12 October 2002

Application of the Systems Engineering Process on High-Risk Programs

By

Kurt A. Himmelreich

A MASTER OF ENGINEERING REPORT

Submitted to the College of Engineering at
Texas Tech University
In Partial Fulfillment of the
Requirements for the Degree of:

MASTER OF ENGINEERING

Approved By

Dr. Atila Ertas

Dr. Tim Maxwell

Dr. Murat Tanik

Dr. John Borrelli

ABSTRACT

Many times, systems engineers find themselves addressing customer needs where available technology does not adequately resolve all goals and constraints. The unique nature of programs seeking the advancement of technology, where unpredictable issues are almost inevitable, often renders them unsuitable for cookbook program execution processes. This paper explores the fundamentals of systems engineering and its application in particularly challenging technical endeavors. The trials and lessons learned of past programs are used to support findings. The work concludes with recommendations on how to better utilize, understand, and adapt systems engineering processes on "high-risk" programs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DISCLAIMER

To avoid proprietary, classified, or otherwise sensitive information, systems engineering is addressed herein from a generic sense based primarily on open literature. Although some specific references are provided in support of findings, the opinions and conclusions expressed in this report are strictly those of the author and do not necessarily reflect the views or policies of Raytheon, Texas Tech University, or any other associated or referenced entity.

CHAPTER - I
1. INTRODUCTION

1.1 Purpose

The art of engineering has been around as long as mankind itself. We marvel at the accomplishments of our current civilizations and those that preceded us. Despite the triumphs, and the wealth of information and tools now available to us, many engineering endeavors still fail. Indeed, very few programs execute without some degree of setbacks. The application of the systems engineering process in such programs is critical to recovery and to the ultimate success of the project.

The process of systems engineering has been defined and refined in considerable detail in the academic and corporate arenas. An internet search of "systems engineering" will return hundreds of thousands of websites. Many thousands of websites can even be found for "systems engineering process." With all the standards and procedures and processes, why do so many programs still experience difficulties?

This study examines the use of the systems engineering process on programs where technical and programmatic setbacks are possible and even probable. Such programs are addressed herein as "high risk" programs. These may include a variety of engineering efforts, including technology development, new applications, etc.

For high-risk programs, it will be shown that systems engineering processes must be applied and controlled with different emphasis than is the norm for routine projects. The fundamentals of these changes are identified and substantiated.

This document contains the required content of the Master's Report for Texas Tech University's Master of Engineering program. It is respectfully submitted in accordance with degree requirements.

1.2 Scope

The heart of this study is the assessment of technical and process challenges associated with high-risk development programs. Both general and specific examples are cited. However, findings and results remain qualitative in nature, as the focus is on the *relationship* between common development obstacles and the fundamental engineering principles and processes used to mitigate them.

1.3 Objectives

The objective of this work is to identify the most important elements of systems engineering processes on challenging development programs and determine means for improvement. Key areas that are addressed include:

- The Fundamentals of Systems Engineering
- A Case Study of a High-Risk Program
- Lessons Learned from Other Programs
- Root-Cause Analysis of Obstacles
- Recommendations for Improvement

Problems associated with systems engineering are examined to produce recommendations applicable to both the given case study program, and to the engineering process in general.

That said, the underlying goal is to gain knowledge that will further the student's ability to lead technical development programs. A topic of generic nature was chosen in hopes that the work may be of interest and value to any systems engineering student.

1.4 Executive Summary

In the most generic sense, engineering processes usually provide a framework that *allows* success. However, their use sometimes emphasizes "how" at the expense of "what" and "why". The intelligence and

discipline that effect “how” a program will be run are great enablers, but also require vision and passion to successfully overcome the challenges of difficult developments.

Considerable information is presented in this report. While every attempt has been made to maintain a logical and understandable flow, remembering the fundamentals will aid in the understanding of conclusions and formulations. The basics of engineering are always important, but they are paramount in situations where interim setbacks are commonplace.

CHAPTER - II
2. BACKGROUND

2.1 The Systems Engineering Process

What is systems engineering? What are the goals of the process? These are obvious questions, but the nature of responses can vary greatly. In general, engineering processes are intended to minimize program risk during execution. Their implementation is generally based on a combination of successes and lessons learned, attempting to script a recipe for duplicating successful programs. Of course, no two programs are alike. Resources, goals, technology, and the business environment all change. While processes attempt to encompass programmatic differences, the widely varying and unpredictable nature of the subject high-risk programs often results in process failures.

If you don't know where you are going, you will wind up somewhere else. - Yogi Berra

The following paragraphs provide an overview of the history and evolution of the Systems Engineering process, its current status, and expectations for the future. Systems engineering will be examined both as a process and as a job function. Information is limited to open literature. Raytheon proprietary processes and elements are not included.

2.1.1 Evolution

The American Heritage Dictionary defines engineering as, "The application of scientific and mathematical principles to practical ends." Webster's College Dictionary says it is "the practical application of science and mathematics." While there is reasonable agreement on this definition, they offer little guidance to the systems engineer, using a circular definition of "an engineer who specializes in the design and implementation of production systems" [Webster's College Dictionary].

A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, and die gallantly. Specialization is for insects. - Robert A. Heinlein

The International Council on Systems Engineering (INCOSE) [1] begins the definition of systems engineering as " an interdisciplinary approach and means to enable the realization of successful systems." In addressing why we need systems engineering, Brad Yelland [2] gets to the root of systems engineering as the "simplification of complex systems." This latter definition is far more timeless than most definitions and supports review of the history of systems engineering.

As stated in the opening of this report, engineering in one form or another has been around since the dawn of time. Although it could even be argued that engineering is not limited to mankind, a look at more recent *human* aspects will suffice for this study. Arunski, et al. [3], gives a list of examples of the history of systems engineering in Table 1 below.

Table 1. Systems Engineering Heritage [3]

Water Distribution Systems in Mesopotamia	4000 BC
Irrigation Systems in Egypt	3300 BC
Urban Systems such as Athens, Greece	400 BC
Roman Highway Systems	300 BC
Water Transportation Systems like Erie Canal	1800s
Telephone Systems	1877
Electrical Power Distribution Systems	1880
British Multi-disciplined Team Formed to Analyze Air Defense System	1937
Bell Labs Supported Nike Development	1939-1945
SAGE Air Defense System Defined and Managed by MIT	1951-1980
ATLAS Intercontinental Ballistic Missile Program Managed by Systems Contractor, Ramo-Wooldridge Corp	1954-1964

Yelland [2] adds more recent developments such as the Apollo mission and military standards in the 1960's, the formalization of software engineering in the 1970's, the emergence of computer tools in the 1980's, and Systems Engineering standardization in the 1990's.

Although systems engineering as a practice is clearly nothing new, it is only in the last few decades that it has become recognized as an engineering discipline in itself. The need for this specialization comes primarily from the growth in complexity of modern systems resulting from the vast information stores and technological capabilities at the disposal of the engineering community. Put simply, many development projects simply involve too much information for total comprehension throughout a development team. Tanik and Ertas put it as follows [4]:

> *"Engineering has never been easy. The speed of introduction of new materials, tools, and techniques are increasing. We are approaching a human processing bottleneck for effective use of these inventions…"*

The information must be decomposed and managed in a disciplined fashion. As a result, INCOSE was founded in 1991, and there are now a number of recognized standards. The most common systems engineering standards are listed in Table 2.

Table 2. Common Systems Engineering Standards

Organization – Document Number • **Title/Description**	**Source**
Electronics Industry Association (EIA) – 632 • Processes for Engineering a System	[5]
EIA/Interim Standard (IS) – 731 • Systems Engineering Capability Model	[6]
Institute of Electrical and Electronics Engineers (IEEE) - 1220 • Application and Management of the Systems Engineering (SE) Process	[7]
European Cooperation for Space Standardization (ECSS) -E-10A • System Engineering	[8]
International Standards Organization (ISO) - 15288 • System Life Cycle Processes	[9]
MIL-STD-499 • Systems Engineering Management	[10]
MIL-STD-499A • Engineering Management	[11]

Figure 1 gives the historical timeline for the development of these standards. Early military standards provided good definition of systems engineering principles, but more recent offerings facilitate the application and measurement of systems engineering performance as an engineering discipline of its own.

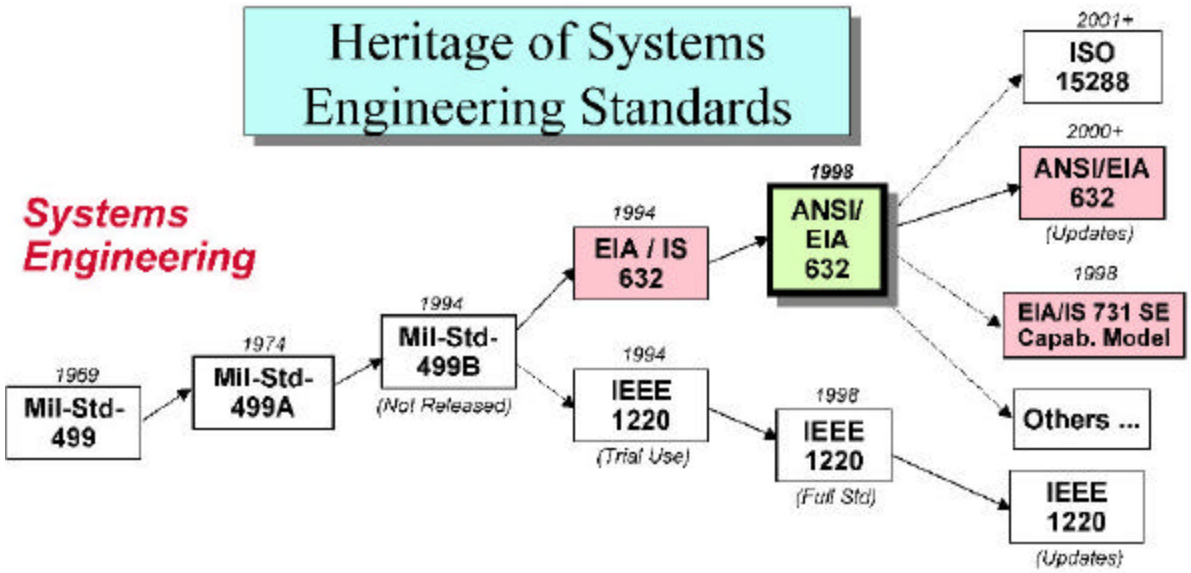


Figure 1. Chronology of Engineering Standards [12]

2.1.2 Overview of Basic Elements

Systems engineering is often introduced in the form of a "V" diagram. The purpose of this approach is to emphasize the symmetry between system definition and system validation. For example, you define the system and partition the subsystems (can be multiple levels), then integrate the subsystems, the system, and verify performance against the definitions. An example "V" diagram is shown in Figure 2.

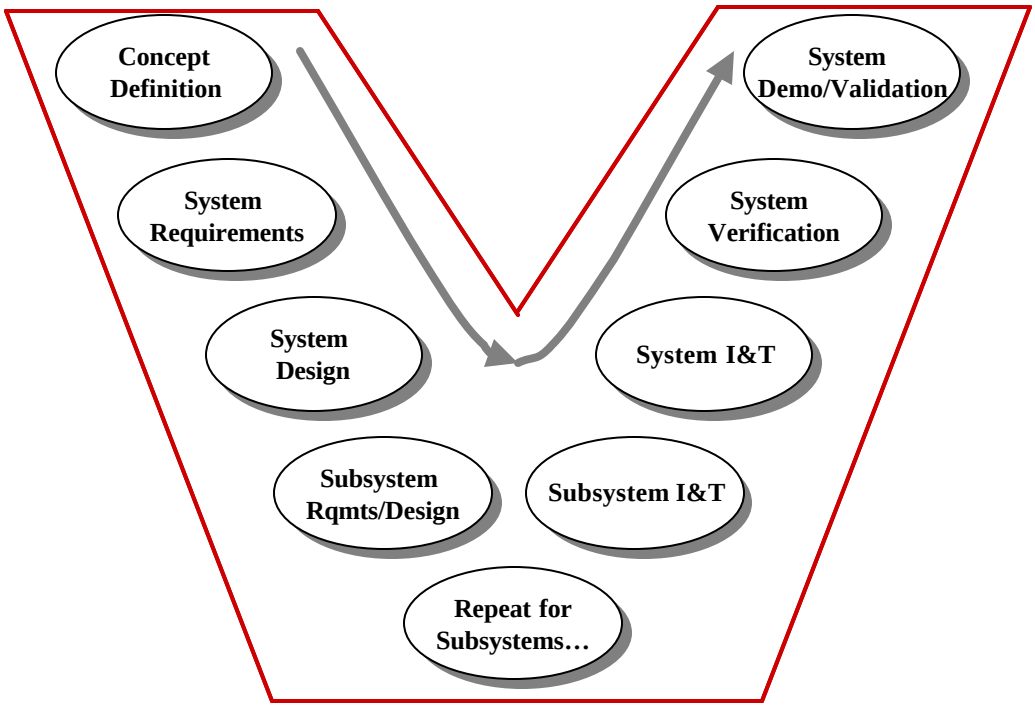


Figure 2. Typical Systems Engineering "V" Diagram

While defining an important aspect of systems engineering, such diagrams do not adequately address many program realities. Much of systems engineering is customer relations, program planning, trade-offs, change management, iteration, etc. Consequently, the systems engineering process requires a more complete definition. This will be achieved through examination of systems engineering job functions in the following section.

2.1.3 Roles and Responsibilities

The relationship of systems engineering with other areas associated with high-risk programs might be described as follows [3]:

- **Science** - Determines what Is
- **Component Engineering** - Determines what Can Be
- **Systems Engineering** - Determines what Should Be

The systems engineer plays a number of roles to ensure that the final product is what it *should be*. These can be categorized in a number of ways, but skills generally include the following:

- *Project Planning – includes process deployment, staffing, controls, etc.*
- *Development Leadership – involves the broad-ranging influence and coordination of teams*
- *Requirements Definition – understanding, capturing, and disseminating goals and constraints*
- *Functional Architecture – establishing the top-level system design based on requirements*
- *Design and Performance Analysis – resolve trade-offs and monitor performance expectations*
- *Integration, Verification and Validation – ensure final product meets stated needs*

The systems engineer is always focused on the end solution and its intended purpose. While one element of this is traditional engineering design, the systems engineer also brings to the table an understanding of corporate/management constraints, teamwork, and perhaps most importantly, a thorough understanding of the customer. This customer understanding is rarely as simple as possession of a document. It more frequently involves knowledge of the customer's culture, including their motivation and even the needs and constraints of the customer's customer.

2.1.4 Future Trends

> *…it is easier to introduce new complications than to resolve the old ones. - Neal Stephenson*

Systems engineering has evolved from a way of logical thinking amongst all "engineers" to an engineering discipline in its own right. As such, many detailed standards, processes and procedures are in place at most system developers. It is now accepted as a field of study at many universities.

At the start of this new millennium, we see the flow of systems engineering principles back into other disciplines. As complexity grows more and more, the top-level systems engineers may not be positioned to make "system" decisions for other disciplines. Instead, the entire design team (or at least a level or two down) must practice good systems engineering techniques. Indeed, part of systems engineering may become the coordinating process for other, discipline-specific development processes.

CMMI (Capability Maturity Model Integration) thrusts are well underway for systems engineering. The purpose of CMMI is to provide guidance for improving processes and the ability to manage the development, acquisition, and maintenance of products and services [13]. This movement is intended to provide a means to measure and evolve systems engineering development processes.

Bill Edwards [14] acknowledges the advances in systems engineering over the last 50 years, but he also points out that it "is so wide and multi-faceted that as of yet there is no applicable single unified approach." He goes on to say of the various models and their relationship to good engineering fundamentals:

> *"…they (the SE process models) are at best approximate representations of the SE effort. Their usefulness depends on how well they help the practitioners understand and solve their problems."*

2.1.5 Study Categories for the Systems Engineering Process

Based on the background provided in the previous paragraphs, the systems engineering process will be parsed into the areas shown in Figure 3 for subsequent analyses. Key activities of each area will be presented with the analysis.

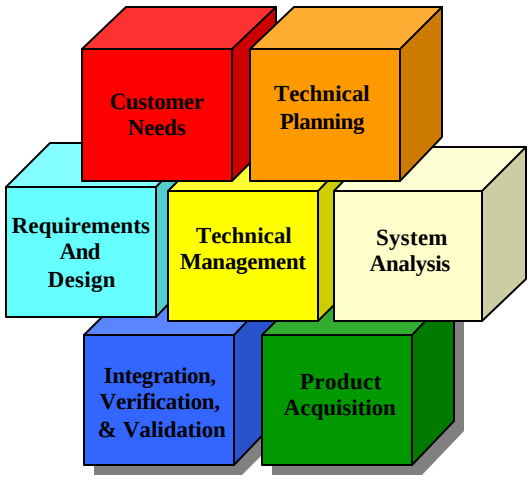


Figure 3. Systems Engineering Functions

The categories in the figure are meant neither to be a complete look at all systems engineering, nor as a rigid organization of team members. Rather, it is a reasonable separation of functions to support the remainder of this study.

2.2 "High Risk Program" Defined

In his paper [15], Dr. Raymond Yeh addresses system development as a "wicked" problem. He defines five symptoms of wicked problems as:

> *"1) The problem formulation and its solution cannot be separated"*
> *"2) There are no rules to determine when a solution is complete"*
> *"3) Symptoms and causes cannot be distinguished"*
> *"4) Wicked problems are substantially unique"*
> *"5) Exhaustive and definite problem formulation are not generally possible"*

Although Yeh is addressing software systems in his paper, his definition of the wicked problem is similar to the "high-risk" programs addressed in this report. Key similarities include:

- Requirements are difficult to define until the system is built
- The root cause o f problems is often difficult to ascertain
- The challenge is new, with limited reuse of product and process
- Even the very need of the customer is hard to nail down

Yeh says the wicked problem is hard to define, hard to plan, hard to manage, hard to schedule, and hard to test. Wicked and high risk indeed! Such a program will be used for the case study and report emphasis.

2.3 Customer Perspective

There are growing process legalities and requirements associated with customer needs and expectations. Engineering organizations are often required to be *ISO-compliant-this* and *SEI-level-that*, just to be *eligible* for a project.

While these aspects impact the systems engineering process, the general nature of most "customers" remains fundamental and almost universal. The customer has a problem, may or may not truly understand the problem, and wants you to fix it. The systems engineer must identify the stakeholders, gather their needs, assess requirements and options, and communicate and iterate. Politics and personal career ambitions may even play a role. Such things must be sorted and managed.

Whenever you find yourself on the side of the majority, it is time to pause and reflect. - Mark Twain

2.4 Case Study Program – Radome De-Icing

Many airborne radar applications require a high likelihood of target detection and very accurate position measurements. These applications may include target recognition, guidance, imaging, traffic control, and flight safety. The precision radars are usually operable in a variety of weather and/or battlefield conditions. Few, however, can withstand the presence of an uncontrolled obstacle immediately in front of the antenna, as is the case with radome ice formation.

