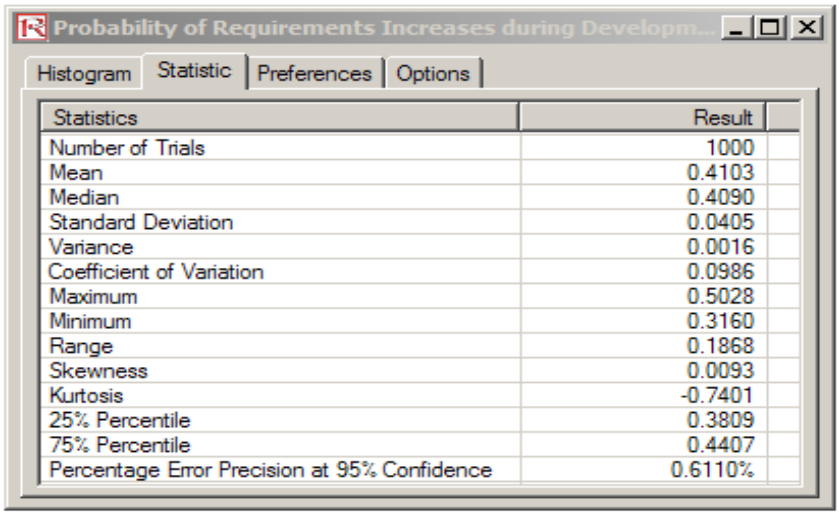


Figure 26. Probability of Requirements Increases during the Development Phase. The frequency histogram (Figure 26 on previous page) shows the frequency counts of values occurring in the total number of trials simulated. The vertical bars shows the frequency of a particular probability of requirements increase during the development phase occurring out of the total number of trials, while the cumulative frequency depicted by the smooth line shows the total probability of all values at and below the maximum probability of requirements increase during the development phase occurring in the forecast.



Statistics	Result
Number of Trials	1000
Mean	0.4103
Median	0.4090
Standard Deviation	0.0405
Variance	0.0016
Coefficient of Variation	0.0986
Maximum	0.5028
Minimum	0.3160
Range	0.1868
Skewness	0.0093
Kurtosis	-0.7401
25% Percentile	0.3809
75% Percentile	0.4407
Percentage Error Precision at 95% Confidence	0.6110%

Figure 27. Statistics of Requirements Increases during Development Phase. The forecast statistics summarizes the distribution of the forecast values of the probability of requirements increases during the development phase in terms of the four moments of a distribution.

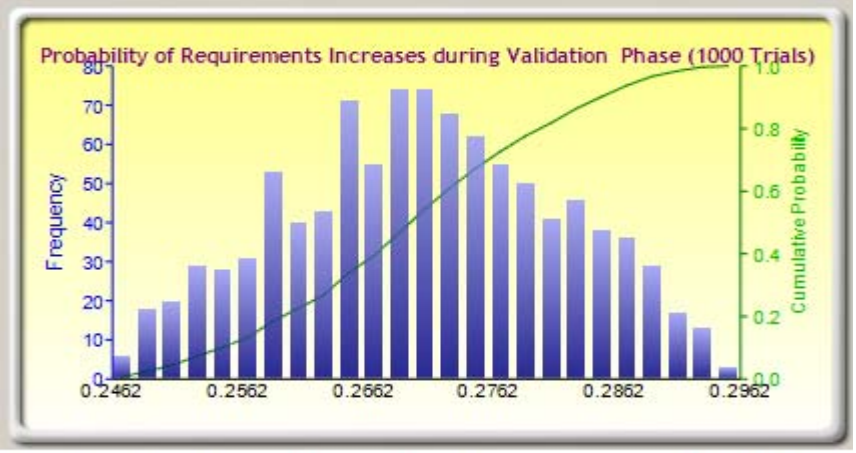


Figure 28. Probability of Requirements Increases during the Validation Phase. The frequency histogram (Figure 28 on previous page) shows the frequency counts of values occurring in the total number of trials simulated. The vertical bars shows the frequency of a particular probability of requirements increase during the validation phase occurring out of the total number of trials, while the cumulative frequency depicted by the smooth line shows the total probability of all values at and below the maximum probability of requirements increase during the validation phase occurring in the forecast.

Probability of Requirements Increases during Validation...

Histogram | Statistic | Preferences | Options

Statistics	Result
Number of Trials	1000
Mean	0.2691
Median	0.2691
Standard Deviation	0.0109
Variance	0.0001
Coefficient of Variation	0.0407
Maximum	0.2953
Minimum	0.2432
Range	0.0521
Skewness	-0.0263
Kurtosis	-0.7022
25% Percentile	0.2612
75% Percentile	0.2777
Percentage Error Precision at 95% Confidence	0.2521%

Figure 29. Statistics of Requirements Increases during Validation Phase. The forecast statistics summarizes the distribution of the forecast values of the probability of requirements increases during the validation phase in terms of the four moments of a distribution.

H. INTERPRETING THE FORECAST RESULTS

To obtain the charts above, we examined two of the three variables in question - requirements increases during the specification and requirements increases during the development phase. We selected the Lognormal distribution from the Risk Simulator Tool due to the multiplicative impact of the variables in question on the overall volatility of the software investment.

The four moments described as the first, second, third and fourth moments which the forecast statistics summarizes are respective measures of the following parameters [14].

1. Measure of Returns

2. Measure of Risk

3. Measure of Skewness

4. Measure of Kurtosis

All these four moments provide a more comprehensive view of the project under analysis and are explained in detail in [14] as follows: The first moment depicted by the mean value measures the expected rate of return on the investment effort, by measuring the location of the different scenarios of the investment effort and the average possibility of the outcomes. The second moment depicted by mathematical functions such as standard deviation, variance, coefficient of variation and volatility provides a risk measure of the investment effort through a measure of the variability of the variable (spread of the distribution). The third moment provides a measure of skewness by taking into account the differences in the skewness of a distribution should a situation arise that two or more different investment projects under consideration have the same mean and standard deviation. The fourth moment provides a measure of the peakedness (kurtosis) of the distribution as a way of differentiating between two or more investment projects with similar mean, risk and skewness distribution (first, second and third moments). This means that although the risk and returns are identical, the probability of extreme and

catastrophic events occurring is higher for a high kurtosis distribution and a higher excess kurtosis value indicates that the downside risks are higher.

The volatility of requirements can be observed in Figures 24, 26 and 28. The kurtosis is given by -0.6317, -0.7401 and -0.7022 for the specification, development and validation phases respectively. From our observations, we can see that that the probability of requirements changes are much higher during the development phase.

I. APPLICATION OF DEMPSTER-SHAFER THEORY

At this point since we have determined the volatility of our software project as well as the associated probabilities of the impact of the risk of requirements changes, the executive would like to further reduce the uncertainty in the objective probability estimates to better estimate the impact of requirements changes, so that the appropriate option can be created where applicable. Therefore we now resort to the DST approach.

We begin by partitioning our problem of requirements increases based on the phases of the software engineering (SE) process, so as to identify the impact of the requirements increases on the different phases of the SE process on the value of the asset. We accomplish this by collecting additional evidence using the Delphi Method to gather more information based on the expert opinions of two independent experts.

We then develop our frame of discernment which consists of the set of mutually exclusive alternatives or possibilities which the risks of requirements increases pose to our current software investment effort. Therefore Θ would be the set consisting of all the possible phases in which the risk of requirements increases affect the value of the software investment. We denote these as follows

- Requirements increase during Software Specification (SS)
- Requirements increase during Software Development (SD)
- Requirements increase during Software Validation (SV)

Thus our frame of discernment is $\Theta = \{SS, SD, SV\}$

Moving along the lines of our example, suppose independent expert's #1 determines subjectively that the initial estimates obtained from the historical sources are 0.90 reliable. i.e. requirements increases during the development phase contributed the most to the volatility of the software-investment. Therefore under DST belief function we can establish a 0.9 degree of belief that increases in requirements during the development phase would lead to a increase in costs, and a 0 (not 0.1) degree of belief that increase in requirements during the development phase would not lead to increases in costs. (Under classical probability theory, the software executive can say that the independent expert's Delphi Method is 0.1 unreliable.)

The 0.9 and the 0, (which do not add to 1) together constitute a "belief function", where the "0" degree of belief does not imply that independent expert #1 is sure that increases in requirements during the development phase would not lead to cost increases, as is the case in classical probability, but simply implying that the historical estimates gives independent expert #1 no reason to belief that increases in requirements during the development phase would not lead to cost increases.

We can then develop our propositions based on the following axioms

(B1) $Bel\ (\emptyset) = 0$

(B2) $Bel\ (\Theta) = 1$

(B3) Given a set of subsets of Θ , $A_1 \ldots\ldots\ldots A_n$

$$Bel\ (A_1 \cup \ldots\ldots \cup A_n) \geq \Sigma\, Bel\ (A_i) - \Sigma\, Bel\ (A_i \cap A_j) + \ldots + (-1)^{n+1} Bel\ (A_i \cap \ldots\ldots \cap A_n)$$

Therefore independent expert #1's belief *Bel*(A) in any subset Θ could then be given as follows:

Bel({SD}) = 0.9

Bel({SD, SS, SV}) = 1

Note that the belief function basically conveys the same information as the basic probability assignments as it represents evidence in favor of the proposition. For example

we can also assign basic probability assignments based on Independent experts #1 opinion as shown below since each element is the sum of the probabilities of the sets enclosed by the element. (the concept of masses), since for example SS could further consist of interface and operating systems requirements as its subset both of which have their associated probabilities.

$m(\{SD\}) = 0.9$

$m(\{SD, SS, SV\}) = 0.1$

We can then compute plausibility using the equation below

$$Pl(A) = 1 - Bel(\overline{A})$$

To further compute $Bel\ (\overline{A})$ as a doubt function

$Dou(\{SD\}) = Bel(\{SS, SV\}) = 0$

$Dou(\{SD, SS, SV\}) = Bel\ (\emptyset) = 0$

Therefore our upper probability function in the form of plausibility for each subset is

$Pl(\{SD\}) = 1 - Dou(\{SD\}) = 1$

$Pl(\{SD,SS,SV\}) = 1 - Dou(\{SS,SD,SV\}) = 1$

To illustrate the concept of combination of information from an additional source of information (expert opinion), we present two scenarios. The first scenario depicts the second expert opinion providing information consistent with the objective estimates and the second providing conflicting evidence against the objective estimates.

In our first scenario, say the independent expert #2's also has a similar subjective probability of 0.90 on the reliability of requirements increases affecting the specification phase based on the assessment of historical estimates. Thus based on the second source of information (independent expert #2's Delphi Method) we can reasonably assign probability assignments to each of the elements as follows:

$m(\{SD\}) = 0.9$

$m(\{SD, SS, SV\}) = 0.1$

and also develop the corresponding belief function

Bel({SD}) = 0.9

Bel({SD, SS, SV}) = 1

Similarly the computation of plausibility mimics the plausibility computation above.

Dou({SD}) = *Bel*({SS, SV}) = 0

Dou({SD, SS, SV}) = *Bel* (Ø) = 0

Therefore our upper probability function in the form of plausibility for each subset is

Pl({SD}) =1 - Dou({SD}) = 1

Pl({SS,SD,SV}) =1 - Dou({SS,SD,SV}) = 1

We now attempt to combine the information from both independent expert #1's & #2's Delphi Method in order to obtain our evidence functions. We begin this process by creating a matrix and multiplying the masses from the associated column and row of both sources of information in order to calculate the combined basic probability assignment for a particular cell. In other words to compute the orthogonal sum in the form $m = m_1 \oplus m_2$ where m_1 and m_2 represent the basic probability on our frame of discernment Θ. Table 5 below shows the basic template we use for our computations.

		Indpendent Expert 1	
	Cost Increase Factors	{SD} m1	{SD, SS, SV} m1
Independent Expert 2	{SD} m2	m1({SD})*m2({SD})	m1({SD,SS,SV})*m2({SD})
	{SD, SS,SV} m2	m1({SD})*m2({SD,SS,SV})	m1({SD,SS,SV})*m2({SD,SS,SV})

Table 5. **Matrix Template Used To Compute Orthogonal Sum Of Basic Probability Assignments**

We apply this template to our scenario and are able to come up with the combined basic probability assignments as shown in the matrix below (Table 6) by using Eqn 19 above given by the equation on the next page.

		Indpendent Expert 1	
	Cost Increase Factors	{SD} m1= 0.90	{SD,SS,SV} m1 = 0.1
Independent Expert 2	{SD} m2 =0.90	0.81	0.09
	{SD,SS,SV} m2 =0.1	0.09	0.01

Table 6. **Orthogonal Sum Of Basic Probability Assignments For Consistent Evidence**

$$\sum_{A_1 \cap A_2 = A} m_1(A_1) m_2(A_2)$$

Since the information provided by independent expert #1's & #2's Delphi Method is consistent. We obtain the following evidence functions

$m_1 \oplus m_2(\{SD\}) = 0.81 + 0.09 + 0.09 = 0.99$

$m_1 \oplus m_2(\{SD, SS,SV\}) = 0.01$

Therefore given the evidence presented by m_1 and m_2, we can say the most probable belief for this universe of discourse is consistent with the probability assignments of both belief functions and no further action is needed besides determining the option to address the risk presented.

In the event that the sources of information contradict each other as is the case in our second scenario, such that while independent expert #1's evidence indicates a 0.9 probability of belief, independent expert #2's evidence indicates a 0.7 probability of belief, we assign the following probability assignments.

$m(\{SD\}) = 0.7$

$m(\{SD, SS, SV\}) = 0.3$

and also develop the corresponding belief function

$Bel(\{SD\}) = 0.7$

$Bel(\{SD, SS, SV\}) = 1$

Similarly the computation of plausibility mimics the plausibility computation above.

Dou({SD}) = *Bel*({SS, SV}) = 0

Dou({SD, SS, SV}) = *Bel* (Ø) = 0

Therefore our upper probability function in the form of plausibility for each subset is

Pl({SD}) =1 - Dou({SD}) = 1

Pl({SS,SD,SV}) =1 - Dou({SS,SD,SV}) = 1

Thus we are able to come up with the following matrix of combined probabilities.

		Indpendent Expert 1	
	Cost Increase Factors	{SD} m1= 0.90	{SD,SS,SV} m1 = 0.1
Independent Expert 2	{SD} m2 =0.70	0.63	0.07
	{SD,SS,SV} m2 =0.30	0.27	0.03

Table 7. **Orthogonal Sum Of Basic Probability Assignments For Conflicting Evidence**

We obtain the resulting evidence functions:

$m_1 \oplus m_2(\{SD\}) = 0.63 + 0.27 + 0.07 = 0.97$

$m_1 \oplus m_2(\{SD, SS,SV\}) = 0.03$

To examine the degree of conflict we revisit Eqn 17 above depicted as:

$$m_{12}(A) = \sum_{B \cap C = A} \frac{m_1(B)m_2(C)}{1-K} \text{ when } A \neq Ø$$

where

$$K = \sum_{B \cap C = Ø} m_1(B)m_2(C)$$

However since we have no null intersections, the sum of all such sets is in our matrix in Table 7 is equal to 0 meaning there is no conflict in the two expert's opinion.

The plausibility (certainty) of this belief is computed based on the following doubt functions:

Dou({SD, SS, SV}) = *Bel* (Ø) = 0

Dou({SD}) = *Bel* (SS, SV) = 0

Thus plausibility is as follows:

Pl({SD, SS, SV) = 1- Dou({SD, SS, SV}) = 1

Pl({SD) = 1- Dou({SD }) = 1

The resulting value of 1 in both cases denotes the upper probability function and expresses how much confidence there is that ({SD, SS, SV}) and ({SD}) risk contributes to volatility. Based on the information derived from the matrix we can establish the following joint beliefs

Joint Belief of Independent expert #1 and #2 estimates in {SD} is 0.97

Joint Belief of Independent expert #1 and #2 estimates in {SD, SS, SV} is 1

Within the context of our assumptions, the joint belief in {SD} of 97% infers that both independent experts have a high belief in the propositions surrounding requirements increases during the development phase. Any variations between inferred probability assignments based on the mass of evidence under this joint belief and our initial volatility estimates would reflect inconsistencies. These variations are captured and used to refine the initial probability estimates to reflect the new "findings" which are then modeled using a Monte Carlo simulation to derive new estimates for the project and an overall volatility for the project.

For example, in our scenario, say after we revise our probability estimates to reflect the findings based on the application of the approach and model these new estimates, we obtain a new volatility estimate of 0.0012 which infers a reduction in our original volatility estimate of 0.0017. This new volatility of the software project would result in an increase of NPV from the initial computed estimate of $248,170,302.58 to $248,762,936.51 meaning that since our project is less volatile than we initially assumed, our NPV would increase to reflect the more accurate valuation in light of the revised volatility of our investment effort. Figure 30 on the next page depicts the histogram reflecting our new data under this scenario.

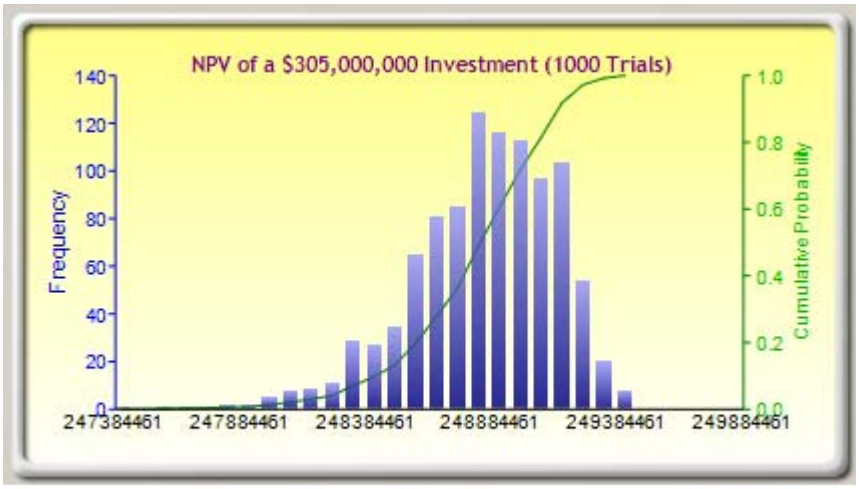
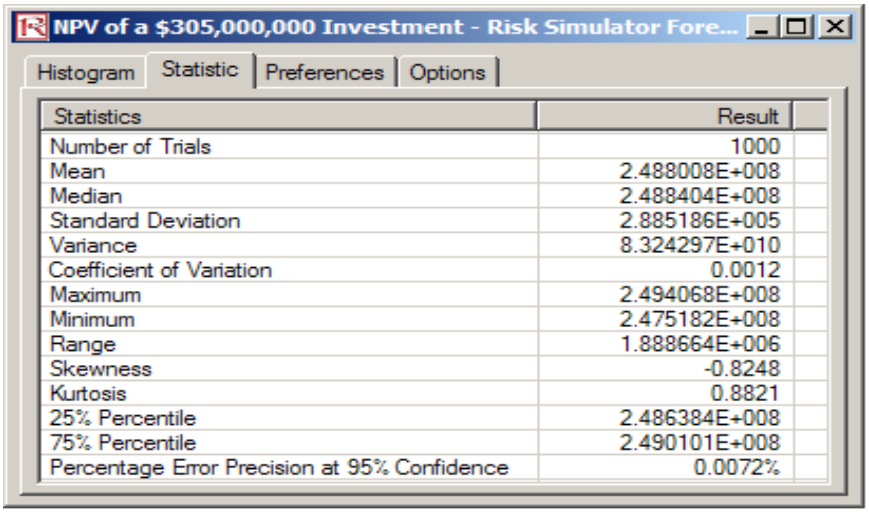


Figure 30. **Revised NPV Returns on $305 million Software Investment. The frequency histogram shows the frequency counts of values occurring in the total number of trials simulated. The vertical bars shows the frequency of the revised NPV returns occurring out of the total number of trials, while the cumulative frequency depicted by the smooth line shows the total probability of all values at and below the maximum revised NPV occurring in the forecast.**



Statistics	Result
Number of Trials	1000
Mean	2.488008E+008
Median	2.488404E+008
Standard Deviation	2.885186E+005
Variance	8.324297E+010
Coefficient of Variation	0.0012
Maximum	2.494068E+008
Minimum	2.475182E+008
Range	1.888664E+006
Skewness	-0.8248
Kurtosis	0.8821
25% Percentile	2.486384E+008
75% Percentile	2.490101E+008
Percentage Error Precision at 95% Confidence	0.0072%

Figure 31. **New NPV Returns on $305 million Software Investment based on refined probabilities. The forecast statistics summarizes the distribution of the revised forecast values in terms of the four moments of a distribution.**

J. ANALYSIS

We can then repeat this whole process as often as needed until we feel confident that we have reduced all our uncertainties as much as possible. It is also important to note that other special cases might also exist related to categorical, completely non-probabilistic information. In these cases, while one is 100% confident in the reliability of the source of information or evidence, the information is still lacking in completely answering our question. For example in hypothetical example drawn along the lines of reasoning in [51] there might be conclusive evidence, that a change in a software requirement might affect one of three places in the overall software, without any clue as to which one. This calls for a belief function that assigns a degree of belief of 100% to the three places as a set, but a degree of belief of 0% to each of the three individually.

In our example, we are dealt with a question that has only two answers (whether or not increases in requirements are associated with increases in costs). In the case that a question has more than two answers, belief functions can be derived by determining the degree of belief for each answer and for each set of answers, although the belief function may be very complex if the number of answers (or the size of the "frame") is large. Now that we have successfully depicted how we estimate volatility, we proceed to the next phase of our process, which is the development of the "real option" to mitigate the risks. We depict this in the following chapter.

V. REAL OPTIONS FRAMEWORK FOR SOFTWARE INVESTMENTS

> …there is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success than to take the lead in the introduction of a new order of thing…
>
> (Niccolo Machiavelli, philosopher, 1469-1527)

A. INTRODUCTION

The underlying premise of the Real Options (RO) approach is the ability to link uncertainty to the value of flexibility within the real options framework. In the previous two chapters, we addressed the critical issue of uncertainty identification and quantification in order to determine volatility, a key input into the real options framework. We also established that uncertainty affects decision associated with software-related capital investments. In this chapter, we build on the body of knowledge established thus far and proceed to establish an RO framework that could be used to address software-related capital investments under the underlying assumption that when properly utilized by a well empowered management team (another precondition for the application of RO theory), it would better guide the decision-making process. However, before we establish the framework we first discuss the issue of valuing the underlying software investment.

B. BASE CASE VALUATION OF THE UNDERLYING ASSET

In Real Options risk-based analysis, the second most important measure besides volatility estimation is the valuation of the underlying asset being considered for acquisition. This is typically the first activity accomplished in a software-related capital investment effort. The RO approach calls for the existence of a financial model before it could be applied. This involves presenting or developing a financial model of the underlying asset, as the real options approach requires the use of an existing discounted cash flow model despite their limitations. In a discussion of the limitations of some of the

investment valuation approaches in the assessment of previous work done in Chapter 2, it was asserted that the limitations of the existing valuation models were primarily due to the lack of consideration of managerial flexibility. Net Present Value (NPV) is measured in today's dollars and its computation is based on the principle of *discounting* in which all projected future cash flows representing estimated costs, cost savings, and revenues at various points during the useful lifetime of the project, are *discounted* utilizing a discount rate. This discount rate captures the *opportunity cost* and the cost of money (based on interest rates) associated with the underlying investment, *back* to the present time under the assumption that one dollar today is worth $(1 + r)^t$ dollars at time *t* in the future [52].

