Design and Management of a System of Systems: A Systems Engineering Approach

By

Bradley Whittington

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

29 June, 2001

ACKNOWLEDGEMENTS

I would like to express my sincerest thanks to Dr. Atila Ertas for taking the time to develop this program and give professionals a chance to further their education once again. His guidance throughout the last year has been invaluable. I would also like to thank Dr. Timothy Maxwell who has also taken a majority of his weekends the past year ensuring that each student reached their fullest potential. Further thanks go to the other members of my committee, Dr. Tanik and Dr. Smith for their helpful criticism and advice.

I would like to express thanks to Raytheon in Garland, Texas for sponsoring this program. As evidenced later in this paper, I believe continuous training to be paramount to success.

I would like to thank my Mother for her love, unrelenting support and encouragement throughout the years.

I would like to thank my Father for his love and support throughout the years. I have worked with industry leaders, Nobel Laureates, rocket scientists and more and, although he chooses not to flaunt this fact, he is one of the most brilliant men I have ever known.

Finally, I would like to thank my wife for her unconditional love and support throughout this entire process. She endured untold late nights, missed weekends and more. She deserves everything that I could give her and more, so, as a start, I dedicate this to her.

TABLE OF CONTENTS

DISCLAIMER

ABSTRACT

In today's world of complex systems, the need for Systems Engineers who understand the system as a whole, plus have knowledge of other "intangibles" i.e. Program Management, Specialty Engineering, Configuration Management, etc. is at a premium. With this system complexity, however, is cost. To offset this cost, more and more customers are taking legacy systems and combining them to make a new system. These "systems of systems" are obviously much more complex than their predecessors. The Systems Engineers that were once responsible for a whole system; are now responsible for a component of the new system. This paper will delve into the design of these systems of systems and discuss how a Systems Engineer, particularly a Lead Systems Engineer (LSE), should design and manage one of these systems. The subject matter will be at a high level so that applicability is universal.

LIST OF FIGURES

LIST OF TABLES

CHAPTER I
INTRODUCTION

Technology and the world around us are changing rapidly. What was once thought of as "high tech" is now non-existent or only thought of in comical tones. This new technological outbreak has developed into a perplexing puzzle for engineers. With this new technology comes an increase in price, how then, is it possible to deliver new systems capable of feats of wonder and still manage the cost of the system?

One way of dealing with this problem is being used frequently now in the military and is gaining recognition in the civilian world as a viable solution. The existing systems that a customer uses are connected creating one large super system – a system of systems! These systems usually offer solutions to the customer's immediate needs while positioning them for future growth and technology surges.

This seems like a simple cost-effective solution, and it can be if the Systems Engineering required to create a system of systems is properly performed. As an example; imagine a cell phone. Certainly it is a sophisticated device and doubtless somewhere there is an engineer who is an expert on that cell phone. He/She understands how this system is mechanically assembled, what frequency it operates at, how the RF radiation translates from it, etc. However, if we imagine this cell phone, about 20 satellites, and 3 million other phones trying to talk together all at once, this system of systems can be complicated. Now add some of the new features that are popular today: voice recognition ("Call John."), Internet, Personal Digital Assistants (PDA), etc, it is not difficult to see that the Systems Engineering team for this project needs to have plans in place on how to deal with various scenarios.

The ability of a Systems Engineer to effectively lead the development and implementation of these systems will depend upon their ability to:

- Design the system such that it is manageable by the person or people whose job it is to manage the system. This person may be the actual system design engineer!
- Effectively interface with the Project Manager and engineer solutions with the Program Manager's concerns in mind.
- Effectively handle the technical challenges that present themselves from the networking of these systems. In a system of systems, all of the interfaces will not necessarily be the same, technically or politically!
- Effectively maintain a configuration of the system.
- Provide for the present and future maintenance of the system.
- Handle the logistics of a large system.

Large-scale systems have become prevalent in our society. “Systems rapidly become too large-scale with respect to the human capacity to observe, comprehend, analyze, steer, amend and tolerate.” [Maxwell, 2001] This report will begin with a discussion on the generic design of the system including how to handle the complexity of such a large system. Next, we will discuss what concerns that are normally associated with Program Management might be the Systems Engineer’s responsibility and vice versa. We will discuss Quality Assurance, Logistics, Maintenance, Configuration Management and more. Every topic that is discussed will be looked at from a Systems Engineering point of view with a particular bent toward these large-scale systems of systems. Finally, the author will give conclusions and recommendations based on his experience of utilizing these concepts on actual systems that are in the field today.

CHAPTER II
BACKGROUND INFORMATION

2.1 LITERATURE REVIEW

There has been and continues to be significant studies about the subjects within the individual chapters of this paper. In this study, we will take these individual components within the chapters and combine them so that the Systems Engineer may have a generic template for the design and management of a system of systems. Further, we will add to these discussions points of emphasis that become more relevant when dealing with a system of systems.

Harrington (2000) presented a white paper to the Society for Maintenance and Reliability Professionals detailing how the concept of a Reliability Centered Maintenance (RCM) Preventative Maintenance (PM) program came about. His six steps for a RCM program were especially helpful and were "tweaked" to be applicable to the subject at hand.

Watts and Mar (1996) performed a study asking professionals in the Systems Engineering field what further training was required for a person to be come a "complete" Systems Engineer. Surveys were sent to between March 19, 1996 to May 15, 1996 to various industry leaders. Portions of their results are presented in Chapter IV. The results are generic as to classes and specific as to topics. These results were helpful in showing the author's contention that maintenance of the technical knowledgebase is critical to the management of a system of systems.