The case-study example of systems engineering in a high-risk program is the development of a means to eliminate or compensate for the radio frequency (RF) degradation experienced with radome ice formations in precision airborne radar applications. Radomes are designed to provide environmental protection for the antenna and other electronics with minimal impact to radar emissions. An ideal radome is illustrated on the left side of Figure 4. This ideal radome is effectively transparent at the electromagnetic frequency of interest, causing no perturbation of the wavefront.

The presence of ice on the radome introduces reflection, refraction, diffraction, and loss of RF energy. This can result in an inability to detect objects and a prism-like bending that causes scattering and angular measurement error. This effect is shown on the right in Figure 4. The result is a conflicting set of goals, as the elimination of ice generally requires the introduction of hardware that also degrades radar and aircraft performance.

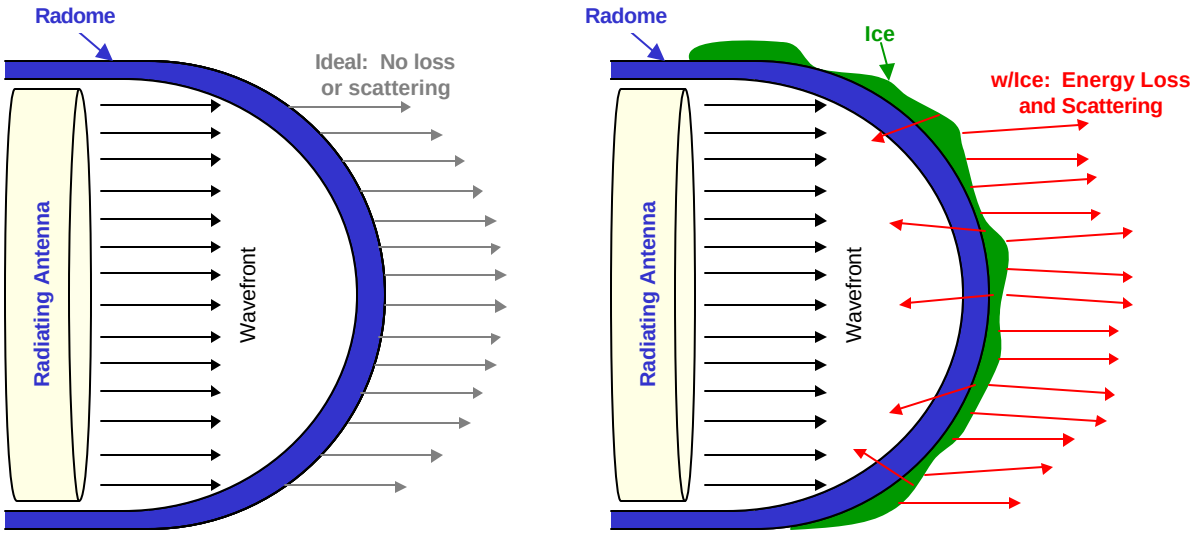


Figure 4. Radome Impact with Ice

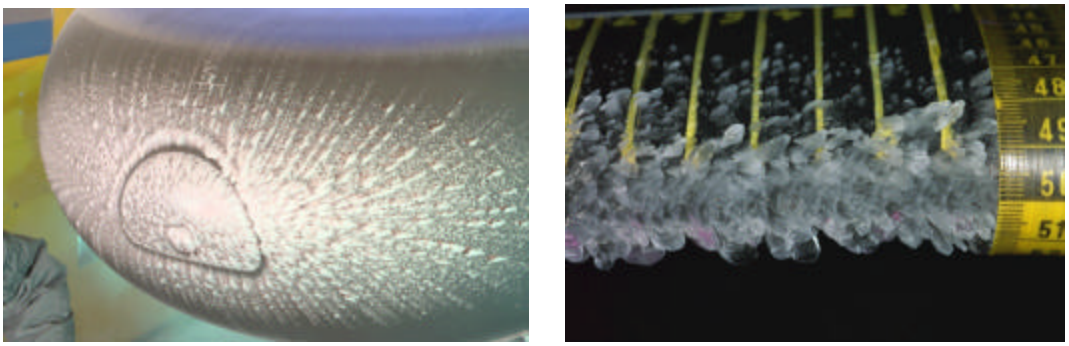
The presence of ice is not new to aircraft or to radomes. However, advances in the capabilities of avionics are making aircrews and battlefield managers increasingly reliant on all-weather sensors such as radar. This trend is expected to increase as unmanned surveillance platforms proliferate the battlefield and as the importance of stealth dictates operation in otherwise poor visibility conditions. A solution to precision radome icing is needed, and a special approach to systems engineering may be necessary to succeed within the various program constraints.

The development of a solution to radome icing is a "high risk" program that will be used as a case study for process analyses in this report.

To avoid inclusion of sensitive information, technical performance parameters are addressed only in a generic sense and will not reflect the actual performance associated with any known Raytheon or customer fielded equipment.

2.4.1 Requirements and Performance Trades

Unlike the familiar "black ice" that often accumulates on roadways and other stationary surfaces, the ice which forms on airborne radomes is effectively unpredictable in its shape and constitution. Radome ice varies with airflow, temperature, moisture density and droplet size. It is not unlike the ice that forms in freezers, but with a more complex and variable shape. Examples of icing from wind-tunnel testing are shown in Figure 5.



[Photos from NASA Glenn Research Center – Cleveland, OH]

Figure 5. Ice Formations on Nose Shape and Wing Edge

While the ice itself has a detrimental effect on radar performance, it is the ice removal equipment that is the subject of development as a case study. Although there are plenty of "outside the box" ideas, most feasible approaches require fundamental change to existing radomes.

The case study involves the retrofit of a fielded system. The base radar system was designed without consideration or requirement to handle ice on the radome. Thus, the radome was designed for optimum RF performance, minimum weight, high reliability, low cost, etc., and such performance information was allocated in the system error budgets to meet customer needs.

The new requirement on the radome has a detrimental affect in virtually all areas. The result is a customer that needs and is willing to pay for the new capability, but has difficulty swallowing the ripple in

system-wide impact. The systems engineer (and the program in general) is faced with conflicting design goals, as feasibility studies indicate that the new de-icing capability can only be achieved with:

- Reduced RF performance
- Higher cost
- Lower Reliability
- Higher Weight
- Higher Power Consumptions
- New and Modified Aircraft Interfaces
- Radar and Aircraft Software Modifications
- Increased Difficulty in Maintenance Actions

Even if the roll-up of these effects to the radar performance was acceptable, it propagates to the next higher level – the aircraft and its mission. A seemingly small, localized problem becomes huge.

2.4.2 Early Approaches

Mitigation of ice on airborne platforms is often addressed in two categories; anti-icing and de-icing. As the name implies, anti-icing systems are designed to prohibit the accumulation of ice. These are often heated surfaces, like the windshield defroster of a car. De-icing systems allow a limited amount of ice to form before removal. These types of systems have been around for decades in the form of inflatable bladders (often called "boots") on leading edge surfaces such as wings. Examples of wing boots are shown in Figure 6. When the wing boots inflate, surface bonds are stressed and the ice is fractured, allowing the pieces to be blown away by the air stream.

[Photos from Leading Edge Aviation - Oklahoma City, OK]

Figure 6. Leading Edge Aircraft De-Icing Boot

2.4.3 New Research and Development

While the physics of ice formation has not changed much, there are some novel approaches to ice removal that have been recently pursued. Some involve the use of advanced chemical formulations that are ill suited for ice formation. These include things such as freezing point depressants (FPDs) that bring the surface freezing point down low enough that atmospheric conditions do not favor ice accumulation (aircraft icing is generally not an issue below -20^{o}C). Other advanced materials exhibit hydrophobic properties such that ice will not stick. Still others use high-shock bursts to shatter the ice.

When the only tool you own is a hammer, every problem begins to resemble a nail - Abraham Maslow

Despite the options available for research and development, anti-icing heating systems and de-icing pneumatic boots remain the most common approaches.

2.4.4 Process Application

The development process for the radome de-icing case study program was managed largely through a "gating" assessment process. This is a series of internal checkpoints during the program where formal assessments are performed (similar to design reviews). Items reviewed include process, design, management, etc. The primary output of the reviews is a risk assessment from specialists and managers outside the program. The underlying program execution process is Raytheon's Integrated Product Development Process (IPDS). Figure 7 shows the eleven assessment gates within the framework of IPDS.

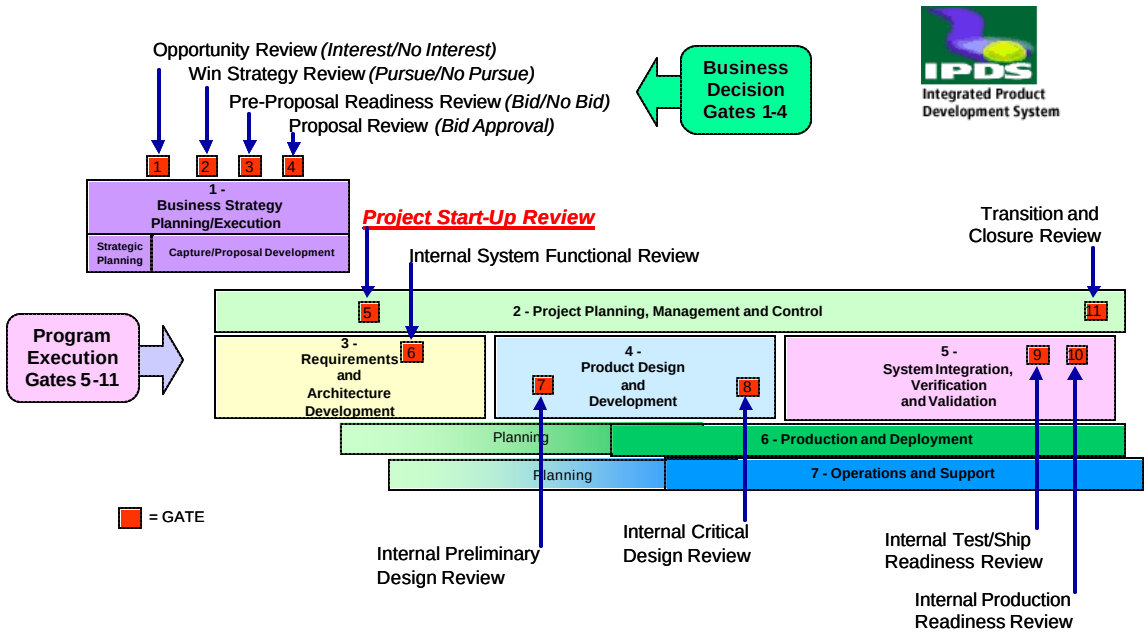


Figure 7. "Gates" within Raytheon's Development Process

2.5 Literature Research

Perhaps the most recent process work applicable to high-risk programs is related to life cycles. The evolutionary life cycle is one often referenced for "high risk" endeavors. The evolutionary model involves successively developing more refined versions of a product. Although it is often unpopular to plan repeated redesigns of a product, this is a necessary evil in many cases. Like any development effort, evolutionary pursuits still follow a set plan toward the desired product functionality and performance. Pertinent benefits include:

- Incorporation of feedback to improve product capabilities
- Elimination of forced and final decisions too early in development that can ultimately result in catastrophic program direction

A simplified illustration of the evolutionary lifecycle is provided below in Figure 8. Addition details on this and other lifecycle representations are included in section 4.4.

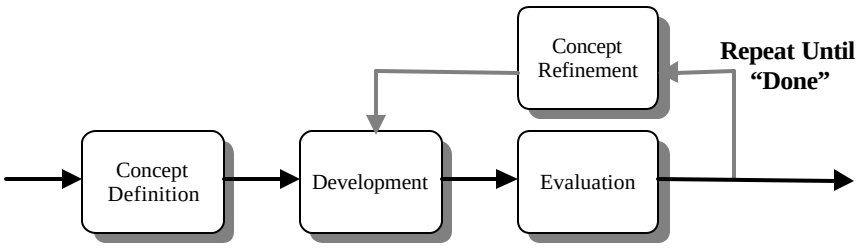


Figure 8. Simplified Evolutionary Life-Cycle Model

The approach to this study spans a broad area of the systems engineering process. Numerous sources have already been cited as background references. There are many papers and books addressing engineering *complex* systems. However, surprisingly few seem to really attack the heart of the problem of engineering in unpredictable (high risk) environments. At this point, it is sufficient to hypothesize that this lack of information might be caused by reasons such as:

- It's hard to write a clean paper when dealing with an inexact outcome
- You can not write a universal script for good engineering
- As defined for this study, many high risk programs span the boundary between engineering and scientific experimentation

The analyses and recommendations that follow in this report will further illustrate the process challenges in this class of development programs, thereby providing a better understanding as to why it is not a popular source of literature.

2.6 Background Summary

This background section has provided (1) a look at Systems Engineering, (2) the definition of High-Risk Programs under evaluation, and (3) a synopsis of the challenges associated with the Radome Icing Case Study program. In the next section, these three elements will be combined to address the key aspects of systems engineering on challenging development programs.

CHAPTER - III
3. PROCESS ANALYSIS

The following analysis is intended to formulate beneficial process changes that supplement process literature and standards with real-world feedback from high-risk programs. This is accomplished by first dissecting the process and the case study program to a common level, then recombining the desired attributes within the format of process categories.

3.1 Process Goals and Content

> *There are two kinds of people, those who do the work and those who take the credit. Try to be in the first group; there is less competition there. - Indira Gandhi*

As discussed in the previous chapter, great work has been done developing the systems engineering process. Basic goals could be stated in many different ways, but they generally include the following interrelated items:

1. Assess alternatives for design – understand risk/benefit trades versus cost, schedule, and performance
2. Provide coordination and motivation between other disciplines
3. Ensure customer needs are met in the final design
4. Promote continuous improvement

These could apply to many endeavors. While it's hard to argue the fundamental goals of most engineering processes, the underlying vision is occasionally lost when specific implementation procedures are developed. It is such areas where the process does not provide sufficient direction and control that are sought in this chapter.

The exact reasons why process definitions and usage do not consistently reflect their underlying goals and principles are beyond the scope of this work. However, the systems engineer should be on guard for such process disconnects, as they are yet another risk to be recognized and managed. Warning signs might include:

- Process groups/managers that are too far removed from the customer and the development task at hand
- Processes that attempt to script a solution to every problem
- The presence of confusing detail resulting from repeated additions of process steps geared toward individual lessons learned cases
- Inconsistent vision and application of process ("too many cooks")

3.2 Mapping of Process Fundamentals and Details

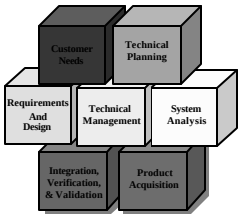
Although detailed process content is not necessary for this report, Raytheon's development process, IPDS, is used as a high-level framework for process evaluation. At the top level, the process is defined as seven stages.

Stage 1. Business Strategy Planning/Execution
Stage 2. Project Planning, Management and Control
Stage 3. Requirements and Architecture Development
Stage 4. Product Design and Development
Stage 5. System Integration, Verification, and Validation
Stage 6. Production and Deployment
Stage 7. Operations and Support

While systems engineering spans all seven stages, engineering *development* activities and related contracts are covered primarily in Stages 2 through 5. Stage 1 is primarily business pursuit and proposal development. Production and support (stages 6 and 7) are often separate contracts.

As presented earlier, the main areas of systems engineering addressed in this study are:

Customer Needs
Technical Planning
Technical Management
Integration, Verification and Validation
System Analysis
Product Acquisition
Requirements and Design



The names of these systems engineering categories are largely self-explanatory. However, it is important to understand the basic systems engineering components clearly for the process analyses later in this chapter. The seven process elements are defined as follows for the remainder of this report:

1. **Customer Needs** – the tasks associated with ascertaining customer and user identities, needs, cultures, goals, priorities, and motivation that will influence program direction and success
2. **Technical Planning** – the efforts related to program organization, characterization, sizing, tailoring, and coordinating in the early stages of development. Plans cover documentation, integration, testing, etc.
3. **Technical Management** – the coordination, monitoring, and control of the various technical pursuits, including management of risk, change, and configurations. Technical management involves both internal development coordination, as well as that of customer expectations and direction
4. **Requirements and Design**– the definition and documentation of various levels of system architecture, functionality, and performance. This includes the planning of requirements verification
5. **System Analysis** – the evaluation of system design trades for performance, cost, and schedule impact. Also includes simulation and modeling of system level operations. System Analysis is often a supporting element of Requirements and Design
6. **Integration, Verification and Validation (IV&V)** – the support and monitoring of integration and test at various levels to ensure requirements, expectations, and constraints are met. Also includes formal verification testing and reporting for the delivered product (generally at the system level) and support for customer operational evaluation and validation
7. **Product Acquisition**– the planning and management of procurement activities, including internal and external suppliers and contracts

These seven categories map into the seven stages of Raytheon's IPDS as shown in Figure 9 below. The illustration of overlap may not be precise for every program, but it is provided to highlight that few aspects of systems engineering stand alone.

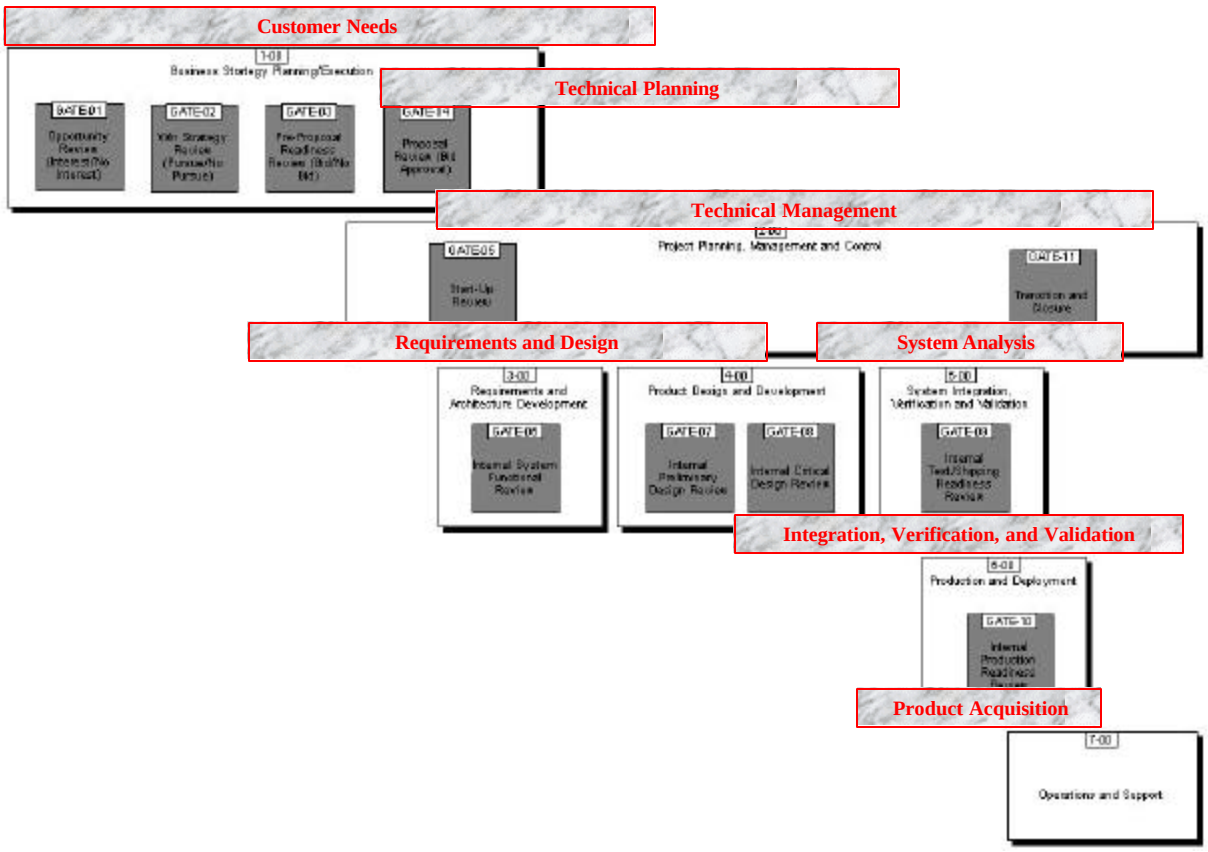


Figure 9. Systems Engineering Study Categories in Raytheon IPDS

Figure 10 summarizes the relationship between the process goals stated in 3.1, the seven systems engineering functions identified for this study, and the approximate timing of the seven stages of Raytheon's development process. The figure captures one omission in this study's systems engineering functionality; that of continuous improvement. Indeed, systems engineers play a considerable role in providing for the future prosperity and efficiency of the company. Their knowledge base must be applied to improve processes and program execution. This link has been left out of the graphic because it is viewed as an all-

encompassing feedback loop that governs (at least partially) the other activities. This very study is essentially part of that feedback process.