The traditional NPV formula is given as

$$\text{NPV} = \sum_{t=1}^{t} \frac{C_t}{(1+r)^t} - C_0$$

Similarly, a discounted cash flow (DCF) analysis can be computed as

$$\text{DCF} = \frac{CF_1}{(1+r)^1} + \frac{CF_2}{(1+r)^2} + \ldots\ldots + \frac{CF_n}{(1+r)^n}$$

where CF = Cash Flow, r = discount rate and t = time

It must however be emphasized that in performing a discounted cash flow analysis the discounting factor is concerned with two things: the discounting convention and the discount rate itself. There are several discounting conventions ranging from continuous versus discrete, midyear versus end-of-year, to beginning versus end-of-period. It is critical to understand the convention being used in order to maintain consistency in results as each of the discounting conventions produce different results. The second factor, discount rate is normally the weighted average cost of capital (WACC) for the cash flow series. However, this is considered to be somewhat problematic because of risks ranging from organizational capital structure and investment policies to the flaws in the Capital Asset Pricing Model (CAPM).

As detailed financial data is not always readily or publicly available for U.S. government investment project, we rely on guidance provided by the U.S. Office of

Management and Budget (OMB) for NPV computation [53]. OMB states that the standard criterion for deciding whether a government program can be justified on economic principles is based on the discounted monetized value of expected net benefits (i.e., benefits minus costs). The guidance further states that "social" net benefits and not the benefits and costs to the federal government, should be the basis for evaluating government programs or policies that have effects on private citizens or other levels of government. Consequently, the emphasis on "gained" social benefits either real or perceived needs to be factored into NPV computations in software related capital investments funded and operated by the government. In the case of non-measurable benefits and costs, further analysis is needed to guide decision makers when they have to weigh the net non-measurable benefits against net measurable costs.

From a DoD perspective, the perceived value created of any software investment is gauged from the ability of the investment to contribute meaningfully to the DoD's Strategic, Operational and Tactical objectives, be it for the war-fighter or the running of the various agencies supporting the war-fighter. While it might be somewhat difficult to assign a monetary value to this benefit, attempts could be made to either estimate or employ qualitative forecasting techniques in the absence of any current or historical data. Expert opinions or relative estimates based on comparable proxies could also be explored. Specifically, the value of DoD software or technological investment could be gauged and estimated on how the investments contributes to or enhances mission or operational readiness, value received in the form of reduction in casualties/injuries amongst U.S. service members, estimated net annual recurring savings, etc. In the case of weapons systems development, the expected benefits realized prior to the development of counter-measures by potential adversaries also need to be factored into the NPV computation. Regardless of the factors explored, it must be emphasized that this is a non-trivial task.

For the purposes of our study, we have chosen to employ relative NPV valuation as depicted in our example in the previous chapter. Under traditional NPV analysis, a project with a positive NPV increases the wealth of the firm, that is, the total value generated through the project's lifetime is superior to the cost of financing it and a higher

NPV is always preferable to a lower NPV, while a negative NPV represents an unacceptable investment. This traditional NPV formula can be tailored to specifically suit a software development project as outlined in [52] in which the NPV for a software development project is computed in terms of five high-level determinants and the equation outlined as follows:

$$\text{NPV} = \sum \frac{(C_t - M_t)}{(1+r)^t} - \text{I}$$

where

$$\sum\left(\frac{CF_t}{(1+r)^t}\right) \geq C$$

and

$$\text{M} = \sum\left(\frac{M_t}{(1+r)^t}\right)$$

where the standard assumption in [52] is (C –M) is always positive and

I is the (initial) *development cost*. It represents the total present value of all negative cash flows from the time the decision to invest is made to the time of the first major positive cash flow (time *T*) with development cost and development time are positively correlated

T is the (initial) *development time* or time to market. It is defined as the elapsed time between the commitment to invest in the project and the time of its first major positive cash flow. This period covers activities leading to the deployment of the end product.

C is the *asset value*. It represents the total value of the positive cash flows that the project is expected to generate during its lifetime, calculated at time *T*. Asset value mainly consists of the revenues from sales in the case of Foreign Military Sales, direct cost savings from using the end product and social benefits derived from acquisition of

the software product. *Direct product costs*, which represent the expenses incurred in proportion to the revenues generated, are deducted from asset value.

M is the *operation cost*. It is the total value of all negative cash flows of the operation phase, calculated at time *T*. This amount consists mainly of regular maintenance costs and pre-planned future investment outlays. Note that direct product costs (which are deducted from asset value) are not included here.

r is the rate at which all future cash flows are to be discounted (the *discount rate*).

Technically speaking we can reasonably conclude that NPV is a Real Options approach that assumes no flexibility in the decision making process. When we factor in flexibility in the form of Real Options created to hedge risk is factored into NPV computation, we arrive at the following formula

$$\text{NPV} = \sum \frac{(C_t - M_t)}{(1+r)^t} - \text{I} + \boldsymbol{\Omega}$$

where

$\boldsymbol{\Omega}$ is the *flexibility (option) premium*. It measures the contribution of the project's inherent strategic flexibility to its base NPV under uncertain conditions. For example, the ability to delay the commitment of certain resources, or to change the maintenance schedule.

The determination of the flexibility premium is dependent on several factors. Before we outline its determination, we need to revisit the issue of RO and how they relate to software related capital investments. First, we outline the approach to refining the value of our underlying asset.

C. REFINING ASSET VALUE USING MONTE CARLO SIMULATION

Traditional valuation methods are usually a static single point estimate which is insufficient for credible analysis. Consequently to better or accurately estimate the actual value of a particular project, a Monte Carlo simulation would need to be run [14].

However, before the Monte Carlo simulation can be run, the key input variables which are deemed highly uncertain in the future need to be identified. This could be accomplished by conducting a sensitivity analysis on the valuation model. Once the key input variables have been identified, the following steps as outlined in [14] are carried out during the Monte Carlo simulation.

1) A probability distribution is defined for each of the key inputs which underlie the cash flows. Initial estimates of the key input parameters, are obtained using either historical data or subject matter experts or a combination of both.

2) An outcome is drawn in each simulation from each distribution and the present value of the cash flows estimated based on the draws

3) The expected value of the project is determined from the mean of the distribution of the expected cash flows which is computed after several simulations and the standard deviation or the variance of the distribution of the cash flows can be used to value the project.

D. REAL OPTIONS

An option gives its owner or option holder the right, but not the obligation, to buy or sell an asset at a certain pre-negotiated price and the most obvious determinant of an option's value is its intrinsic value, or what it would be worth if it were immediately exercised [6]. This amount, defined as the option price less the exercise price, ultimately determines how much money the option holder makes. An option can still be valuable even if it has no intrinsic value because the possibility exists that the option could be profitably exercised in the future. The value of this possibility is an option's time value and is determined by three factors: volatility, length of time before an option expires, and risk-free-rate [6]. In other words, an option is essentially a precise contract between the software executive and the contractor (in the case of a government acquisition) to buy or sell a specific capability known as the options underlying instrument. In the case of software investment options from both a managerial and technical perspective, the underlying instrument is the software itself. The "contract" has an associated strike price

at which the contract or option may be "exercised" or acted on as well as an expiration date. In general, options can be grouped into two categories: "calls" and "puts", both of which could be bought and sold.

1. Buying Call and Put Options

In the event that a software executive buys a call option, the software executive would have the right but not the obligation to buy the underlying instrument at the strike price on or before the expiration date which would more than likely should be tied to the software development or delivery schedule in the case of software acquisitions. In the event that a put option is bought, the software executive would have the right but not the obligation to sell the underlying instrument on or before expiration of the option.

2. Selling Call and Put Options

On the other hand, if the software executive decides to sell a call option, the executive becomes obligated to sell the option or underlying interest at the strike price to the buyer should the buyer so decide and in the case that the software executive decides to sell a put option, the software executive becomes obligated to buy the underlying interest. In this case, since the software executive is the "writer" of the options, the executive would really has no control over whether or not the option is exercised.

Both cases involve a fee called the "premium", with the purchase price of the option being the premium in the event that an option is bought, and the selling price being the amount that is received. Once the appropriate options have been identified, they need to be valued accordingly.

E. IDENTIFYING STRATEGIC REAL OPTIONS

As discussed in the previous chapter, we implement our methodology which incorporates the Dempster-Shafer theory (DST) approach to volatility estimation which generally involves solving two related problems.

1) First, we must sort the uncertainties in the problem into a *priori* independent items of evidence.

2) Second, we must carry out the Dempster's rule computationally by combining the evidence obtained from all sources

We then model our results stochastically using the Risk Simulator and determine the correlation of variability. Once we have determined the volatility of our software investment effort, we can now identify possible options which we could used to manage the risk associated with the software investment effort. Generally speaking, over the course of the examination of a software investment effort, several independent options or combinations of options (compound options) can be identified to mitigate or hedge risk as it emerges. While options come in various forms such as American, European, Asian, and Bermudian option, the underlying concepts between each of these forms remain the same with the only differences being in the period or time of execution of the options. To applying options theory in this study, we explore various types of options for analyzing and addressing uncertainties associated with software-related capital investments which we broadly categorize into

1) Expand/growth options,

2) Wait/Deferment options and

3) Contract/Switch/Abandon options.

We depict these categories in Figure 32 below.

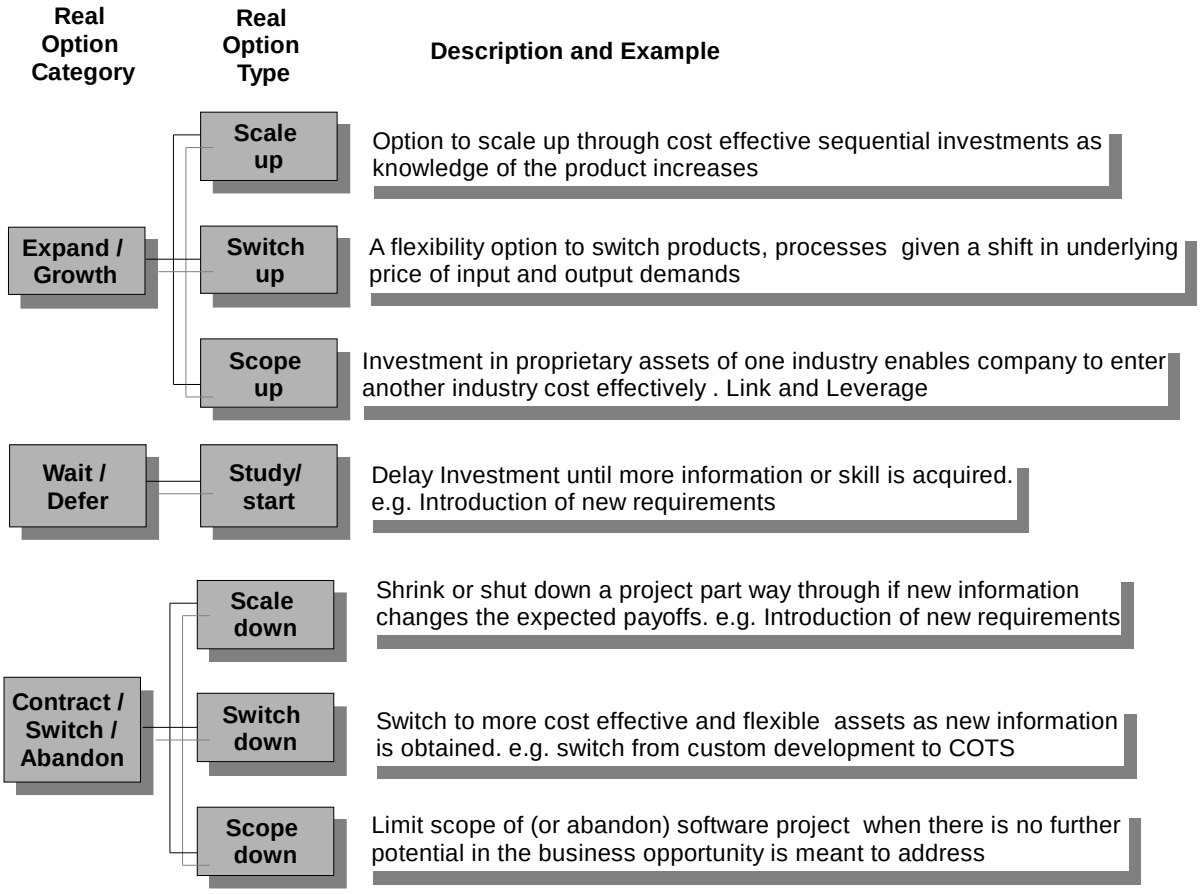


Figure 32. Sample Options to Address Software Investments (From: [6]).

In addition we also explore variants or combinations of options in the form of Sequential and Simultaneous compound options and explore a valuation methodology for the options.

F. PARTITIONING: DECOMPOSING THE SOFTWARE SOLUTION

To take advantage of the options identified above, we address the issue of software design. From a Real Options perspective, the decomposition of the software into components, modules or subsystems serves to introduce flexibility which the software executive or program manager could exploit and benefit from. Software design is a key activity aimed at conceiving how a software solution would solve a particular problem and it involves dealing with critical design decisions such as the decomposition/ partitioning of the software product under consideration into components to make its

acquisition and development more manageable. Given the impact which design decisions have on a software product, managerial concerns/uncertainties ranging from work-force capability to maturation/market availability of the proposed technology and technical concerns/uncertainties ranging from maintainability to reusability should be factored into design decisions. It must be noted that while we break this down into both managerial and technical concerns, these concerns can and do overlap. In the case of large-scale capital-intensive software-investment efforts, decomposing or partitioning the overall software product into well-defined, independent components based on functionality has the benefit of allowing each of the independent components to be developed by different vendors with the understanding that the ultimate end-goal would be to integrate each of the independent components to arrive at a fully functional software product that meets the customers needs. However it must be noted that design decisions are mostly made under uncertainty, and while these uncertainties might have been factored in the design during the high-level functional decomposition of the software, decomposition does not imply that uncertainty no longer exists, but merely reduced. This is because by virtue of software being an intangible artifact, it is impracticable to predict or have access to perfect information to key factors associated with the operation of the software when deployed at the time of the design. Hence uncertainties might still remain in a component or across components after decomposition stemming from both development and operational perspective concerns. Furthermore, while it is impossible to totally eliminate uncertainty, reduction strategies could however be employed such as the use of high fidelity modeling and simulation tools explicitly tailored to address uncertainty. By modeling and simulating the areas of concern within and across the components uncertainty, especially uncertainty arising from requirements uncertainty, operational uncertainty and uncertainty about the emergent attributes could be either reduced or resolved and ultimately help increase the degree of certainty associated with the decomposed components and the overall design of the software solution.

Given the unresolved issue of requirements uncertainty in our example in the previous chapter, the software executive needs to figure out the development approach to utilize for the software development effort. After considering several software

development options, the executive decides a stage-gate development approach based on concepts of both iterative development approaches and standard portfolio modeling techniques would be the most optimal development approach.

The approach would allow the software executive to partition/decompose the proposed software into different independent subsystems or components (e.g., the flight control subsystem and the avionics subsystem) and form a "portfolio" of the subsystems around the solution space (the software that needs to be developed). By doing this, the development of the subsystems can begin at the same time using iterative development techniques to guide each subsystem towards success while at the same time deferring subsystems that still face requirements uncertainty. The iterative concepts employed would facilitate shorter feedback cycles by cycling through the development phases, from gathering requirements to delivering functionality in a working release.

In essence this approach allows the software executive to recast the KC-X software program as a series of options to start and defer the development of a subsystem when requirements uncertainty is encountered, reallocate resources and then resume as uncertainty becomes resolved a strategy which supports the following two propositions [54].

- Some projects that look attractive on a full investment basis may become even more attractive if the project is partitioned/decomposed into components because we are able to reduce downside risk at the lowest possible level.

While we made the assumption that our software investment was attractive in our example, if we had made the assumption that it was unattractive, the second proposition below would also hold if we were to adopt the stage-gate-development approach

- Some projects that are unattractive on a full investment basis may be value creating if the firm can invest in stages.

Therefore by embracing a stage-gate development approach, the executive hopes to exploit the basic concepts of standard portfolio theory in order to reduce risk by

beginning the development of all the subsystems simultaneously with the hope that most of them would not be affected by requirements uncertainty and that in case some of them are affected by requirements uncertainly, the completion of those subsystem can be deferred until the uncertainty becomes resolved, an approach which is basically an "Option" on an "Option", with the options being the Option to Defer and the Option to Switch. Thus we frame our Real Options as follows:

1) Deferment Options: Option to defer the completion of a subsystem until more information becomes available

2) Switching Options: Option to switch resources to the development of other subsystems if the development of a subsystem they are working on has to be deferred until more information becomes available.

Given the relationship between our options in this case, we can call them a Compound Option where the value of each option is dependent upon whether the previous option was exercised or not and could also provide a method of stress testing or sensitivity testing of the final results by treating the options as alternatives within the option pricing model [14]. Moving along the lines of our example in the previous chapter, we can then develop a strategy tree showcasing the options. We assume in our example that the software executive partitioned the solution space of the proposed software for the KC-X program into three independent systems capable of functioning independent of the other systems: the Flight Control system, the Avionics system and the Navigational system, which we denote as component A, B and C respectively, and assume that the development of all of these component start simultaneously. In the event that the development of any component needs to be stopped due to requirements uncertainty, our proposed uncertainty elicitation model in Chapter III (Figure 9) could be applied to attempt to resolve the uncertainty. This approach is useful in that it might also help identify requirements that had not been considered but have been uncovered due to the partitioning of the proposed software. Consequently we can develop a strategy tree based on seven possible strategies. The strategy tree is given in Figure 33 on the next page.

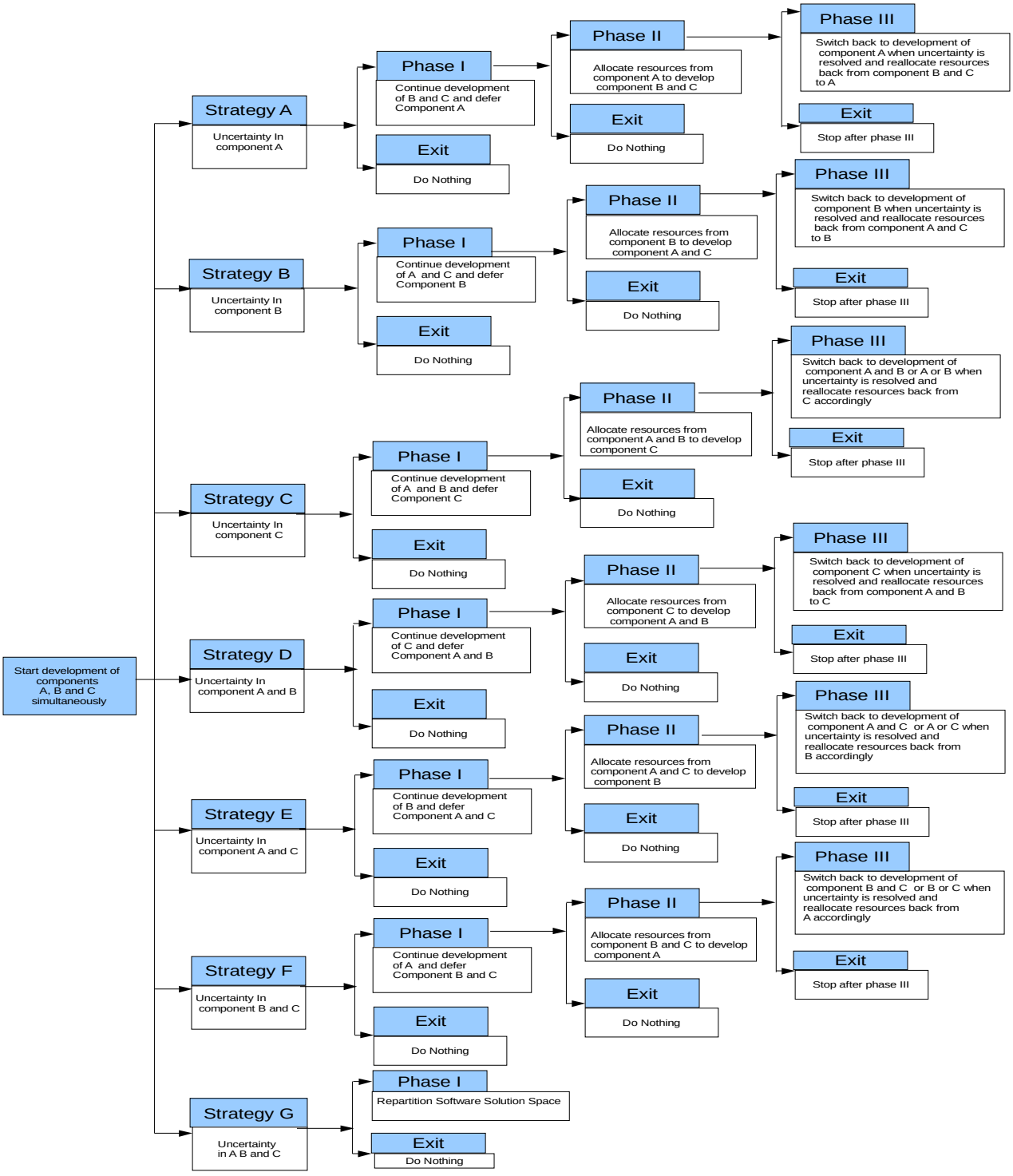


Figure 33. Strategy Tree depicting strategic pathways for the Software Executive. Real Options framework around the KC-X software program and shows the different strategies the software executive or the decision maker can adopt to hedge risk in order to mitigate cost and schedule overruns.

G. ANALYSIS OF STRATEGIC OPTIONS

Strategy A calls for the development of components B and C (Avionics system and the Navigational system) and deferring the development of component A (Flight Control system) using a "deferment option" until the uncertainty is resolved in component A. In this strategy, the resources that were initially allocated to component A are reallocated to components B and C using a "switching option" during the period of uncertainty resolution and are then allocated back to component A using a "switching option" once the uncertainty in A is resolved .

Strategy B calls for the development of components A and C (Flight Control system and the Navigational system) and deferring the development of component B (Avionics system) using a "deferment option" until the uncertainty is resolved in component B. In this strategy, the resources that were initially allocated to component B are reallocated to components A and C using a "switching option" during the period of uncertainty resolution and are then allocated back to component B using a "switching option" once the uncertainty in B is resolved .

Strategy C calls for the development of components A and B (Flight Control system and the Avionics system) and deferring the development of component C (Navigational system) using a "deferment option" until the uncertainty is resolved in component C. In this strategy, the resources that were initially allocated to component C are reallocated to components A and B using a "switching option" during the period of uncertainty resolution and are then allocated back to component C using a "switching option" once the uncertainty in C is resolved .

Strategy D calls for the development of component C (Navigational system) and deferring the development of components A and B (Flight Control system and the Avionics system) using a "deferment option" until the uncertainty is resolved in components A and B. In this strategy, the resources that were initially allocated to components A and B are reallocated to component C using a "switching option" during the period of uncertainty resolution and are then allocated back to components A and/or B using a "switching option" once the uncertainty in A, B, or A and B is resolved .

Strategy E calls for the development of component B (Avionics system) and deferring the development of components A and C (Flight Control system and Navigational system) using a "deferment option" until the uncertainty is resolved in components A and C. In this strategy, the resources that were initially allocated to components A and C are reallocated to component B using a "switching option" during the period of uncertainty resolution and are then allocated back to components A and/or C using a "switching option" once the uncertainty in A, C, or A and C is resolved .

Strategy F calls for the development of component A (Flight Control system) and deferring the development of components B and C (Avionics system and Navigational system) using a "deferment option" until the uncertainty is resolved in components B and C. In this strategy, the resources that were initially allocated to component B and C are reallocated to component A using a "switching option" during the period of uncertainty resolution and are then allocated back to components B and/or C using a "switching option" once the uncertainty in B, C, or B and C is resolved .