Kauffman (1998) presented a white paper at the 8th Annual International Symposium of the International Council on Systems Engineering regarding how Systems Engineering and Program Management overlap. For his comparison, he examined the differences between various Engineering standards and the Project Management Book of Knowledge. Mr. Kauffman's conclusions were largely generic and must be modified slightly to manage a system of systems, however his greatest conviction of "plan early" lends credence to the author's most important point of the chapter. This paper is discussed in detail in Chapter IV.

Ginac (1997) did work in the area of Software Quality Assurance. He spends a great deal of effort expanding the concepts of SQA to four domains outside the "realm" of SQA. Developers, Testers, Production Services and Program Management. His concepts on how the Program Manager should utilize SQA was of great interest.

The Center for Maintenance Technologies (2001) has a website (http://www.amsinc.com/cmt/OVERVIEW.HTM) that discusses the benefits

Warfield (1994) created a textbook called *A Science of Generic Design*. This book's main focus is on managing system complexity through generic design. Mr. Warfield develops generic design as a science and then applies the science back to design. Its principles are found throughout chapter three.

Vigilinx (2001) created a white paper on system and network security. This white paper is written with selling Vigilinx security software in mind. However, its introduction to system security was especially useful as background information as is read in chapter seven.

Ken Wheeler (2001) created a power point presentation on the DIICOE initiative for computer interoperability as demonstrated by the military. The information provided as to the internal structuring of the runtime segments was useful in this paper as means of explanation as well as in the office where one project that is currently being bid requires DII COE level 5 compliance.

2.2 BACKGROUND

The thought process for this paper originated with the realization that the United States Military is shrinking in size. Not only are they shrinking in manpower, but in budget as well. This poses a problem for the military due to the fact that in order to stay ahead of the world militarily means maintaining and extending our technological advantage. In order to accomplish this goal, they must take existing systems and create them into "supersystems" or a system of systems. Through this protocol, money can be saved and the effective technological life of a system may be furthered.

Systems Engineering has long been in charge of creating and maintaining systems, however, these new systems of systems create some new problems for the Systems Engineer. This paper has taken components of system integration that may, in the past, have been on the technical sidelines to the engineers (with the exception of Quality and Safety) and brought them to the forefront of thought. It is imperative that today's Lead System Engineer be able to speak to all areas of the system, not only the technical ones, for that is what in today's world of a system of systems.

CHAPTER III
GENERIC DESIGN OF THE SYSTEM OF SYSTEMS

The principal purpose of theoretical research…is to find the point of view from which the subject appears in its greatest simplicity.

- J. Willard Gibbs

3.1 THE NEED FOR DESIGN

"Large systems have become prevalent in society. They have a pervasive influence on people everywhere. Their growth can be compared to making a quilt, A variety of components appearing in a variety of forms share an attribute: susceptibility to being linked. As linkages appear, systems grow rapidly in scale.

The impact of a failure of a quilt is normally quite confined, so that the comparison of a large system with a quilt stops when we look at the failure. As systems get larger, the impact of failure also grows larger. Not all failures are visible, but many of them become highly-visible. What we learn about those failures that we happen to see or hear about leads us to believe that there are many others that we do not learn about, and still others waiting to happen." [Warfield, 1994]

As shared in the introduction, systems are gaining complexity. Such is the scenario with our system of systems. Depending on the scenario, an individual system may be complex, however, when we start linking these individual systems together, the complexity induced must be managed or the complexity will overwhelm the process. As is a theme throughout this paper, we will attempt to manage this complexity through extensive initial planning and management through system design (which is a form of early planning).

3.2 COMPLEXITY DEFINED

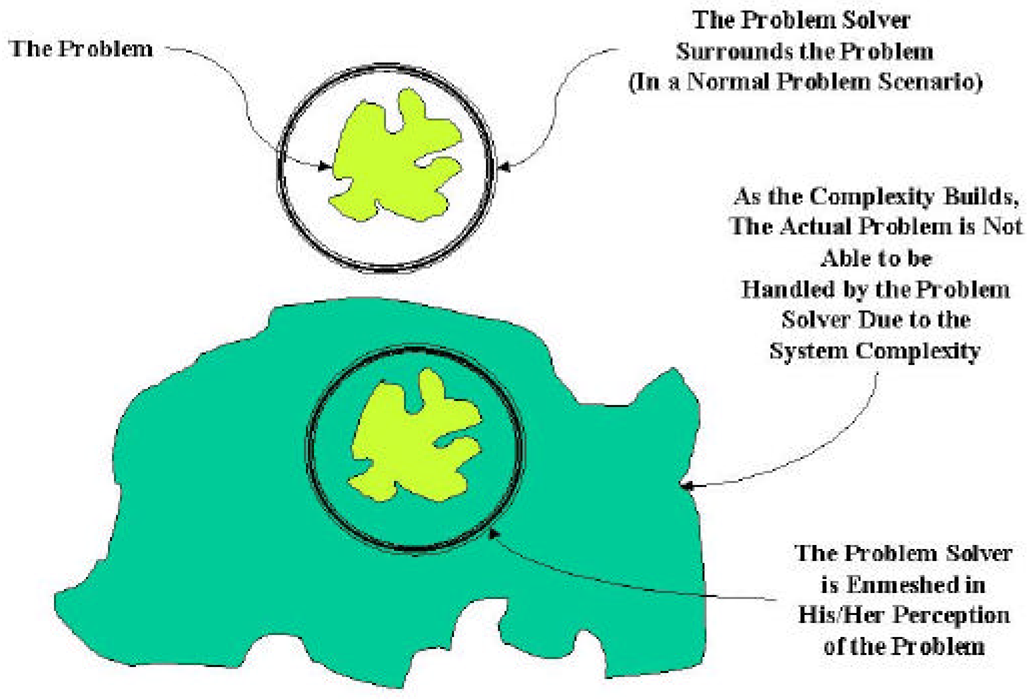
An example, given by Dr. Tim Maxwell in his lecture notes for the class "Generic Design," reverts back to the proverbial tree falling in the woods. The question is if no one is around, does the tree make a noise when it falls. The answer depends upon the definition of sound that one chooses to embrace. If one is referring to the ability to make sound waves, then the answer is "yes," however, if one defers to the physical definition of sound requiring a sender, a carrier and a receiver, then the answer is "no." This example leads us to the two types of complexity that can obscure our reasoning.