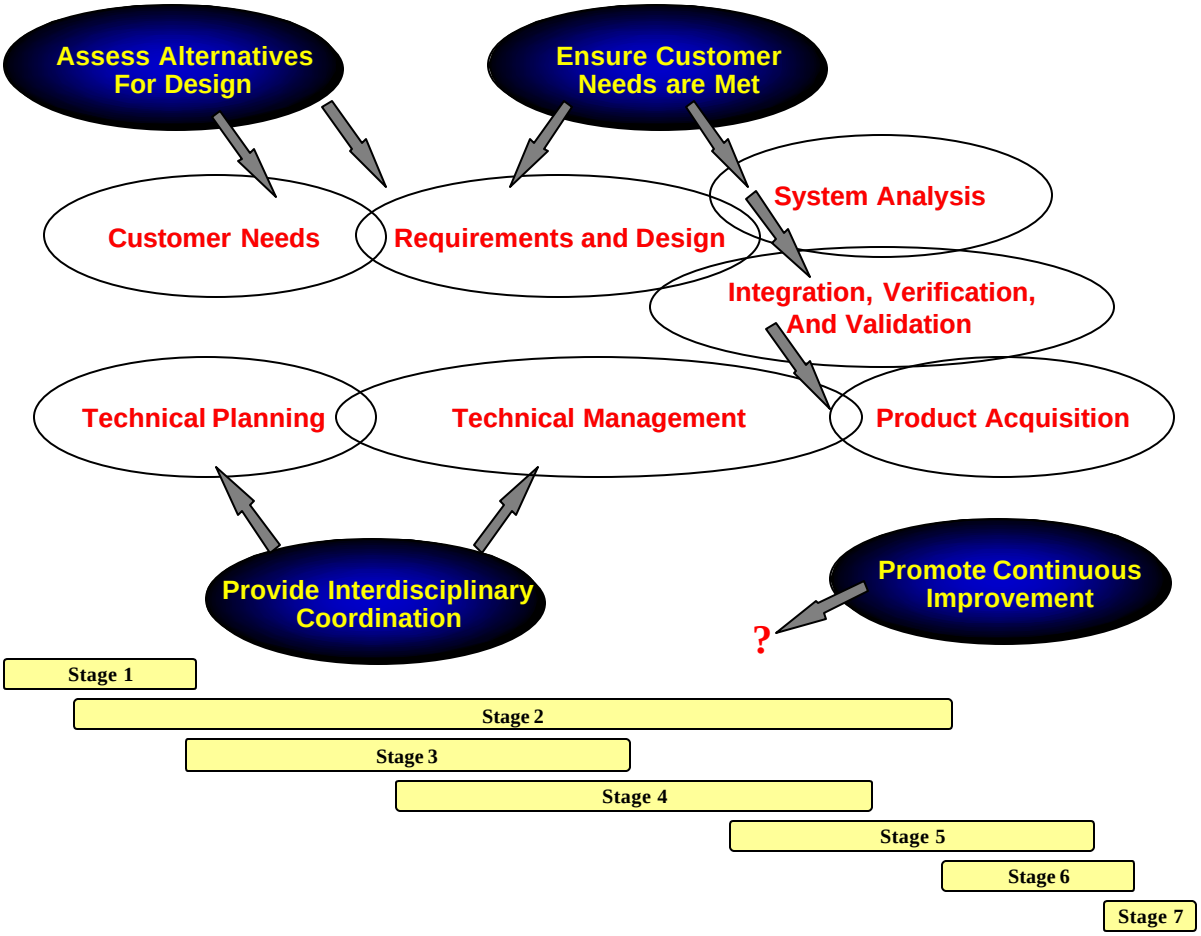


Figure 10. Mapping of Goals to Functions and Development Process

3.3 Case Study Technical Trades

Once the customer acknowledged the need for a solution to radome icing, a trade study was initiated to evaluate alternatives in advance of any proposal or development. Despite a relatively short timeline, the study performed a thorough look at radome options to eliminate ice formations. Existing technologies, as well as many new concepts, were evaluated. Options were scored according to a customer-approved weighting system. This weighting system addressed primarily the "usual suspect" list of program metrics,

such as cost, performance, weight, and power. From the scoring, a short-list of candidates was selected for further pursuit. Table 3 contains a representation of the original study output. Candidates shown were scored from 1 (poor) to 4 (good) for each metric.

Table 3. Initial Case Study Technical Trades

	RF/Radar Perf.	Icing Effectiveness	Technical Risk	NRE $	Recurring $	Schedule	Rel. & Maint.	A/C Impact	Weight	Environ. & Safety	Electrical Power	Total
Weight →	25	15	10	5	5	10	5	10	5	5	5	100
Thin Neoprene Boot	3	2	2	3	3	3	1	3	2	4	3	66.25
Boron Nitride Surface	3	1	1	3	2	2	3	4	4	4	4	65
Pneumatic Teflon Boot	4	1	1	3	3	2	1	3	2	4	3	63.75
Standard Neoprene Boot	1	3	4	4	3	3	1	3	2	4	3	63.75
Electrical Heating – Nichrome Wires, etc.	2	2	2	2	1	2	3	4	4	4	2	60
Cutoff Point for Continued De-Icing Study												
Coatings	Primary Failing → Inability to avoid/shed ice											
Teflon Shell / Sandwich Heating	" → Excessive inlet temperature required for heat delivery											
De-Icing Fluids	" → Inability to accurately control fluid delivery											
Altered Radome Shape	" → Excessive cost and schedule risk											
Artificial Dielectrics	" → Insufficient RF energy available for absorptive heating											
Internal Heating	" → Inadequate heat transfer through radome											
Thinned Radome with Pneumatic Boot	" → Excessive cost and structural risk											
Other Pneumatic Materials	" → Little or no advantage over neoprene materials											
Fluted Radome with Heated Air Circulation	" → Excessive cost and risk of radome mechanical failure											
Scraper – Rotation	" → Complex mechanical structures											
Electrical Heating – FSS	" → High cost and long prototype lead time											
Solid FPD Coating	" → Lack of proven substances											
External Radiation	" → Excessive impact to aircraft											
High-Current Expulsion	" → Poor RF performance											
Expendable Layers	" → Poor maintainability and high reliability risk											
External Heated Air	" → Inability to accurately control delivery											

The exact names and values used to assess technical trades are not important. What was key, however, is the fact that there were no solutions that were highly superior to others. The conflicting requirements/goals were well balanced. Although the highest weights were placed on performance parameters associated with mission success, many candidates were quickly eliminated due to cost and schedule constraints. This was clearly a challenging pursuit.

3.4 Case Study Development Summary

Upon completion of the initial study, considerable effort was placed on simulation and prototyping of the top solution candidates. For risk mitigation, two solutions were pursued. Both involved the removal of ice by pneumatic inflation of a bladder (boot). One solution was based on current neoprene technology used on aircraft wings (refer back to Figure 6). This approach offered low-risk de-icing, but considerable radar performance impact. The other solution involved the use of Teflon, which has good radar performance, but had not been used before for leading-edge aircraft de-icing. Photos of each boot option in the icing wind-tunnel are shown in Figure 11 below (prior to de-icing inflation).

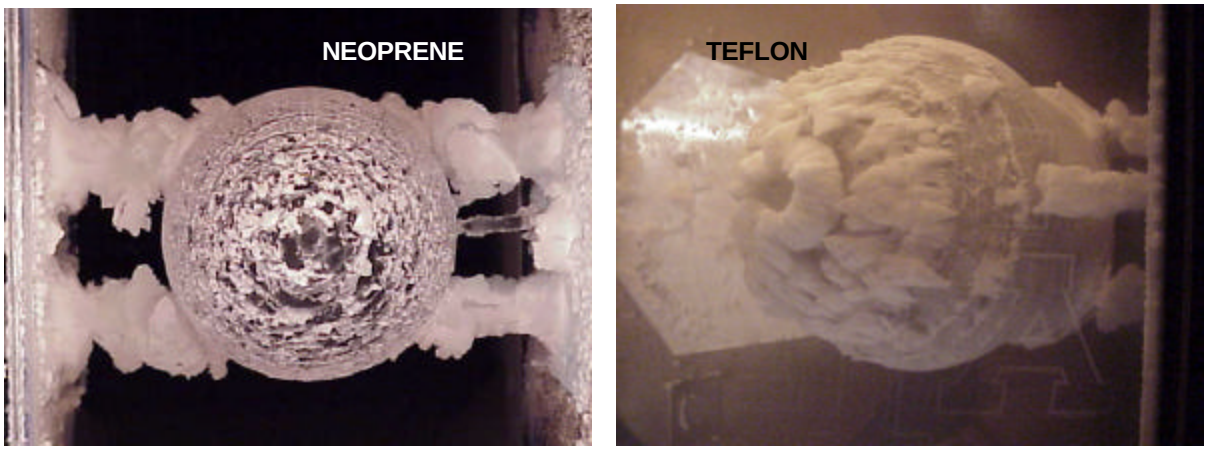


Figure 11. Radome Boots in Icing Wind-Tunnel

The perceived challenges were to reduce the RF loss in the neoprene approach and/or to design a means to use Teflon in a fashion similar to that proven for neoprene. Early prototypes of each solution were created to quantify performance and development challenges prior to formal contract award.

Upon contract award, the aircraft integrator/manufacturer was actively engaged to identify and quantify system control, test, maintenance, and installation constraints. Meetings and reviews were held on a regular basis. With every new piece of information, program direction was updated.

While the radome itself was clearly the number one technical challenge, the good work that was done early in the program to evaluate radome options was not accompanied by a thorough evaluation of real versus perceived constraints. The development was initiated with the assumption that requirements, constraints, and the conceptual design of the low-tech part of the program would be worked out in due time. Although over dramatized, Figure 12 illustrates the focus before and after contract award. The radome remained the highest-risk item, but the additional programmatic and technical complexities introduced another dimension to the problem.

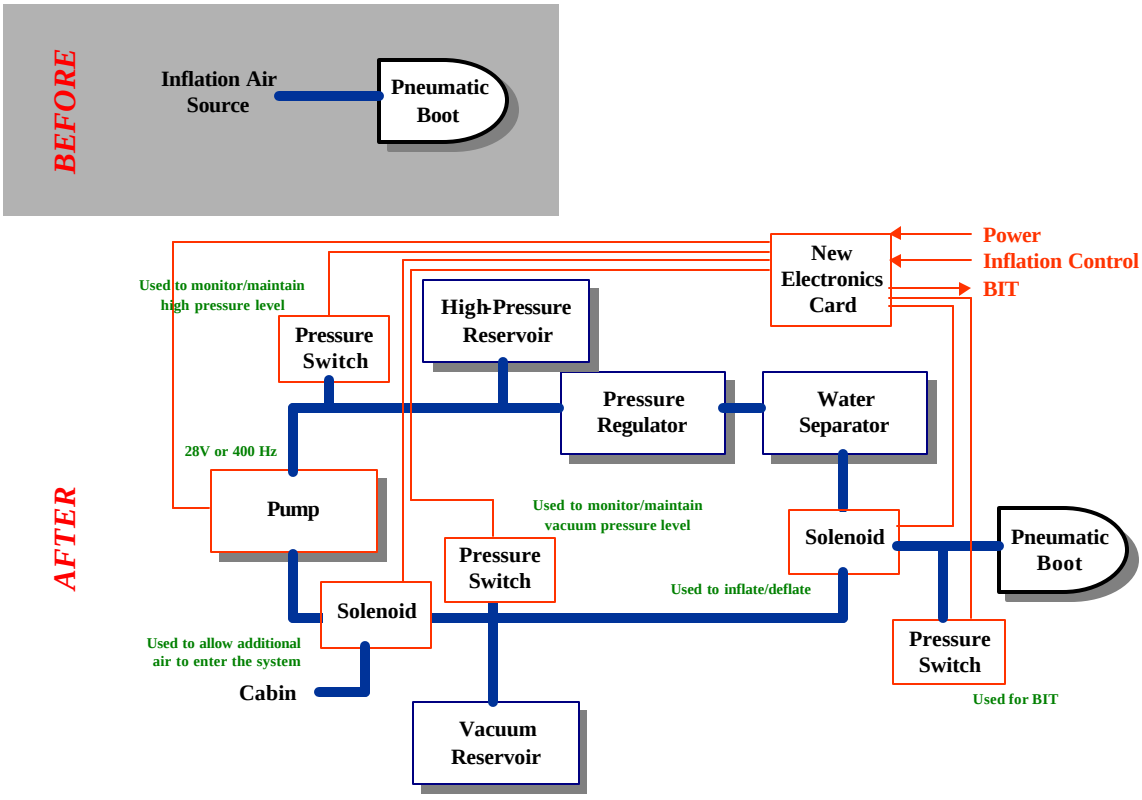


Figure 12. Notional Complexity Realization After Contract Award

In the end, development was canceled before critical design review. This was generally attributed to expanded work-scope that negated many of the earlier findings, decisions, and estimates. Many requirements and constraints changed and evolved considerably, but others (e.g., schedule and cost) were not

allowed to vary to accommodate new needs. Although it took a while to get there, the right decision was made. Many of the previously firm constraints were removed and the available concepts and options were revisited. New pursuits are now underway but are beyond the scope of this study.

3.5 Case Study Problem Areas

By doing just a little every day, I can gradually let the task completely overwhelm me. - Oscar Wilde

A detailed look at virtually any development program will produce a number of problems experienced along the way. Some will be bigger than others, but they are inevitable. The radome de-icing development effort had many successes that could be praised. However, the purpose of this case study involves the examination of problems. The external "symptoms" of problems that surfaced during the development are described in the subparagraphs below. A root-cause analysis is initiated in section 3.6.

3.5.1 Lack of Suitable Air Source

The program was started with the recognition that the air source for boot inflation was not identified. Early evaluations determined that no suitable source of compressed air was available on the aircraft, and options were very limited in power, size, and weight for the installation of a compressor. This low-tech need quickly became a substantial obstacle.

3.5.2 Need for Constant Vacuum

For reliability and performance reasons, the radome boot was required to be held under vacuum when in flight. Only when pressure was applied for inflation and ice removal was the vacuum released. This further complicated the air source design and control, reducing reliability and increasing average power consumption.

3.5.3 Need for Air Dryer

To avoid possible moisture in the boot (which would greatly reduce RF performance), as well as in other parts of the system where it could freeze (and lock up) components, an air "dryer" had to be installed at the inlet. In addition to the size, cost, and weight impact, periodic maintenance for such an item was a violation of existing radar requirements. Furthermore, no readily accessible space (for routine maintenance) was available near the radome.

3.5.4 Lack of Space for Electro-Mechanical Assemblies

Real estate in the nosebay of many aircraft is scarce. The program was initiated with the mutual understanding that "no" space was available for new equipment. It was hoped and assumed that new equipment would be minimal, such that it could be integrated within exiting avionics chassis. As a result, design options and component selection were severely limited.

3.5.5 Unplanned Impact to Radar Performance Requirements

The radar system had been design and integrated without a requirement for de-icing. "Forcing" inclusion of the functionality (even in a new assembly) on prior radar requirements greatly degraded the previously compliant areas of built-in-test (BIT), mean-time-between failure (MTBF), weight, power consumption, etc. These negative factors further complicated program perceptions, both inside and outside the program Integrated Product Teams (IPTs).

3.5.6 Schedule Slip

Development efforts to produce fully functional prototypes were initially constrained to one year. Technical difficulties and scope change caused delays. This forced shortcuts that proved problematic, putting even more pressure and constraint on development schedules.

3.5.7 Development Cost Growth

Better-than-expected performance of pre-contract radome prototypes precipitated the underestimation of the development challenge. These technology obstacles and scope changes regularly increased the estimates for development completion. Additionally, simplistic conceptual designs for control equipment were negated by changing mission and aircraft-level constraints.

3.5.8 Unsatisfactory RF Performance

Despite the promising results from early prototypes, some level of RF performance was sacrificed to maintain reasonable boot reliability. As an example, boresight error (angular error) became worse in the middle of the program when boot construction changed to increase its reliability. Although Figure 13 shows that improvements were made later in the program, the trade between RF performance and boot reliability remained a difficult one throughout.

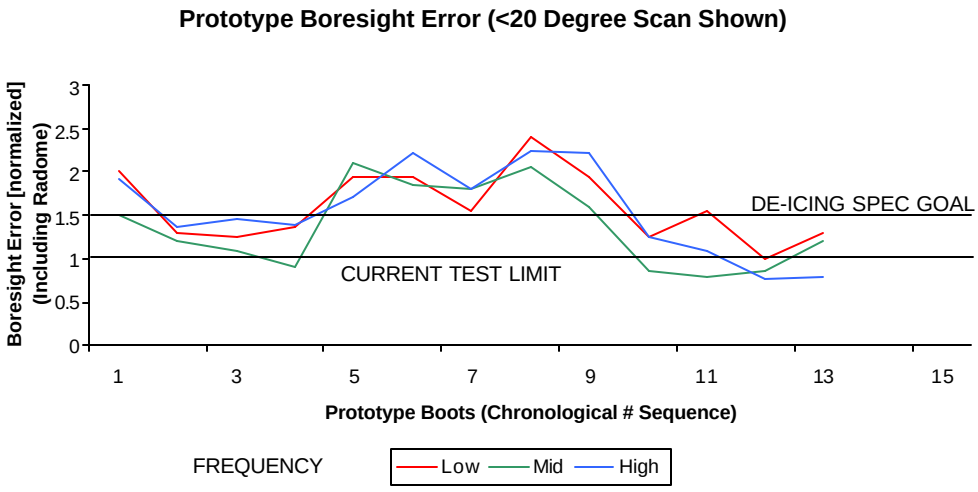


Figure 13. Typical Retrofit Angular Accuracy Measurements

3.5.9 System Noise

As stated in 3.5.1, the air source was given little attention before contract award. Constraints and designs evolved to the stage where a new compressor was planned. With this use of a fairly powerful compressor for inflation and vacuum maintenance, system noise increased well beyond that of the baseline radar. This was especially unattractive to the user community, who hadn't considered this effect until the Preliminary Design Review (PDR) timeframe.