Strategy G calls for repartitioning the software solution space along alternative partitions to the current set of partitions (Flight Control system, Avionics system and Navigational system) to see if there is sufficient information in any single subset to allocate resources and proceed with development of that subset while exercising a "deferment option" to wait until more information becomes available about the other subsets.

Now that we have identified the appropriate options to hedge risk, the question that arises is how much does the software executive pay for the option? The following section addresses these concerns.

H. MECHANICS OF OPTIONS VALUATION: OPTIONS PREMIUM

Valuation plays a central part in any acquisition analysis. Options are usually valued based on the likelihood of the execution of the options. In general, there are three approaches to valuation, all of which have different outcomes depending upon which approach is used. These valuation approaches are discounted cash flow valuation/relative valuation, which are estimates based on the value of an asset by looking at the pricing of

'comparable' assets relative to a common variable such as earnings, cash flows, book value or sales, and lastly contingent claim valuation.

Since an option represents the *right* but *not the obligation* to make an investment, the payoff scheme to the option-holder is asymmetric and the options are only exercised if they have a positive value and are left unexercised if worthless [6]. While there are several methods for computing and valuing real options such as employing the use of closed-form models, partial differential equations, and the binomial approach. The binomial approach utilizes risk neutral probabilities elicits a great appeal due to its simplicity, ease of use and the ability to solve all forms of options (e.g., European, American).

When a software executive buys a call option, the executive expects that the benefits derived from the option would translate into cost savings when the option is exercised in the future because at the time at which the option is bought, the software executive is fully aware that the money spent on purchasing the option might not be recovered if the option is not exercised or transferred to another similar government investment effort since technically speaking the option cannot be sold for profit as a last resort, since the government is not in the business of making profit, but rather saving tax payers money. Therefore if the option is not exercised and is not transferred to another government entity that could benefit from it, the cost of the premium must be factored in the computation of realized benefits by subtracting the premium (loss) from the projected realized benefits. On the other hand, the seller of the option has a net credit [55] because of the premium that was received for the option, since they would always keep the premium even if the option is never exercised. From a sellers perspective, who more often than not might be a private or commercial organization, this call option translates into a cost increase thereby impacting their profits if the option is exercised. Hence, they receive the option premium as a compensation for their loss.

Several factors affect the value of an option. These factors could range from the maturity of the option to the volatility of the underlying asset. Since premiums are not fixed and constantly change, the fluctuations reflect the compromise between what the buyers are willing to pay for the option and what the seller is willing to receive for the

option with the buyers and sellers being the software executive and contractor respectively or vice-versa as the situation might warrant at which point an agreement is reached on the price of the transaction.

Regardless, the options premium has two main components: intrinsic value and time value, both of which contribute to the valuation of the underlying software investment. In the case of call options, the call option is said to be "in-the-money" if the strike price of the call option is less than the current market price, meaning it becomes optimal for the option holder to exercise the option and benefit from factors such as realized cost savings in the form of inflation adjusted labor rates between the time the option was bought and the time the option is exercised. Likewise, in the case of a put option, if the strike price of the put option is greater than the current market price of the underlying interest, the put option is also said to be "in-the-money" [55]. If the events above do not hold, i.e. strike price is less than current market price for the call option or strike price is greater than current market price for the put option, the option is said to be "out-of-the-money" and should the strike price equal the current market rates, the option is say to be "at-the-money". The amount by which a call or put option is in-the-money at any given moment is called its intrinsic value and any amount by which an options total premium exceeds the intrinsic value is the time value portion of the premium [55]. It is the time value portion of the options premium which is affected by factors such as the volatility of the software investment.

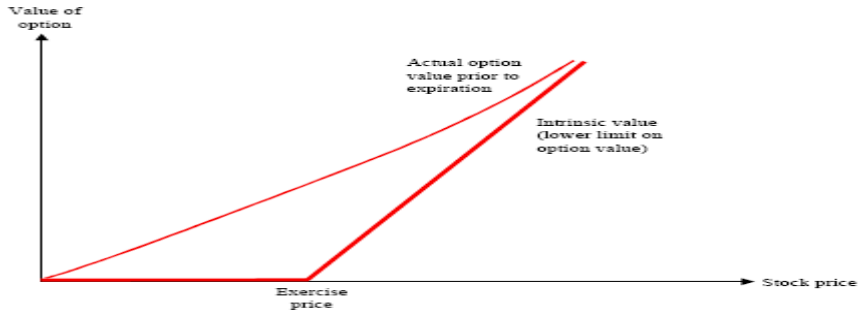


Figure 34. Relationship between Stock Price and Call Option (From[6]).

Thus in the case that an option is out-of-the-money, the option has no intrinsic value and the time value is the total option premium. As can be seen from the figure

above depicting the relationship between a call option and the stock (option) price, a call option's intrinsic value increases as the stock price increases, but never falls below zero. This is analogous to a software investment. Since a call option's time value increases as stock price volatility increases, so would a call option's time value increase as the volatility associated with software investments increase *because the* longer an option holder has before expiration, the higher the probability that the value of the overall software investment will end up above the exercise price, thereby making options with longer "lives" more valuable than similar options with short lives.

Furthermore the more volatile a stock, the higher the chance that the option will be profitable as can be seen from Figure 35 below depicting two scenarios, with one showing an option on a low-volatility stock with a narrow price distribution clustering around the exercise price and the second showing a similar option on a high-volatility stock with a wide price distribution. All things equal, then, higher stock price volatility translates into a higher time value [6].

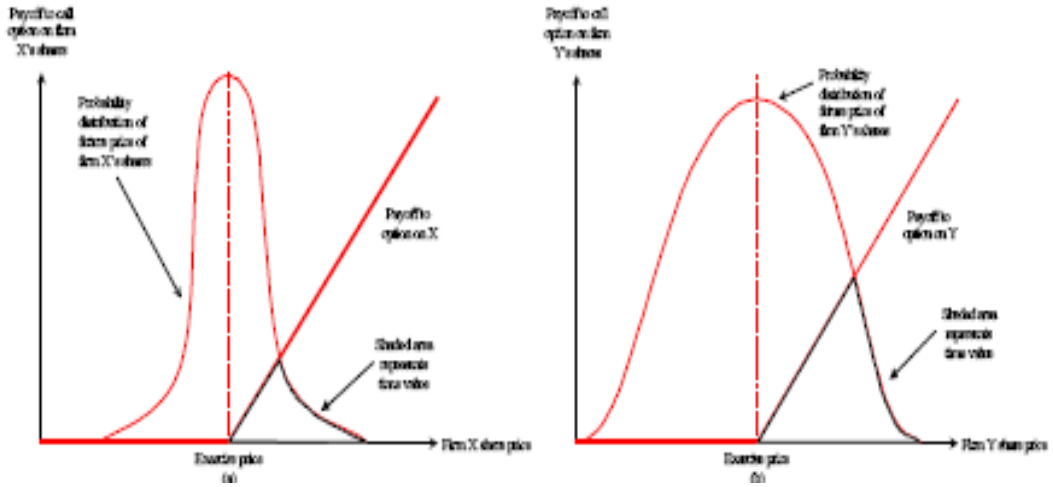


Figure 35. Scenario A: Low stock price volatility Scenario B: High stock price volatility (From: [6]).

Given the way an option is valued, we summarize the key points in Table 8 on the next page.

	Call Option	Put Option
In-the-money	Strike price of the call option is less than the current market price	Strike price of the put option is greater than the current market price
At-the-money	Strike price of the call option is the same as the current market price	Strike price of the put option is the same as the current market price
Out-of-the-money	Strike price of the call option is greater than the current market price	Strike price of the put option is lower than the current market price

Table 8. **Value of an Option.**

I. VALUATION COMPUTATIONAL METHODS

As mentioned earlier, to determine the value of our option created to hedge the risk presented by the uncertainties surrounding the software investment process requires the computation of several factors. The basic inputs are the present value of the underlying asset (S), present value of the implementation cost of the option (X), volatility (cv), time to expiration in years (T), risk free rate or rate of return in a riskless asset (r) and are summarized in Table 9 on the next page.

Symbol	Real Option on Software Acquisitions Project	Description
S	Value of underlying Asset (Asset Price)	Current Value of expected cash flows. (Expected benefits realized from investing in the software effort (NPV))
K	Exercise Price / Strike Price	Price at which the created option would be realized (Investment Cost of investing in options, which is an estimation of the likely costs of accommodating changes)
T	Time-to-expiration	The useful life of the option. (Time until the opportunity disappears or the maturity date of the option contract)
r	Risk-free Interest Rate	Risk free interest rate relative to budget and schedule (Interest rate on US Treasury bonds)
cv	Volatility	Uncertainty of the project value and fluctuations in the value of the requirements over a specified period of time (Volatility in requirements, cost estimation and schedule estimation based on DST of Evidence)

Table 9. **Factors Affecting Value of an Option.**

Although the Black-Scholes Formula is a more popular option pricing model, its underlying principle is based on European Call options, a key limitation since it cannot be used to price American Call options which allow the option holder to exercise the option any time up to the maturation of the option, hence the need for an alternative pricing method in the form of the Binomial Lattice.

The binomial lattice approach could be based either on market replicating portfolios or risk neutral probabilities however the latter is much more easier to perform [14], since the risk neutral probability approach is simply a type of discrete simulation of

the cone of uncertainty and uses a "discrete-time" model of the varying price over time of the underlying financial instrument.

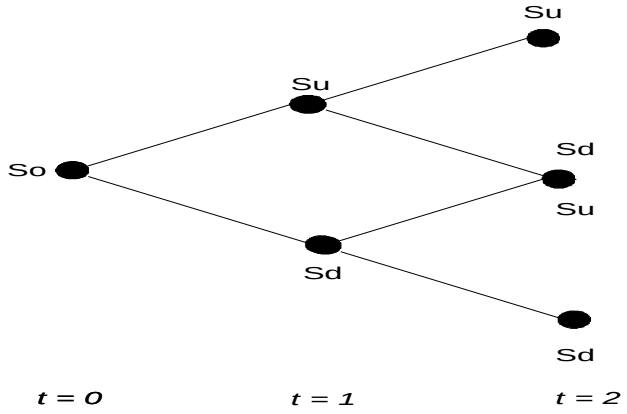


Figure 36. Binomial Lattice.
So is the current asset value; the value moves up to *Su* with probability *p* and down to *Sd* with probability 1-p in any time period.

Option valuation is then computed via application of the risk neutrality assumption over the life of the option, as the price of the underlying instrument evolves [56]. A binomial lattice could be graphically represented in its simplest form as shown in the Figure 36 above. It requires two basic computation: the *up* and *down* factor denoted by *Su* and *Sd* and the risk neutral probability measure with the *up* factor being the exponential function of the cash flow returns volatility multiplied by the square root of time steps (∂t) (the time scale between steps) [14]. The risk neutral probability is simply computed as the ratio of the exponential function of the difference between the risk-free-rate and dividend multiplied by the time steps, less the down factor, to the difference between the up and down factors [14].

The binomial lattice approach to option valuation can be described as a three step process [56].

1) Asset Value tree generation

2) Calculation of option value at each final node

3) Progressive calculation of option value at each earlier node with the value at the first node is being the value of the option.

A pre-requisite for its application is the existence of at least two lattices with one lattice representing the underlying asset and the other representing the options valuation. It could be used to visually describe price movements over time, where the asset value can move to one of two possible prices with associated probabilities and the precision of the results is correlated to the number of steps that run in the simulation. Given the factors that are involved in computing the value of an option using the binomial lattice approach, the following formulas can be deduced:

$$Su = e^{\ell \sqrt{\partial t}} \quad \text{.....(20)}$$

$$p = \frac{e^{r(\partial t)} - Sd}{Su - Sd} \quad \text{.....(21)}$$

$$Sd = e^{-\ell \sqrt{\partial t}} \quad \text{.....(22)}$$

Therefore moving along the line of our example from the previous chapter, we can compute these factors based on the volatility of our software project of 0.0012, with a step size of 0.2 years (one year expiration divided by five steps):

$$Su = e^{\mathbf{0.0012}\sqrt{0.2}} = 1.000536800340363$$

$$Sd = e^{\mathbf{-0.0012}\sqrt{0.2}} = 0.999463487659644$$

$$p = \frac{e^{0.03(0.2)} - 0.999463487659644}{1.000536800340363 - 0.999463487659644} = 6.1068396117618535$$

Subject to the following underlying assumptions:

- The present value of the underlying asset is $305 million
- Implementation Cost is $55 million

- Net Present Value is $250 million
- The risk free rate is the 3.0
- Volatility of our project cv = is 0.0012
- Duration is 12 Months (1 year)

(We use coefficients of variation as a proxy of standard deviation because it is a relative measure of the standard deviation expressed as a percentage of the mean.)

For the purposes of our example, we run the Super Lattice Solver 3.0 provided by Real Options Valuations Inc to compute the value of our option. The first step in the Super Lattice Solver 3.0 involves the creation of the lattice evolution of the underlying asset. What the model essentially does is multiply the up and down factors by the present value of the underlying asset at time zero to create the binomial lattice. Therefore we multiply the underlying value of the asset ($305 million) by 1.000536800340363 and 0.999463487659644 to get the up and down factors respectively of the lattice. This is depicted in Figure 37.

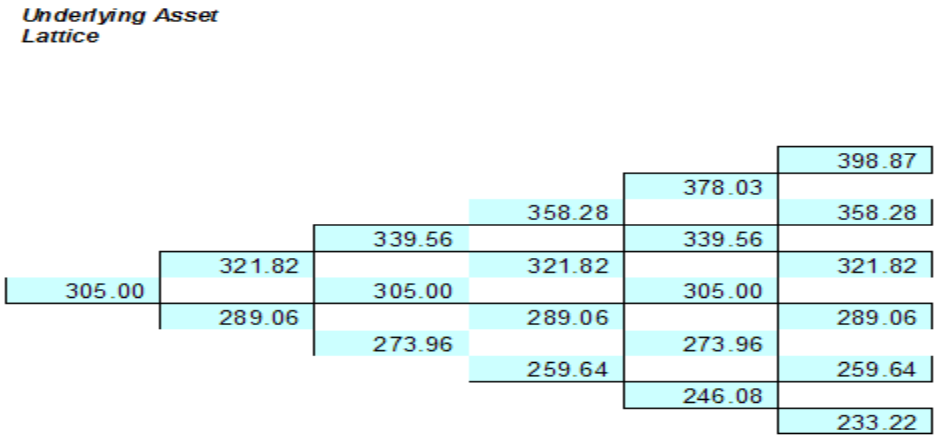


Figure 37. Underlying Asset Lattice

The second step in the Super Lattice Solver 3.0 is to create the option valuation lattice (Figure 38) using the values computed in the lattice of the underlying asset to generate the value of the option.

This requires two steps, the valuation of the terminal node and the valuation of the intermediate nodes using a process called backward induction [14].

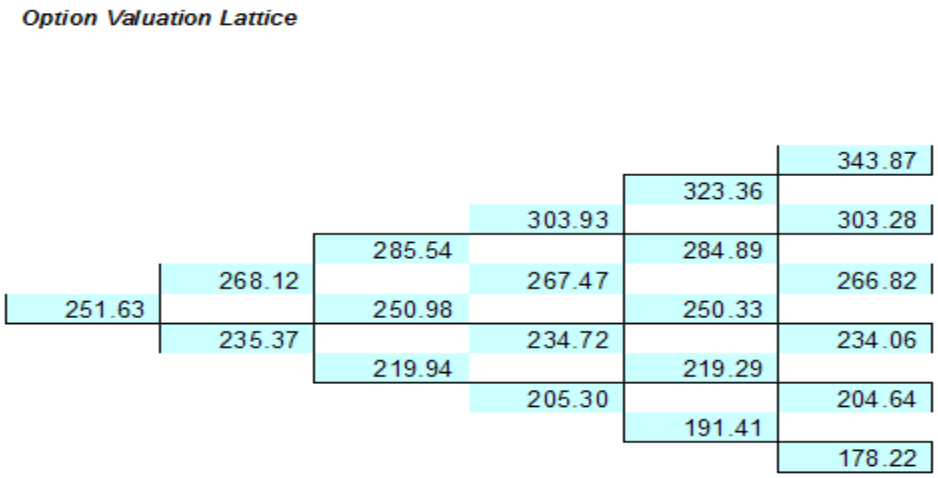


Figure 38. Option Valuation Lattice

The final node in the lattice represents the expiration of the option and the option value is simply the intrinsic or exercise value and represents the fair price of the Real Options with all the nodes leading up to the final node representing the value at a particular point in time given the evolution in the value of the software to that point. The terminal values in our lattice are the computed values that occur at maturity, while the intermediate values in the lattice are the computations that occurs at all periods leading up to maturity and. All these values are computed using backward induction.

This value basically represents the value of the option if it were to be held as opposed to exercised at that point. Now it is worth noting the differences between the different types of call options. With a European option, there is no option of early exercise, and the binomial value applies at all nodes, for an American option, since the option may either be held or exercised prior to expiry, the value at each node is: Max (Binomial Value, Exercise Value), and for a Bermudan option, the value at nodes where early exercise is allowed is: Max (Binomial Value, Exercise Value); at nodes where early exercise is not allowed, only the binomial value applies.

Therefore given the values we obtained above, the binomial option valuation comes out to be $251.63 million. Since the NPV of the investment is $250 million, our total option value (premium) in this case is $1.63 million.

J. REAL OPTIONS ANALYSIS USING MONTE CARLO SIMULATION

The final step in our methodology is to conduct a RO analysis in order to value the payoffs associated with our RO. This analysis involves the modeling of the value of the underlying asset which includes the option premium. We follow the steps outlined above. However in this case the expected value of the software investment effort is determined from the mean of the value which is computed after several simulations and the standard deviation or the variance of the value can be used to value options on the project.

K. REALIZED REAL OPTIONS FRAMEWORK

All the work we have conducted this far has been aimed at developing the foundations on which we could now use to establish a RO Framework (Figure 39) for supporting the strategic decision making associated with software related capital investments.

1) We begin this process by computing the base case present value of the software investment effort using the NPV formula we discussed earlier in this chapter once the operational needs have been identified. Since we are computing NPV at time = 0, we omit the option premium factor **Ω** .Thus NPV computation equates as follows:

$$NPV = \sum \frac{(C_t - M_t)}{(1+r)^t} - I$$

(we can choose to refine our asset value using a Monte Carlo simulation)

2) We then proceed to identify the uncertainties associated with the software investment effort by utilizing our proposed "2**T**" approach of uncertainty

elicitation from Chapter 3 and feeding these uncertainties into a risk model. The steps associated with the "2**T**" approach are as follows:

a) Identify pragmatic uncertainties and associated input drivers.

b) Identify Heisenberg-type and Gödel-like uncertainties and associated input drivers

c) Quantify and feed the input drivers of both types of uncertainties into the Risk Simulator provided by Real Options Valuations Inc.

3) We then proceed to measure the risks posed by these uncertainties in the form of volatility by utilizing the Real Options Inc risk simulator in order to model the impact of the risks posed by the uncertainties.

4) To "fit" our data, we attempt to identify sources of information that we could use to better understand these uncertainties and the risks they present. Specifically we look at both historical data and utilize the Delphi Method. In the case where historical data is not available, we rely on the Delphi Method or any other applicable proxies.

5) Once we have gathered a better understanding of the uncertainties based on the techniques employed in Step 4, we proceed to quantify these uncertainties based on the risk they pose in order to estimate the volatility of the software investment program.

6) We then solicit additional evidence from different independent sources and employ the Dempster-Shafer Theory of Evidence utilizing belief assignments and the Dempster's combination rule to combine the evidence we obtained from our different sources.

7) We utilize the evidence we obtained using Dempster-Shafer Theory to modify or refine our initial probability estimates obtained from historical data and obtain a refined volatility estimate of our investment effort.

8) After completing all these steps, we are now ready to conduct our analysis.

9) We begin our analysis of the risks by proceeding to formulate the appropriate Real Options to hedge against the risks and develop a strategy tree depicting the strategic pathways.

10) We value the options created in Step 8 using the Binomial Lattice approach and determine our option premium.

11) We add this value of the option premium to our investment costs to compute the new asset value using

$$NPV = \sum \frac{(C_t - M_t)}{(1+r)^t} - I + \Omega$$

We conduct a RO analysis by using random variations in a Monte Carlo simulation to model different scenarios as applicable

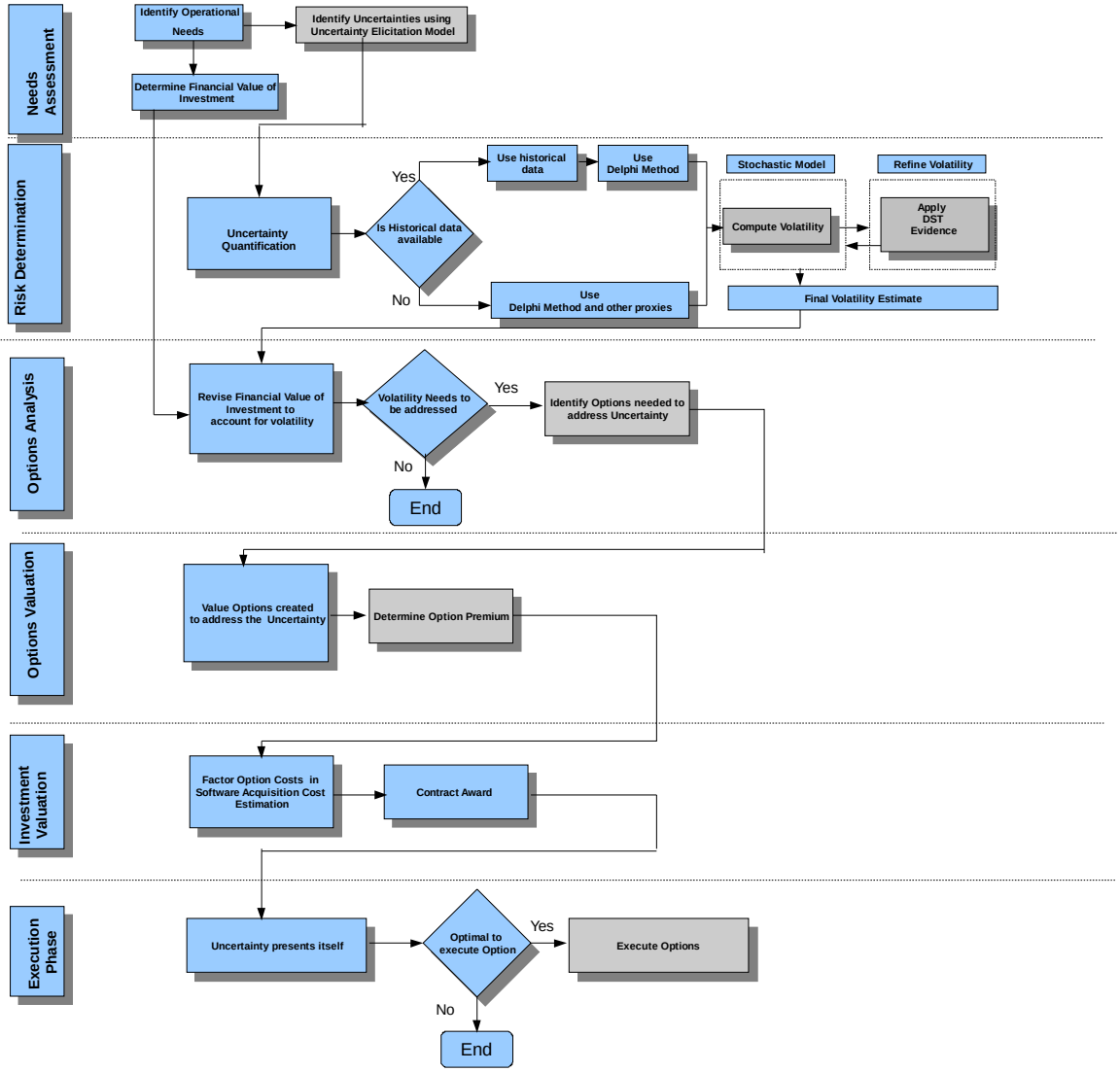


Figure 39. Realized Real Options Framework

Now that we have established our real options framework we attempt to validate our approach to determine its fit to software-related capital investments.