Situational complexity tends to hide the interception of phenomena from the mind, whereas cognitive complexity are aspects of the situation that make the interpretation of interpreted data difficult. The combination of these complexities leaves us with a generic definition:

Complexity: *The limitation of the human mind and reasoning capabilities*

As a rule, cognitive complexity is not an issue until the mind begins to consider the problem or situation. When many different imputs are considered that are not easily understood the complexity can not help but escalate. As linkages are identified/mis-identified, the situation becomes more complex.

For normal, smaller problems, the answer is simple, we are able to surround the problem, understand the problem and see the linkages and thus solve the problem readily. For large problems, it is not possible to extract oneself to an extent that the problem may be surrounded. Thus, the problem solver becomes part of the problem! Figure 1 graphically demonstrates this perspective.



The Problem
The Problem Solver Surrounds the Problem (In a Normal Problem Scenario)
As the Complexity Builds, The Actual Problem is Not Able to be Handled by the Problem Solver Due to the System Complexity
The Problem Solver is Enmeshed in His/Her Perception of the Problem

Figure 1 – The Problem Solver

As shown, and as is obvious, when the system becomes more complex, we can often get in our own way. When dealing with the system of systems, it is imperative that it be broken down into its most basic components and rebuilt, stopping at such a level that the System Engineer in charge of that particular section will be able to firmly grasp the operation of that section. Of course, the LSE must surround themselves with a team that is capable of understanding these levels and relaying the pertinent information back to the LSE. It is when these interactions take place that we see the next level of difficulty "pop up."

Human perspective can be a good thing. Indeed, when solving a problem and one person is stumped, a new perspective may lead to the resolution of the problem. However, just like the old saying, "too many cooks spoil the broth," too many perspectives may confuse matters even worse. Every team member of the technical team is going to bring their own unique perspective and interpretation of the situations encountered to the table. This is one type of linkage escalation. A second type of linkage escalation occurs with an enlarged set of problem perceptions and the process related difficulties of getting a group to work as a group.

We are now in a double loop escalation. There is the original perception of the problem solver coupled with (linked to) the new components that have been brought to the table by the technical team. While the group members may individually posses content knowledge that is relevant to the problem, they may not be knowledgeable in how to make this group of people an effective problem solving team. Also, dual roles in groups are difficult. "If one is contributing knowledge to a solution, while trying to steer the group's activity, a conflict of interest is likely to be perceived by the group." [Warfield, 1994] This dynamic within the technical team proves a difficult task for the LSE. Generally, a person becomes a LSE due to the experience and management skills that they posses. However, as we have just stated, if the LSE leans to much on their experience and applies it to the problem, the group may see a conflict of interest. When the initial design of the system of systems is occurring,

Figure 2 gives a graphical approach to the double loop approach to problem solving. Remember Figure 1 had the problem solver that initially had their arms around the problem and understood it completely prior to outside variables increasing the complexity to such a point that they were overwhelmed? For this initial problem and simpler issues, the first loop may be sufficient. However, as the system grows more complex, the second loop may be required- this may be due to a misunderstanding of the situation and thus the selection of governing variables or options. According to Maxwell [1994] however, there are significant problems with the double loop process:

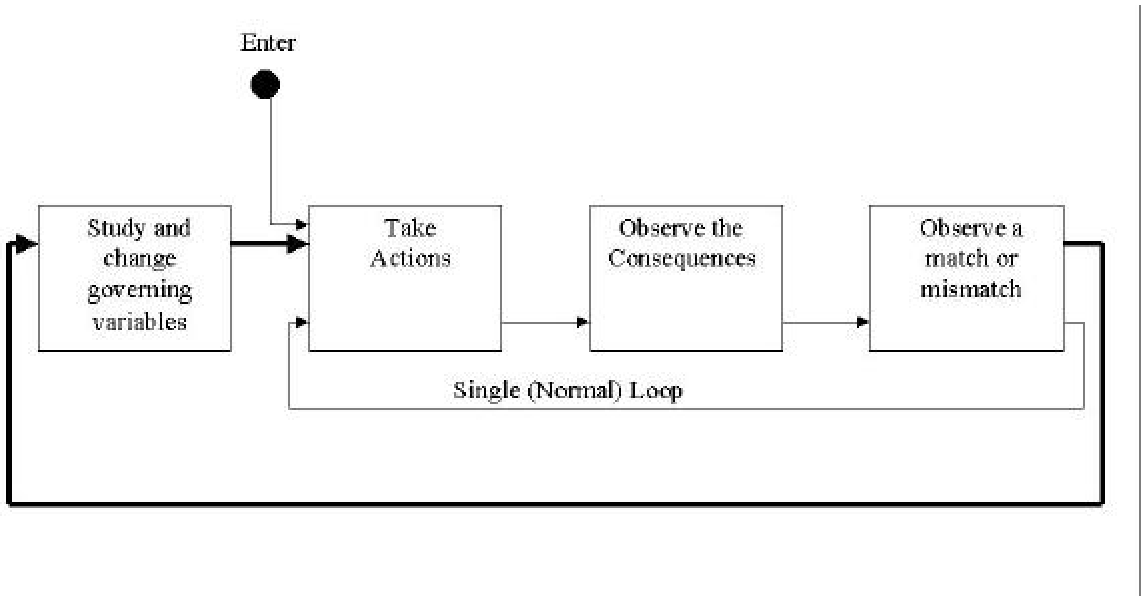

Enter
Study and change governing variables
Take Actions
Observe the Consequences
Observe a match or mismatch
Single (Normal) Loop

Figure 2 – Double Loop Model

- Many people do not know how to conduct a double loop
- They do not know that they do not know
- They do not know that they have designs in their heads to ensure the first two options
- They do not know that they cannot learn to do so with their present skills.