3.5.10 System Weight

Previous paragraphs have explained the growth of the development task from its original concepts. As system complexity increased with scope changes and aircraft considerations, weight grew accordingly. This trend met with fierce resistance, as it would for nearly any aircraft equipment program.

3.5.11 System Reliability

Both the air compressor and boot had predicted reliability performance that caused considerable impact to the total radar values. Although these were not complete surprises to anyone, when they were put in terms of mission availability and likelihood of success, reliability moved quickly to near the top of the issue list. Again, this "new" problem was aggravated by the late recognition of user perception.

3.5.12 Aircraft Structural Impact

Possible impact to external aircraft structures could have forced considerable regression design and testing. Although work stopped before this risk was realized, it was significant to the program. Changes were needed to accommodate the larger radome diameter, and to enable interconnects between the radome and icing control subsystems.

3.5.13 System Maintainability/Inspection

The existing radome for this radar is composed of highly durable materials and coatings that have been proven on hundreds of aircraft. The neoprene surface used for the de-icing radome introduced the possibility of minor damage to the external surface that could not be detected by built-in-test (BIT) features. This required assessment and consideration from maintenance personnel.

3.5.14 Lack of Support from External Entities

Although the basic goals for RF and icing performance were achievable, the development effort's predicted and actual results rippled from the design of the radome through the aircraft performance metrics, and even to the users and maintainers. Such is the nature of many programs. While mitigation of radome ice was a user requirement, no one wanted to accept the negative baggage that came with it. Thus, pro-active support and meaningful trades and compromises were scarce.

3.6 Root-Cause Analysis of Process Problems

> *I never blame myself when I'm not hitting. I just blame the bat, and if it keeps up, I change bats. After all, if I know it isn't my fault that I'm not hitting, how can I get mad at myself? - Yogi Berra*

When a program experiences problems, the first finger usually points to a failure to follow process. Unfortunately, the recommendation for corrective action is often to apply more resources to following processes, rather than attempting to identify any inherent incompatibilities between the process and program goals. The following subparagraphs and sections decompose the case study problems in section 3.5 to increasingly base causes, to arrive at process priorities and recommendations in section 3.8.

3.6.1 Case Study Problem Mapping

The fourteen identified problems from the case study (sections 3.5.1to 3.5.14) are evaluated in this section to determine process failings. The fundamental case study problems are first linked to the seven previously identified systems engineering categories as shown in Table 4. These are combined and generalized in the next section to facilitate process recommendations and analysis.

Table 4. Case Study Mapping of Problems to Process Areas

Customer Needs	Lack of Space for Electro-Mechanical Assemblies
	System Noise
	System Reliability
	System Maintainability/Inspection
Technical Planning	Lack of Suitable Air Source
	Need for Air Dryer
	Schedule Slip
	Development Cost Growth
	System Weight
Technical Management	Aircraft Structural Impact
	Lack of Support from External Entities
Requirements and Design	Need for Constant Vacuum
	Unplanned Impact to Radar Performance Requirements
System Analysis	Unsatisfactory RF Performance
IV & V	None identified
Product Acquisition	None identified

3.6.2 Case Study Process Breakdown: Root-Cause Categories

Below, the specific program obstacles that are described in section 3.5 and listed in Table 4 are restated in more general terms and combined to identify the underlying failings.

3.6.2.1 Customer Needs

Problems previously described that relate to Customer Needs fall into the categories below:

1. Identifying both spoken and unspoken needs, requirements and goals
2. Understanding details of the product environment (not just the spec)
3. Understanding customer requirements allocation (how things roll up)

3.6.2.2 Technical Planning

Problems associated with Technical Planning can be generalized as follows:

4. Validating technical assumptions
5. Challenging unachievable program constraints
6. Identifying hidden requirements and constraints

3.6.2.3 Technical Management

Obstacles that can be attributed to Technical Management are:

7. Customer communications (the right information and the right customer)
8. Customer buy-in (including user and support communities)

3.6.2.4 Requirements and Design

Problems in Requirements and Design are categorized as:

9. Evaluation of end-to-end mission considerations
10. Customer agreement of requirements bases

3.6.2.5 System Analysis

The failing in the System Analysis area might be the only one that was "acceptable", or at least unavoidable. Simply put, there were technical challenges that were not overcome to the desired extent. This is sometimes a fact-of-life that must be handled through the risk mitigation process. The lesson learned was simply that predictions of manufacturing process improvements were overly optimistic.

11. Accurate prediction of process capabilities

3.6.2.6 Product Acquisition and IV & V

The case study program involved development phases only through design and prototyping. Limited work was performed on Product Acquisition and Integration, Verification, and Validation. Problems experienced in these areas were minor in nature and do not require further discussion.

3.7 Case Study Analysis Summary

The root-cause results can be further simplified to a handful of general principles. The simplified causes of the eleven problem areas identified in the previous section are shown in Figure 14.

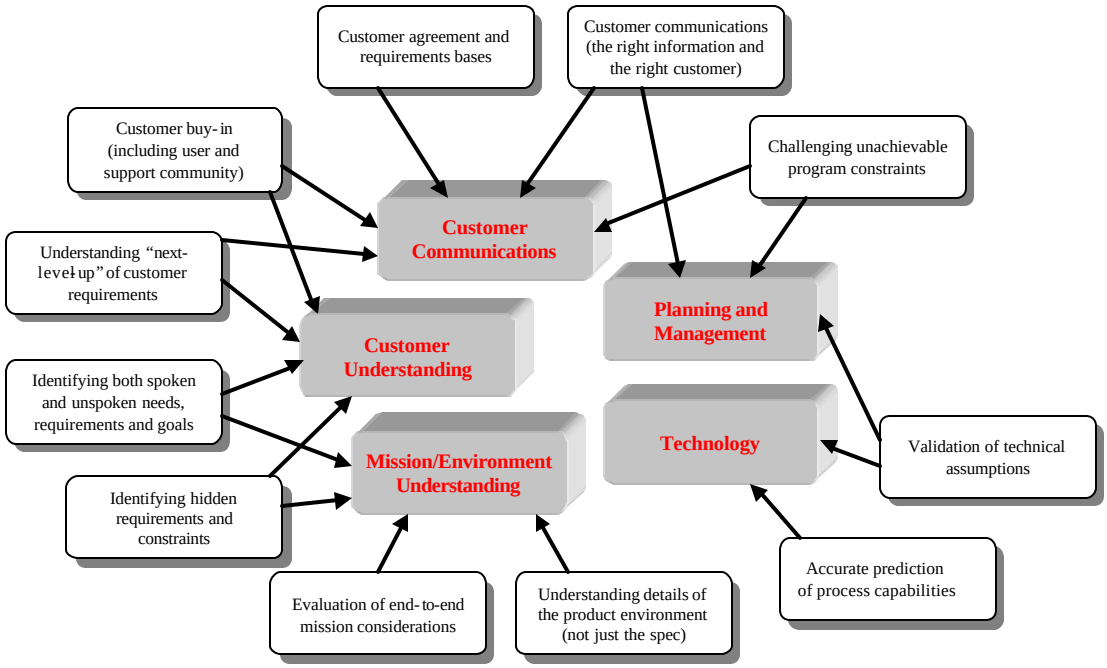


Figure 14. Root-Cause Results

It is apparent in the figure that problems are dominated by customer issues. This is not surprising in the "high risk" environment under study, where fundamentals are a dominant driver for success. The five areas are discussed further below. Although many issues shown in Figure 14 span multiple core areas, they are mapped to only one in the following subparagraphs.

3.7.1 Customer Communications

Unfortunately, a web search for "System Engineering For Dummies" revealed nothing. If such a source existed, it would likely begin and end with customer issues. Many of the customer problems experienced in programs do not come from bad systems engineering processes. They come from bad execution. The processes will all say "know you customer's needs." Unfortunately, engineers often take

such a process step only to the level required to pass an internal review or audit. Once they have a Statement of Work (SOW) and a specification, they (we) claim success and move on.

3.7.1.1 Getting the Right Information to and from the Right Customer

The fundamental problem is often the quality of systems engineering your *customer* is performing. Do *they* have all *their* stakeholders involved? Do *they* understand *their* user's needs? Are *they* communicating effectively internally and externally? All too often, the answer is no.

Systems engineers are the representatives of the end user. Regardless of intermediate bureaucracies, the systems engineer must make sure the right voices are heard in customer communications and ensure that all are in agreement on the path forward. This may even include the facilitation of communications within the customer's organization.

3.7.1.2 Acknowledging Risk

"Everyone loves a hero" and "heroes die young". Systems engineers must avoid the temptation to always be the good soldier and the naïve optimism that goes with it. Once the customer believes there is the slightest possibility for success, they tend to latch onto it. The systems engineer must educate the customer and user on the realities of risk areas, and challenge any constraints that could help mitigate those risks. In the case study, there were numerous programmatic constraints that minimized opportunity for success. These included the normal cost and schedule issues, along with design constraints intended to minimize corporate risk.

3.7.2 Customer Understanding

3.7.2.1 Identifying Hidden Requirements

As mentioned in the previous section, the system engineer's job is not done once a specification is in hand. Sometimes a squirm in a chair, a concerned grunt, or an overheard comment foreshadows customer

direction, even though no official word is given. The systems engineer must root out the identity and needs of all customer representatives, willing or not. Even the customer's customer may need to be solicited. Depending on organizational complexity, it may be very difficult to know when this job is done.

A related challenge is in conflicting requirements. These may come from independent customer organizations, or represent customer needs to satisfy their system's performance parameters. Regardless, conflicting objectives must be quickly evaluated to reach concurrence on the planned approach and goals.

3.7.2.2 Ensuring "Buy-In"

"One man's trash is another man's treasure." The radome de-icing program experienced a difficult challenge relating to the customer-based IPTs. Although the contracting group within the customer's organization had the responsibility to develop a solution to comply with this one requirement, other aircraft-level teams were not particularly interested. On their scale, this was a small problem and they had seemingly little interest in rocking their bigger boat by relaxing pre-existing constraints. This is understandable given all the analyses and tests that might need to be re-run on a new aircraft development, just because someone added a box, or some weight, or changed a leading edge shape. Once again, this is not an easy challenge, but the mass of constraints in the high-risk program often requires the battle to be fought.

3.7.2.3 Understanding Technical Budgets

The case-study program required knowledge of the weight, power, and reliability budgets at the aircraft level. Although it was relatively simple to calculate the changes resulting from the added equipment, no one wanted to adopt the degradation. From the radar manufacturer's point of view, it was undesirable to have a good system saddled with more weight and lower reliability just because this one platform needed it. Although not insurmountable, this had a perceived impact on marketability for the system. From the aircraft manufacturer's perspective, adding weight or reducing aerodynamics beyond existing budgets had a

potentially devastating effect on analyses associated with key performance parameters like payload and mission range.

3.7.3 Mission/Environment Understanding

3.7.3.1 End-to-End Comprehension

This title needs little explanation. Suffice it to say that the systems engineer must maintain an all-inclusive vision of how the product will be used. This may not always involve detailed specifications and analyses, but rather common-sense thought experiments. What will it go though during the course of a mission? What will happen when it is damaged or replaced? What *else* can go wrong?

3.7.4 Planning and Management

Several root-cause items related to planning and management have already been discussed. On the surface, you could blame nearly everything on planning and management. Special discussion is warranted for a few additional items.

3.7.4.1 Disciplined Planning

Few development programs are awarded without a history to build on. While past experiences are clearly useful, program plans must not be force-fit into previous expectations. To the extent possible, risk databases, schedules, and technical plans should be allowed to evolve from scratch before historical findings and data are imposed. Indirectly, this was one of the problem areas of the radome de-icing program in that it evolved from a study. The lack of bottoms-up planning contributed at least partially to the over-aggressive schedule.

3.7.4.2 Monitoring and Control

Most companies have set procedures for internal program reviews. These are highly desirable, provided they are sufficiently tailored to the specific program. Findings from reviews and other process checks must be communicated without prejudice. It's all too easy for well-intentioned engineers to manipulate situations to minimize recognition and perception of potential problems. Managers and lead systems engineers must foster (or create) an environment that rewards problem recognition and mitigation.

Likewise, the systems engineer must avoid blinders. While it may be a business necessity to cover the legal bases, failure for any reason is failure. Whether the fault of the customer or the contractor, systems engineers must notify stakeholders when analyses or tests indicate potential problems at any level of the design.

Problem reaction should be based on analytical information where possible. By definition, the subject high-risk programs will experience trials. Decisions on program direction must be consistent with the fundamental vision of the program. Winning battles does not guarantee winning the war. Good daily decisions can lead you astray if the overall customer and user needs are not overtly considered.

3.7.4.3 Personnel Management

Near constant attention is needed by one or more individuals from day one of the program. The human mind remains a cost-effective processor and database for coordination and direction. However, the more processors and databases that must be coordinated, the more complex the interface control will be. Part-time management of resources results in part-time efficiency.

While time-sharing of talent is commonplace, progress on a program often necessitates dedicated resources, especially in systems engineering. Sometimes 25% of four people is better than 100% of one, but not always. While multiple perspectives are great in concept exploration, managing the interface between multiple parties generally creates an inefficiency in detailed tasking, such as requirements management and coordination.

Processes and controls for personnel management and assignment are noteworthy, but will not be addressed further due to their place in program management, rather than systems engineering. Indeed, systems engineers often play this role, but it is not a needed change to the systems engineering process resulting from the case study for high-risk programs.

3.7.5 Technology

While technology development may be the root element of the program, it should rarely be the root cause of failure. Certainly, technology does not always accomplish what is desired, but early recognition of risks and mitigation plans can avoid programmatic failure. In the case study, there were some desired radar performance improvements that were not achieved. There were, however, plans in place to provide varying levels of compensation in other system areas. More importantly, the case study program involved several technical assumptions that were not validated and communicated in a timely manner to ensure cross-company concurrence. Clearly, this had a detrimental affect.

3.8 Process Improvement Opportunities

> *When you can measure and express it in numbers, you know something about it: when you cannot, your knowledge is of a meager and unsatisfactory kind – Lord Kelvin, 1891*

The general process issues identified thus far can be simplified to seven areas of focus. These will be defined in the following subparagraphs. It is recognized that many of the programmatic issues addressed to this point span the boundary between systems engineering and program management. However, the following process improvements are identified for the generalized program to address the aforementioned issues in the case study from the systems engineer's perspective.

3.8.1 Re-Plan Early and Often

Do not initiate the program with legacy baggage. Perform initial planning without pre-conceived constraints, and then address and manage those constraints as risks. Likewise, don't be afraid of significant changes in plans. The high-risk programs are going to change. Hiding your eyes will not help. Early recognition and adaptation will.

Technical and programmatic risks should not only be identified and weighed, but mitigation and contingency plans should be developed and communicated to the customer. This will create an environment more conducive to change.

3.8.2 Utilize Technical Performance Measurements and Reviews

Everyone has Technical Performance Measurements (TPMs) and program reviews, but how are they really used? Are they a necessary evil to be manipulated for management, or are they a tool used to recognize and fix problems? They should be the latter for any program, but it is especially important in high-risk programs to set good TPMs and eliminate any motivation for personnel to "cook the books". Reward team members that acknowledge and report problems and risks.

3.8.3 Establish Customer Communication Emphasis

Good customer communication usually comes more from hard work than from process definition. However, disciplined use of the process will help identify the important information that needs to be shared and agreed upon. Both the systems engineer and the customer must have a common understanding of the program objectives. This includes continuous agreement on the current state of affairs, as well as the goals and expectations of the development.

The customer must be as committed to success as the contractor. This can sometimes be gained or verified by interaction with the end user or the customer's customer. When necessary, facilitate

communications between customer entities. If such problems are anticipated, include this type of task in the proposed work effort and team structures.

3.8.4 Formalize Customer Penetration Verification

…more hard work. Too often, single-point communications with the customer result in a single-point perspective. The systems engineer must verify that the customer and user communities have been sufficiently briefed and solicited for their expectations. This can be facilitated through the creation of detailed organizational information (e.g., org charts) about relevant customer branches. Obtain formal agreement on the list and make sure everyone is informed and involved.

Contact with the right customer personnel is relatively straightforward. Ensuring each group is motivated for the success of your program can be a different story. Although it is desired to have buy-in from all customer entities, questionable areas can at least be tracked as a risk and mitigated through customer communications as described in section 3.8.3.

3.8.5 Document Mission Usage Early

Although not really a process *change*, it is another step that is often left out once a SOW and a specification are in hand. As yet another planning tool, the product usage needs to be evaluated over all environment and mission conditions, including storage, maintenance, and the like. Customer concurrence should again be sought.

3.8.6 Obtain Early Feedback

Yelland [2] says, "all models are wrong; some models are useful." On high-risk programs, engineering and management optimism can lead to disaster. Even simulations and models may be creatively interpreted to bias the estimates. Early tests and prototypes are mandatory on these programs. Like

everything else, however, testing must follow good engineering plans, and the results must be dealt with analytically and objectively so that programmatic course corrections can be made.

Feedback is important from non-test areas as well. Maintenance concepts, environmental concerns, and many other areas can and should be conceptualized and modeled for customer/user consideration and comment early in the project.

3.8.7 Employ Scientific Experimentation Principles

The very nature of the "high-risk program" is that there are both known and unknown obstacles that will arise. This uncertainty must be managed in a disciplined fashion in order to minimize risk. Internal and external pressures often temp the development team to attempt total compliance in one shot. For some efforts, this may be achievable. However, the high-risk program generally involves technology development and/or a complex trade space of system level requirements. If total compliance were likely in one shot, it would not be considered a high-risk program, per the definition used for this study.

The complex trade space must be managed for risk. While virtually any test can be concocted that yields useful data, careful forethought must be applied in the test planning to ensure that ALL parametric trades are addressed in a timely fashion.

Part of the answer to this dilemma lies in the aforementioned elements of planning, TPMs, and early feedback. The associated investigations for trade-space evaluation must follow fundamental scientific experimentation principles.

The goals of such evaluations are usually obvious, including things such as determining which parameters are most influential on system performance and which have no impact. In the high-risk program, it is important to separate the key drivers from those less significant, so that more focused test, evaluation, and risk mitigation can be performed. Montgomery [16] gives the following seven steps associated with designing experiments:

1. Recognition of and statement of the problem – coordinate with all concerned parties
2. Choice of factors and levels – identify range of conditions, keep it simple
3. Selection of the response variable – determine measurement
4. Choice of experimental design – identify sample size, test order, etc.
5. Performing the experiment – follow procedures and plans, and document results
6. Data analysis – utilize statistical methods for evaluation
7. Conclusions and recommendations – draw practical conclusions and communicate

Although these seven steps are highly correlated with typical systems engineering activities, detailed explanation of experimental design is beyond the scope of this report. Numerous literature sources are available that detail procedures for determining controls, parametric delimiters, sample sizes, statistical methods, etc. However, the systems engineer must recognize when he or she is in a program requiring experimentation, as is often the case with technology development. If experimentation is needed, it should be accomplished according to proven principles that will yield incremental knowledge and understanding toward the end goal. Depending on the complexity involved, it is usually an iterative process.