VI. VALIDATING THE REAL OPTIONS FRAMEWORK

Cost is the value of the foregone option....Unknown

In an attempt to validate our proposed approach, we decided to apply the framework to the software component (Future Combat Systems Network) of U.S. Army Future Combat System (FCS). The decision to select this case study as a validation mechanism was based on the recent nature of the project, the high-risks associated with software development due to the advanced technologies involved, the challenge of networking all of the FCS subsystems together so that FCS-equipped units can function as intended and the associated outcome had a Real Options approach been applied. The associated key contributions of this chapter are as follows:

A. KEY CONTRIBUTIONS

What we show in this chapter using our Real Options methodology is how the decision maker can develop a series of "Call Options" which can be incorporated into the Future Combat Systems Network acquisition contract that would give the decision-maker the right but not the obligation to exercise the options when the need arises. Essentially what the decision maker would be doing by embracing our methodology is identifying and paying for risk up-front at today's price and taking advantage of cost savings dependent on if the Call Option is "in-the-money" on the upside that the risks which the Real Options were created to address presents itself, otherwise the decision maker forfeits the option. Specifically these cost saving would be realized on "in-the-money" Call Options in the form of savings on differences in prevailing market labor rates actor in software development), since at this point in time, the exercise price would be less than what was paid for the Call Option. We start by providing an overview of the Future Combat Systems program, the challenges facing the Future Combat Systems Network and the application of our methodology within the constraints of two assumptions we make during the application of methodology.

B. FUTURE COMBAT SYSTEM (FCS) OVERVIEW

The concept of the U.S. Army Future Combat System (FCS) was first introduced in October 1999, by then Chief of Staff of the Army (CSA) General Eric Shinseki in an attempt to convert all of the Army's divisions (called Legacy Forces) into new organizations called the Objective Force with the intent of making the Army lighter, more modular, and most importantly more deployable [57]. These strategic considerations, were in line with enhancing U.S. Military operations and allowing US dominance in future conflicts across a full spectrum of threats in a global context. As part of this transformation, the Army adopted the Future Combat System (FCS) as a major acquisition program to equip the Objective Force [57]. Figure 40 below offers a visual depiction of the FCS. This transformation, due to its complexity and uncertainty, was scheduled to take place over the course of three decades, with the first FCS-equipped objective force unit reportedly becoming operational in 2011 and the entire force transformed by 2032.

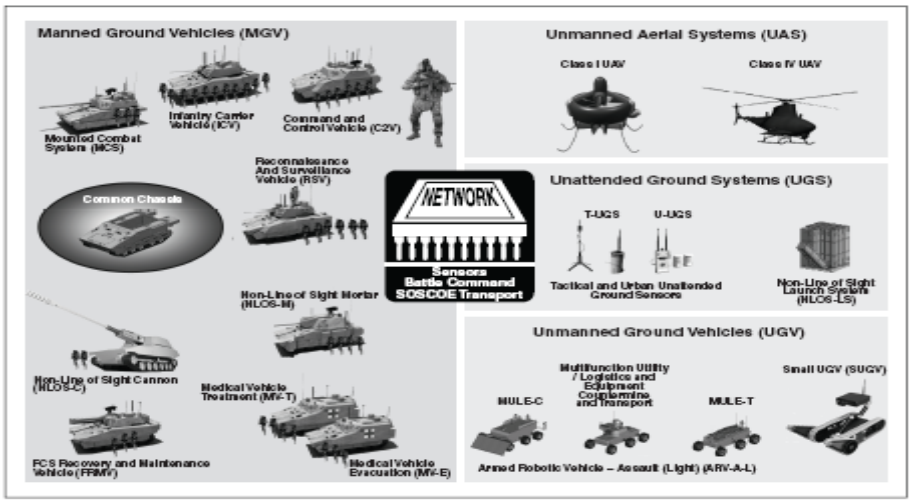


Figure 40. FCS Core Systems (From: [58]). Core systems of the Future Combat Systems depicting the software component (Future Combat Systems Network) as the "heart" of the overall program.

The vision for the FCS was such that it would consist of four major components: Manned Ground Vehicles (consisting of 8 platforms), Unmanned Systems (which includes the Unmanned Aerial Vehicles (UAV), Unattended Systems, and Unmanned Ground Vehicles (UGV)), FCS Network (the software platform that provides the

communication and automation that creates the battle command environment), and Soldiers (who are ultimately empowered with the use of robotics and technological advantages).

The Future Combat System (FCS) when deployed is intended to replace such current systems as the M-1 Abrams tank and the M-2 Bradley infantry fighting vehicle by tying together what was initially expected to be 18 manned and unmanned systems via an extensive communications and information network. However, over the last few years, this program has been plagued by program development issues with some technologies advancing quicker than anticipated, others progressing along predicted lines, while still others experiencing significant delays, and skyrocketing costs, with initial estimates of the FCS program jumping first from $91.4 billion in 2003 to $127 billion, and now to over $160 billion [63]. Coupled with the skyrocketing costs, comes the issue of requirements increase with source lines of code (SLOC) almost doubling from an estimate 33.7 million SLOC initially, to 64 million SLOC and more recently tripling to 95 million SLOC. While those costs have grown and grown, the list of equipment the FCS is actually supposed to deliver has been getting shorter, with the number of systems dropping from 18 to 14. Table 10 shows FCS code measured in source lines of code (SLOC) from inception until 2007.

	Original Estimate (May 2003)	Estimate as of Jan 2006	Estimate as of August 2007	Percentage Increase
SLOC	33.7 million	63.8 million	95.1 million	182

Table 10. **FCS Software Growth Estimates.**

C. SOFTWARE COMPONENT: FCS NETWORK

Currently, there are many radio and computer systems using various different software which makes it difficult to communicate during both peacetime and combat operations. The intent of the FCS Network is to overcome this issue by enabling leaders at all levels to see first, understand first, act first, and finish decisively by connecting FCS

platforms to the Soldier at every echelon, from Brigade to Squad while at the same time giving the ability to integrate communications with other Department of Defense agencies and US allies [58]. The software component of the FCS program is intended to provide a communication platform for soldiers to communicate through a wireless network in near real-time with multiple other "assets" such as hovering drones, remotely control robots to defuse bombs, firing laser-guided missiles at enemies on the move, conducting a video teleconference in a tank rumbling about 40 mph in the haze of battle utilizing the same network at the same time.

The design is based on ensuring the availability of the communication network at all times so that even if a soldier should lose their connection, the software would automatically "correct itself," retrieving the information within seconds without rebooting by finding and utilizing an efficient mathematical algorithm to reconnect the soldier. The FCS Network has its own operating system known as the System-of-Systems Common Operating Environment with over 100 interfaces or software connections to systems outside FCS. The FCS Network is made up of 5 layers: Sensor/Platform Layer, Application Layer, Services Layer, Transport Layer, and Standards Layer. These layers provide diversity in waveform, frequency and environment to ensure there are multiple paths to transport the data. Each network is tailored to support the specific needs of the end users. Depending upon the communication configuration most users will be provided with multiple layers of access. The FCS software is currently being developed in four discrete stages, or blocks, with each block adding incremental functionality spanning approximately eight functional areas (command and control, simulation, logistics, training, manned ground vehicles, unmanned aerial vehicles, unmanned ground vehicles, and warfighting systems) [64].

D. BENEFITS OF FCS

Besides the operational benefits in terms of tactical advantages and increased situational awareness, cost savings are expected to be realized from the FCS program because the new combat vehicles in the FCS are designed to share common hulls and 80 percent common parts. This would manifest into fewer spare parts and streamlined

training and functions of mechanics; instead of needing specialists for a mixed fleet. Furthermore, the reduced weight of the combat vehicles also translate into reduced airlift requirements in terms of capacity (Air Force Cargo aircrafts would be able to carry more combat vehicles due to reduced weight) in the case of operational deployments, which also translate to reduce wear and tear on Air Force aircraft and fuel requirements. While the FCS Network is just one component of the overall program, it is the underlying platform on which the success of the FCS program depends, hence the need to apply a disciplined approach to its acquisition and development. Before identifying the challenges facing the FCS Network from both a managerial and technical perspective, we first we state our key assumptions.

E. ASSUMPTIONS

Due to data limitations at the level of granularity at which we would have hoped we made the following two assumptions which we justify later on in this chapter at the applicable situation.

1) We estimated the future benefits of the Future Combat Systems Network (Asset Value) under traditional NPV and assumed it was positive (i.e., benefits outweigh costs) and, further later on in the study, transposed the costs of the overall FCS program by making it the cost of the FCS Network (software component) since detailed breakdown of cost data was not available for comparables.

2) We assume the independent assessments provided by the Cost Analysis Improvement Group (CAIG) and the Institute of Defense Analysis (IDA) include belief assignments. In other words, we assume that an executive, during the decision making process, is provided not only with a raw set of the risk factors of requirements creep, integration risk, performance risk, but with additional measures: belief in the estimation of the risk factors and certainty of the estimation.

F. TECHNICAL CHALLENGES

To date, the software development component of the FCS Network represents the largest software development effort in DoD history with a current projection of 95.1 million SLOC providing 95% of the FCS functionality. From the onset, its development has been plagued by technology maturation issues since most of the capabilities were conceptual in nature and needed to be demonstrated so that the associated software requirements developed. Furthermore, while the concepts of the FCS does have its operational merits, it has been compounded by requirements challenges in which meeting some requirements has the unintended consequence of working against another requirement [63].

It is also currently envisioned that the 95.1 million lines of software code of the FCS program would be based on three categories (Figure 41) new code, reused code, and commercial-off-the-shelf code (COTS). Given the huge spike in the Source Lines of Code (SLOC) estimates from 33.7 million in 2003 to today's estimate of 95.1 million, there is a high probability that the number of lines of code required for the program would increase as the Army learns more about the technology and its design concepts.

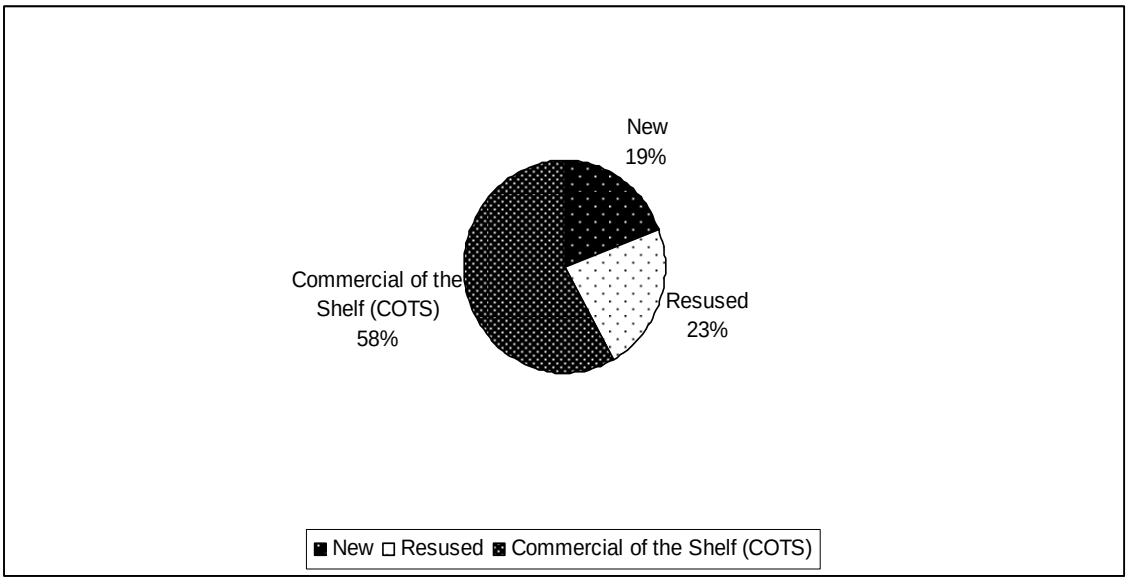


Figure 41. FCS Projected Software Lines of Code (in thousands) (From: [58]).

Furthermore, while new software code represents the greatest challenge of all the three categories of code due to the fact that it has to be written from scratch, it is also highly speculative that the amount of software code that could either be reused or adapted (COTS) is inaccurate. Hence these estimates which might be somewhat conservative and could easily translate to greater time and effort to develop software than is currently planned therefore resulting in cost overruns.

Currently the FCS warfighter operational requirements stand at 544 which translates to approximately 11,697 system-of-systems requirements. Of the system-of-system requirements, 289 have "to be determined" items, 819 have open issues to be resolved individual system level requirements, resulting in 51,944 system level requirements [58]. The system level requirements provide the specificity needed for the contractors to fully develop detailed designs for their individual systems. Figure 42 below illustrates how the FCS requirements are translated from the warfighter to the individual systems.

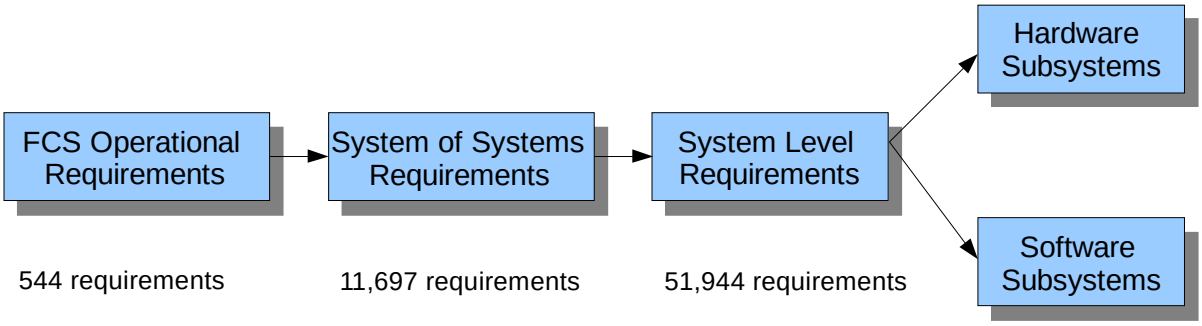


Figure 42. Flow of FCS Overarching Requirements to System-Level Requirements (From: [58]).

The development strategy of the FCS Network is based on the evolutionary approach as can be seen in Figure 43 in which the practice is to overlap builds more than the traditional spiral model does, by beginning the requirements phase of the next build before the testing phase is completed on the previous build so that the next build is ready for design by the time the former build has completed testing.

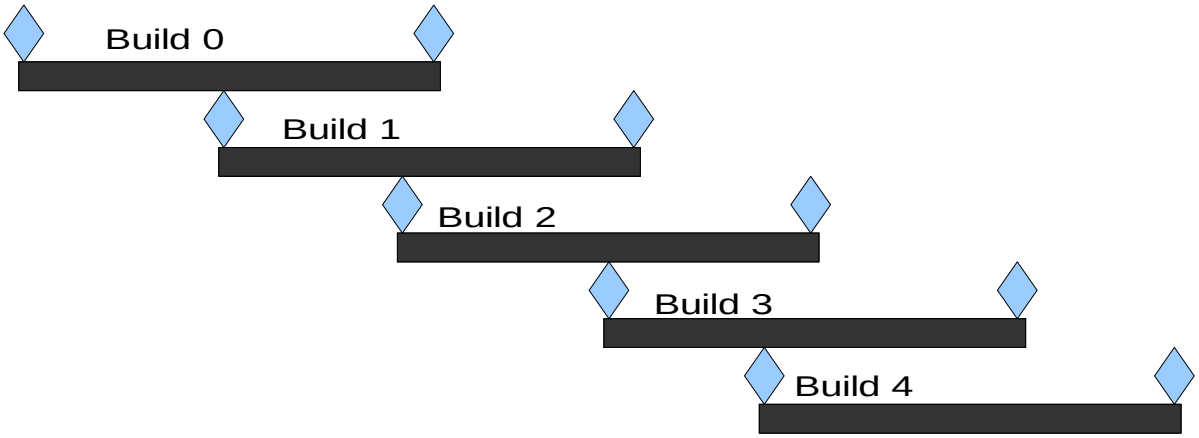


Figure 43. FCS Spiral Development Strategy and Software Life Cycle Reviews (From: [64]).

However, in the midst of evolving requirements, this approach has caused developers to interpret and implement changes in requirements well into the design and code phases, compromising the amount of time allotted for testing. This is not to say that the requirements should have been defined more quickly, rather, the requirements instability is amplified by the relative immaturity of the program, coupled with its aggressive pace, the pronounced overlap of the FCS builds and the cascading effect on software development [64].

G. MANAGEMENT CHALLENGES

Program costs and schedule concerns driven by technical risks and uncertainty form the basis of the managerial challenges. From an original modest cost of $92 billion for the whole program for all 18 systems, the costs of the FCS has skyrocketed to over $163.7 billion dollars for only 14 of the 18 systems according to Army estimates [64]. Based on the current development and delivery schedule, about 10% of the software which is considered to be the most difficult part of the development effort by program officials is planned to be delivered and tested after the early 2013 production decision which would limit the knowledge available to decision makers at that point [64].

Block	Percentage of total Software completed	Delivery Date
0	5	Sep-05
1	30	Dec-07
2	61	May-10
3	90	Oct-11
4	100	Oct-13

Table 11. **FCS Software Blocks, Percentage of Completion, and Delivery Dates (From: [64]).**

Currently, the greatest risks faced by the program, besides the unknown total costs of the system, stems from concerns that the overall FCS would not deliver the required capability within estimated resources [61]. Compounding these problems is the lack of a top-level requirements and architecture definition which in effect leaves system-level requirements incomplete until the preliminary design review in 2009, which further affects the accuracy of projected SLOC. Therefore given all the challenges facing the software component of the FCS program (FCS Network), we summarize our key finding of sources of uncertainty of the FCS Network based on the current challenges.

H. MANAGERIAL UNCERTAINTIES

As mentioned earlier and depicted in our "2**T**" model, constraints of people, time, functionality, budget, and resources form the basis of the uncertainties faced the program manager. Therefore using this paradigm, based on the information we have gathered thus far, we can frame managerial uncertainties along two lines: Estimation (size and costs) Uncertainty and Schedule Uncertainty. We discuss this further.

1. Estimation Uncertainty

Generally speaking, poor size estimation is one of the main reasons that major software-intensive acquisition programs fail. The overall costs of the FCS Network program continue to be plagued by estimation difficulties associated with changing requirements, immature architecture, with the U.S. Army, the Institute of Defense Analysis and DoD's Cost Analysis Improvement Group (CAIG) reporting independent and different cost estimates of $163.7 billion, $166.7 and a range of $203 - $234 billion

respectively. The difficulties associated with accurate software estimating is an indication that complexity increases as the design is better understood, resulting in the increases in the level of effort and possible increases in the development schedule, and ultimately resulting in increased costs. The general consensus as reported in reports authored by the Government Accountability Office (GAO) is that the Army's estimates are limited by the low level of knowledge in the FCS program today since the Army does not have a base of mature technologies and well-defined system-level requirements. Therefore, the Army resorts to making significant assumptions about how knowledge will develop [58]. Table 12 below highlights the trend of cost growth from the inception of the program.

	Army Original Estimate	Current Estimate	Independent Cost Estimate
Base year 2003 dollars	May-03	Dec-06	May-06
Total	$91.4	$163.7	$203.3 - $233.9

Table 12. **Comparison of the Original Cost Estimate and Recent Cost Estimates for the FCS Program (in billions of dollars) (From: [64]).**

2. Scheduling Uncertainty

In comparison to the Joint Strike Fighter (JSF) which has only approximately 24 million SLOC [64] with a delivery timeline of 12 years using a similar 5 block incremental delivery approach, we believe that the FCS Network development schedule to be too risky as the software development and testing schedules appear to be based on the paradigm of "development success", with have little margin to accommodate delays. Furthermore, in light of the uncertainty surrounding the estimation of the size of the software, we believe that the schedule does not necessarily or adequately reflect the potential impact that uncertainty in size could have on the acquisition schedule.

I. TECHNICAL UNCERTAINTIES

There is a preponderance of technical uncertainties surrounding the development of the FCS Network by virtue of its reliance on the successful development of conceptual critical technologies. For example, technological maturation issues such as the limitations associated with wirelessly transmitting still images, video and audio have plagued the program. Using our "2**T**" paradigm we frame technical uncertainties along the following categories: Requirements Uncertainty, Integration Uncertainty, and Performance uncertainty. We expand on these uncertainties below.

1. Requirements Uncertainty

The lack of adequately defined requirements (Table 13) is one of the leading problems in the software development effort. Without adequate definition and validation of requirements and design, software engineers could be coding to an incorrect design, resulting in missing functionality and errors. This problem is further compounded by the lack of top-level requirements and architecture definition which in effect leaves system-level requirements incomplete until the preliminary design review in 2009, further affecting the accuracy of projected lines of code.

	Poorly defined requirements	Late requirements	Missing requirements
Software Partitions	X		X
Combat Identification	X	X	X
Battle Command and Mission Execution	X	X	X
Network Management System	X	X	X
Small Unmanned Ground Vehicle	X	X	X
Training Common Components	X	X	X
System of Systems Common Operating Environment	X	X	

Table 13. **Software packages and associated requirements problems (From: [59]).**

2. Integration Uncertainty

There are over 100 software vendors involved in the development of software programs for FCS, including 14 first-tier contractors, and many other sub-contractors [59]. Due to the amount of contractors involved, there is a lot of uncertainty surrounding how successful the different components would integrate due to the amount of collaboration that the effort would involve. This issue is further compounded by the inherent competition amongst software suppliers and their unwillingness or inability to share information and rapidly negotiate changes in products and interfaces, which could lead to missed delivery schedules, significantly reduced operational capabilities, and less dependable system performance.

3. Performance Uncertainty

Requirements changes, integration issues and late testing pose the risk of that the FCS Network might not yield fully functional software that performs as desired due to the complexity and functional scope. Furthermore, security is also a major concern from an operational perspective, with worries about the possibility of hackers implanting viruses and malicious code, thereby increasing the possibility of software failure when fielded in an operational mode.

Based on the uncertainties we have identified thus far, we can summarize the risks affecting the value of the FCS Network as being:

1. Requirements Creep
2. Estimation Accuracy (size and cost of the software)
3. Performance Risk
4. Software Integration

With this information in hand we now searched for historical data on a similar software development (size and scope) effort as the FCS Network that exhibits the same risk factors as the FCS Network effort. In this case we chose to select the Joint Strike Fighter program (JSF).

J. BASIS FOR SELECTING THE JOINT STRIKE FIGHTER PROGRAM

In our examination of historical data, we were able to determine that the historical data presented by the JSF program closely reflected the risks and challenges faced by the FCS program. Hence we elected to compare the JSF program to the FCS program for the following reasons.

1. JSF Technology Maturation Risks

Requirements instability perpetuated by technology maturation issues plagued the JSF program from the onset just like the FCS program, with only two of the JSF's eight critical technologies being fully mature and three other technologies being immature even after design review had been completed. This is a similar situation to the FCS program where only two of the program's 44 technologies are fully mature and 30 are nearing full maturity. Maturing critical technologies during development led to cost growth in the JSF program and is also the case in the FCS program.