If this is the case, trying to implement this concept will escalate the complexity of the problem in the name of reducing it. On top of all this, the design target may change, requirements documents may change, resources may disappear, schedules may slip, etc.

If more than one iteration of learning is required from the double-loop model, one may choose to use an enlarged model called the poly-loop model as seen in Figure 3.

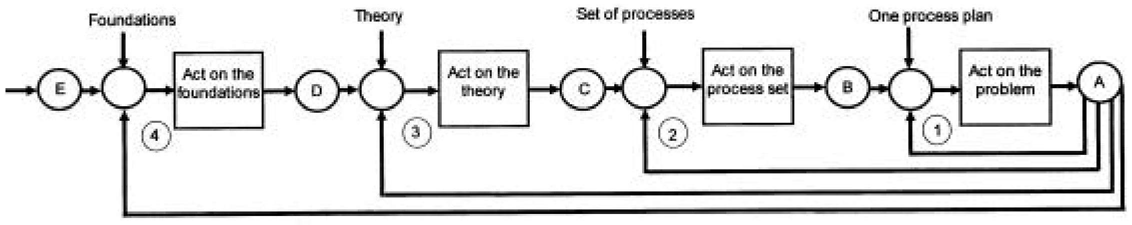


Figure 3 – Poly-Loop Model

The following is John Warfield's [1994] description of the poly-loop model, modified slightly for application to our case of the system of systems:

0-LOOP Enter with a process plan. Act on the problem (as perceived via the plan) and achieve result A. This usually only happens with the most trivial problem cases.

1-LOOP Enter with desired outcome at B and A process plans. Act on the problem (as perceived via the plan) and achieve results A. Feed back the results and compare with what is desired. Try the plan a second time, using new information to provide additional input to the process, and achieve results A. Keep doing this until it is decided that the problem is not getting solved.

2-LOOP Enter with desired outcome at C and a set of process plans. Act on the set to pick a new plan that promises to outdo the original plan. Now use the new one to act on the problem. Feed back the results to compare with desired outcome C. Repeat with new information supplied to the new plan. Iterate as long as necessary until either the problem is solved or it is concluded that the set of processes is not adequate.

3-LOOP Enter with the desired outcome at D, and a theory. Act on the theory to develop a new process set. Provide this set as entry to the 2-LOOP situation, and keep trying until either the problem is solved or it is decided that the new theory is inadequate. Iterate with a new theory, and try again. If the problem is still not able to be solved, go back to point E.

4-LOOP Enter with a set of foundations to accompany the desired outcome entered at point E. Act on the foundations to produce a new theory. Act on that to produce new processes. Act on the problem and observe the outcomes at point A. Keep trying until you get a set of foundations, theory, and a methodology that works.

This problem plan was developed for a new form of science but its application extends to the system of systems. When designing the system, if the foundation, theory and processes are not adequate, the system design will come crashing down! Having a well thought-out, well laid-out design is the best was to control complexity in the design. **This being said, escalation of complexity will be the one of greatest challenges that the Systems Engineer will face when building a system of systems.**

3.2.1 MANAGING COMPLEXITY

When we attempt to deal with the complexities that have entered our design and integration of the system, we realize that in order to manage the complexity, we must understand it. Obviously if it were easy to understand it would not be that complex and therein lies the conundrum, complexity always involves elements of the unknown. When we are faced with complexity, John Warfield [1994] gives four approaches to managing it:

- Control the situational escalation
- Reduce the personal cognitive burden
- Eliminate the situational detraction set
- Provide a personal enhancement set

Let us look at ways to control the situational escalation. What are some factors that produce situational escalation?

- Varying perception among team members
- Difficulty in managing team efforts
- Presence of organizational or cultural constraints
- Difficulty in communicating solutions to implementers
- Change in problem situation in time
- Lack of actors to field needed roles
- Difficulty of dealing with the foregoing when several or all occur in combination

"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." [Warfield, 1994]

As we have demonstrated, a system of systems can add up complexity on a monumental scale. There is the need for order when we design, not only in the final product, but in the process itself. This need for order is the need for design, the only way to reduce the complexity of the system is to plan ahead, which is to say, design ahead.

When we design, we will follow various plans and processes. How useful are these processes in helping to manage the complexity of the system? Dr. Tim Maxwell [2001] states that designed processes must be designed to:

- Formulate situations in the form of sets
- Limit situational escalation
- Reduce personal cognitive burden
- Eliminate most situational enhancements

In doing this, significant areas include:

- Information handling
- Idea generation

- Environment design and control
- Role specification
- Leadership (facilitator, members, organization)
- Quality Control
- Cultural Environment
- Institutional Change Strategy

As always, putting a little forethought into these processes used to design the system of systems will eliminate complexity and enhance the design.

3.2.2 TOOLS FOR MANAGING COMPLEXITY

There are several “tools” to help manage the complexity in a system. Within a system of systems, however, everything and everyone may or may not be co-located, thus further adding to the complexity of the system and its subsequent design. We will now present one of these “tools” as an example. The tool that we will present is called DELPHI. Ideawriting and Nominal Group Technique (NGT) are two further “tools” that the reader may choose to investigate.