3.9 Process Improvement Summary for Case Study Program

Based on the previous analysis, the radome de-icing case study program is believed to benefit from the combined recommendations in section 3.8. These are re-stated below for summary purposes and will be evaluated further in the next section for validation and refinement.

Table 5. Candidate Improvement Areas

1.	**Re-Plan Early and Often**
2.	**Utilize TPMs and Reviews**
3.	**Establish Customer Communication Emphasis**
4.	**Formalize Customer Penetration Verification**
5.	**Document Mission Usage Early**
6.	**Obtain Early Feedback**
7.	**Employ Scientific Experimentation Principles**

CHAPTER - IV
4. VALIDATION

The objective of this section is to substantiate and refine process improvement recommendations identified in the previous section from the case study. Both specific and generic validation sources and means are used.

4.1 Lessons Learned from Other Sources

"Lessons learned" information was pulled from ten open literature sources on a pseudo-random basis to further explore process breakdowns in engineering. Sources were limited to relevant topics, and individual lessons that did not relate to systems-engineering activities were culled. Many sources included data from multiple programs. As the intent was only to capture lessons, not rank them, the quantity of occurrences were not tracked.

To conserve space, the items from all ten sources were grouped into the relatively short list shown in Table 6. Judgment was applied to adapt wording for this report and eliminate project-specific items. Sources are listed in the table for detailed reference. Appendix B has the paraphrased entries from each source, the listing of sources, and the mapping to the "Summary Lessons" in Table 6.

Table 6. Lessons Learned from Outside Sources

Sources	Summary Lesson
[17] through [22]	1. Team Buy-In and Commitment
[17], [19] through [23]	2. Management Support and Participation
[17] through [26]	3. Emphasis on Early Planning
[17], [21] through [26]	4. Process Control Measures (e.g., TPMs)
[17] [18] [20] [21] [23]	5. Improved Leadership
[17] through [21], [23] through [26]	6. Improved Communication
[17], [20] through [26]	7. Enhanced Stakeholder Involvement
[17]	8. Recognition and Reward
[17] through [22], [26]	9. Consistent Vision and Objectives
[18] [20] [22] [24]	10. Adequate Resources
[20] [24] [26]	11. Simplify Where Possible
[21] [24] [26]	12. Maintain Flexibility
[22] [23] [25] [26]	13. Clear and Reasonable Requirements
[23] [24] [26]	14. Emphasis on Testing

4.1.1 External Lessons Learned Problem Mapping

Similar to the approach used for the case study, the issues identified in external lessons-learned sources are first linked to the systems engineering categories as shown in Table 7. It is interesting to note that whereas the case study "problems" were concentrated in Customer Needs and Technical Planning, the more general nature of items researched in open literature produced an emphasis on management areas. This is due primarily to the more generic nature of the programs identified for outside research and possibly their pre-categorization (or bias) of problem causes.

Table 7. Lessons-Learned Mapping of Problems to Process Areas

Customer Needs	Enhanced Stakeholder Involvement
	Consistent Vision and Objectives
	Maintain Flexibility
Technical Planning	Emphasis on Early Planning
	Adequate Resources
	Simplify Where Possible
Technical Management	Team Buy-In and Commitment
	Management Support and Participation
	Process Control Measures (e.g., TPMs)
	Improved Leadership
	Improved Communication
	Recognition and Reward
Requirements and Design	Clear and Reasonable Requirements
System Analysis	None identified
IV & V	Emphasis on Testing
Product Acquisition	None identified

To simplify the continuing analyses, the fourteen lessons-learned problem areas are further mapped to the previous case study process recommendations to identify any new areas for consideration. This mapping is shown below in Figure 15, with the case-study process improvement candidates in the thicker boxes with black font, and the new problem areas in thinner boxes with red font.

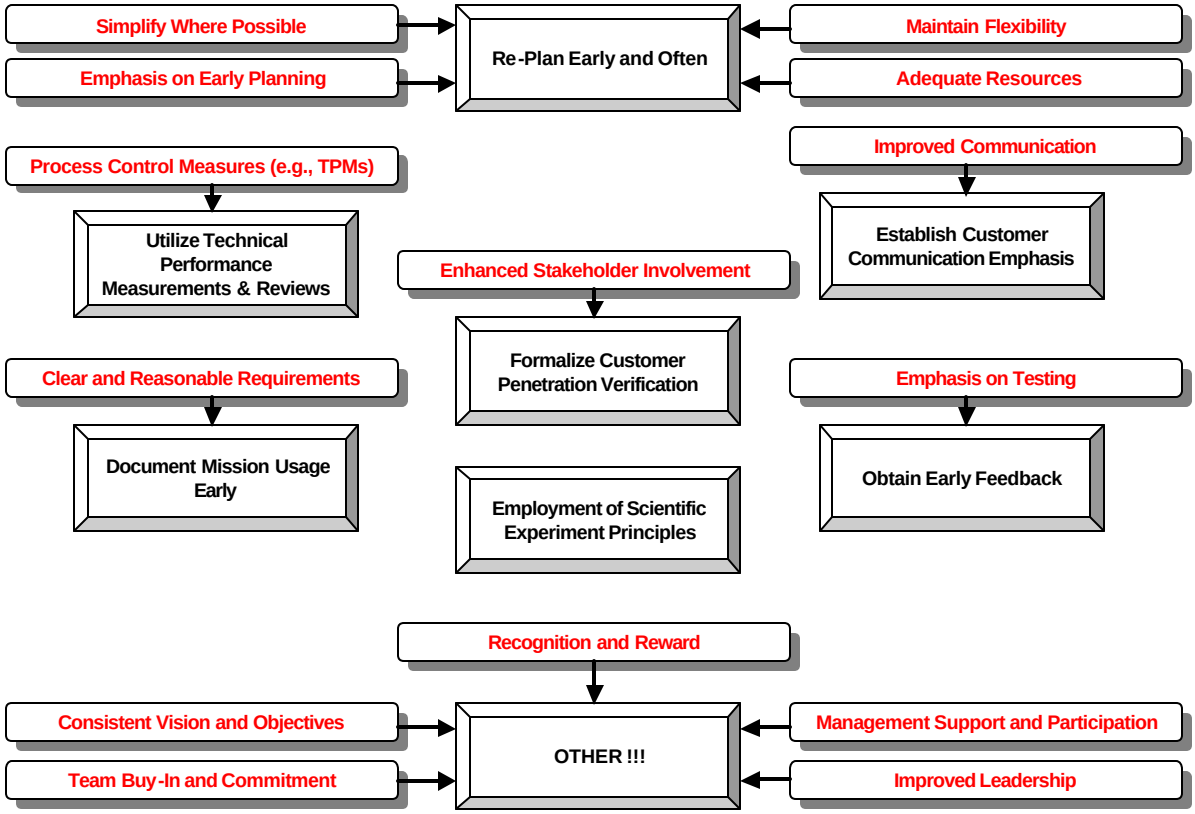


Figure 15. Lessons Learned Problem Mapping

The figure shows that in addition to the confirmation of many case study findings, the lessons-learned search resulted in five new items for consideration, as shown at the bottom of the figure mapped to OTHER. Of these, "Management Support and Participation" will not be considered further, as it relates to upper levels of program management, which is beyond the scope of this systems engineering study. The four remaining issues for process improvement are then:

1. Consistent Vision and Objectives
2. Team Buy-In and Commitment
3. Recognition and Reward
4. Improved Leadership

While it could be argued that these are not areas unique to the systems engineering process, they are nonetheless critical for program success, and program success IS largely the systems engineer's responsibility. The four items can be simplified in two areas for process consideration; Vision and Passion. Further descriptions of these two items are provided in the following section.

4.1.2 Fundamental Success Drivers

You do not teach someone to play chess by writing a script of actions and reactions. Instead, you teach them the fundamentals – the goals, the pieces, the rules, etc. Likewise, most systems engineering processes make every attempt to avoid the details of process implementation. However, they do not, and perhaps can not, address the even more basic ingredients of desire. Good luck to those that try to teach or instruct someone that has no interest.

Many successful pursuits (engineering and otherwise) can be characterized as the combination of vision, passion, discipline, and risk.

Vision is the understanding of where you want to go;
Passion is the desire to get there;
Discipline is the controlled fashion by which you maximize opportunity to succeed; and
Risk is the willingness to attempt the journey

The systems engineering process does not deny these elements, but their nature is such that they can hardly be taught or documented. They must come from the individual and corporate culture. In actuality, the systems engineering process primarily addresses the "discipline" part of the equation, as defined above. Vision and passion should flow from the program leadership throughout the team. Each job function may (and should) have a different vision, but the collective vision should be that of the systems engineer as the

representative of the user. Risk is a little harder to justify as desirable, but on programs where interim "failures" are virtually unavoidable, risk is part of the equation. If the risk is unacceptably high, the systems engineer has the responsibility to point out the likelihood of failure. Not all development programs should be attempted.

Thus, it is recommended that two items be added to the list of "process" improvements developed in section 3.8. The two items relate to vision and passion and are defined in the following subparagraphs. Again, these border on program management, but they are also critical to the successful implementation of systems engineering on high-risk programs.

4.1.2.1 Share the Vision

Corporate executives often see fit to develop vision and mission statements to guide their company. These are usually simple messages of extremely broad nature so as to apply to a large group of programs and individuals. At the program level, vision is rarely explicitly captured. Instead, the team may be left with only the legalities of contracts and hopes for career incentives to motivate performance.

While there is not always a need for formal documentation of a vision statement, it is recommended that program leaders periodically remind their teams of the expected end result, as well as the grand plans and opportunities relying on the success of their work.

4.1.2.2 Recognize and Reward Passion

Fear of dismissal or poor financial rewards can provide some short-term motivation, but it more often leads to cover-ups and escalating problems. True desire to succeed, on the other hand, will promote teamwork and ingenuity to overcome setbacks. As a program leader, the systems engineer must share that passion and ensure that it spreads to the leaders of development teams.

4.2 Validation of Process Improvements on Case Study Program

The combined result of the case study analysis and lessons-learned research is a set of nine candidate systems engineering improvement areas.

1. Re-Plan Early and Often
2. Utilize Technical Performance Measurements and Reviews
3. Establish Customer Communication Emphasis
4. Formalize Customer Penetration Verification
5. Document Mission Usage Early
6. Obtain Early Feedback
7. Employ Scientific Experimentation Principles
8. Share the Vision
9. Recognize and Reward Passion

Looking back at the list of problems on the case study de-icing program (section 3.5), the nine fundamental process improvements above are re-assessed to determine if they meet the need of that challenging radome development. As the case study was used as an input to create the process improvements, this is something of a circular analysis. It is, however, considered an important validation step, as the recommendations were iterated and refined based on additional literature and program lessons-learned. The fourteen case study problem areas are mapped to the nine improvements as shown in Figure 16.

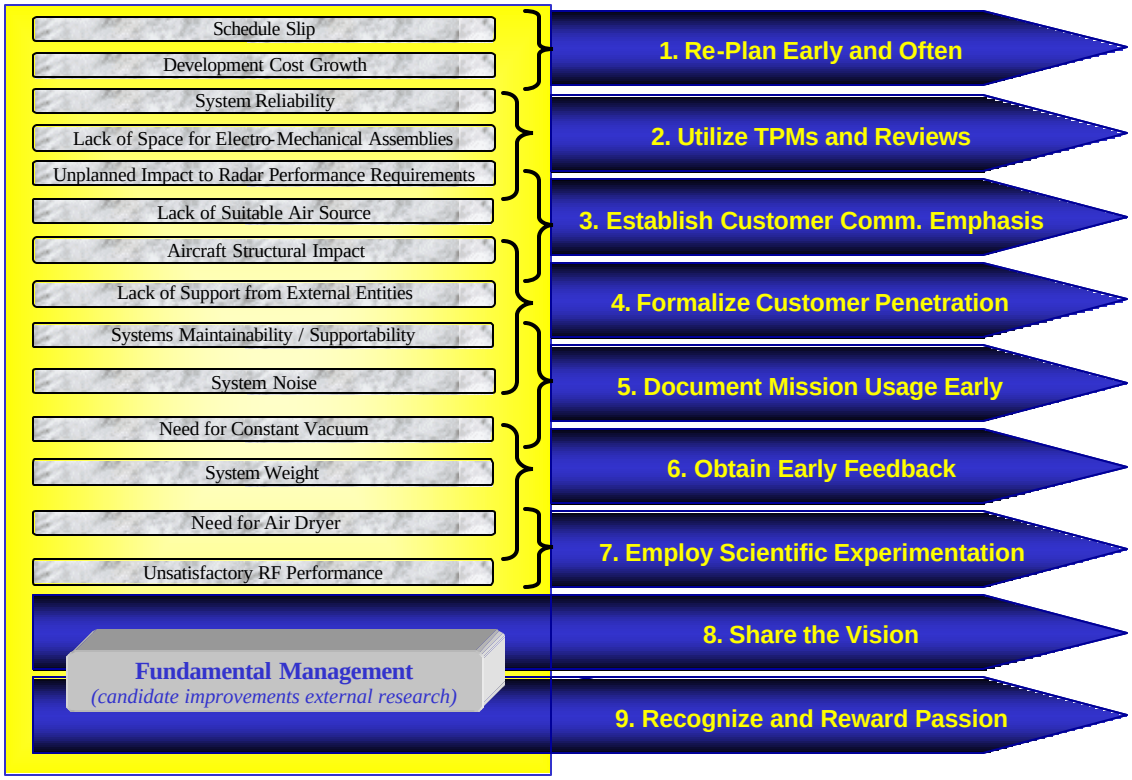


Figure 16. Application of Candidate Improvements on Case Study

The generalized nature of the process improvement descriptions results in case-study problems that span multiple areas. Although the figure above attempts logical alignment, the important aspect is that all problem areas are addressed by one or more process improvements. As expected, the last two candidate improvements (numbers 8 and 9) are not mapped to any case-study problems. This is because they were not identified as key issues on that program. The need to "Share the Vision" and "Recognize and Reward Passion" came out of the lessons-learned research on outside programs.

Nine steps to guarantee success? Not necessarily. As Edwards [14] points out, a formal systems engineering process "is necessary, but not sufficient for good SE implementation." He says "techniques that

work in one situation will not necessarily work in another." Emphasis on the process improvements stated thus far would certainly have helped in the case study program, but success is never guaranteed.

> *Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning.*
> *- Rich Cook*

4.3 Validation of Generalized Process Improvements

Thus far, seven candidate recommendations were generated from the analysis of the case study program. These seven were largely validated through the "lessons learned" information found in open literature. These latter "lessons learned" also spawned two new recommendations, bringing the total to nine. This section will address these nine items independently in a somewhat subjective fashion, calling on literature references to further validate assertions. Focus will be on generalized application. Issues clearly unique to any single program (including the case study) will not be discussed henceforth in this chapter.

4.3.1 Re-Plan Early and Often

Few would argue the importance of planning to the success of any program. DeFoe [27] recommends plans that are "success oriented, achievable, defendable, and cost effective but which can absorb the changes that will come." He also says, "Change the plan as soon as experience shows a better way to do a task." These seemingly obvious concepts are often lost in the bureaucracy of program cost and schedule pressures.

> *In preparing for battle I have always found the plans are useless, but planning is indispensable. - Dwight D. Eisenhower*

McClinton [28] gives 25 "Unwritten Laws of Systems Engineering." Many of these also deal with planning – work planning, contingency planning, test planning, planning documentation, etc. On *re-planning*, his 5th law goes so far as to say, "never be afraid to start over."

Plans on the high-risk program must not only be thoroughly conceived, but also dynamically adjusted throughout the critical stages of development. While "routine" developments might hope to change plans infrequently, the high-risk program must accept change more readily, and even perform considerable advance planning for contingencies.

4.3.2 Utilize Technical Performance Measurements and Reviews

Whatever they are called, measurable controls must be in place to monitor technical performance and risk. TPMs on airborne systems (like the case study) often include things like weight, power, size, MTBF, growth capability, etc. TPMs must be more than just report fodder. They must be integrated into the management decision chain for the high-risk program.

Like the other process recommendations provided herein, TPMs should focus on the fundamentals, representing customer goals and constraints first and foremost. They should be kept simple, meaningful and unambiguous. DeFoe [27] suggests TPMs have "demonstrable links to customer/consumer needs and system requirements." Reviews should also focus on basic program elements and risks.

Managing risk is one of the primary uses of the TPMs and reviews. For useful evaluation on the high-risk program, the relative nature of risk must be recognized. On high-risk programs, "smaller" risks can be significant and must be tracked. Tosney [24] ranks risks from highest to lowest as shown in Table 8. A quick assessment of this table against a typical "high-risk" program shows that most aspects of such a program fall into the highest risk categories. Were models tested, as is required for the lower risks, it would not be considered a high-risk program, per the definition in this study.

Table 8. Risk Assessment Levels

Highest Risk		Technology Development
Level 1	Basic principles observed and reported	
Level 2	Conceptual design formulated	
Level 3	Conceptual design tested analytically or experimentally	
Level 4	Critical function/characteristic demonstration	
Level 5	Component/brassboard tested in relevant environment	Advanced Development
Level 6	Prototype/engineering model tested in relevant environment	
Level 7	Engineering model tested in space	
Level 8	"Flight-qualified" system	Flight Systems
Level 9	"Flight-proven" system	
Least Risk		

4.3.3 Establish Customer Communication Emphasis

Much of the communication with customers is documented in formal means such as specifications, meeting minutes, and contracts. These are important, as stakeholders tend to forget verbal agreements and understandings, unless they are of direct importance to their everyday activities. However, reliance solely on legal documents will usually result in a failed program or a solution to the wrong problem.

"A verbal contract isn't worth the paper it's written on," – Sam Goldwyn

When done properly, customer communications is a continuous process throughout program life. It is a trust-based relationship that begins with the merging of goals and objectives. Care must be taken to avoid compromising realistic expectations for the sake of customer relations. Be open an honest about risks and expectations. DeFoe [27] affirms,

- "Work with the customer to identify the consumer (user) groups that will be affected by the system"
- "Use a systematic method for identifying the needs and solution preferences of each consumer group"
- "Don't depend on written specifications and statements of work. Face-to-face sessions…are necessary"
- "State as much of each need in quantified terms as possible"
- "Clarify each need…relative to the customer's larger purpose"

4.3.4 Formalize Customer Penetration Verification

There's no master's degree for this one. The systems engineer must identify the customer, the customer's customer, the user, and anyone else that touches the product throughout its life. Learn their culture, listen carefully to what they say and don't say. Document as much as possible and get their concurrence on agreements and their potential impact or risk.

Yeh [29] says, "honor your customer and your customer's customer…Zero time companies know what drives their customers. It is their customer. Get to know what *their* customers are demanding of them."

4.3.5 Document Mission Usage Early

This "improvement" was previously recognized as an existing element in the systems engineering process. EIA/IS-731.1 [6] calls for the development of "operational concepts and scenarios, which include functionality, performance, maintenance, support and disposal as appropriate." For this study, the description is included to note its particular value in high-risk programs. It may not be enough to think it through. Thoroughly document usage and actively seek broad agreement.