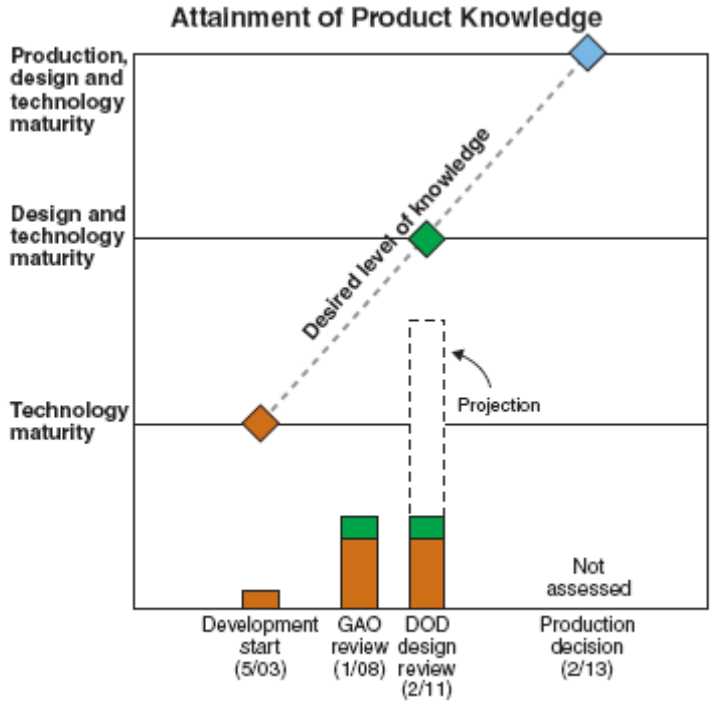


Figure 44. FCS Program Management (From: [62]).

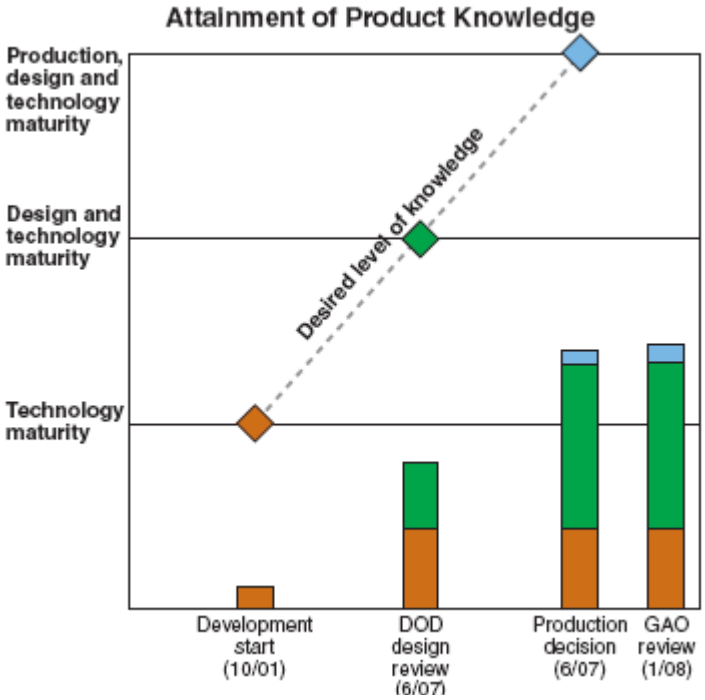


Figure 45. JSF Program Management (From: [62]).

2. JSF Program Management (Cost Risks)

The JSF and FCS contracts were both awarded on a cost reimbursable basis for 12 years. These contract vehicles allowed payments to the contractors on the basis of time spent on the project at pre-determined man-hour rates rather than a firm fixed price. Hence, the risk for all cost growth over the estimated value rests with the DoD. As of fiscal year 2007, DoD anticipates having to reimburse the prime contractors on these two programs nearly $13 billion more for their work activities than initially expected [62].

3. JSF Software Size (SLOC)

The JSF program represents the second largest software-intensive program behind the FCS. Figure 46 and Table 14 below highlight the SLOC, and cost and development schedule for the JSF.

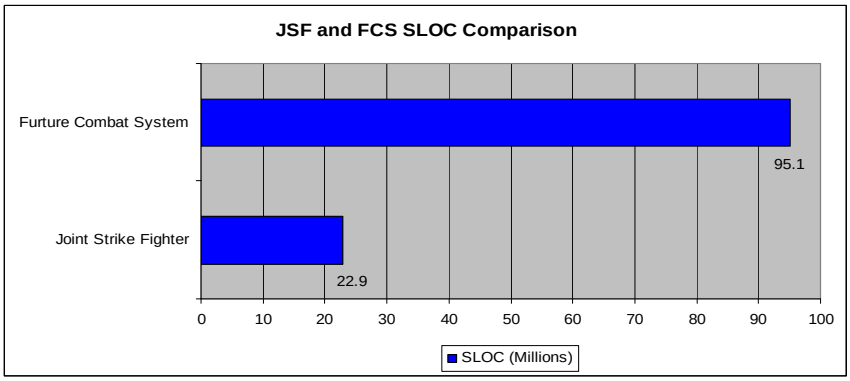


Figure 46. SLOC of Historical Acquisition programs of Software-Intensive-Weapons Systems.

	Program Inception November 1996	System development start (October 2001)	Program Rebaseline (December 2003)	Data as of December 2005
Cost Estimates (Then Year $ in billions)				
Development	$24.8	$34.4	$44.8	$44.5
Estimated Delivery Date				
First Operational Aircraft delivery	2007	2008	2009	2009

Table 14. **Joint Strike Fighter Cost and Schedule Increases from program inception.**

Our goals and objectives in utilizing the JSF historical data is to investigate, analyze and incorporate any lessons learned in the JSF program since development of the JSF began prior to the FCS.

K. RISKS IMPACT ON FCS NETWORK

In software development, requirements instability have been found to have a profound impact on a program's schedule and drive up costs due to Research, Development, Test & Evaluation (RDT&E) costs increases associated with the requirements changes. Therefore, given the situation with the FCS Network, we can depict the relationship between the risks as follows.

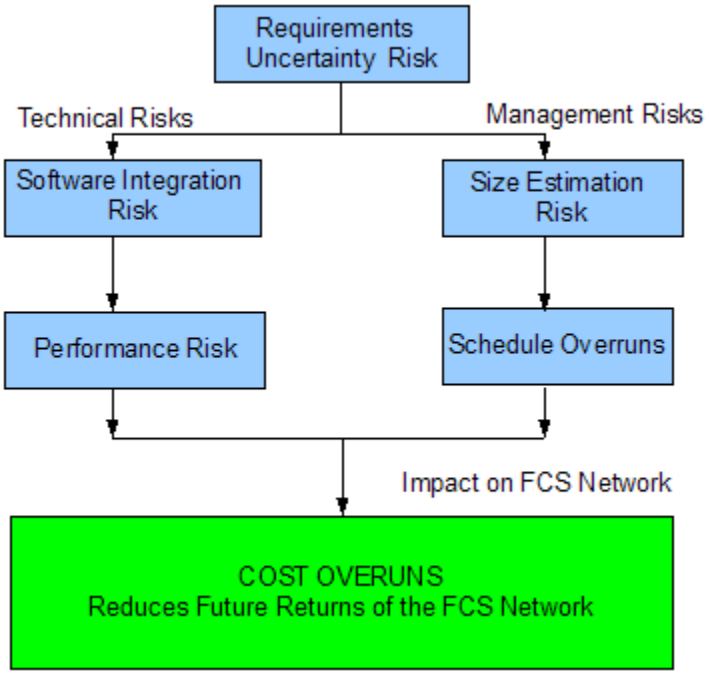


Figure 47. Impact of Risk on FCS Network.

Given the impact that requirements instability has on costs, we attempt to determine the rate of change of requirements or the volatility of requirements. To accomplish this, we utilize the Caper Jones' approach which is a transposition from the financial industry. Jones asserts that the method of average percentage of change of the overall requirements volume lacks information, because it does not give any information on the time in which the change occurred, a key factor that is important to determine in software engineering, since requirements changes become more expensive to implement, the farther we are into the software development process.

Jones therefore uses the compound monthly requirements volatility rate [67] to express the time aspect. Calculating monthly requirements volatility rates, as defined by Jones, is a transposition from the financial world. The time value or future value of money is well-known in the field of accounting as compound interest or CAGR, short for compound annual growth rate. By transposing from compound growth rate in finance we assume that requirements are compounded within a project [67]. The basic financial equation is given as follows:

$$r = \left(\sqrt[t]{\frac{SizeAtEnd}{SizeAtStart}} - 1 \right) \times 100$$

which translates to

$$r = \left(\sqrt[t]{\frac{SLOCAtEnd}{SLOCAtStart}} - 1 \right) \times 100$$

where t is the time period in years during which the estimates were observed

However, SLOC is not a suitable proxy for measuring requirements volatility because more often than not, it is dependent on the type of programming language being used and also does not take COTS into consideration, which represents a sizeable portion of the FCS program. Ideally we would have preferred to use an alternative proxy such as Function Points, which is a better metric for the size of the software requirements irrespective of how the software will be developed. However due to lack of data to determine the number of Function Points associated with the FCS Network program, we resorted to use SLOC to demonstrate our approach in determining requirements volatility. Note that the same approach can be used to determine requirements volatility using Function Points if we had information on the number of Function Points in the FCS program at the time of our study.

We compute the requirements volatility of both the FCS Network and the software component of the JSF using the SLOC data, and summarize the information we have thus far in Table 15, and compute the volatility of requirements using the formula above.

Program (Period)	**Beginning SLOC (Millions)**	**Ending SLOC (Millions)**
FCS (2003 – 2008)	33.7	95.1
JSF (2001 – 2008)	17.2*	22.9

Table 15. **Comparison between JSF and FCS SLOC**

*We estimate the initial SLOC of JSF program by utilizing the average of 25% increase in requirements reported by GAO in their assessment of selected weapons systems [62] and determine the initial SLOC estimate of the JSF to be 17.2 million SLOC

Program	Beginning SLOC (Millions)	Ending SLOC (Millions)	r (Volatility)
FCS (2003 – 2008)	33.7	95.1	12.37754868
			0.282516275
JSF (2001 – 2006)	17.2	19*	

Table 16. **Volatility computation using Caper Jones approach.**

*We utilize the SLOC of 19 million as reported in GAO assessments in 1996. We utilize this value because the FCS Network growth reported in this table is based on a 5 year period of SLOC growth, hence our need to size the JSF program accordingly as it represents a 5 year period of SLOC growth

We further estimate that over the 5 year period of observation depicted in Table 15, the FCS Network was 5.0 times the size of the JSF program (i.e. 95.1/19). Therefore, to adequately size the JSF program, we multiply its current volatility by 5 and get a resulting volatility estimate of requirements of 1.440833

In comparison with the averages reported by Jones' research on industrial averages (Table 17), we can see that requirements volatility in the FCS Network exceeds traditional averages of military software, making it a risky venture that is approaching the "out of control" category.

Software Type	Average volatility (%)	Maximum volatility (%)	Out of control (%)
MIS Software	1.2	5.1	> 5
Systems Software	2	4.6	> 5
Military software	2	4.5	> 15
Commercial software	2.5	6	No info
Civilian government software	2.5	No Info	No info
Web-based software	12	No Info	No info

Table 17. **Caper Jones' industry averages and maximum rates in various industries (From: [67]).**

Since the SLOC estimates of the FCS program are nearly 5 times that of the JSF program, we extrapolate the JSF data to mirror that of the FCS program using our Risk Simulator modeling toolkit. We first determine the value of the FCS Network as after all, the intent of running a Monte Carlo simulation is to determine the volatility of the returns on the FCS Network based on the modeled risks.

L. VALUATION OF THE FUTURE COMBAT SYSTEMS NETWORK

In order to value the FCS Network, we need to use a financial model and the development of the financial model requires that we either have or compute the following four factors.

1. Costs of developing the Future Combat Systems Network
2. Quantifiable benefits
3. Discount Rate
4. Time period

As of 2008, the overall FCS program cost was estimated at $163.7 billion [59], with the software component alone providing provides 95% of the FCS functionality [65]. Furthermore determining the discount rate and the time period is quite simple as this information is readily available. For discount rate, we use the spot rate on the U.S. Treasury bill and a time period of 10 years reflects the software development/delivery schedule.

However quantifying the benefits of the FCS Network in terms of a monetary value is a very complicated task. While the U.S. Army provides guidance on quantifying benefits based on factors such as cost savings, cost avoidances and productivity improvements, the numeric benefits cannot be easily estimated. Even though the overall FCS program would feature affordable sustainment costs, reduced logistics requirements, and a decrease in crew size as compared to the current systems as gathered from the initial justification for investment in the FCS program, we did not have sufficient data that reflected these benefits in the form of cost savings or cost reduction at the time of our study. To compound this, it is very difficult to proportionately allocate the cost savings to the FCS Network. Although the FCS Network is a critical component of the overall FCS program, these cost savings cannot be allocated to the FCS Network alone because the FCS Network is just a technology platform providing the underlying functionality of the overall FCS program, with many other components such as lighter manned vehicles contributing to the cost savings.

Consequently we make an assumption, the first of two assumption we make during the validation of our approach.

1. Assumption 1

We assume that the benefits could far exceed the costs under traditional NPV analysis or with the goal is to manage the development (investment) costs within the constraints of the given budget without reducing required functionality.

2. Justification

We believe this to be a reasonable assumption because of the perceived benefits obtained over an extended time period in the form of cost savings due to the replacement of legacy systems, vehicles and the tactical advantages the troops on the battle field would enjoy.

We take the approach that the regardless of the future benefits provided by the FCS Network, the value is derived based on how much we can successfully manage the software development or investment costs so that we can maximize the returns of the investment by keeping costs low via successfully planning and paying a risk premium up front for risks. In other words, we take an investment cost management approach towards maximizing future returns.

Since the government model is to provided the needed capability at least cost to the taxpayer, our focus on “managing” the investment costs to make sure that we do not overrun the costs and therefore maximize returns. Furthermore in actuality, since the program is currently underway, we can say with 100% certainty that a financial model, a pre-condition for the application for Real Options, was developed before the commencement of the acquisition effort.

Based on this underlying assumption we claim that under traditional NPV analysis, the NPV of the FCS Network using our formula below is positive.

$$\text{NPV} = \sum \frac{(C_t - M_t)}{(1+r)^t} - \text{I}$$

I is the (initial) development cost and represents

T is the (initial) development time or time to market,

C is the asset value and represents the total value of the positive cash flows that the project is expected to generate during its lifetime,

M is the operation cost and is the total value of all negative cash flows of the operation phase, calculated at time T.

(C –M) is always positive

r is the rate at which all future cash flows are to be discounted (the discount rate).

3. Assumption 2

For cost estimation purposes we assume that the overall FCS program is software, therefore we use the costs of the overall FCS program in our computation.

Thus based on our assumptions, we develop our financial model to mirror a positive NPV based on the following key input factors

I = $163. 7 billion

T = 13 years

r = 3.0 %

C – M = $10 trillion

Since we did not have detailed data, we assume an asset value of $10 trillion and operating cost of $0. Based on these values, we are able to compute the NPV of the FCS Network as being approximately $6.4 trillion as shown in Table 18.

	NPV of Future Returns on a Investment of $163.7 billion in an Asset Valued at $10 Trillion
FCS	$6,481,505,349,050.37

Table 18. **NPV of FCS Network.**

4. Rationale for Assumption

We assume these cost estimates reflect the estimates of the FCS Network as opposed to the overall FCS program for the purposes of our study since we cannot clearly delineate between software costs and hardware costs from the inception of the program. We therefore treat the overall FCS program as consisting of "software only" for cost purposes and assume the overall costs of the FCS Network is the cost of the overall FCS program. By doing this we can then develop a model based on the following:

Value of Army's overall FCS cost estimates = Army's FCS Network estimate

Value of CAIG's overall FCS cost estimates = CAIG's FCS Network estimate

Value of IDA's overall FCS cost estimates = IDA's FCS Network estimate

We made these assumptions because at the time of our study, we did not have access to the detailed data of both IDA's and CAIG's estimates of only the FCS Network. The data we reviewed at the time of our study did not provide information at the level of detail that was beneficial to our study, consequently we were not able to isolate the FCS Network (software component) from the overall FCS program based on their independent CAIG's assessment. Therefore, for the purposes of validating our framework, we assign the values of the overall FCS program provided by the Army, IDA and CAIG as our software development/investment costs. In other words we now assume that the estimates provided by the Army, CAIG and IDA of $163.7 billion, $218.5 billion (average of range provided) and $166.7 billion respectively represent the costs of the FCS Network investment.

Now that we have addressed the issue of asset valuation, we proceed to model our risks and determine the volatility of the returns on the investment using our modeling toolkit.

M. VOLATILITY DETERMINATION

Based on the uncertainties we have identified thus far, we now feed risks into our risk model in the Risk Simulator (Table 19) to observe the impact on the value of the future returns on the overall FCS program using the JSF program as a proxy in our data fitting. Based on our computation volatility of each of the risk factor we identified using the Caper Jones' approach, we are able to develop probability estimates, and run a Monte Carlo simulation. Since we are dealing with random quantities which are positive in nature and whose values cannot fall below zero we select the lognormal distribution which utilizes the following four mathematical constructs to compute four key measures known as "moments" (returns, risk, skewness and kurtosis) of our logarithm distribution These measures are computed using the following formulas as outlined in chapter IV of this study.

$$\text{f(x)} = \frac{1}{x\sqrt{2\pi \ln(\sigma)}} e^{-\frac{[\ln(x) - \ln(\mu)]^2}{2[\ln(\sigma)]2}} \quad \text{for } x > 0; \mu > 0, \sigma > 0$$

$$\text{mean} = \exp\left(\mu + \frac{\sigma^2}{2}\right)$$

$$\text{standard deviation} = \sqrt{\exp(\sigma^2 + 2\mu)[\exp(\sigma^2) - 1]}$$

$$\text{skewness} = \left\lfloor\sqrt{\exp(\sigma^2) - 1}\right\rfloor(2 + \exp(\sigma^2))$$

$$\text{excess kurtosis} = \exp(4\sigma^2) + 2\exp(3\sigma^2) + 3\exp(2\sigma^2) - 6$$

The model is run and the simulation calculates numerous scenarios of the model by repeatedly picking values from the lognormal distribution for the uncertain variables using the values in the model.

					Risk Factors							
	NPV of Future Returns on a Investment of $163.7 billion in an Asset Valued at $10 Trillion	Current Requirements (SLOC)	Planned Schedule (Months)	Planned Costs	Requiements Creep Risk	Intergration Risk	Schedule Overrun	Performance Risk	Software Cost Estimation	Unit Cost	Monthly Cost	Final Costs Due to Risk Factors
FCS	$6,138,586,402,574.14	95000000	156	$163,700,000,000.00	190.90%	0.56%	-0.02%	9.38%	91.37%	$1,723.16	$1,364,166,666.67	$670,926,997,357.64

Table 19. **Screen capture of risk model developed in the Risk Simulator.**

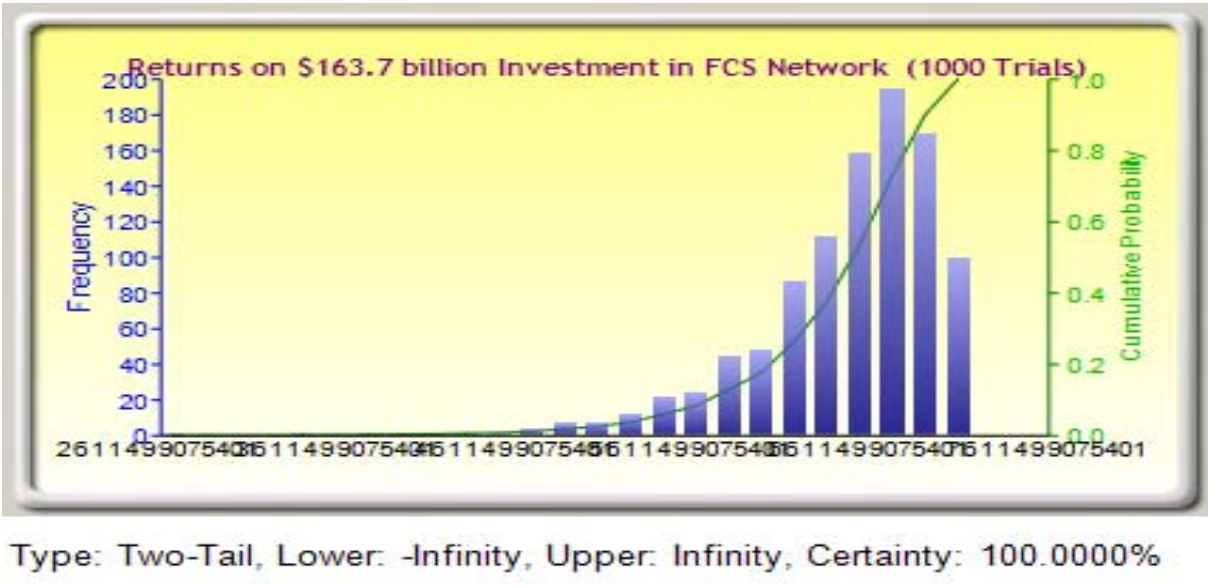


Figure 48. Risk Simulator output depicting returns on a $163.7 billion Investment in the FCS Network. The frequency histogram shows the frequency counts of values occurring in the total number of trials simulated. The vertical bars shows the frequency of a particular NPV occurring out of the total number of trials, while the cumulative frequency depicted by the smooth line shows the total probability of all values at and below the maximum NPV occurring in the forecast.

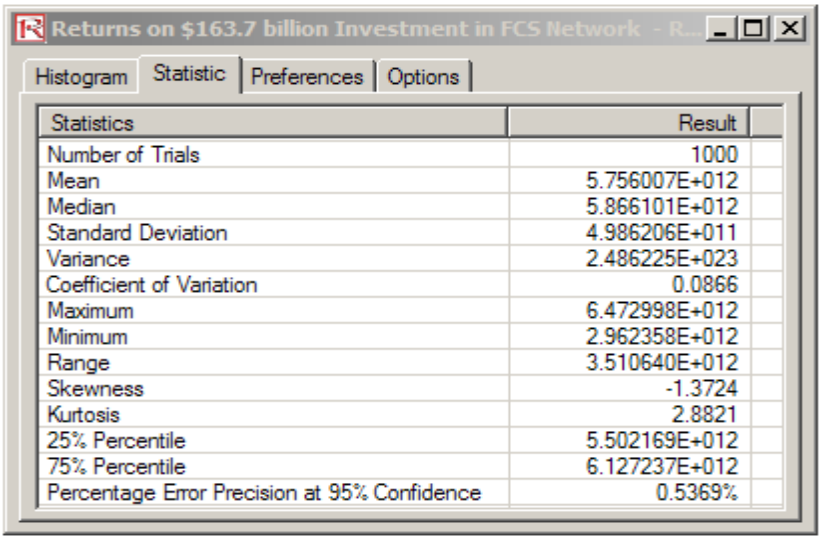


Figure 49. Volatility of the returns on the $163.7 billion Investment in the FCS Network. The forecast statistics summarizes the distribution of the forecast values in terms of the four moments of a distribution.

In Figure 49, we observe that the volatility of the returns associated with the FCS Network investment, as indicated by the correlation of variation, has a value of 0.0866, and there is a decrease in the NPV from $6.4 trillion to $6.1 trillion, a direct reflection of the risk factors of requirements volatility, integration volatility, schedule overrun, performance risk and software cost estimation risk which we factored into our valuation of the FCS Network.