DELPHI is defined as “A method for getting views of people when there is not a practical means for bringing the people together in one place and having them engage in constructive dialog. A method for systematic development of the views of a panel of individuals, generally experts, with regard to some issue. To minimize the influence of dominant personalities, to remove graphic limitations, to allow input from persons who might otherwise not be able to participate, and to allow anonymity among the panel (if desired), written responses to a sequence of correlated questionnaires are obtained. The results of each questionnaire are fed back to the panel by the monitoring group, which strives gradually to develop consensus among the panel members.” [Warfield, 1994] Figure 4 is a pictorial flow of the DELPHI method. The following description of the DELPHI method is given by John Warfield [1994].

3.2.2.1 APPROPRIATE CONDITIONS FOR USE

There is a need to collect and evaluate ideas, forecasts or opinions relative to some issue. Face-to-face participation and interactions is either not feasible or not desired. Expert opinions and knowledge are the prime sources of information. The experts should learn from each other during the process.

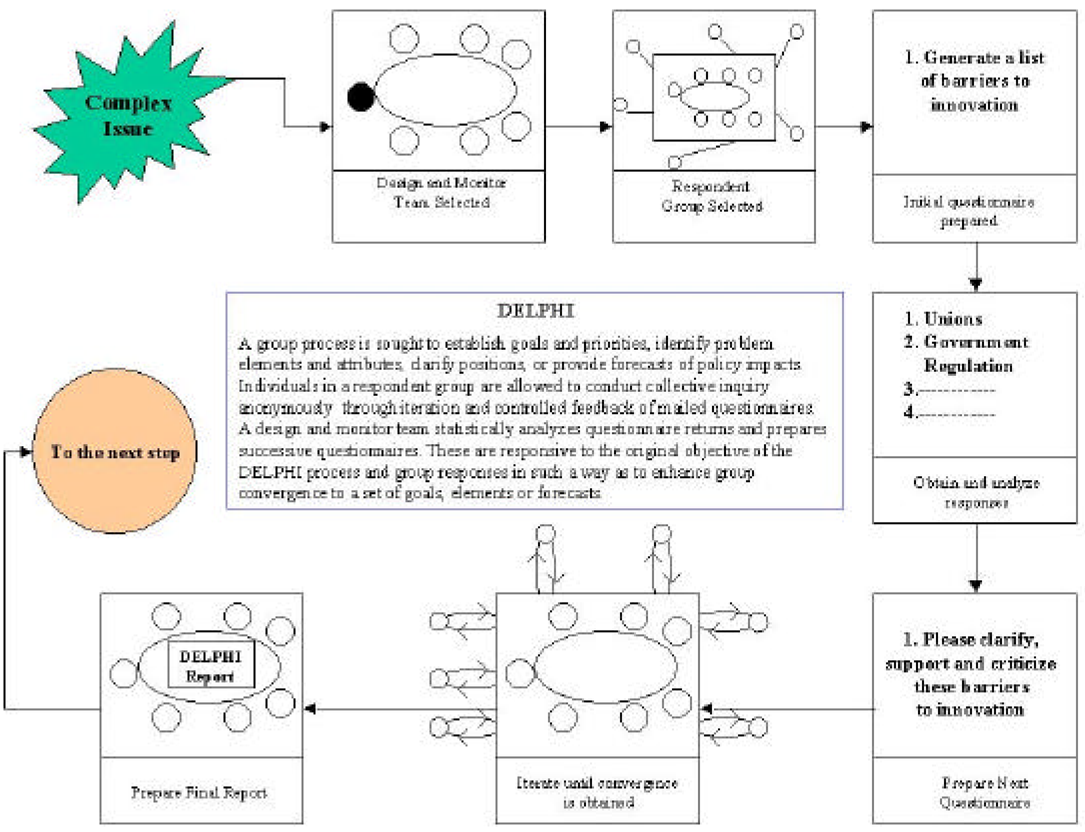

Complex Issue
Design and Monitor Team Selected
Respondent Group Selected
1. Generate a list of barriers to innovation
Initial questionnaire prepared
1. Unions
2. Government Regulation
3.------------
4.------------
Obtain and analyze responses
DELPHI
A group process is sought to establish goals and priorities, identify problem elements and attributes, clarify positions, or provide forecasts of policy impacts. Individuals in a respondent group are allowed to conduct collective inquiry anonymously through iteration and controlled feedback of mailed questionnaires. A design and monitor team statistically analyzes questionnaire returns and prepares successive questionnaires. These are responsive to the original objective of the DELPHI process and group responses in such a way as to enhance group convergence to a set of goals, elements or forecasts.
To the next step
1. Please clarify, support and criticize these barriers to innovation
Prepare Next Questionnaire
Iterate until convergence is obtained
DELPHI Report
Prepare Final Report

Figure 4 – DELPHI Method

3.2.2.2 APPLICATION AREAS

The DELPHI method is appropriate for idea generation, description and evaluation. Examples of application include technological and social forecasting, water resource management and technology assessment.

3.2.2.3 RESULTS

Increased understanding, clarification of positions, and explanations of differences of opinion among experts. A supply of elements, events and possible dates of occurrence for further analysis. A final report summarizing the process.

3.2.2.4 RESOURCES NEEDED

Design and monitor group of 1 to 10 people. Supporting staff to type and mail questionnaires, receive and process results. A respondent panel of 5 to 100 people whose judgements are sought, and who posses reasonable written communication skills. A process leader to coordinate the design and monitor group, the supporting staff and the expert panel. Physical facilities for housing and supplying the monitor group.