4.3.6 Obtain Early Feedback

Despite the availability of advanced simulation tools, growth in system complexity often prohibits prediction of complex behavioral interactions between integrated subsystems. Analyses and intelligent speculation are fine, but testing must be a primary focus area in the high-risk program. There are numerous literature references to support this assertion. McClinton [28] states it succinctly (and humorously), "any analysis will be believed by no one but the analyst who conducted it – any test will be believed by every one but the person who conducted it." George Polya [30] says, "If you want a description of scientific method in three syllables, I propose: Guess and Test." Always test!

Ric Sylvester [31] further acknowledges that fast delivery of technology requires "rapid acquisition with demonstration technology" and "full system demonstration before commitment to production." Tanik and Ertas give three axioms associated with engineering complex systems [4], all of which relate to systems engineering. Axiom 2 specifically calls for early and repeated testing (validation) for system refinement. Their three axioms are:

1. *Axiom of Hierarchy – recognizes the need for a logical decomposition of complex problems resulting from the inability of humans correctly specify all levels of a system at the beginning of development*
2. *Axiom of Feedback – addresses the need for rapid and repeated validation of the design for both specification refinement and design improvement*
3. *Axiom of Automation – calls for the use of automated tools to expedite tasks which the human (or the culture) is not well suited*

4.3.7 Employ Scientific Experimentation Principles

Almost as a corollary to the previous section, testing must be performed in an analytical fashion that allows incremental progress and building of the knowledge base. Montgomery [16] confirms, "It is usually a major mistake to design a single, large, comprehensive experiment at the start of a study…we do not perfectly know the answers to these questions, but we learn about them as we go along." Luftig and Jordan [32] also recognize the problem of thoughtful test foundations, saying, "lack of rigor and discipline displayed in many companies in the conduct of industrial research is astonishing."

Wasted or inefficient testing is not the only danger. While the high-risk program is characterized by possible setbacks that must be managed objectively, Warfield [33] points out the dangers of *interim successes*, saying:

> *"It is reasonable to be optimistic about overcoming these factors, but only if they are dealt with as a set, because resolutions to one factor are not necessarily resolutions of others, and may even escalate them."*

This clearly points out the need not only for a thorough and logical test/experiment program, but also a disciplined analysis effort.

4.3.8 Share the Vision

Everyone should have a clear understanding of where their efforts are leading, both individually and collectively. Martin [34] recognizes that the systems engineer is not only the "keeper of the process", but also "establishes the shared vision of the system solution." Ertas and Jones [35] go on to say, "If an individual has a vision, believes that he or she has the potential to be successful, and is willing to *perspire* (work), almost any goal can be realized."

"When you set yourself on fire, people love to come and see you burn." - John Wesley

Communication of vision and purpose is a key element of program leadership. Although not overtly addressed in most engineering processes, it is fundamental to success.

4.3.9 Recognize and Reward Passion

Yeh, et al., [29] reports the importance of employee motivation, saying:

> *"…each person accepts the mantle of authority and responsibility for leadership. Nothing less will enable a company to navigate the complex, high-speed, and geographically far-flung digital marketplace."*

Corporate, program, and team cultures should encourage the human will to succeed. Yelland [2] sums it up by saying, "even knowing all the theory, engineering is still 95% hard work, 5% inspiration."

But were do we get this motivation to accept responsibility and do the "hard work"? Ertas and Jones [35] suggest that the secret lies in leadership. They reference studies indicating that effective supervision requires genuine interest in subordinates, saying:

"When employees take responsibility for their work, see that it is recognized favorably, and believe that they are properly rewarded, they will normally perform to the maximum of their ability."

Even EIA/IS-731 [6] states that proper implementation of the standard requires that "skilled personnel are used to accomplish the purpose of this Interim Standard…" The process alone is of little value without the passion to use it successfully.

4.4 Supplement to Generalized Process Improvements

The pieces are all there. The nine candidate process improvements for high-risk programs have been confirmed in the lessons learned of programs and validated in literature on the subject. The importance of customer emphasis and planning is nearly universal, along with the people skills for group vision and passion to succeed.

While not absent in other programs, the reliance on TPMs, mission needs, early feedback, and disciplined experimentation is heightened in high-risk pursuits. These items all relate to change and risk management. In a sense, the entire systems engineering process for such programs IS risk management.

There is no free lunch, but one tool that can help tie these latter recommendations together is the appropriate lifecycle model as described below.

4.4.1 Lifecycle Models

Life-cycle models can play an important role in the planning process, and thereby in the overall program success. While traditional viewpoints used the Waterfall Model, this "over-the-wall" method of systems engineering has been taboo for years in most complex programs. As Boehm [36] puts it, the waterfall model assumes that "requirements are knowable in advance of implementation." This violates the definition of the high-risk program used herein. Boehm even says explicitly of the Waterfall Model, "the requirements have no unresolved, high-risk implications."

Concepts of IPTs, concurrent engineering, and even transdisciplinary engineering have improved coordination amongst disciplines in many stages of development programs. For the high-risk program, this must be taken to yet another level, to coordinate not only the teams, but also the development phases in which the teams operate. This is where modern lifecycle approaches can help.

There is a growing awareness of lifecycle models in engineering literature. This development of more realistic lifecycle representations has sought to address several issues associated with complex and high-risk programs. No longer is the Waterfall Model acceptable for risky programs. Instead, evolutionary or spiral developments are preferred. The Evolutionary and Spiral Models utilize an iterative approach conducive to change and to *growing* requirements throughout the program. Stroup and Naylor [23] say,

> *"Big-bang development is not an effective means to develop highly complex safety critical systems. The name of the game is "risk reduction," which means it must be developed to an evolutionary life cycle process."*

Figure 17 shows a simple representation of the three lifecycle approaches discussed above.

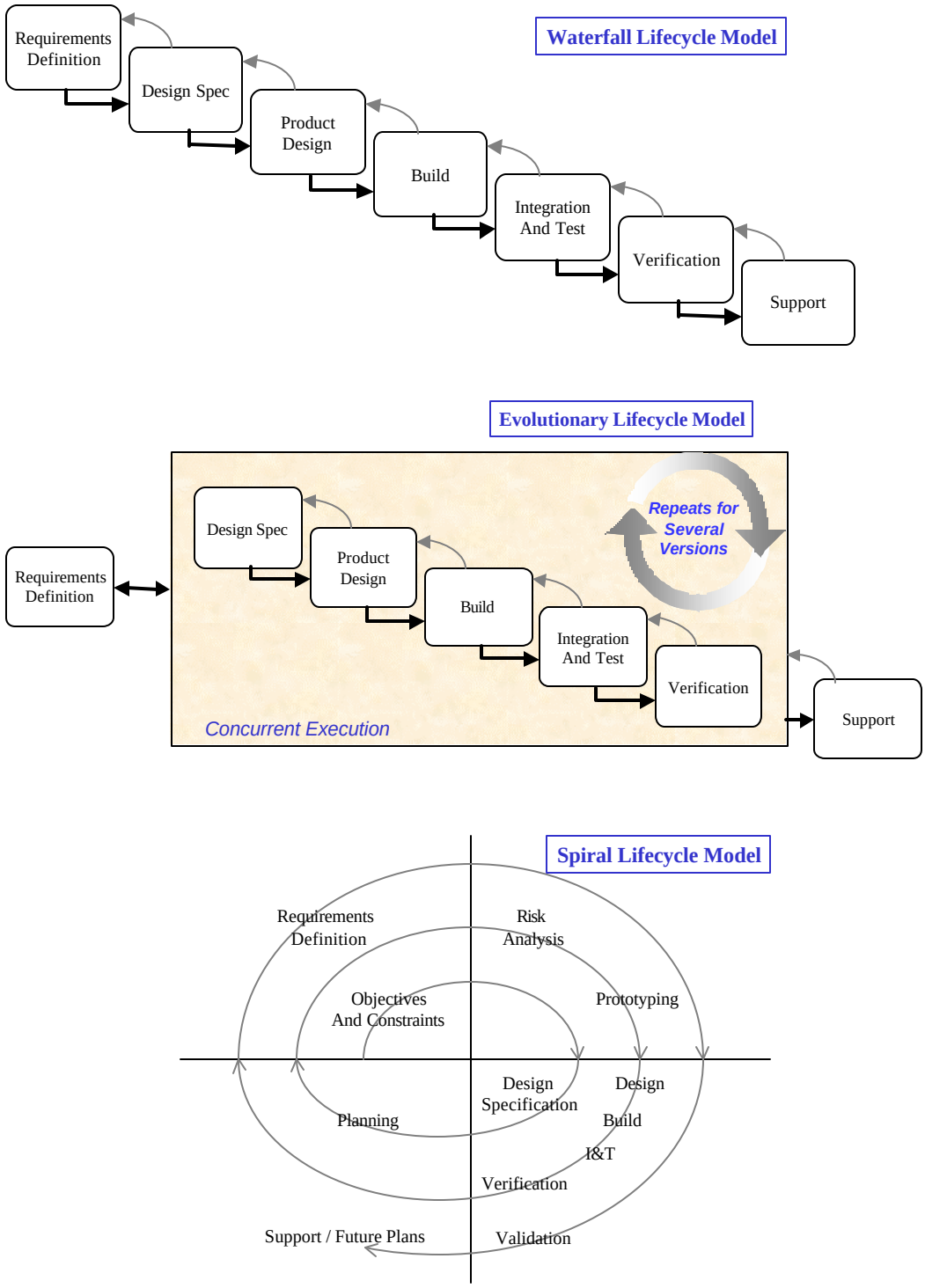


Figure 17. Common Lifecycle Models

Unfortunately, Evolutionary and Spiral models look good in theory, but are often not suited for the development and political constraints of particular programs. Schedule, cost, and even culture may dictate a more traditional looking plan. Redmiles [37] provides an attractive alternative for high-risk programs in his "Prototyping Model". An adaptation of this model is shown in Figure 18 below. The prototyping model is similar to the waterfall model except that it imposes a rapid development and test stage (prototyping) to aid in requirements and specification development. This provides an opportunity for both the acquisition of early feedback and the use of scientific experimentation principles, both of which have been identified as significant risk mitigators for high-risk programs.

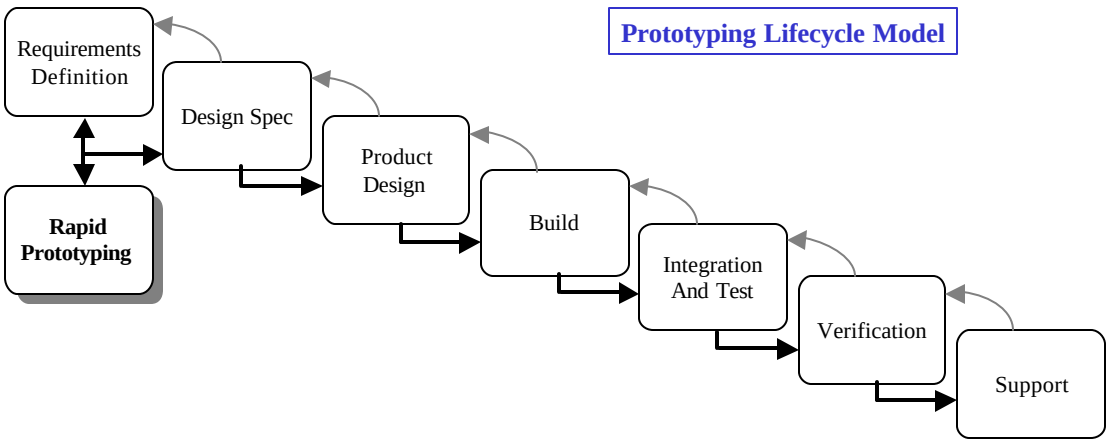
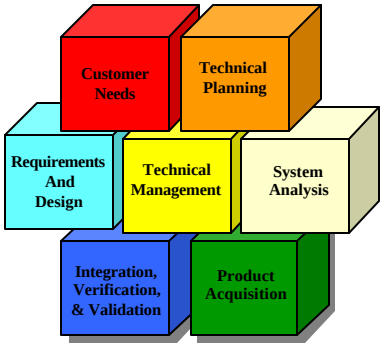


Figure 18. Prototyping Lifecycle Model

4.5 Feasibility of Process Change

For high-risk programs, it has been shown that fundamentals must be stressed. This starts with customer understanding, proceeds to technical planning, and propagates throughout. Based on the relative severity and quantity of lessons learned from the high risk programs presented herein, the seven elements of systems engineering could be qualitatively ranked in decreasing order of emphasis as follows:

1. Customer Needs
2. Technical Planning
3. Technical Management
4. Requirements and Design
5. System Analysis
6. Integration, Verification and Validation
7. Product Acquisition



Clearly, failure in any area can spell doom for a program. The "ranking" above is meant only to highlight those areas that are repeat offenders in higher-risk programs, not as a measure of effort or overall importance. It should not surprise any experienced systems engineer or program manager to see that the ranked order is identical to the order that these items were presented for definition in 3.2. The order is effectively chronological for program execution. For many programs, the up-front investment is critical. For risky programs, where change is likely, it is even more important to understand and manage the fundamentals up front. *But is action justified or needed to bring about change?* The answer is, "it depends."

DeFoe [27] reminds us that we must "Maintain process integrity but never let the process prevent the 'best' solution from being discovered or used – do whatever it takes to build in product quality." From that perspective, one could argue that almost any process is good enough – just adapt it to ensure success. This is the systems engineer's responsibility, regardless of the process detail.

Madness is rare in individuals, but in groups, parties, nations, and ages it is the rule. - Friedrich Nietzsche

Justifying the need for corporate-wide process changes for high-risk programs is highly dependent on the company, the product, and even the economic environment. As such, it is beyond the scope of this study. However, a few areas that would require analysis include:

Entrepreneurial – Is the ability to execute high-risk programs fundamental to the corporate strategy, or is it the exception? Can market share or technology leadership be enhanced?

Cost – As always, cost will likely dominate the decision. The cost of process change implementation must be weighed against the risk-based cost of program delays and failures.

Timeliness – With changing standards and procedures, it often seems that there's a new cure-all process every few years. Even without drastic overhauls, corporate development processes are always being updated and improved. Any special considerations for high-risk programs would need to be properly time-phased and coordinated with other process changes.

4.6 Process Change Recommendations

Nine "needs" have been qualitatively identified and validated in this report. While all are of substantial importance in the high-risk program, only a few are truly appropriate for specific process change or tailoring. The others are more reflective of the program culture, and the discipline to follow existing processes. Those items that generally do NOT require explicit process modification are:

Utilize TPMs and Reviews – As stated early, nearly every developer of complex systems has processes and reviews for minimizing development risk. These typically require, or at least suggest, tangible measures of progress and performance. The challenge is to make the TPMs meaningful and maintain program flexibility to adapt.

Establish Customer Communication Emphasis – This is more effort that process. While certain metrics might be imposed to force compliance, meaningful communications require the mutual understanding of importance and the commitment to follow-through.

Document Mission Usage Early – There is nothing to stop you. Program-wide emphasis on a thorough understanding of product requirements is needed in advance of design. It must also be updated throughout if dependent on design decisions.

Recognize and Reward Passion – The "process" associated with this would be better applied to program management. However, the systems engineer still has the responsibility to provide technical and team leadership. This includes the need to recognize the hard work of team members and foster their development and pride.

The four items above can be sufficiently accomplished within the framework of most standard systems engineering processes. The remaining five areas are more conducive to process *change*. Process changes associated with these items are addressed in the following subparagraphs.

4.6.1 Re-Plan Early and Often

Technical planning was given early in this report as one of seven fundamentals of systems engineering. For the high-risk program, flexibility in plans is critical, as the program is unpredictable by definition. Thus, detailed attention to sequential program steps may be of little value and can even lead the effort astray. Program planning should utilize the principles of evolutionary, spiral, or prototype development lifecycle models given in section 4.4.1 and associated literature. The entire process, including requirements definition, design, and integration/test, should be arranged accordingly. The underlying assertion in the planning is to *expect change*. In many cases, existing process steps will still apply. They must simply be timed appropriately for the phase of iterative development and progress.

4.6.2 Formalize Customer Penetration

While the processes all tell you to "know your customer," more emphasis is needed to ensure this goal is achieved. Although implementation varies from program to program, this could be accomplished with something as simple as the creation of a detailed org chart, or it could require face-to-face meeting between development team members (leads) and their counterparts within the customer organization. This latter approach (face-to-face) is preferred, but may not be necessary in all cases. Formal documentation and management review of the outputs of these tasks are required to ensure compliance. Process checklists could be created and/or adapted to verify that a thorough understanding of needs has been attained.

4.6.3 Obtain Early Feedback

Although is seems like a no-brainer, early prototyping and tests are essential. Yet, they are sometimes delayed by bureaucracy or omitted altogether for cost "savings". Knowing the high-risk program will experience considerable change along the way, process details should not delay or prohibit early experimentation. Often, it is the other important element of "planning" that is *used* to delay prototyping. The challenge is to balance thorough planning with early experimentation. Planning processes should be

tailored to gain and use technical data from prototypes. The process for the high-risk program should insist on the early feedback and allow tailoring flexibility on how it is achieved, as timing and objectives will vary greatly from program to program.

4.6.4 Employ Scientific Experimentation Principles

In an unpredictable situation, as is the case for the high-risk programs, change is a certainty. Analyses and test programs must be carefully planned to facilitate incremental progress in the face of this change. The foundation of knowledge must be established before the program is channeled toward a single solution or approach. This can be accomplished through process control that limits the scope of experimentation to a minimal number of variables per cycle. By following a logical pattern of smaller, lower risk exercises, steady progress is more likely, and the overall program will have higher confidence of maintaining plans and schedules.

Training may also be necessary, as Luftig and Jordan [32] point out:

> *"1. There is a general lack of formal education on the part of managers and supervisors in the area of industrial research methods."*
> *"2. Engineers…often have little education in experimental design methods, and even less in research design technology."*
> *"3. …the use of research and experimental methods on a day-to-day basis for data-based decision making is simply not common"*

4.6.5 Share the Vision

Program contracts and specifications may establish the overall development "vision", but they do not ensure each subordinate team and individual has a vision of their contributing work product. The lead systems engineer should communicate not only the collective program vision, but also the vision for each team, as decomposed from the top level. IPTs are usually established across function or physical boundaries

that help delineate deliverables and responsibilities. The vision of each IPT could be captured and documented alongside these IPT and/or organizational charts.

While team "visions" may be obvious in some programs, others may get lost in the details of volumes of governing specifications and reports. Care must be taken to create a vision that provides a simple mantra for everyday efforts and decisions. If this can not be readily accomplished, the purpose and objectives of the team are not sufficiently defined and performance of the team may be disjointed.

4.7 Academic Application of Principles

The concepts identified through research and analysis in this study are not foreign to any seasoned systems engineer. In fact, they are likely reflective of personal experiences, even outside the realm of engineering.

The process improvement topics are also not new to instruction in systems engineering. The following subparagraphs discuss how they are included in the fundamental principles of select courses taught during Texas Tech University's masters program through their Institute for Design and Advanced Technology (IDEATE) [38].