We use coefficients of variation as a proxy of standard deviation because it is a relative measure of the standard deviation expressed as a percentage of the mean. Even though we have identified five possible risks as being the key contributors to the volatility of the returns of investing in the FCS Network, we model only the requirements risk and attempt to determine that, given the volatility of requirements, how does this impact the schedule and consequently affect the value returns of the FCS Network.

Next we attempt to refine our volatility estimates using Dempster-Shafer Theory. Since a key proposition of Dempster-Shafer Theory is that all sources of information be independent, we satisfy this proposition by utilizing independent estimates provided by the Cost Analysis Improvement Group (CAIG) [64] and Institute for Defense Analysis (IDA) [59]. We review the background of both organizations in the next section.

N. BACKGROUND OF THE EXPERTS

The Institute for Defense Analyses (IDA) is a non-profit organization first established in 1947 responsible for providing independent technical analyses of weapons systems and programs. They administer three federally funded research and development centers to provide objective analyses of national security issues, particularly those requiring scientific and technical expertise, and conduct related research on other national challenges. The second independent entity, the Cost Analysis Improvement Group (CAIG) was formed at the behest of the Deputy Secretary of the Department of Defense. In 2006, the deputy secretary issued a policy directive (5000.04) directing that any Major Defense Acquisition Program (MDAP) undertaken by the DoD undergoes an independent review and assessment. As an objective entity in Office of the Secretary of Defense, the CAIG was identified as the independent body responsible for conducting independent cost assessments of Major Defense Acquisition programs and serving as the principal advisor on matters of program life-cycle cost.

O. REFINING VOLATILITY USING DEMPSTER SHAFER'S THEORY

To compute the volatility of FCS Network based on the risks identified above, we first establish the frame of discernment Θ which consists of the set of mutually exclusive alternatives or possibilities these risks pose to the FCS Network. In the case of CAIG, they provide a range of $203 - $234 billion for their estimate, thus we take the average to reflect the median ($218.5 billion) of their estimate. The IDA, on the other hand, provides an estimate that is $3 billion over the Army's estimate. An important point worth mentioning is that the Army does not agree with both assessments as program officials believe that the independent estimate of research and development costs is too high because it is too conservative regarding risks [64].

We summarize the independent estimates provided by both organizations in the Table 20 below.

Source of Assessment	FCS Estimates (billions)
Army	$163.7
Cost Analysis Improvement Group	$218.5
Institute of Defense Analysis	$166.7

Table 20. **Summary of the three independent estimates of the FCS program**

The first step in application of DST is to establish belief functions. To accomplish this, we examined the primary differences between the CAIG estimate, IDA estimates and the Army's estimates. CAIG's estimate appeared to be the most conservative estimate. They determined that the FCS software development would require more time and effort to complete than the Army had estimated, resulting in several additional years and additional staffing beyond the Army's estimate to achieve initial operational capability [64]. IDA found that Army plans for developing FCS, including the network, were optimistic with regard to time and money needed for the program and therefore projected at least $3 billion in additional FCS development costs due to unplanned software effort including code growth, software integration difficulties, and longer development schedules [59]. However based on our studies, it appeared the general consensus was that there was no question regarding the issue of requirements volatility. In other words, based on our assessment of GAO reports, we were led to believe that the probability of requirements creep was fairly consistent across all sources. Therefore, while the probability of schedule impacts might be debatable across all the different estimates, the probability of requirements creep was the same.

We now establish the frame of discernment for both the CAIG's and IDA's assessment of the FCS Network based on the set of the risk factors affecting the investment as follows:

$\Theta = \{SE, SR, SI, SS, FP\}$

where

Estimation Accuracy Risk (SE) - Under/overestimation risk

Requirements Creep Risk (SR) - Requirements volatility risk

Software Integration Risk	(SI)	- Integration risk
Delivery Risk	(SS)	- Risk of schedule overruns
Performance risk	(FP)	- Risk of software not meeting user needs

CAIG Belief functions are as follows[3]:

$Bel(\{SR\}) = 0.90$

$Bel(\{SR, SE\}) = 0.90 + 0.08$

$Bel(\{SR, SE, SI, SS, FP\}) = 1$

We assign $Bel(\{SE\})$ the next highest probability assignment because the effects of requirements creep on the cost estimation of the FCS Network.

The belief functions were based on the masses of evidence gathered as follows

$m(\{SR\}) = 0.90$

$m(\{SR, SE\}) = 0.08$

$m(\{SR, SE, SI, SS, FP\}) = 0.02$

Similarly we come up with the following Belief assignments based on IDA's assessment using the same analogy above.

$Bel(\{SR\}) = 0.95$

$Bel(\{SR, SE\}) = 0.95 + 0.03$

$Bel(\{SR, SE, SI, SS, FP\}) = 1$

$m(\{SR\}) = 0.95$

$m(\{SR, SE\}) = 0.03$

$m(\{SR, SE, SI, SS, FP\}) = 0.02$

3 To determine CAIG's subjective probability assignments on the Army's estimate, we need to examine the mass of evidence that CAIG is able to gather to either validate or discredit the Army's estimate. Since this information was not explicitly available at the time of this study, we resort to a different approach for illustrative purposes.

We now attempt to combine the information we have so far. I.e., CAIG's assessments and IDA's assessment by computing the orthogonal sum in the form m = $m_1 \oplus m_2$ where m_1 (CAIG) and m_2 (IDA) represent the basic probability on our frame of discernment Θ. (Table 21)

		Cost Analysis Improvement Group Estimate (CAIG)		
	Cost Increase Factors	{SR} m1 = 0.90	{SR, SE} m1 = 0.08	{SR, SE, SI, SS, FP} m1 = 0.02
Institute of Defense Analysis (IDA)	{SR} m2 = 0.95	0.855	0.076	0.019
	{SR, SE} m2 = 0.03	0.027	0.0024	0.0006
	{SR, SE, SI, SS, FP} m2 = 0.02	0.018	0.0016	0.0004

Table 21. **Screen capture of orthogonal matrix.**

We obtain the following Evidence Functions

$m_1 \oplus m_2$ ({SR}) = 0.855 + 0.076 + 0.019 + 0.027 + 0.018 = 0.995

$m_1 \oplus m_2$ ({SR, SE}) = 0.0024 + 0.0006 + 0.0016 = 0.0046

$m_1 \oplus m_2$ ({SR, SE, SI, SS, FP }) = 0.0004

To examine the degree of conflict we revisit Eqn 17 above depicted as:

$$m_{12}(\mathrm{A}) = \sum_{\mathrm{B}\cap\mathrm{C}=\mathrm{A}} \frac{m_1(\mathrm{B})m_2(\mathrm{C})}{1-K} \text{ when } \mathrm{A} \neq \emptyset$$

where

$$K = \sum_{\mathrm{B}\cap\mathrm{C}=\emptyset} m_1(\mathrm{B})m_2(\mathrm{C})$$

However since we have no null intersections, the sum of all such sets is in our matrix in Table 21 is equal to 0. Therefore $1 - K = 1$

The plausibility (certainty) of these beliefs are computed based on the following doubt functions

Dou({SR, SE, SI, SS, FP}) = *Bel* (Ø) = 0

Dou({SR, SE}) = *Bel* (SI, SS, FP) = 0

Dou({SR }) = *Bel* (SE, SI, SS, FP) = 0

Thus plausibility is as follows:

Pl({SR}) = 1- Dou({SE, SI, SS, FP}) = 1

Pl({SR, SE }) = 1- Dou({SI, SS, FP}) = 1

Pl({SR, SE, SI, SS, FP}) = 1- Dou({SR, SE, SI, SS, FP}) = 1

We can establish the following joint beliefs

Joint Belief of Independent expert #1 and #2 estimates in {SR} is 0.995

Joint Belief of Independent expert #1 and #2 estimates in {SR, SE} is 0.9996

Joint Belief of Independent expert #1 and #2 estimates in {SR, SE, SI, SS, FP} is 1

Since SR is the dominating risk factor within the context of our assumptions, the joint belief in SR of 0.995 infers that both CAIG and IDA have similar if not exact beliefs on the question at hand regarding SR.

The next step in the analysis is to determine the "direction" of the belief. For the purposes of our study, we assume that the DST results imply a 99% degree of belief that the Army underestimated the risk of requirements creep, and infer that requirements volatility is actually 20%[4] based on both the CAIG and IDA's estimates as opposed to the 12% volatility value determined by the Army

Consequently with the belief that an increase in requirements risk would adversely impact cost and schedule estimation, we readjust the risk factor off requirements volatility by increasing the probabilities of its occurrence by 8% in our model. (20% - 12%) and assign readjust the other risk factors of (SE, SS, SI, FP) such that there sum is 1.

4 While our assumption of the risk of requirements increases is consistent with the findings of both the CAIG and IDA as reported in their testimony to Congress based on the cost estimates provided by the Army, we further assume in this study that the preponderance of evidence gathered by both independent experts served as a basis for determining a 20% volatility based on the proportionate recomputation of all the other risk factors surrounding the FCS program such that the sum is of the probabilities of all the other risk factors is 1 or 100%.

We rerun a Monte Carlo Simulation and as shown in our model (Table 22).Our new results are depicted in the new output charts:

					Risk Factors							
	NPV of Future Returns on a Investment of $163.7 billion in an Asset Valued at $10 Trillion	Current Requirements (SLOC)	Planned Schedule (Months)	Planned Costs	Requirements Creep Risk	Intergration Risk	Schedule Overrun	Performance Risk	Software Cost Estimation	Unit Cost	Monthly Cost	Final Costs Due to Risk Factors
FCS	$5,738,251,785,507.12	95000000	156	$163,700,000,000.00	413.17%	0.61%	0.20%	9.30%	81.52%	$1,723.16	$1,364,166,666.67	$1,071,261,614,424.66

Table 22. **Screen capture of risk model developed in the Risk Simulator with revised estimate.**

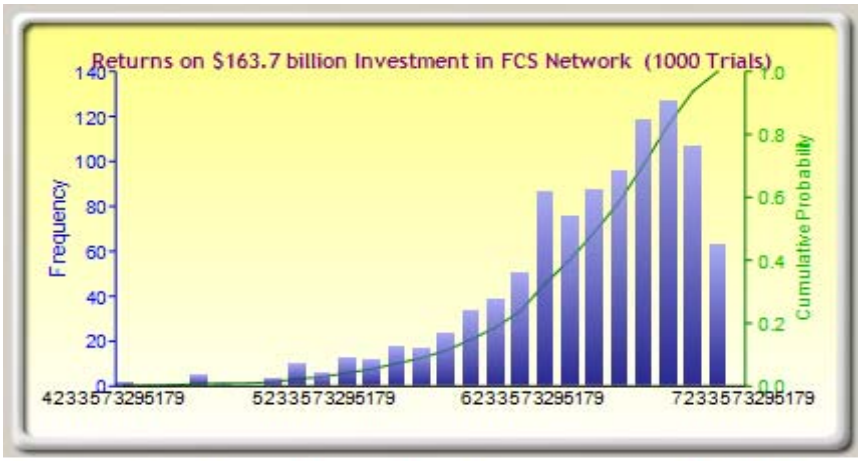
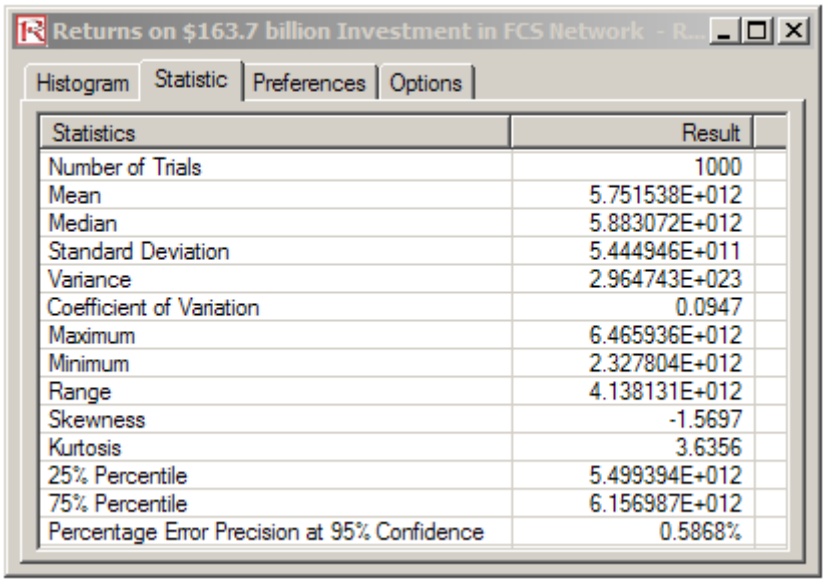


Figure 50. Risk Simulator output depicting revised returns on a $163.7 billion Investment in the FCS Network with revised probability estimates. The frequency histogram (Figure 49) shows the frequency counts of values occurring in the total number of trials simulated. The vertical bars shows the frequency of a particular NPV occurring out of the total number of trials, while the cumulative frequency depicted by the smooth line shows the total probability of all values at and below the maximum NPV occurring in the forecast.



Statistics	Result
Number of Trials	1000
Mean	5.751538E+012
Median	5.883072E+012
Standard Deviation	5.444946E+011
Variance	2.964743E+023
Coefficient of Variation	0.0947
Maximum	6.465936E+012
Minimum	2.327804E+012
Range	4.138131E+012
Skewness	-1.5697
Kurtosis	3.6356
25% Percentile	5.499394E+012
75% Percentile	6.156987E+012
Percentage Error Precision at 95% Confidence	0.5868%

Figure 51. Revised volatility of the returns on the $163.7 billion Investment in the FCS Network. The forecast statistics summarizes the distribution of the forecast values in terms of the four moments of a distribution.

In comparing our new volatility estimates of 0.09477 to the previous volatility estimate of 0.0866, we observe that the volatility of the returns on the FCS Network increased, while the returns of our investment declined from $6.1 trillion to $5.7 trillion - a direct consequence of the increase in volatility of our investment effort. Therefore given the volatility of the returns on the FCS Network investment the decision-maker with executive oversight on the FCS Network needs to develop and incorporate the appropriate Real Options. We now proceed to formulating our Real Options.

P. IDENTIFYING OPTIONS

Now that we have determined the volatility of the returns of the FCS Network, we develop Real Options to address the risk factors we have identified so far. The strategy taken is that in the light of requirements uncertainty which is beyond the control of the program manager or executive, how does s/he decrease costs or how do they stay within the authorized budget. We therefore recast the FCS Network development effort as a series of Deferment/Learning Options and Investment/Growth Options during which the Option to Start, Stop, and Options to scale up staff and scale down staff or defer development in the face of requirements uncertainty is utilized. This whole strategy is

based on reallocating resources based on the periods of lulls in software development activities based on uncertainties, i.e. maximizing resources (e.g. staff). This concept is based on a stage-gate development approach utilizing the concepts of both iterative development approaches and standard portfolio theory. Its success relies on the successful and correct partitioning/decomposition of the FCS Network into the appropriate subsystems which the Army appears to have done already. We therefore partition the FCS Network solution into the six independent systems as identified in Table 13 above as follows:

1) Combat Identification
2) Battle Command and Mission Execution
3) Network Management System
4) Small Unmanned Ground Vehicle
5) Training Common Components
6) Systems of Systems Common Operating Environment

If further partitions of each of these subsystems are warranted, they should be developed accordingly. Now that we have our FCS Network "portfolio" of all the required systems, to the extent possible, development of each system can begin at the same time, using iterative development techniques to guide each system towards success while at the same time deferring systems that still face requirements uncertainty. The iterative concepts employed would facilitate shorter feedback cycles by cycling through the development phases, from gathering requirements to delivering functionality in a working release.

Since there are six possible software systems, we examine the 63 possible combinations of one or more components may have uncertainties in them at the same time, resulting in a strategy tree based on 63 possible strategies.

Rather than develop a complicated strategy tree showing our 63 strategic pathways, we examine one possible scenario with two different strategic options.

1. Scenario

All six systems are facing uncertainty with the exception of one system. That is, at least one system out of the six systems is not facing requirements uncertainty. We therefore develop two types of options in response to this scenario 1) Compound option 2) Deferment Option

a. Compound Option

In the event that at least one of the systems is not facing requirements systems, with all the others facing requirements uncertainty, an option could be developed to "scale down" the resources (staff) allocated to the other systems. The staff could then be switched to work on the system that is not facing requirements uncertainty, while the uncertainty is addressed using our uncertainty elicitation model. The assumption with this approach is the system development effort which the staff engineers are being reallocated to work on is not already behind schedule and hence does not violate *Brooks Law*[5]). If the development effort which the staff are being assigned to work on is late (behind schedule), the number of staff, experience level and role which the added staff would play in the software development effort must be taken into considerationWe therefore frame the Real Options in this case as:

To illustrate the reasoning behind the development of our Real Options "Option to Scale down from a system with uncertain requirements, Option to Switch resources to another system, Options to Scale up staff assigned to the development of a system not facing uncertainty we use an analogy based on the concept of "switching lanes" on a highway.

When a driver feels a lane is going too slow, he has the option to switch to the fast lane. However this option comes with a premium in form of faster burn rate of fuel and training associated with being able to drive in the fast lane in order to keep up with the moving traffic. The driver remains in the fast lane until he runs into another driver ahead

[5] Brooks law states that adding people to a late project makes it later.

of him, slowing down the traffic. He then exercises his option to switch back to the slow lane.

In the case of the FCS Network, by relocating resources from unsuccessful systems (systems facing significant uncertainty) to "successful" systems, i.e. systems not facing requirements uncertainty, the development of the successful systems could be accelerated based on the increased level of effort. Granted previous research by Robert Glass has shown that this might not be the best way to accelerate delivery (i.e. by throwing bodies to project), we explicitly acknowledge this by developing a "Switching Option" that acknowledges the costs that might be incurred in the logistics related to training and reassigning staff to the successful systems.

Therefore given our approach in this scenario, what we have essentially done is develop a "Sequential Compound Switching Option", an "Option" on a "Option".

b. Deferment Option

In the event that five of the six systems are facing requirements uncertainty, another option could be developed to stop and defer all development until uncertainty is resolved in the single system. What we have essentially done is develop a Deferment Option.

Now that we have determined the appropriate Real Options, our next step is to develop a strategy tree depicting the strategic pathways. As mentioned in the previous chapter under the partitioning (decomposition) of the software solution, the success of these strategies hinges on the following.

> The software solution has been partitioned (decomposed) properly, taking into consideration uncertainty both within the subsystem and across the subsystems and uncertainty either reduced to lowest level possible or resolved using modeling and simulation tools as advocated for in our discussion.

We depict our strategy tree in Figure 52.

Q. STRATEGY TREE OF COMPOUND AND DEFERMENT OPTIONS

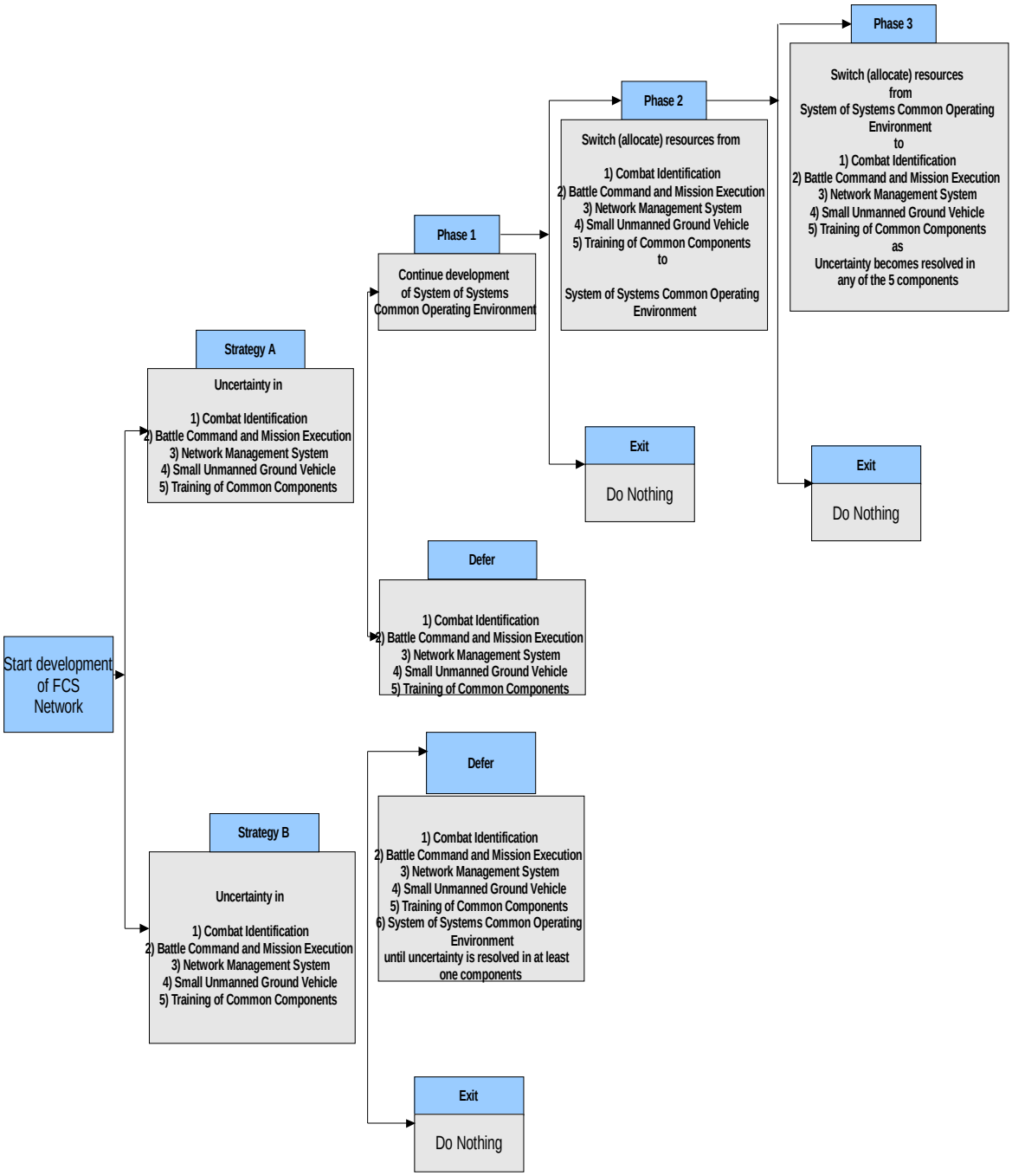


Figure 52. Strategy tree depicting the types of options for scenario involving 5 Component Systems of the FCS facing Uncertainty.

R. OPTION VALUATION

As the next step in our methodology, we examine strategy A to value the "options" we created to hedge against risks using the Binomial Lattice approach. To accomplish this, we used the Super Lattice Solver 3.0 provided by Real Options Valuations Inc. The simulator uses a two step approach.

1. The first step involves the valuation of the underlying asset.
2. The second step involves the creation of the option valuation lattice using the values computed in the lattice evolution of the underlying asset.

We selected the "Multiple Asset Supper Lattice Solver" choice in the tools menu because our strategy approach is based on sequential compound options with multiple phases (3 phases based on our strategy tree).

1. Real Options Assumptions

Since we already made an assumption that the FCS Network developments costs $163.7 billion, and have also determined that the NPV of the FCS Network reduced from a initial value of $6.4 trillion to $5.7 trillion due to a volatility of 0.0947%. We apply these assumptions in our model based on a risk free rate of 3%.

a. Strategy A

Phase 1. Since we have uncertainty in 5 of the components, we proceed to develop the remaining component and defer the other 5 components until uncertainty is resolved in the 5 components.

Phase 2. We allocate the resources from the 5 components facing uncertainty to component 1.