3.2.2.5 HOW THE METHOD IS APPLIED

An issue and a process leader are chosen. The design/monitor group is chosen, and the panel is chosen. An initial questionnaire is developed and mailed to the panel. Panel replies are received and analyzed. A revised questionnaire is mailed to the panel. The process continues iteratively until convergence is achieved or sufficient information is obtained. A final DELPHI process report is prepared and distributed.

3.2.2.6 IMPORTANT FEATURES AND ATTRIBUTES

Individuals in the respondent panel participate on an equal and anonymous basis. Iteration and controlled feedback between iterations are provided. Statistical group response by aggregation of individual opinions of the panel can often be developed. Does not require respondent panel travel or prolonged effort at any particular time, since the time allowed typically varies from one month to one year.

3.2.2.7 RELATED METHODS

If anonymity is not required and short-term results are essential, one may use Ideawriting, Charette or Nominal Group Technique in place of DELPHI. When sufficient data and theory are available, various modeling methods can be used as alternatives to DELPHI questionnaire data.

3.2.3 CONCLUSIONS AND SUMMARY ABOUT COMPLEXITY IN THE SYSTEM

As we approach the future, the systems that we build will have a tendency to interconnect and form a system of systems. These systems will be of such complexity that the Systems Engineer will need to minimize it in order to successfully complete the system. The complexity in design is not limited to the design itself. This complexity will either be managed, or it will overwhelm the individual and the system.

John Warfield [1994] states that the management of complexity refers to an overt, integrated mode of operation that:

- Identifies detractors to human mental activity that diminish conceptual power.
- Recognizes the classes of detraction, and eliminates those that can be eliminated.
- Recognizes possible enhancements to human mental activity
- Acknowledges process requirements for providing such enhancements, including role requirements and environmental requirements to support the roles and the processes.
- Studies human performance in complex situations, and recognizes bad outcomes from such performance and the sources and commonalties of these outcomes.
- Involves the establishment of a dedicated (maxi-min) environment that maximizes the enhancements and minimizes the detractions through conscious environmental design.
- Recognizes the division of labor among roles, to provide the variety of expertise needed to implement the processes within the maxi-min environment.
- Provides for the education and training needed to fill the entire set of roles.
- Provides the resources based on the rule that even a very small percentage of the potential losses that accrue in the event of failure can wisely be dedicated to a systematic and thorough attack on complexity itself.
- Assigns special importance to Quality Control.

Necessary conditions for managing complexity include control of escalation of complexity, the reduction of the cognitive burden on the designers, the elimination of the detraction set and the provision of the enhancement set.

Specially designed processes can deal with information sets, can limit or eliminate escalation, can reduce cognitive burden, can eliminate many detractors and can provide many enhancements.

3.3 GENERIC DESIGN

In the design of our system of systems, our final goal will be well defined roles and environments. The theory of generic design must provide the requisite steering to the definitions of role and environment. Figure 5 pictorially shows this implementation.

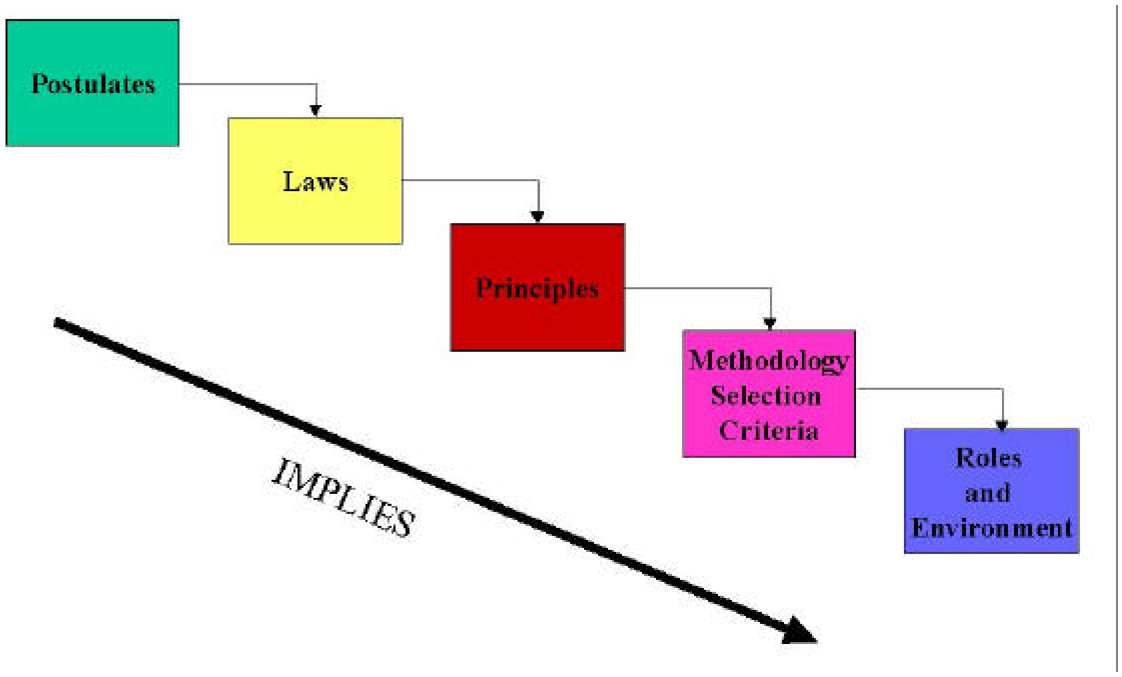


Figure 5 – Theory of Generic Design

3.3.1 OPTIONS FIELD AND A TRIPLY-STRUCTURED QUAD

Within a generic design, the generic theory of situations and processes introduce the Options Field as key products of the design activity. Figure 6 is a representation of an Options field as a Triply-Structured-Quad (TSQ). It is a quad due to the four levels. These four levels are represented by