4.7.1 Systems Engineering Principles Course

Key goals of the Systems Engineering class were based on the general application of systems engineering principles, not specifically on the high-risk programs studied in this report. Much of the course material involved Raytheon's proprietary processes. However, course lead-in material [39] states that the systems engineer:

- "ensures the communication and coordination of requirements, design and interfaces among the implementing disciplines."
- "is the glue in a successful program"
- "acts as the user's advocate"
- "coordinates with the customer"

These are clearly related to the candidate process improvements associated with (1) customer communications, (2) customer penetration, (3) usage documentation, and (4) vision. Furthermore, course assignments involved detailed planning activities, hitting a fifth area.

4.7.2 Fundamentals of Transdisciplinary Design and Process Course

The Transdisciplinary Design Class objectives were primarily associated with identifying fundamentals associated with complex design problems involving multiple teams and disciplines. Thus, this course addressed approaches highly relevant to high-risk programs. Topics in the course material and presentation [40] included:

- A *good* design, from the user's perspective
- Requirements decomposition beginning with customer needs and constraints
- Establishing independence of requirements
- Decoupling existing designs
- Identification of the customer and understanding their environment
- Finding requirements overlooked by the customer

Much like systems engineering, these can be easily mapped to the candidate process improvements of (1) customer communications, (2) customer penetration, and (3) experimentation principles. Additional lecture during that class addressed the "psychology of designing" which included instruction related to (4) vision and (5) recognition.

Furthermore, assignments included Quality Functional Deployment (QFD) techniques that involved the use of technical measures and requirement interrelations akin to (6) TPMs. There was even a required submittal associated with the final masters project for (7) early feedback!

4.7.3 Technical Management Course

The final class to be specifically addressed is "Technical Management". The objectives for this course were broad in nature, including process measurement and control, risk management, organization,

and leadership. Class material [41] focused on actively seeking out program constraints, the importance of the planning phase, the need for tangible control measures, and leadership principles for motivating various personality types. This rounds out the presence of candidate process improvements by hitting (1) customer communications, (2) customer penetration, (3) planning emphasis, (4) TPMs, (5) vision, and (6) recognition and rewards.

4.7.4 Coursework Conclusions

Looking at just a few classes, all nine process improvements were touched upon at some level. Several were covered by *all three* classes! Even in the more specific courses of this masters program, like Image Processing and Decision Making, instructors do not unnecessarily load their students with details of topics until they have the fundamentals. The foundation must first be laid. Just like development processes, there is an optimum mix of detail and fundamentals.

The resulting conclusion from coursework evaluation is that the suggested process improvements are not prohibited by our collective understanding of engineering. Rather, they were affirmed in one or more classes. Their practical use is not so much a matter of new concept implementation, but rather the emphasis and disciplined pursuit of fundamental ideas.

4.8 Limitations of Findings

Use of a single program to critique and refine a general development process is clearly an inadequate sample. While hindsight may easily adapt a process for one program, it does not guarantee widely applicable recommendations. In addition to the application of experience on one program, preliminary recommendations were evaluated against additional programs and literature to refine and gain confidence in findings. It is left to the reader, however, to decide if and how the findings can be used in any given situation.

4.9 Validation Summary

The only source of knowledge is experience – Albert Einstein

Do you want a roadmap or a list of directions? Sure, in some cases you might provide specific instruction to a person and get them from here to there. In general however, you're better off teaching them how to read a roadmap and letting them go. Only in that way will they be able to negotiate the bumps in the road that will be common in the high-risk program. Nine recommendations have been provided to help.

In the very first paragraph of his #1 Bestseller, "All I Really Need to Know I Learned in Kindergarten" [42], Robert Fulghum acknowledges his quest to write a personal statement of belief; a Credo. When he was young, the author says, the statement ran for many pages, trying to cover every base, with no loose ends. He says, "It sounded like a Supreme Court brief, as if words could resolve all conflicts about the meaning of existence."

In the second paragraph, he discusses the simplification of the Credo that came in more recent years. He recalls the events that lead to the realization that the truly useful knowledge isn't complicated at all, and he learned it long ago. Hence, the title of his book. He says, "wisdom is not at the top of the graduate-school mountain, but there in the sandpile at Sunday School." (Additional excerpts from Fulghum are provided in Appendix E.)

Likewise, good systems engineering on high-risk programs is not about the minute details, but rather the passionate pursuit of the basics we all know. Both specific and general recommendations for improvement in this area have been stated and substantiated in this chapter. The result was nine generalized items to consider, four of which require only process emphasis, and five which may include process change.

Surprisingly, these five *changes* contain a common thread – plan for change. This requires an in-depth understanding of program fundamentals, as well as a development approach that promotes steady progress, feedback, and adaptation in all areas. Know where you want to go, learn and adapt!

CHAPTER - V
5. CONCLUSIONS

Like a struggling football team, engineers on high-risk programs must *practice* the basics. A 200-page playbook is of little value to a team that is having difficulty blocking and tackling. Indeed, there is a knee in the curve where additional process detail becomes counter-productive if over-emphasized. Since such a trend in process effectiveness changes from program-to-program, even day-to-day, there is little value trying to quantify the results for this discussion. Figure 19 provides a notional illustration.

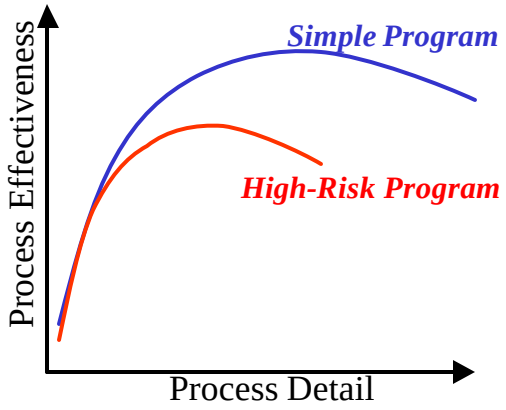


Figure 19. Notional Process Effectiveness Trend

The effect of diminishing returns on process application is not so much because the details are unimportant, but rather because available resources must be properly applied for the phase and risk of a program. As an exaggerated example, if you do not know who your customers are or what they want, you do not yet need to worry about subsystem specification control processes. McClinton [28] affirms, "Since we do not have unlimited resources, it is vital that resources are allocated to the critical tasks and not spent on tasks that become the playground of the analyst and designers…a thing not worth doing is not worth doing well."

Another way to view the figure above is to liken it to the traditional bathtub curve of reliability for hardware products. When you start from scratch, process improvements and detail pay big dividends. As more and more resources and constraints are put on the process application work, the initial intent and flexibility is lost and the process (or program) begins to break down. High-risk programs must be especially aware of the basics, as the details will be in flux as the design progresses. The process should be tailored to the appropriate level, based on the command of program fundamentals.

Re-visiting the football analogy, the playbook also doesn't score touchdowns. Process flexibility must permit systems engineers to engineer systems, even in difficult circumstances. Yelland [2] reminds us that a good systems engineering process is not a silver bullet, a cookbook approach to success, or an excuse to stop thinking! Tanik and Chan [43] also recognize the role of people in their designs, saying, "In designing large-scale systems we should do everything possible to integrate the human element as a fundamental part in our development process models…"

Success requires vision, passion, discipline, and risk. For high-risk programs, these are emphasized through the nine improvement areas that have been identified and substantiated herein.

1. *Re-Plan Early and Often*
2. *Utilize Technical Performance Measurements and Reviews*
3. *Establish Customer Communication Emphasis*
4. *Formalize Customer Penetration Verification*
5. *Document Mission Usage Early*
6. *Obtain Early Feedback*
7. *Employ Scientific Experimentation Principles*
8. *Share the Vision*
9. *Recognize and Reward Passion*

CHAPTER - VI
6. RECOMMENDATIONS

6.1 Philosophical

In addition to process priorities, one thing that was learned from the creation of this report it is to spread the vision and passion about a project throughout the development team with as much or more investment as you spread corporate processes and management control. Once this is done, the skills of the motivated engineer will be the most valuable risk mitigation tool, not the policing of process details. The systems engineer is the root of the systems engineering process. Not the other way around.

The whole of science is nothing more than a refinement of everyday thinking – Albert Einstein

In Stephen Covey's popular book [44], he gives us the "the seven habits of highly effective people." While these don't map directly into any one set of process improvements that have been presented, they do substantiate topics that have been shown to be important on high-risk programs. Covey's widely acclaimed "habits" are provided in parentheses below, along with their relation to the findings herein:

1. "Be proactive" relates to the leadership and passion needed for a successful program
2. "Begin with the end in mind" stresses the importance of the common vision
3. "Put first things first" indicates the need for early planning with emphasis on fundamentals
4. "Think win/win" confirms the need for common goals and understanding with the customer
5. "Seek first to understand, then to be understood" reiterates the value of a thorough understanding of customer and user cultures and program objectives
6. "Synergize" is the teamwork and management required to execute efficiently
7. "Sharpen the saw" relates to the process improvement aspect of continuing programs

Even the creation of this master's report is an excellent example of the findings presented herein. Early in the program, students were asked to select a topic, then later to develop the outline, requirements, and even a detailed schedule. Few were likely successful in accurately predicting schedule tasks. This was

not for lack of logical thought or lack of process. Surely most schedules were entirely achievable. The problem with this early scheduling (at least for one student) was the simple "fact" that the schedule involved the creation of something new, the exact nature of which could not be predicted at the time. It was not until work began on various phases of the project that the true challenges and results could be predicted – a.k.a., a wicked problem.

Consequently, the flow of execution steps for the development of this report was not accomplished through the task-by-task processing of premature schedule items. Rather it was accomplished through the nurturing of the more basic elements of vision, passion, discipline, and risk. A **vision** of the overall project is needed before anything can begin. In this case, it was needed to overcome "writer's block". Once you have the big-picture vision, **passion** promotes action. Procrastination is a very real part of engineering. It can be mitigated through force (e.g., deadlines), but instilling a sense of passion about the positive goals of a project will always result in a better end product and higher team morale. Finally, **discipline** is required to maintain efficient and controlled activity. This is the one area where traditional development processes can help, even on high-risk programs. Of course, the **risk** was accepted when students signed up for the master's program!

6.2 Process Improvements Summary

The impetus for this study was the idea that processes do a disservice to their users when excruciating detail is included. It was thought that process breakdowns could be attributed to a few simple concepts and even to the presence of too much process. Although this was partially validated in qualitative terms for an extreme case, the outcome was not as expected. *High-risk programs fail from reasons that are more attributable to the systems engineer than to the systems engineering process.*

Even so, several items have been identified that warrant consideration on high-risk programs to improve process success. These nine items were addressed in several sections in this report. It is recommended that they be carefully considered when embarking on any risky pursuit. Each program will

have to adapt the definitions and goals to their circumstances, but care must be taken to avoid omission or shortcuts associated with any of the following, which were detailed in section 4.6.

1. Re-Plan Early and Often
2. Utilize Technical Performance Measurements and Reviews
3. Establish Customer Communication Emphasis
4. Formalize Customer Penetration Verification
5. Document Mission Usage Early
6. Obtain Early Feedback
7. Employ Scientific Experimentation Principles
8. Share the Vision
9. Recognize and Reward Passion

Detailed implementation, training, and costs associated with company-specific process changes are beyond the scope of this study. However, it is believed that much of the benefit can be gained by simply understanding the history of similar programs, and planning for flexibility and the mastery of systems engineering fundamentals. One enabler for such planning was found to be the use of evolutionary or spiral lifecycle models that reflect the evolving nature of may high-risk programs. Although these models use iterative approaches to development that may suggest increased cost and schedule, they often have the opposite effect in high-risk programs, due to their risk-averse nature.

The bitterness of low quality still remains after the sweetness of low cost is forgotten –author unknown

This report looked at systems engineering from a particular angle. Many of the basic aspects are not explicitly mentioned, but are nonetheless important for success. Good documentation management, thorough requirements flowdown, IV&V planning, etc., can not be overlooked in favor of the nine emphasis areas above. The nine items simply highlight common problems specific to high-risk programs.

6.2.1 Opportunities for Further Study and Applications

If I had my life to live over again, I'd be a plumber. – Albert Einstein

During the course of research and analysis, several topics were encountered that would have furthered the understanding of systems engineering on high-risk programs, but were well beyond the scope of this report. A few of the most applicable ones are stated below for possible future expansion of the subject:

1. Statistical evaluation of lessons learned from a larger sample of high-risk programs to prove and quantify re-occurrence of many basic problems
2. Identification of metrics to facilitate recognition of high-risk programs and/or measure program nature for lifecycle application
3. Identification and evaluation of specific process metrics to measure implementation of recommendations herein.
4. Detail means for various lifecycles to be instantiated in program-level processes, development contracts, and specifications for high-risk programs.

6.3 Closing

This project has been something of an iteration of the basic systems engineering process to adapt it to particularly challenging programs. The trials of coping with and eliminating radome ice in an airborne environment were used as a medium to critique and refine the process. As a result of this case study, and research of open literature, several process areas were found to require special emphasis and possible modification for high-risk development efforts.

Not everything goes by the book. Often it seems nothing goes by the book. High-risk programs must utilize development processes in a manner that acknowledges interim failures and accommodates course changes. The development must be treated almost as an iterative scientific experiment, and not just as a turn-the-crank design task. The popular cartoon in Figure 20 rounds out these findings. There is no *one-size-fits-all* process. On the high-risk program, the application of process detail must facilitate, but not complicate or delay, progress towards fundamental program understanding and objectives.

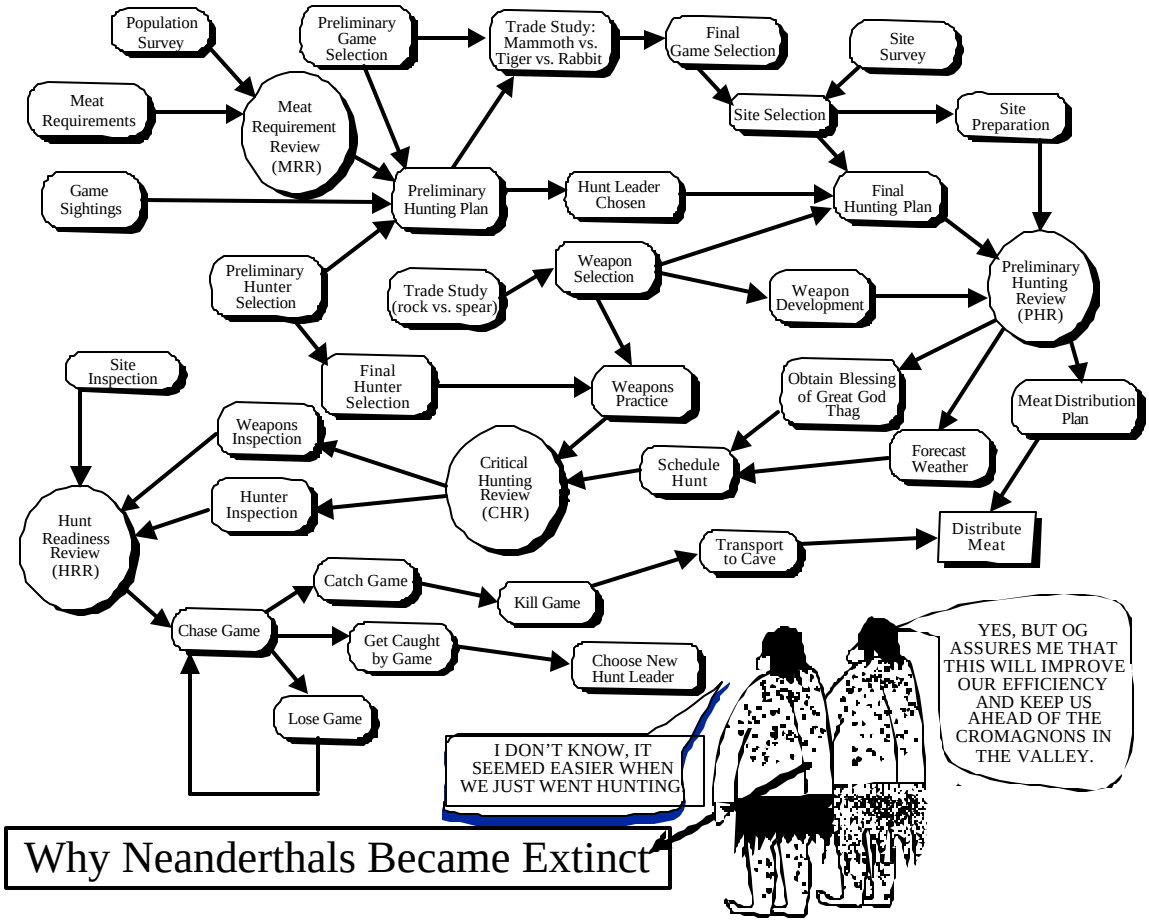


Figure 20. Right-Sizing the Process [45]

Thankfully, you can't replace people with process. If you're not real hungry and you don't know "how to hunt," the figure above may have some merit. The skilled hunter might, however, be rendered useless. Likewise, the systems engineer remains the key element of the systems engineering process. On high-risk programs, particular attention should be applied to the people, in an effort to motivate execution of the fundamentals. Assemble a skilled workforce, and then treat them as such in the process and program tailoring. Plan for steady progress on the high-risk program to ensure ultimate success!

Imagination is more important that information. – Albert Einstein

7. REFERENCES

[1] INCOSE, General Website Information, http://www.incose.org

[2] Yelland, Brad, "Introduction to Systems Engineering", University of Sydney, Australian Center for Field Robotics, May 2002, http://cannabis.acfr.usyd.edu.au/teaching/1st-year/mech1540/material/lectures/Systems Engineering Hand Outs.pdf

[3] Arunski, Karl, et al., "Systems Engineering Overview", Presentation to the Texas Board of Professional Engineers, INCOSE, September 1999, http://www.incose-wma.org/info/docs/2000/feb00pres.ppt

[4] Tanik, Murat and Ertas, Atila, "Design as a basis for Unification: System Interface Engineering", PD-Volume 43, pp. 113-114, Computer Applications and Design Abstraction, ASME, 1992.

[5] EIA-632, Processes for Engineering a System, Dec 1998, http://www.geia.org/sstc/G47/page5.htm

[6] EIA/IS-731, *Systems Engineering Capability Model*, January 1999, http://www.incose.org/stc/news731.htm

[7] IEEE 1220, Application and Management of the Systems Engineering Process, 1998.

[8] ECSS-E-10A, System Engineering, April 1996, www.estec.esa.nl/ecss/standard/status.html

[9] ISO 15288, System Life Cycle Processes, JTC1/SC7/WG7 (under development).

[10] MIL-STD-499, Systems Engineering Management, July 1969, http://www.incose.org/stc/mil499.htm

[11] MIL-STD-499A, Engineering Management, 1974, http://www.incose.org/stc/mil499A.htm

[12] INCOSE, Standards Evolution, http://www.incose.org/stc/standards_evolution.htm

[13] Carnegie Mellon Software Engineering Institute, CMMI General Information, http://www.sei.cmu.edu/cmmi/general/general.html

[14] Edwards, Bill, "A Systems Engineering Primer for Every Engineer and Scientist", Lawrence Berkeley National Laboratory, First Edition, December 2001. http://engineering.lbl.gov/sy/documents/SEPrimer.pdf

[15] Yeh, Raymond T., "System Development as a Wicked Problem", International Journal of Software Engineering and Knowledge Engineering, Volume 1, No. 2: pp. 117-130, February 1991.