Phase 3. We reallocate the resources back to the 5 components once uncertainty is resolved. We account for the administrative costs associated with reallocating resources back to the development of the 5 components. Thus we recoup the

overall benefits of the $6 billion set aside in phase 1 for administrative and resource allocation logistics issues.

With this assumption in hand, we go ahead and set up our model as depicted below in Figure 53 below subject to the following underlying assumptions:

Implementation Cost of FCS Network is $163.7 billion

Value of Underlying Asset is $6.4 trillion

The risk free rate is the 3.0

Volatility of our project cv = is 0.0947

Duration is 13 Years

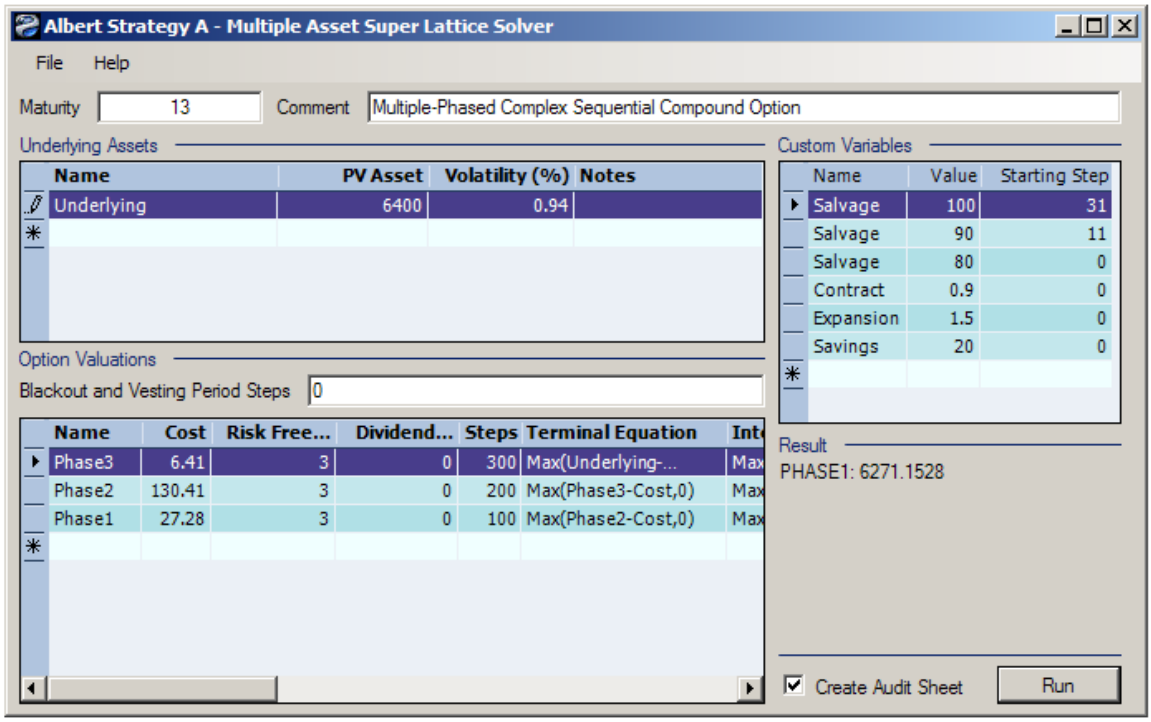


Figure 53. Screen Shot of our Model in Super Lattice Solver 3.0.

We execute the model and obtain the lattice of our underlying asset as well as the lattices associated with each of the three phases

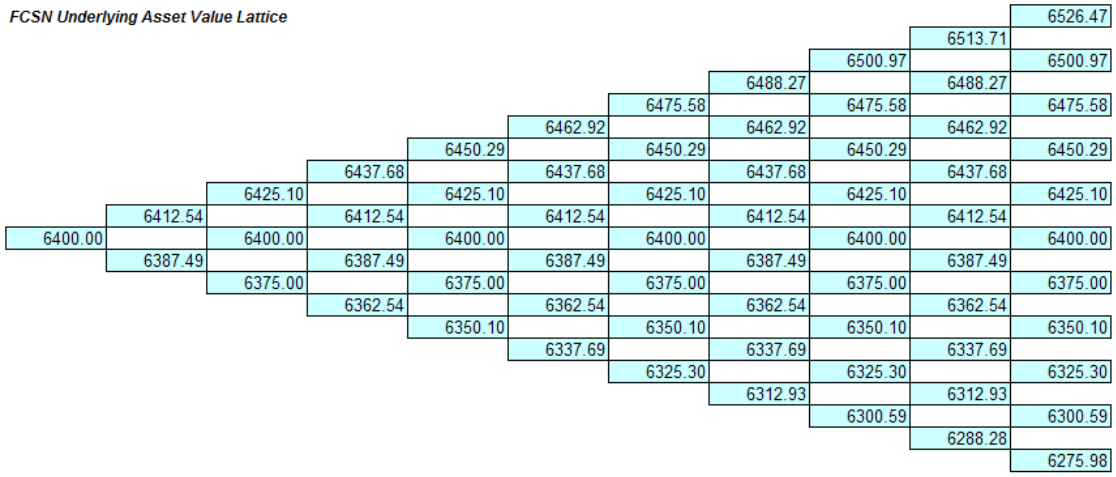


Figure 54. Lattice of Underlying Asset (FCS Network).

The value of the underlying asset was computed as $6.4 trillion (Figure 54).

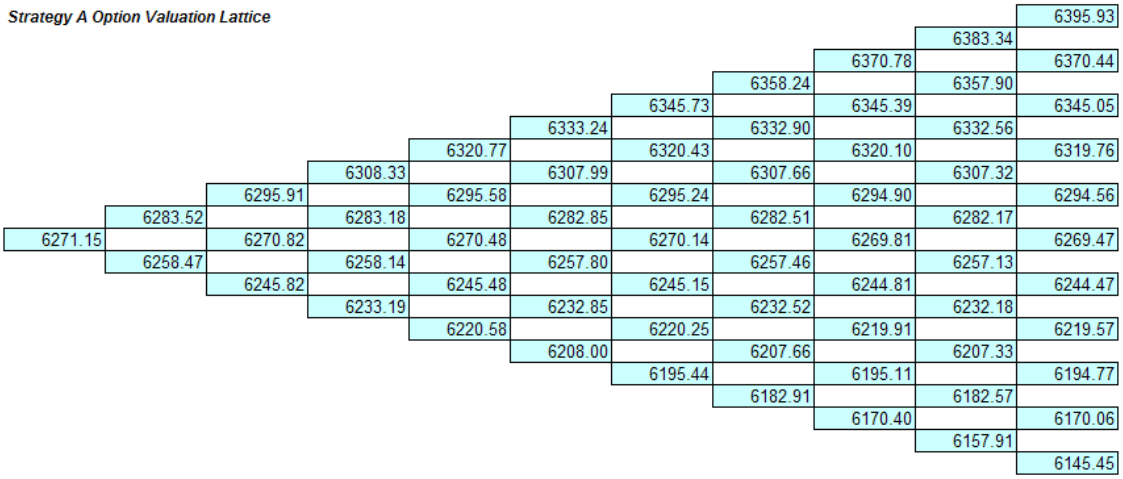


Figure 55. Phase 1 Option Valuation Lattice.

The option analysis which represents the value of the option under strategy A returned a value of $6.27 trillion (Figure 55). The lattices are created and values computed through backward induction working backwards from phase 3 through phase 2 to phase 1. Given the option value of $6.27 trillion, the intrinsic value of the option (option premium) is determined to be $6.27 trillion – $5.7 trillion = $570 billion .

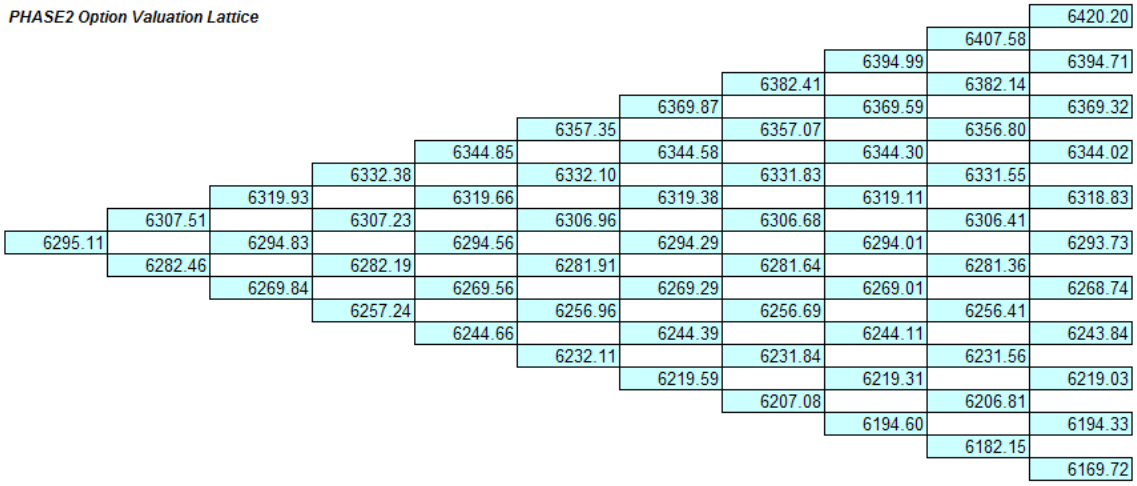


Figure 56. Phase 2 Option Valuation Lattice.

The phase 2 option analysis returned a value of $6.29 trillion (Figure 56).

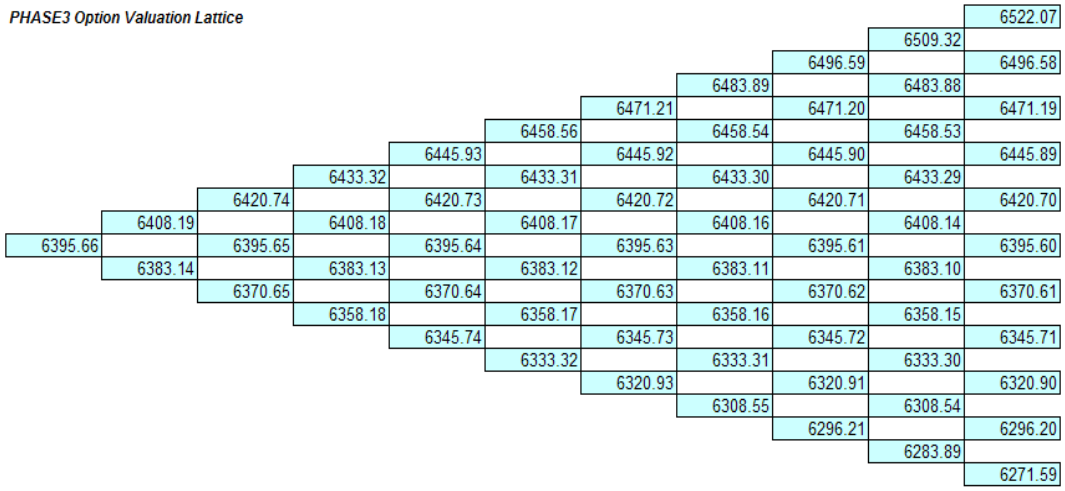


Figure 57. Phase 3 Option Valuation Lattice.

The phase 3 option analysis returned a value of $6.39 trillion (Figure 57).

Option Valuation Lattices

Phase3

Cost	$6.41	*Dividend*	0.00%
Riskfree	3.00%	*Steps*	300

Terminal Equation	Max(Underlying-Cost,0)
Intermediate Equation	Max(Underlying-Cost,OptionOpen)
Intermediate Equation (Blackout)	OptionOpen

Phase2

Cost	$130.41	*Dividend*	0.00%
Riskfree	3.00%	*Steps*	200

Terminal Equation	Max(Phase3-Cost,0)
Intermediate Equation	Max(Phase3-Cost,OptionOpen)
Intermediate Equation (Blackout)	OptionOpen

Phase1

Cost	$27.28	*Dividend*	0.00%
Riskfree	3.00%	*Steps*	100

Terminal Equation	Max(Phase2-Cost,0)
Intermediate Equation	Max(Phase2-Cost,OptionOpen)
Intermediate Equation (Blackout)	OptionOpen

Custom Variables

Name	Contract	Expansion	Salvage	Salvage	Salvage	Savings			
Value	0.90	1.50	80.00	90.00	100.00	20.00			
Starting Step	0	0	0	11	31	0			

Figure 58. Audit Sheet For Strategy A.

We also captured the "audit" trail of the values in our model which is depicted in Figure 58 above. The Terminal equation depicted in our model is the computation that occurs at maturity, while the intermediate equation is the computation that occurs at all periods leading up to maturity and is computed using backward induction hence the reduction in the valuation of the options from $6.39 trillion to $6.29 trillion and to $6.27 trillion across phases 3, 2 and 1 of the options valuation lattices respectively.

b. Strategy B

In strategy B, which calls for deferment, the assumption is made that the duration for deferment option would be 3 years. (This means that the project is deferred and not commenced until uncertainty is resolved within the 3 year period.) The model is set up (Figure 59) using the same assumptions below.

Implementation Cost of FCS Network is $163.7 billion

Value of Underlying Asset is $6.4 trillion

The risk free rate is the 3.0

Volatility of our project cv = is 0.0947

Duration is 3 Years (Maturity of deferment option)

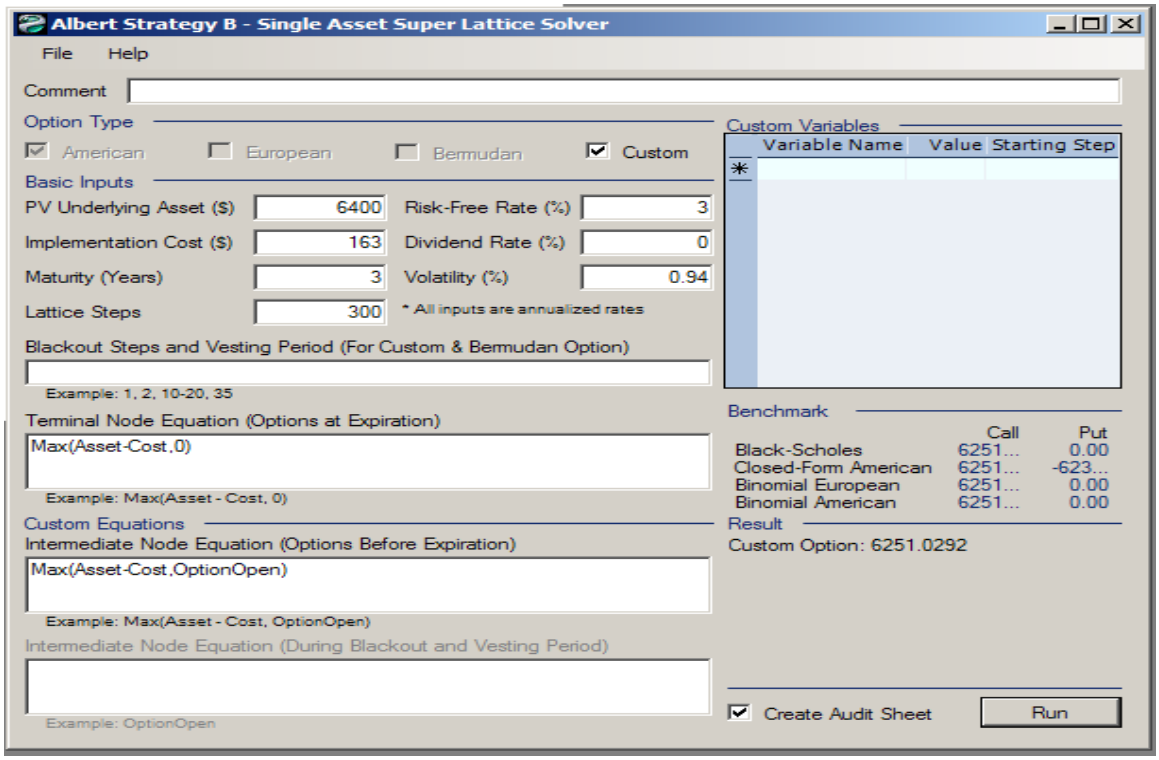


Figure 59. Real Options Super Lattice Solver Deferment Model.

We also captured the "audit" trail associated with our model which is depicted in Figure 60 below and also provide an explanation of the equations in the analysis of our results.

Option Valuation Audit Sheet

Assumptions	
PV Asset Value ($)	$6,400.00
Implementation Cost ($)	$163.00
Maturity (Years)	3.00
Risk-free Rate (%)	3.00%
Dividends (%)	0.00%
Volatility (%)	0.94%
Lattice Steps	300
Option Type	Custom

Intermediate Computations	
Stepping Time (dt)	0.0100
Up Step Size (up)	1.0009
Down Step Size (down)	0.9991
Risk-neutral Probability	0.6594
Results	
Auditing Lattice Result (10 steps)	6251.03
Super Lattice Results	6251.03

Figure 60. Audit Trail of Option to Defer Model.

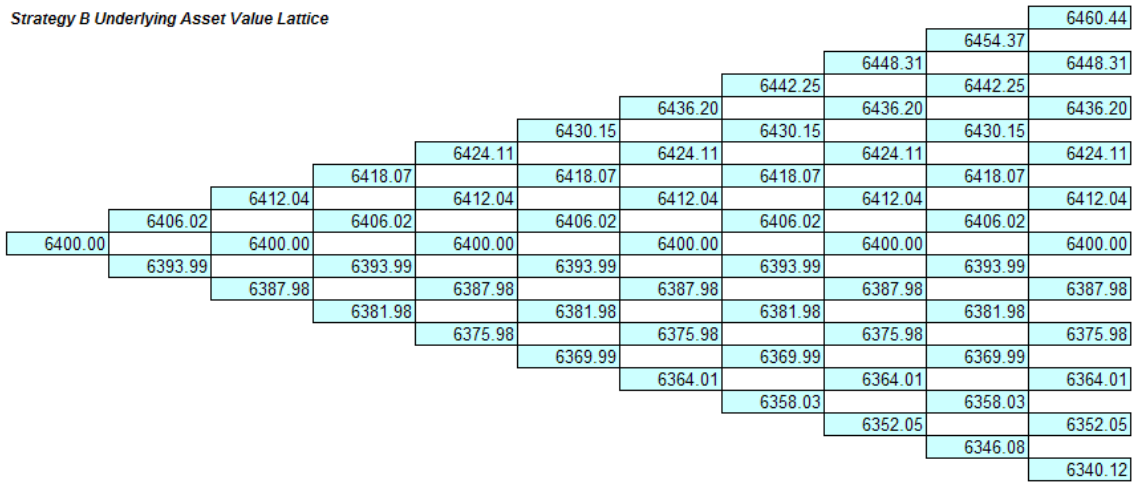


Figure 61. FCS Network Underlying Asset Lattice of with Deferment Option.

The model is executed and similar to strategy A, the value of the underlying asset was computed as $6.4 trillion (Figure 61). The option analysis on the other hand returned a value of $6.25 trillion (Figure 62). Thus the intrinsic value of the deferment option under strategy B is determined to be $6.25 trillion – $5.7 trillion = $550 billion.

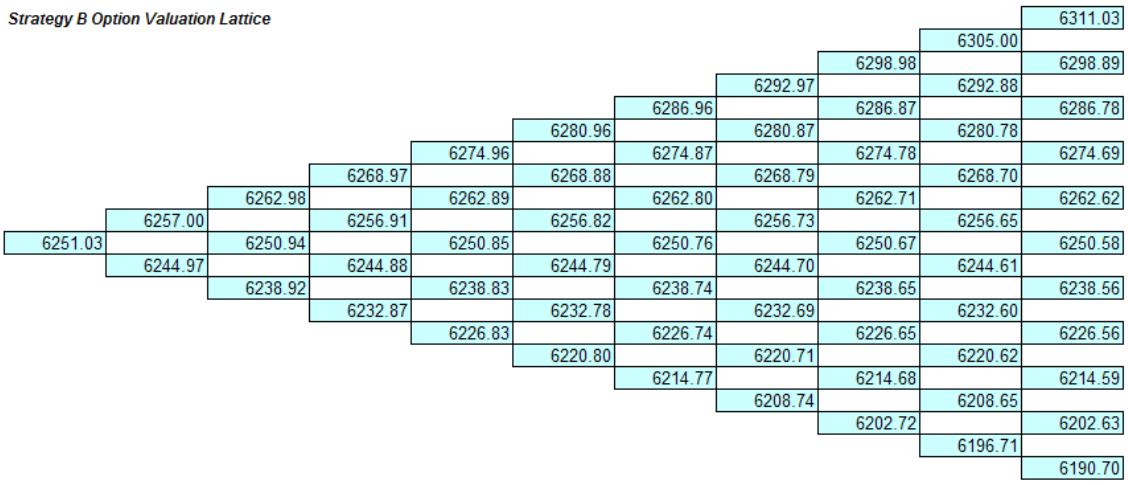


Figure 62. Options Valuation Lattice under Deferment.

S. ANALYSIS OF RESULTS

In comparing the results of both strategies A and B, we observe that while strategy A has an option value of $570 billion, strategy B has an option value of $550 billion. What this implies, is that under both strategies A and B, the software executive should be willing to pay no more, (and hopefully much less than) the option value of $570 billion and $550 billion respectively in addition to the initial investment cost of $163.7 billion to increase the chances of receiving the projected NPV of $6.27 trillion under strategy A and $6.25 trillion under strategy B for the FCSN as opposed to the current projected NPV of $5.7 trillion in light of the risks caused by the uncertainties in five of the six software components.

In analyzing both strategies, strategy A is more attractive in the sense that instead of waiting to invest until after 3 years (after which uncertainty would hopefully have been resolved) and then proceeding to spend $163.7 billion at once, the staged phase approach that adopts spending some little first and then investing more over time with a proof of concept safer is worth more. Therefore under these conditions, strategy A which employs the compound sequential options is the optimal approach.

T. KEY BENEFITS OF APPROACH

The key benefit of the approach that can be observed in the analysis of results is the recognition of the added flexibility which the software program manager has, to make decisions during the life of the FCS Network effort. The Real Options approach viewed the FCS Network as a series of sequential compound options, with each option depending on the exercise of those that preceded it.

This approach is very beneficial to the FCS program in that the current acquisition strategy is based on a cost reimbursable strategy in which contractors are reimbursed for their expenses regardless of the sometimes unreasonable justification of incurring the costs. By adopting our approach, we posit that the manager or decision maker would be able to actively manage the investment effort by exercising the applicable Call Options at the appropriate time, a key benefit towards controlling cost overruns that the current cost

reimbursable acquisition strategy facilitates. In comparison of our approach to traditional Discounted Cash Flow techniques such as Net Present Value, we can see that the investment costs of the FCS Network investment effort is lower under NPV ($163.7 million) than the Real Options approach ($163.7 million + Option Premium), because the NPV approach assumes there is no uncertainty in the FCS Network and thus does not account for risk in the decision making process.

Secondly, we have shown how the volatility of the FCS Network can be refined in the event that the risk factors are either underestimated or overestimated by the Army by using Dempster-Shafer Theory to refine the Army's estimate. This is particularly important because it serves as a cursory check of the Army's assessment. By refining the risk factors using our approach, the decision maker would be in a better position to develop and price the associated Real Options with a higher degree of confidence.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. SUMMARY

A. OVERVIEW

In the Department of Defense (DoD), the typical outcome of a software acquisition program has been massive cost-escalation, slipping planned delivery dates and making major cuts in the planned software functionality to guarantee program success. To counter this dilemma, the DoD put forth a new weapons acquisition policy in 2003 based on an evolutionary acquisition approach to foster increased efficiency while building flexibility in the acquisition process. However, the evolutionary acquisition approach relies on the spiral development process, which assumes the end-state requirements are known at the inception of the development process, which is a misrepresentation of reality in the acquisition of DoD software-intensive weapons systems. Hence the need for a framework as proposed in our study at a higher level of abstraction (acquisition decision-making level) aimed at mitigating risks associated with the technology objectives, constraints, and alternatives as put forth by the customer.