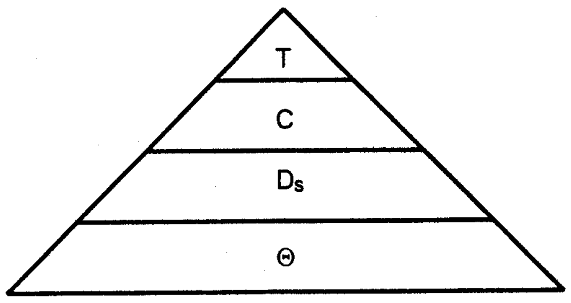


Figure 6 – Triply-Structured Quad

1. T – Target
2. C – Cluster
3. D_S =Dimension
4. Θ = Option

It is triply structured due to the fact that the structure incorporates three distinct relationships. The first relationship is the relationship among the options. This is known as "membership in a direction." This relationship is indicative of the options for attaining the Target fall into a collection of sets – each of which represents one dimension. The second relationship is an "interdependent" relationship where the independent dimensions form clusters. The third relationship is a time preference relationship showing which time sequence choices should be made from among the dimensions. This relationship is developed after interdependence is established so that each set of interdependent dimensions ca be treated as a cluster.

It may be necessary to knit to a TSQ at its lowest (fourth) level another overlapping TSQ. Figure 7 shows what a "knitted tapestry" looks like pictorially. In a tapestry, the option from the first TSQ becomes the Design Target for the second TSQ, This "knitting" process may be repeated as needed with additional TSQ's to produce Design Tapestry.

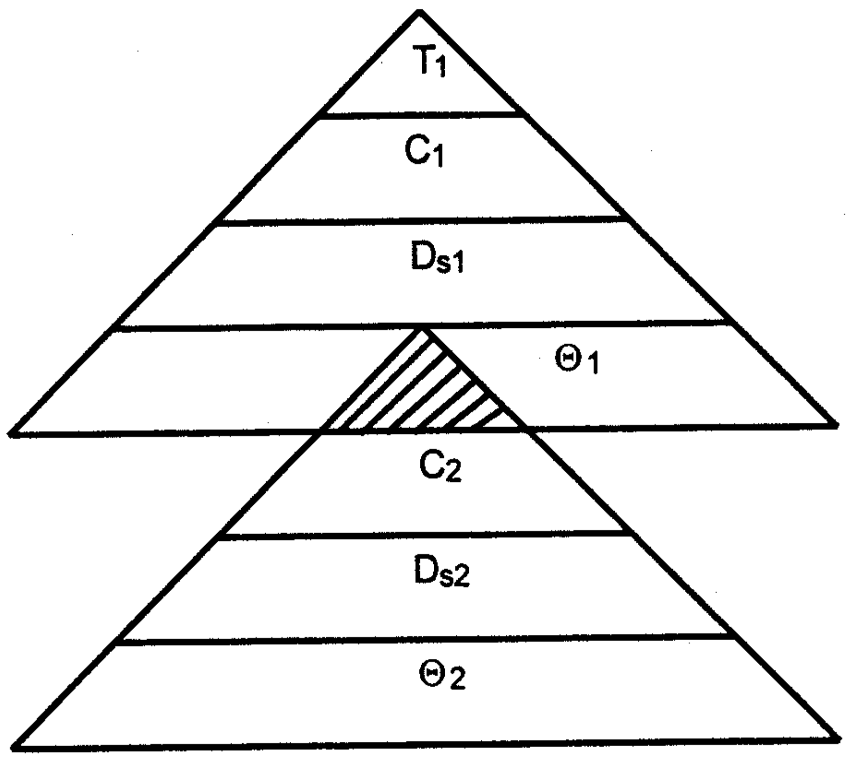


Figure 7 – A Knitted Tapestry

The final structural portrait of the Target Design may be a Tapestry of TSQ's arrayed hierarchically**. It is in this fashion that we will pictorially design the system of systems.** It is important to note that the top level will belong to the LSE. Under that Tapestry, the next TSQ will belong to SE Group Leads (SEGL), the next to another SE, etc. until all TSQs are attached to a Systems Engineer that is responsible for it.

3.4 AN EXAMPLE OF GENERIC DESIGN

It would be beneficial at this point to demonstrate the thought process for the development of a generic design. The following is an example that will demonstrate a knitted tapestry for the development of an Electronic Light Table (ELT) program. Thus, the target will be a completed ELT package. As the example is read, it will be important to notice (as you become familiar with what an ELT is), that although the Target for the example is a completed ELT program, this ELT program could easily be an option for a higher level TSQ!

3.4.1 BACKGROUND FOR THE EXAMPLE

Within the United States Military, there exists a vital need for a system that is able to view tactical reconnaissance imagery and develop usable intelligence from it. Several ground stations are currently available to the military that can accomplish this feat.. The Military's Ground Stations have the ability to fully exploit the aforementioned tactical reconnaissance. An image analyst will review the mission with the aircrew to identify assigned targets and search for targets of opportunity that were covered by the flight by utilizing a screening tool that will allow the data to be viewed on a computer monitor. When these targets of opportunity are observed at the screener, the analyst will "cut out" the selected target and export the "cut out" image to an electronic light table (ELT) program. An example would be the following; at the screening tool, an image analyst is viewing tactical reconnaissance footage of the desert when he/she sees a tank. Wanting to get this information to their commander, they "cut out" the tank and a small amount of the surrounding area for the commander to view as opposed to the entire tape of reconnaissance imagery. It is this small "cut out" picture that is exported to the ELT program. Within the ELT program, the "cut out" picture may be fully exploited with graphics, annotation, etc to help the commander make an informed decision as to how to utilize this exploited imagery. Within these ELT packages lies the ability to exploit imagery in several ways. Imagery may be exploited by utilizing one of the core functions inherent to the ELT program or by using one of the modules that enhance the capabilities of the ELT package.