[16] Montgomery, Douglas C., Design and Analysis of Experiments, 3rd Edition, John Wiley and Sons, New York, NY, 1991, pp. 9-11.

[17] Cusick, Kerinia, "A Collection of Integrated Product Development Lessons Learned", International Council on Systems Engineering Conference, 1997, http://www.secat.com/download/locked_pdf/ipd_ll_lkd.pdf

[18] Weiss, Joseph and Wysocki, Robert, 5-Phase Project Management: A Practical Planning & Implementation Guide, Perseus Publishing; January 1992.

[19] Nevins, James L. and Winner, Robert I., "Ford Motor Company's Investment Efficiency Initiative: A Case Study", Institute for Defense Analyses, April 1999, http://www.cit.gu.edu.au/~bernus/taskforce/Detroit97worksh/presentations/fordcase.pdf

[20] "Project XL 1999 Comprehensive Report", U.S. Environmental Protection Agency, 1999, http://www.epa.gov/projectxl/report6.pdf

[21] Mitretek Systems, "Key Findings from the Intelligent Transportation Systems (ITS) Program", U.S. Department of Transportation, Federal Highway Administration, 1996, http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/bt01!.pdf

[22] Kasser, Joseph and Cook, Stephen, "A Framework for Requirements Engineering in a Digital Integrated Environment", University of South Australia, Version 1.4, April 2000, http://www.seecforum.unisa.edu.au/Presentations/FREDIE.ppt

[23] Stroup, Ron and Naylor, Warren, "Cost & Schedule – The Overlooked Hazards", Federal Aviation Administration, http://www.faa.gov/aio/common/documents/Safety/CostSked.doc

[24] Tosney, W.F., "Faster, Better, Cheaper; An Idea without a Plan", The Aerospace Corporation, November 2000, http://www.incose-wma.org/docs/chapter/Apr02CentrevillePresentation.pdf

[25] George Mason University, "A Tale of Systems Engineering", Hypothetical Scenario, Systems Engineering Operations and Research Department, http://www.gmu.edu/departments/seor/insert/story2/lessons.htm

[26] Evans Engineering, "Space Engineering Lessons Learned", Engineering Directorate, November 1989, http://users.erols.com/ee/sysengll.htm

[27] DeFoe, J.C., "Pragmatic Principles of Systems Engineering", INCOSE SE Practice Working Group, WMA Chapter, January 1993, http://www.incose.org/workgrps/practice/pragprin.html

[28] McClinton, David F., "The Unwritten Laws of Systems Engineering", Fourth Annual Symposium of the National Council on Systems Engineering, 1994, http://users.erols.com/ee/sysenlaw.htm

[29] Yeh, Raymond, et al., Zero Time: Providing Instant Customer Value – Every Time, All the Time!, John Wiley & Sons, Inc., New York, NY 2000, pp. 140, 186, 255.

[30] Polya, George, "Guessing and Scientific Method", Mathematical Discovery, John Wiley and Sons, 1981.

[31] Sylvester, Ric, "Acquisition Reform Experience w/new DoD 5000 Model", 34th Annual DoD Cost Analysis Symposium, 2001, http://www.ra.pae.osd.mil/adodcas/Presentations%202001/DoDCAS_AcqRef.PDF

[32] Luftig, Jeffrey T. and Jordan, Victoria S., Design of Experiments in Quality Engineering, McGraw-Hill, New York, NY, 1998, p. 1.

[33] Warfield, J., A Science of Generic Design: Managing Complexity Trough Systems Design, Second Edition, Iowa State University Press, 1994.

[34] Martin, James, "Overview of the EIA-632 Standard – Processes for Engineering a System", EIA-632 Working Group Chairman, September 1998, http://www.geia.org/sstc/G47/632-web.pdf

[35] Ertas, Atila and Jones, Jesse C., The Engineering Design Process, 2nd Edition, John Wiley & Sons, Inc., New York, NY, 1996, pp. 7, 52-53.

[36] Boehm, Barry, "Spiral Development & Evolutionary Acquisition: Where are We Today?", University of Southern California, Center for Software Engineering, September 2000, http://www.sei.cmu.edu/cbs/spiral2000/september/Boehm/sld001.htm

[37] Redmiles, David F., "ICS 121 Software Tools and Methods", University of California at Irvine, Fall 1999, http://www.ics.uci.edu/~redmiles/ics121-FQ99/lecture/three.pdf

[38] Ertas, Atila, "Institute for Design and Advanced Technology (IDEATE)", Texas Tech University, Undated Brochure.

[39] Norby, Greg, "Systems Engineering Principles", Course Notes, Texas Tech University, December 2001.

[40] Tate, Derrik, "Fundamentals of Transdisciplinary Design", Course Notes, Texas Tech University, January 2002.

[41] Smith, Michael F. "Technical Management: ENGR 5000", Course Notes, Texas Tech University, April 2002.

[42] Fulghum, Robert, All I really need to know I learned in Kindergarten, Ivy Books, 1988.

[43] Tanik, Murat M. and Chan, Eric S., Fundamentals of Computing for Software Engineers, Van Nostrand Reinhold, New York, NY, 1991, pp. 233-234.

[44] Covey, Stephen R., The 7 Habits of Highly Effective People, Simon & Schuster, 1st edition August 1990.

[45] Davis, Gary, "A Demonstration of the Evaluation Process for the U.S. Terminal Area", CNS/ATM Focused Team Analysis Process, United Airlines, March 1997, http://www.boeing.com/commercial/caft/reference/meetings/97_03_19/07_garydavis.ppt

[46] Sowa, John F., "Writing Advice", IBM Systems Research, (undated IDEATE class handout).

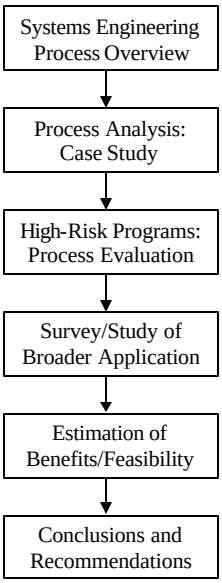
8. ACRONYMS AND ABBREVIATIONS

BIT	Built-in-Test
CMMI	Capability Maturity Model Integration
DEMO	Demonstration
DoD	Department of Defense
ECSS	European Cooperation for Space Standardization
EIA	Electronics Industry Association
FPD	Freezing Point Depressant
IDEATE	Institute for Design & Advanced Technology
IEEE	Institute of Electrical and Electronic Engineers
INCOSE	International Council On Systems Engineering
IPDS	Integrated Product Development System
IPT	Integrated Product Team
IS	Interim Standard
ISO	International Standards Organization
I&T	Integration and Test
IV&V	Integration, Verification, and Validation
NASA	National Aeronautics and Space Administration
ORG	Organization
PDR	Preliminary Design Review
QFD	Quality Functional Deployment
RF	Radio Frequency
RQMTS	Requirements
SE	Systems Engineering
SEI	Software Engineering Institute
SOW	Statement of Work
TPM	Technical Performance Measurement

9. APPENDICES

APPENDIX A: INITIAL PROJECT ORGANIZATION

Although not meant to duplicate the Table of Contents, the following outline provides the initial structure of this report. It is included here for reference to provide the summary framework for the analyses.



Background

- Research and Explain the Systems Engineering Process
- Define the Generic "High-Risk" Program
- Provide the Customer Role/Perspective on Such Programs
- Identify a Candidate Program for Case Study
- Report on Literature Search of Similar Efforts

Process Analysis

- Capture Basis of Primary Process Goals and Content
- Map and Contrast Process Fundamentals to Process Detail
- Detail the Nature of the Technical Trades on the Case Study
- Identify Problem Symptoms Encountered in Case Study
- Speculate on Root-Cause of Problems from Process Perspective
- Summarize Analysis Findings
- Evaluate Pros and Cons of Process Improvements
- Identify Likely Process Changes for Case Study Program

Validation

- Research Lessons Learned from Other Sources/Programs
- Estimate Benefits of Process Changes on Case Study Program
- Define and Refine Generalized Process Improvement Candidates
- Examine Feasibility of Change
- Finalize Change Recommendations
- Examine Generic Application

Conclusions

- Summarize Findings

Recommendations

- Discuss Systems Engineering Process Improvements
- Identify Areas for Further Study and Application

APPENDIX B: LESSONS LEARNED PROBLEM TABULATION

Below, Table 9 contains source information for external lessons-learned research. On the next few pages, Table 10 categorizes information from all ten sources into fourteen summary items. The "Source" and "Table Code" columns in Table 9 provide a mapping between the source list in Section 7 of this report, and the source designators used in Table 10.

Table 9. Lessons-Learned Sources

Source	Table Code	Author	Title
[17]	A1	Cusick, Kerinia	"A Collection of Integrated Product Development Lessons Learned"
[18]	A2	Weiss and Wysocki	5-Phase Project Management
[19]	A3	Nevins and Winner	"Ford Motor Company's Investment Efficiency Initiative: A Case Study"
[20]	A4	U.S. EPA	"Project XL 1999 Comprehensive Report"
[21]	A5	Mitretek Systems	"Project XL 1999 Comprehensive Report"
[22]	A6	Kasser and Cook	"A Framework for Requirements Engineering in a Digital Integrated Environment"
[23]	A7	Stroup and Naylor	"Cost & Schedule – The Overlooked Hazards"
[24]	A8	Tosney, W.F.	"Faster, Better, Cheaper; An Idea without a Plan"
[25]	A9	George Mason Univ.	"A Tale of Systems Engineering" (hypothetical)
[26]	A10	Evans Engineering	"Space Engineering Lessons Learned"

Table 10. Lessons-Learned Summary Mapping

	COMBINED CATEGORIES	Team Buy-In and Commitment	Management Support and Participation	Emphasis on Early Planning	Process Control Measures (e.g., TPMs)	Improved Leadership	Improved Communication	Enhanced Stakeholder Involvement	Recognition and Reward	Consistent Vision and Objectives	Adequate Resources	Simplify Where Possible	Maintain Flexibility	Clear and Reasonable Requirements	Emphasis on Testing
Source	**LESSONS LEARNED**														
A1	Focus on People and Personal Commitment	X													
	Organization Consistent with Company Goals		X							X					
	Emphasis on Planning			X											
	Focus on Measurement and Processes				X										
	Careful Monitoring of Decision Making Process		X		X										
	Leadership Dedicated to IPD		X			X									
	Communication and Data Sharing		X				X	X							
	Stakeholder Involvement							X							
	Rewards and Recognition								X						
A2	Program Strays from Its Original Goals									X					
	Team not Communicating					X	X								
	Program not Tracked to a Plan			X											
	Insufficient Resources										X				
	Program Under-Budgeted										X				
	Program Plan Lacks Detail			X											
	Program is Unstructured			X		X									
	No One is in Charge					X									
	Only Program Team Interested in the Result	X													
	A Solution in Search of a Problem									X					
A3	Changing Mind Sets	X	X												
	Understanding Need for Change	X								X					
	Strengthening Management Support		X												
	Creating Aligned Objectives			X						X					
A4	Stakeholder Education and Training						X	X							
	Technical Assistance							X							
	Early Site visits			X			X	X							
	Trade Studies Underestimated			X							X				
	Broader Employee Involvement	X													
	Involvement of Too Many Slowed Negotiations		X			X									
	Active Support Needed from Management		X												
	True R&D is Costly and Time Consuming			X							X				
	Clear Project Goals Early			X						X					
	Clear Lines of Communication and Decision					X	X								
	Build Stakeholder Involvement							X							
	Establish Trust						X	X		X					
	Simplify Process					X						X			
	Involve Program Offices Early and Throughout			X				X							
	Meet face-to-face Frequently						X	X							
	Speed Management Review Times		X									X			
	Well-Defined Team Roles and Responsibilities	X		X			X								

Table 10. Lessons-Learned Summary Mapping (continued)

Source	LESSONS LEARNED	Team Buy-In and Commitment	Management Support and Participation	Emphasis on Early Planning	Process Control Measures (e.g., TPMs)	Improved Leadership	Improved Communication	Enhanced Stakeholder Involvement	Recognition and Reward	Consistent Vision and Objectives	Adequate Resources	Simplify Where Possible	Maintain Flexibility	Clear and Reasonable Requirements	Emphasis on Testing
	COMBINED CATEGORIES														
A5	Partnerships require building trust, understanding, commitment, and communication	X						X							
	Roles and Responsibilities Identified Early		X	X											
	Good Leadership and Commitment Essential					X									
	Integrators/Evaluators Brought in Early	X					X	X							
	Initiate Eval Process in Planning Phase	X		X											
	Complex Projects Require Flexibility												X		
	Contracting Flexibility is Important												X		
	Operational Tests Need Mgmt Buy-In		X												
	Inter-agency Cooperation is Facilitated by having an Advocate in Every Key Agency						X	X							
	Demonstratable Benefits are Critical to All Participants	X								X					
	Participate by All is Critical to Success	X					X	X							
	It is Important to Make Progress (keep on schedule)				X										
A6	User Involvement							X							
	Executive Management Support		X												
	Clear Statement of Requirements													X	
	Proper Planning			X											
	Realistic Expectations									X				X	
	Smaller Project Milestones				X										
	Competent Staff	X									X				
	Ownership	X													
	Clear Vision and Objectives									X					
	Hard-Working, Focused Staff	X													
	Understand What is Needed									X				X	
	User/Stakeholder Engagement Throughout Product Life							X							
	Capture User Focus in Clear Requirements							X						X	
	Have Process to Manage Inevitable Evolution/Change				X										
A7	Don't compromise testing for cost and schedule														X
	Create Realistic Plans			X										X	
	Provide Good Development Oversight		X		X	X									
	Establish Clear and Realistic Requirements													X	
	Consistent Goals Among Stakeholders							X							
	Coordination Among Stakeholder						X	X							
	Proper Contracting		X	X											
	Evolutionary Life Cycle				X										

Table 10. Lessons-Learned Summary Mapping (continued)

Source	LESSONS LEARNED	COMBINED CATEGORIES	Team Buy-In and Commitment	Management Support and Participation	Emphasis on Early Planning	Process Control Measures (e.g., TPMs)	Improved Leadership	Improved Communication	Enhanced Stakeholder Involvement	Recognition and Reward	Consistent Vision and Objectives	Adequate Resources	Simplify Where Possible	Maintain Flexibility	Clear and Reasonable Requirements	Emphasis on Testing
A8	Insufficient Resource Allocation											X				
	Adequate Sometimes Better than Optimized												X			
	Schedule Drives Technical Decisions				X	X								X		
	Costs Saved Through Teaming of Engineering and Science													X		
	Contract Delay for Planning did NOT Impact Overall Schedule				X											
	Need for More Testing is Recurring Theme															X
	Oversight and Communication Between Contractors							X	X							
A9	Follow Design/Development Model				X	X										
	Improve Team Communications							X								
	Baseline Requirements w/Stakeholder Agreement								X						X	
	Assess Feasibility as Part of Planning				X										X	
	Improve Configuration Management				X	X		X								
A10	Lack of Clear Requirements														X	
	Poor Interface Definition							X							X	
	Inadequate Test PlanS and Consideration of Testability				X											X
	Failure to "Think the Design Through"										X				X	
	Failure to Model/Simulate before Fabrication				X	X										
	Insufficient Mission Ops Consideration in Rqmts Def.								X							
	Poor Design Documentation							X								
	Failure to Ensure Qualified Parts Available					X										
	Unnecessarily Complex Design												X			
	Improper Functional Partitioning							X	X						X	
	Lack of Unbiased Trade Evaluations								X					X		
	Inadequate Test at Subsystem Level															X
	Lack of Communications Within Development Team							X								
	Bowing to Schedule Pressure					X										

APPENDIX C: TOPIC SELECTION BASES

Topic selection for this report occurred early in the master's program. Per course handout [46], a topic of general nature was chosen so as to be of general interest to the master's class. Final selection involved the brainstorming of candidate subjects, then parsing them based on academic, professional, and company benefit. The use of case studies to evaluate Systems Engineering on high-risk programs was selected as it promised a good combination of all three categories.

Academic: Scope and deliverables satisfy Texas Tech University requirements and are conducive to demonstration of student's systems engineering skills.

Professional: Approach requires in-depth look at internal and external Systems Engineering processes, definitions, and standards.

Company/Job: Conclusions and findings support improvement of Raytheon's leadership position in the associated customer community

APPENDIX D: PRELIMINARY REPORT REQUIREMENTS

In the Systems Engineering Class, Top-Level Requirements for the report were developed from the perspective of the graduate program, minimizing topic-specific content. Although there are a few self-imposed items, the list of top-level requirements could apply to nearly any student. The majority of the verification responsibilities fall to TTU-selected faculty (i.e., compliance permits degree completion). It is provided as an appendix for consideration of project plans versus results.

Requirement	Verification Responsibility
Satisfy TTU Report Requirements	
Formal written report	Faculty
20+ hours of internet communication	Faculty
IDEATE prescribed format	Faculty
Submit report prior to final site visit	Faculty
Contain no classified information	Raytheon Mgmt
Deliver no later than mid-October	Faculty
Topic is well-defined and consistent with program	Faculty
Student develops and follows plan for completion	Faculty
Creativity is evident in student's work	Faculty
There is interaction w/literature and experts	Faculty
Work provides comprehensive and detailed presentatio n of idea to be developed	Faculty
Satisfy TTU Presentation Requirements	
25 minute oral presentation on project	Faculty
5 minute Q/A	Faculty
Limit to 20 main slides	Author
Limit to 7 bullets/points/slide	Author
Presentation due mid-October	Faculty
Provide Academic and/or Professional Benefit	Author
Adequately Demonstrate Mastery of Subject	Faculty
Involve Topic Pertinent to Raytheon Business	Author
Include Backup Slides for Q/A in Presentation	Author

APPENDIX E: EXCERPTS FROM FULGHUM

Fulghum [42] shares these and other pearls in his book "All I Really need to know I learned in Kindergarten." Despite their broad nature, they can be creatively interpreted for many systems engineering assignments.

- Share Everything
- Play Fair
- Don't Hit People
- Put Things Back Where You Found Them
- Clean Up Your Own Mess
- Don't Take Things That Aren't yours
- Say You're Sorry When You Hurt Somebody
- Wash Your Hands Before You Eat
- Flush
- Warm Cookies and Cold Milk are Good for You
- Live a Balanced Life
- When you go out into the World, watch out for traffic, hold hands, and stick together
- Be aware of wonder

Fulghum also gives the following about hedging your bets (or risk management!):

- Always trust your fellow man. And always cut the cards
- Always trust God. And always build your house on high ground.
- Always love thy neighbor. And always pick a good neighborhood to live in
- The race is not always to the swift, not the battle to the strong, but you better bet that way
- Place your bet somewhere between turning-the-other-cheek and enough-is-enough-already
- Place your bet somewhere between hast-makes-waste and he-who-hesitates-is-lost
- About winning: It isn't important. What really counts is how you play the game
- About losing: It isn't important. What really counts is how you play the game
- About playing the game: Play to win!