Our approach addresses these issues by taking a proactive/preemptive approach to risk management by planning and paying for risks up front. This is not to say that risk management strategies are not being adopted today, but rather a failure of management to take a strategic approach towards risk management. The status quo today is to employ reactive risk management strategies that often result in the reduction of much needed functionality from the scope of the software investment effort. Therefore we proposed a more proactive decision making-framework to guide management in the form of our Real Options framework.

The Real Options approach is based on the concepts of financial options theory, and in our study we have shown how it could be used as proactive risk management tool that could be employed at the strategic decision-making level (executive level) pre-acquisition, further complementing the spiral development approach at the "tactical level". The Real Options approach builds on several tried and proven approaches of management. In this study we have shown using the U.S. Army Future Combat Systems

program as an example that the traditional Real Options methodology, when enhanced and properly formulated around a proposed or existing software-investment, could provide a framework for guiding software acquisition decision-making by highlighting the strategic importance of managerial flexibility. This flexibility offers management the ability to balance the satisfaction of a customer's requirements within the realms of the associated cost and schedule constraints thus validating our hypothesis by developing the appropriated options during the acquisition decision making phase and executing the options when it becomes optimal to do so.

B. FINANCIAL MODEL AND ASSET VALUATION

To validate our hypothesis, we developed correlations and establish compliance between the established Real Options pre-conditions and the software investment decision-making process. While we were able to establish compliance with the Real Options pre-conditions, this was not an easy task, mostly due to lack of detailed data -- when data was available, it was not available at the level of granularity which we would have desired. Consequently, we had to make some assumptions regarding the data in order to validate our hypothesis. However, we believe that our hypothesis would still hold true when employed with real "data".

The main Real Options pre-condition which we had to make assumptions about was the precondition of the existence of a basic financial model. While we were able to create a financial model based on our beliefs and intuitions, the lack of a comprehensive methodology for valuing the returns of a potential DoD software investment proved problematic in valuing the software asset. Further compounding the problem was a lack of the clear delineation between software and hardware from both a cost and realized benefits perspective.

C. UNCERTAINTY IDENTIFICATION AND RISK QUANTIFICATION

The second most import pre-condition of Real Options, is the existence of uncertainties. Uncertainties must be present, otherwise the Real Options approach becomes useless. We focus on this issue in our study because uncertainties have the

consequence of introducing risks to the software investment effort, and the risk must be properly quantified, because it is a key factor in determining the price of the options we would need to create to hedge against the risk. Thus by properly quantifying risk, we are increasing our degree of confidence in the volatility of our software investment effort and consequently the price of the Option being developed. To satisfy this precondition, we proposed the Uncertainty Elicitation Model based on our "2**T**" approach explicitly aimed at capturing uncertainty. This model introduces an explicit, uncertainty elicitation phase after the requirements elicitation phase and captures uncertainties along both managerial and technical dimensions. Once these uncertainties are captured, we quantify the uncertainties as risks by developing subjective probability estimates and modeling them stochastically using a Monte Carlo simulation, in order to determine the overall volatility of the returns of the software investment effort.

Next we proposed the use of the Dempster-Shaffer Theory of Evidence (DST) as a volatility refinement technique to further refine our initial volatility estimates due to its ability to address epistemic uncertainties and reflect ignorance in the risk probability estimates. To satisfy the constraint of the independence of the sources of information imposed by DST, we utilized the data provided by two independent sources, the Cost Analysis Improvement Group (CAIG) and the Institute for Defense Analysis (IDA), to improve our initial volatility estimates. We formulated the appropriate Real Options in response to the risks using the revised volatility estimates obtained using the DST methodology thereby satisfying precondition 4, which calls for the ability of management to have the flexibility or option to make mid course corrections when actively managing the project. The Real Options are presented in a strategy tree, to highlight all the possible strategic pathways for that could be employed to address the risk at hand.

Given the uncertainties and risks we identified in the FCS program using our uncertainty elicitation model, we proposed two "call" options based on a scenario which assumed that of the six FCS component systems, one is not facing uncertainty while the remaining five software components were facing uncertainty. The proposed "call" options allowed the software executive to either "defer" the development of all the components of the FCS Network to include the single system that is not facing

uncertainty or employ a deferment and switching approach, by deferring development of the system facing uncertainty until uncertainty is resolve and at the same time continuing the development of the remaining five components.

D. OPTION PRICING

To price the Real Options created to hedge against risk, we modeled the options reflected in our strategy tree in a simulator called the Super Lattice Solver 3.0 provided by Real Options Valuation Inc.

We input all the necessary parameters to include the key parameters of volatility and asset value which we had computed earlier on during our analysis and elected to use the binomial lattice approach to value our option. It must be noted that while the Black and Scholes method is probably the most popular options pricing approach, we choose the binomial lattice approach because of its ability to account for American style options, which is an advantage over the Black and Scholes approach. It is also important to emphasize that while there are several types of options, labeled after geographic regions (e.g., American, European and Bermudian options). The name of these options are just labels for the options and do not imply geographical restrictions. An American style option can be used in Europe, just as a European style option could be used in the United States. The names associated with each of these options only have to do with when they may be exercised by the option holder.

Associated with the task of pricing the options is also the timing of the execution of the option, with the goal being to find the optimal revenue drivers, which is where we further make the assumption, that under the laws of rational judgment, management would time and execute the options when it is optimal to do so thereby satisfying precondition 5, which addresses the issue of managerial judgment from a competence perspective requiring that management must be smart enough to execute the Real Options when it becomes optimal to do so.

E. CONCLUSION

Uncertainties associated with software-related capital investments lead to unnecessary and sometime preventable risks. Since DoD often sets optimistic requirements for weapon programs that require new and unproven technologies, the application of the Real Options methodology would be beneficial as it would enable DoD to incorporate the appropriate "Options" into the acquisition contracts. Barring the use of an explicit uncertainty elicitation phase as proposed in our research and the development of "Options" to hedge against the risk as they appear, we believe the current acquisition process would continue to be plagued by the risks of cost and schedule overrun. By employing our proposed approach, DoD would be able to optimize the value of their strategic investment decisions by evaluating several decision paths under certain conditions to lead to the optimal investment strategy.

As it stands today, all the proposals put forth by the Congressional Budget Office (CBO) call for the reduction of the overall scope of the FCS program, and while we cannot affirm or disprove their recommendations due to the limited data we had at the time of our study, we believe that the DoD can benefit from exploring our approach by utilizing precise data with our framework to examine the credibility of the recommendations put forth by CBO. Having validated our proposed framework (subject to the assumptions we made), we believe that if the Real Options methodology had been applied to the FCS program from the onset by acquiring the right to execute switching options (within the constraints of our assumptions) to hedge against risks, the FCS program would most likely not be facing the current calls for the reduction of the overall scope of the FCS program due to cost escalation resulting from the inadequate upfront risk management planning prior to the investment decision. Hence we believe our framework could serve as basis for future work in terms of the expansion of its capabilities to address risk and decision-making within the software engineering domain.

F. FUTURE WORK

As part of the future work in connection with this research, we would like to formalize and create an automated software acquisition decision-making tool explicitly

aimed at managing the risks associated with software-related capital investments using out Real Options approach. Specifically, we would like to gather historical information on previously completed software acquisition programs depicting the number of requirements planned at the onset of the acquisition effort and the number of requirements delivered at the end of the software acquisition effort, as well as the associated cost and schedule information for each of the acquisition programs. We would use all of this data to establish a repository of historical programs which would serve as a basis of comparison with current/future acquisition programs to help provide some insight into the issue of requirements volatility and its associated impact on cost and schedule overruns. By gathering historical information into a centralized repository, we hope to alleviate the assumptions we made in our study due to the data gathering problems we encountered in this study. We would incorporate the DST volatility refinement technique into our software tool and link our automated software acquisition decision making tool to the repository containing historical data of previously completed software acquisition programs to provide a one "stop-shop" modeling toolkit to better facilitate the acquisition decision making process.

LIST OF REFERENCES

[1] Ian Sommerville, Software Engineering, Seventh Edition,. p. 6, 2004.

[2] Hadra Ziv, Debra Richardson, "*The Uncertainty Principle In Software Engineering"* Submitted To ICSE '97, 19th International Conference On Software Engineering, August 1996. Retrieved November 10, 2007 from http://Jeffsutherland.Org/Papers/Zivchaos.pdf.

[3] Rangaswami, Ken Berryman, "*Unifying The Ecosystem*", Software 2006 Industry Report, Sand Hill Group & McKinsey, Software 2006. Retrieved August 26, 2007 from http://www.sandhill.com/conferences/sw2006_materials/SW2006_Industry_Report.pdf.

[4] The Importance of Software in our Society http://en.perseidestech.net/index.php/le_reve/les_logiciels_libres/l_importance_des_logiciels, Perséides Technologie, 2008. Last accessed May 11, 2008.

[5] Elizabeth Starrett, "*Software Acquisition in the Army*", Crosstalk Journal of Defense Software Engineering, May 2007. Volume 20 No 5.

[6] Michael J. Mauboussin, "*Get Real: Using Real Options In Security Analysis*", Credit Suisse First Boston Corporation, June 23,1999. Retrieved December 10, 2007 from http://www.capatcolumbia.com/Articles/FoFinance/Fof10.pdf.

[7] Patience Wait, "*Weapons Projects Misfire On Software*", Government Computer News, Vol. 25 No. 18**,** March 6, 2007**.**

[8] Boehm, Boehm. and K.J. Sullivan. "*Software Engineering Economics: A Roadmap*." The Future of Software Engineering, ed. A. Finkelstein. 2000. ACM Press.

[9] Dixit, A.K. and Pindyck, R.S., "*The options approach to capital investment*", Harvard Business Review, May-June 1995.

[10] Craig Fields, "*Task Force On Defense Software*", Defense Science Board, November 2000. Retrieved January 15, 2008 from http://www.acq.osd.mil/dsb/reports/acqreformfoursub.pdf.

[11] Craig Myers, Patricia Oberndorf, "*Managing Software Acquisition: Open Systems and COTS product*". Software Engineering Institute Series, 2001.

[12] Richard K. Sylvester and Joseph A. Ferrara, "*Conflict And Ambiguity Implementing Evolutionary Acquisition*", Acquisition quarterly review, Winter 2003. Retrieved February 1, 2007 from http://www.dau.mil/pubs/arq/2003arq/Sylvesterwt3.pdf.

[13] Victor Fey, Norman Bodine, Eugene Rivin, " *Strategy for Effective Investment*" White Paper, TRIZ GROUP, 2000. Retrieved February 1, 2007 from http://www.trizgroup.com/articles/StrategyforEffectiveTechInvestment.pdf.

[14] Jonathan Mun, "*Real Options Analysis*" Second Edition, Wiley, 2006.

[15] Wikipedia, "*Net Present Value*" Last Accessed June 30, 2008. http://en.wikipedia.org/wiki/Net_Present_Value.

[16] *Tools for Decision Analysis: Analysis of Risky Decisions*, Last Accessed June 30, 2008 from http://home.ubalt.edu/ntsbarsh/opre640a/partIX.htm#reodam.

[17] *Net Present Value*, Last Accessed June 30, 2008. http://www.algebra.com/algebra/homework/Finance/Net_PresentValue.wikipedia.

[18] Utility, Principia Cybernetica Web, Web Dictionary of Cybernetics and Systems, Last Accessed June 30, 2008. http://pespmc1.vub.ac.be/ASC/UTILITY.html.

[19] James Alleman "*Real Options: Overview*", University of Colorado & PHB Hagler Bailly, Inc, 1999. Retrieved February 1, 2007 from http://www.colorado.edu/engineering/alleman/print_files/real-options-slides.pdf.

[20] Eduardo S. Schwartz[1] and Carlos Zozaya-Gorostiza[2]"*Valuation of Information Technology Investments as Real Options*", University of California, Los Angeles Anderson Graduate School of Management, [2]Instituto Tecnológico Autónomo de México, Feb 2000. Retrieved February 1, 2007 from http://www.realoptions.org/papers2000/ZozayaSchwartz.pdf.

[21] Sven Ove Hansson "*Decision Theory, A brief Introduction*" Department of Philosophy and the History of Technology, Royal Institute of Technology (KTH), Stockholm, August 1994. Retrieved February 1, 2007 from http://www.infra.kth.se/~soh/decisiontheory.pdf.

[22] Bell DE, Raiffa H. Tversky A, "*Decision Making: Descriptive, normative, and prescriptive interactions*". Cambridge, UK: Cambridge University Press; 1988. 562-568.

[23] *Dempster Shaffer Theory*, Last Accessed August 20, 2008 http://en.wikipedia.org/wiki/Dempster-Shafer.

[24] Rami Bahsoon, "*Evaluating Architectural Stability with Real Options Theory"*, PhD Dissertation, Faculty of Engineering Sciences, University of London, U.K., October 2005. Retrieved December 12, 2006 from http://www-users.aston.ac.uk/~bahsoonr/BahsoonThesisFinal2005.pdf.

[25] KS96 Kevin J. Sullivan. "*Software Design: The Options Approach*", ACM 1996.

[26] JM06 Johnathan Mun, "*Real Options Analysis Versus Traditional DCF Valuation in Layman's Terms*"Real Options Valuation, Copyright 2006.

[27] Hakan Erdogmus, "*Valuation of Complex Options in Software Development*," ICSE'99 Workshop on Economics Driven Software Engineering Research (EDSER1), Los Angeles, CA, May 17, 1999.

[28] Mousumi Bhattacharya, Patrick M. Wright, "*Managing Human Assets in an Uncertain World: Applying Real Options Theory to HRM*", Working Paper 04 – 03, Cornell Center For Advanced Human Resource Studies March 2004.

[29] R.V. Field, Jr.†, A. Urbina S.F. Wojtkiewicz†, M. S. Eldred and J.R. Red-Horse, "*Uncertainty Quantification In Large Computational Engineering Models*", American Institute of Aeronautics and Astronautics", 2001.

[30] *Alex Dekhtyar*, Jane Hayes, Judy Goldsmith, "*Uncertainty as a source for knowledge transfer*", *position paper*, First International Workshop on Living with Uncertainty in Software Engineering (IWLU'2007), Atlanta, GA, November 2007.

[31] Svetlana V., Poroseva, Julie Letschert** and M. Yousuff Hussaini , "*Application Of Evidence Theory To Quantify Uncertainty In Forecast Of Hurricane Path*", American Meteorological Society 86th Annual Meeting 29 January-2 February 2006 (Atlanta, GA).

[32] *Uncertainty Quantification*, Last Accessed June 23, 2008. http://en.wikipedia.org/wiki/Uncertainty_quantification.

[33] Phillip A. Laplante, Colin J. Neill, "*Uncertainty: A Meta-Property of Software*," sew,pp.228-233, 29th Annual IEEE/NASA Software Engineering Workshop, 2005.

[34] Daniel P. Johnson "*Multiplying Uncertainty, Uncertainties in Cost Estimation*", From a series of occasional essays on topics in research and development. 2005.

[35] Neal Sample, Pedram Keyani, Gio Wiederhold, "*Scheduling Under Uncertainty: Planning for the Ubiquitous Grid*", Fifth International Conference on Coordination Models and Languages, 2002.

[36] Chapter 15: Software Design Condensed GSAM Handbook, 2003. Retrieved September 23, 2007. http://www.stsc.hill.af.mil/resources/tech_docs/gsam4/chap15.pdf.

[37] *Cone of Uncertainty*, Last Accessed April 20, 2008. http://www.construx.com/Page.aspx?hid=1648, 2008.

[38] Futrell R.T., Shafer D.F., Shafer L.I.: "*Scheduling uncertainties (Quality Software Project Management*", Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[39] David Anderson, "*Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results (Coad Series)*", Prentice Hall 2004.

[40] Brian P. Gallagher, "*Software Acquisition Risk Management Key Process Area (KPA)—A Guidebook*," CMU/SEI-99-HB-001, Version 1.02 October 1999.

[41] Daniel Berry ,"*Academic Legitimacy of the Software Engineering Profession*" SEI/CMU Technical Report 92-TR-034;1992, p. 38.

[42] *Pivotal Insights*, Last Accessed January 13, 2008. http://www.pivotal-insight.com/insight/august2005/program_management.php.

[43] Mathiassen, T. Seewaldt, and J. Stage. In: J. Stage et. al. (Eds.). "*Prototyping and Specifying-Principles and Practices of a Mixed Approach*". Quality Software-Concepts and Tools. Aalborg University, 1994.

[44] Mathiassen and J. Stage. "*The Principle of Limited Reduction in Software Design*". Information, Technology and People, Vol. 6, No. 2, 1992.

[45] Elaine Hall, "*Managing Risks: Methods for software systems development*", SEI Seires in Software Engineering.

[46] Kathleen Hevert, "*Real Options: Valuing Flexibility in Strategic Investments*" Last Accessed March 3, 3007 from http://www.babsoninsight.com/contentmgr/showdetails.php/id/116 Babson College 2008.

[47] Barry Boehm, Chris Abts, Sunita Chulani, "*Software Development Cost Estimation Approaches – A Survey*" University of Southern California Los Angeles, CA 90089-0781.

[48] Lynn M Stuter,*"Delphi Technique*, *What is it ?",* Retrieved March 3, 3007 from http://www.learn-usa.com/transformation_process/acf001.htm.

[49] Stefan Arnborg**,** Kungliga Tekniska Hogskolan "*Robust Bayesianism: Relation to Evidence Theory*", Journal of Advances in Information Fusion Vol. 1, No. 1 July 2006.

[50] Kari Sentz, "Combination of Evidence in Dempster-Shafer Theory", PhD Dissertation, Systems Science and Industrial Engineering Department Thomas J. Watson School of Engineering and Applied Science Binghamton University, 2002.

[51] Glenn Shafer, " *The Art Of Causal Conjecture, Belief Functions Introduction*" Chapter 7, 1996.

[52] Hakan Erdogmus and Jennifer Vandergraaf, "*Quantitative Approaches for Assessing the Value of COTS-centric Development*" Institute for Information Technology, Software Engineering Group National Research Council of Canada, 28 October 2004.

[53] James R. White, "*IRS Guidance on Economic Analyses in Investment Business Cases Guidance on Economic Analyses*", United States General Accounting Office.

[54] Aswath Damodaran, "*Tools and Techniques for Determining the Value of an Asset*, " Second Edition, Chapter 29.

[55] *Options,* Last Accessed April 20, 2008. http://www.investorsobserver.com/Docs/SP/whatis.pdf.

[56] Binomial Pricing Options, Last Accessed April 20, 2008. http://en.wikipedia.org/wiki/Binomial_options_pricing_model.

[57] Congressional Budget Office, CBO TESTIMONY before the Subcommittee on Tactical Air and Land Forces Committee on Armed Services U.S. House of Representatives, Statement of J. Michael Gilmore, Assistant Director The Army's Future Combat Systems Program, April 4, 2006.

[58] Government Accounting Office, "*2009 Is a Critical Juncture for the Army's Future Combat System*" GAO Report 08-408.

[59] Government Accounting Office, "*Defense Acquisitions, Report to Congressional committees: Significant Challenges Ahead in Developing and Demonstrating Future Combat System's Network and Software*", GAO Report 08-409, March 2008.

[60] Congressional Budget Office based on data from the Department of the Army and U.S. Army, Project Manager, Combat Systems, "*Combat Systems: Where We Are, Where We Are Going*", Briefing at the National Defense Industrial Association Combat Vehicles Conference, September 22, 2005).

[61] Andrew Feickert ,Congressional Reporting Service report to Congress The Army's Future Combat System (FCS): Background and Issues for Congress Updated October 11, 2007.

[62] Government Accounting Office, "*Defense Acquisitions, Assessments Selected Weapon Programs*", Report to congressional committees. GAO Report 08-467sp. March 2008.

[63] Government Accounting Office, "*Defense Acquisitions, Testimony Before the Subcommittee on Tactical Air and Land Forces, Committee on Armed Services, House of Representatives, The Army's Future Combat Systems' Features, Risks, and Alternatives*" Statement of Paul L. Francis, Director, Acquisition and Sourcing Management, April 2004. GAO Report 04-635T.

[64] Government Accounting Office, "*Defense Acquisitions, Report to Congressional Committees, Key Decisions to Be Made on Future Combat System*", GAO Report 07-376.

[65] Pamela Hess, "Army Must Afford FCS", Pentagon Correspondent Washington, (UPI) Aug 15, 2006, Last Accessed January 24, 2008. http://www.spacewar.com/reports/Army_Must_Afford_FCS_999.html.

[66] A Congressional Budget Office Study, "The Army's Future Combat Systems Program and Alternatives, August 2006.

[67] G.P. Kulk, C. Verhoef, "*Quantifying requirements volatility effects*", Science of Computer Programming 2008.

[68] Evgenia Vogiatzi, "*Problems in Regression Analysis and their Corrections*", Retrieved October 5, 2008 from http://www.geocities.com/qecon2002/founda10.html, 2002.

[69] *Least Square Methods,* Last Accessed October 2, 2008. http://en.wikipedia.org/wiki/Least_squares.

[70] *Durbin-Watson Statistic*, Last Accessed October 2, 2008. http://en.wikipedia.org/wiki/Durbin%E2%80%93Watson_statistic.

[71] *Dempster Shafer Theory*, Last Accessed October 2, 2008. http://individual.utoronto.ca/weisberg/phl2110/Readings/Dempster_Shafer.pdf.

[72] *The boxer, the wrestler, and the coin flip: A paradox of robust Bayesian inference and belief functions*, Andrew Gelman, Last Accessed October 2, 2008. http://www.stat.columbia.edu/~gelman/research/published/augie4.pdf

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Col William T. Cahoon
   459th ARW/CC
   U.S. Air Force Reserve
   Andrews AFB, MD 20762

4. Dr. Peter Denning
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

5. Dr. Man-Tak Shing
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

6. Dr. Johnathan Mun
   Department of Information Sciences
   Naval Postgraduate School
   Monterey, California

7. Dr. Mikhail Auguston
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

8. Dr. J. Bret Michael
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

9. Dr. Tarek Abdel-Hamid
   Department of Information Sciences
   Naval Postgraduate School
   Monterey, California

10. RADM James B. Greene, USN (Ret)
Graduate School of Business and Public Policy
Naval Postgraduate School
Monterey, California

11. Dr. Keith Snider
Graduate School of Business and Public Policy
Naval Postgraduate School
Monterey, California

12. Ms. Karey Shaffer
Graduate School of Business and Public Policy
Naval Postgraduate School
Monterey, California

13. Dr. Richard Riehle
Department of Computer Science
Naval Postgraduate School
Monterey, California

14. Dr. Bert Lundy
Department of Computer Science
Naval Postgraduate School
Monterey, California

15. Dr. Daniel A. Nussbaum
Department of Operations Research
Naval Postgraduate School
Monterey, California

16. Dr. Thomas Housel
Department of Information Sciences
Naval Postgraduate School
Monterey, California

17. Dr. John Osmundson
Department of Information Sciences
Naval Postgraduate School
Monterey, California

18. Dr. Carl Jones
Department of Information Sciences
Naval Postgraduate School
Monterey, California

19. Dean S. S. Sritharan
Graduate School of Engineering & Applied Sciences
Naval Postgraduate School
Monterey , California

20. LTC Thomas S. Cook
Department of Electrical Engineering and Computer Science
United States Military Academy
West Point, New York

21. Capt. Albert Olagbemiro
22 Laurence Brooke Road
Catonsville, Maryland