Outside of the core packages within an ELT program are several modules. It should be noted at this point, that the group of "core" functionalities themselves creates a module. Within the classes of the ELT system, these modules would be on the same order class as the base package. An example of a typical ELT program would be VITec ELT by British Aerospace (BAE Systems). The following are some examples of modules for an ELT system (http://www.vitec.com, June 2000):

Image Compare – This module allows change detection and comparison of two images. Image sources can be of any type, allowing comparison of two images taken on different days, or any combination of visual, SAR, IR or map imagery. Images can be locked roamed, flickered and merged. Images can be laid on top of each other and a porthole cut out of the top image for viewing down into the bottom image, similar to an image swipe. Rub through capability can be used to remove cloud cover from images for improved presentation.

Precision Targeting – This module is comprised of many functions. The HTML function is a browser-based client capability that allows the user to access a subset of national or tactical targeting capabilities. The Polynomial Registration Tool allows the user to perform a point transfer from tactical imagery to DPPDB or NTM imagery when tactical ESD/telemetry data is unavailable. The sensor models will be NIMA validated and certified. The National Source Selection function allows the user to "Dial-an Accuracy" and select a subset of NTM or DPPDB images from a database that provides the desired accuracy. The NTM Single Ray Intersection with Terrain Geopositioning function allows the user to extract target geocoordinates and accuracies using a single NTM image and DTED terrain data. Finally, the NTM Multi Image Geopositioning (MIG) function allows the user to extract target geocoordinates and accuracies from multiple NTM images.

Special Exploitation Tool – Spectral Exploitation Tool reads imagery in NITF, SPOT, LANDSAT and SYERS internal format. This module performs a variety of Multi Spectral and Hyper Spectral functions such as: Spectral angle search, which locates materials in multi/hyperspectral images that are spectrally similar to a selected sample. This is used, for instance, to find other examples of target paint given one sample. The Spectral Angle Material Classification tool is a supervised classification to segment an image into a set of similar materials based upon samples, creating, for example a terrain categorization (TERCAT) map.

Import – This module allows users to quickly and easily accesses, view, manipulate and convert data for use within the ELT program; roam and pan through CD-ROM overview image data interactively; and use geographic coordinates to select a subset of the map. In addition, this module allows users to save the map data as an 8-bit Pseudo Color file, significantly reducing the amount of disk storage required for each map.

Search – This module expands the capabilities to provide fast, smooth, flicker-free, tear-free, roam and pan of extremely large images. The user can ingest full frame national imagery, smoothly roam images at full resolution, conduct broad area searches at user defined rates, drop markers to target additional analysis, and create any size chips on the fly.

OrthoMosaic – OrthoMosaic will perform the following tasks, in any combination: Build a Digital Terrain Model (DTM) from overlapping images (Automatic Terrain Extraction (ATE)), Register overlapping images together (Triangulation), Rectify an image (Ortho Rectification) and Mosaic images together. This module is a background process that automatically triangulates, rectifies, and extracts terrain, There is no user interaction other than selecting files to rectify.

Register/Mosaic – This module allows tactical users to warp or rubbersheet aerial reconnaissance images to maps (ADRGs) or other images and to quickly determine estimates of ground locations and their coordinate points on the image. As the two images are interactively merged, a single window on the screen shows the combined features of both: the satellite or aerial reconnaissance image along with the map's latitude and longitude or Universal Transverse Grid (UTM) coordinates. After loading the images into the module, the user is quickly able to pick features common to both images (roads, forest boundaries, intersections, etc.) and use them as control points to warp or register one image to the other.

The core packages of the ELT software are the standard features that come with the software when purchased. The following are some of the features found in a typical VITec ELT package (http://www.vitec.com, June 2000):

Enhance – Execute interactive histogram adjustments, region of interest (ROI) processing, look up table (LUT), modifications and digital filtering. Create an unlimited number of ROIs for selective image enhancement and manipulate them before applying to source image. Includes brightness/contrast, color enhancement, sharpness, edge enhancement, etc.

Paint/Airbrush – Perform digital photo retouching, add or delete features to existing maps or images. New roads and features may be added and outdated data deleted.

Annotate – Add, edit or overlay text, icons and graphics on the image then manipulate the annotations independent from the underlying image. Annotations may be saved with or without the image. Certain templates are available.

Measure – Perform Mensuration for all sensors modeled by the NIMA Ruler software. Also perform mensuration on all NITFs format imagery not modeled by NIMA Ruler. Supports the adding of math models to support mensuration of other sensors.

Chip – Chip to scale. Chip to map scale. Chips can be saved at resolutions different from the source imagery.

Grid – Primary/Secondary grid, User selected Coordinate Type, Auto/User defined Spacings, Interactive Line Thickness/Color Control, Interactive Font Size/Color Control. Grid types include Lines, Dashed Lines, Ticks and Reseaus.

Save – Write imagery to a variety of file formats, including NITFS, GIF, JPEG, TIFF, Sun Raster, BMP, XWD, etc.

Print – Supports most printers with easy to use application interface. Print module incorporates NIMA's printer lookup tables for Kodak, Cannon and HP printers. These look-up tables match the hardcopy output to the color values on the screen, providing a "what you see is what you get" capability for hardcopy outputs.

Cut/Paste – Cut out potions of the image and incorporate into the source image of a different image, optionally creating drop shadows. Allows maps to be added to images and vice versa.

3.4.2 APPLICATION OF ELECTRONIC LIGHT TOOL TO GENERIC DESIGN

Figure 8 depicts the "Two-quad Tapestry" concept (Kaake, 2001) applied to Military Intelligence and Electronic Light Table Programs.

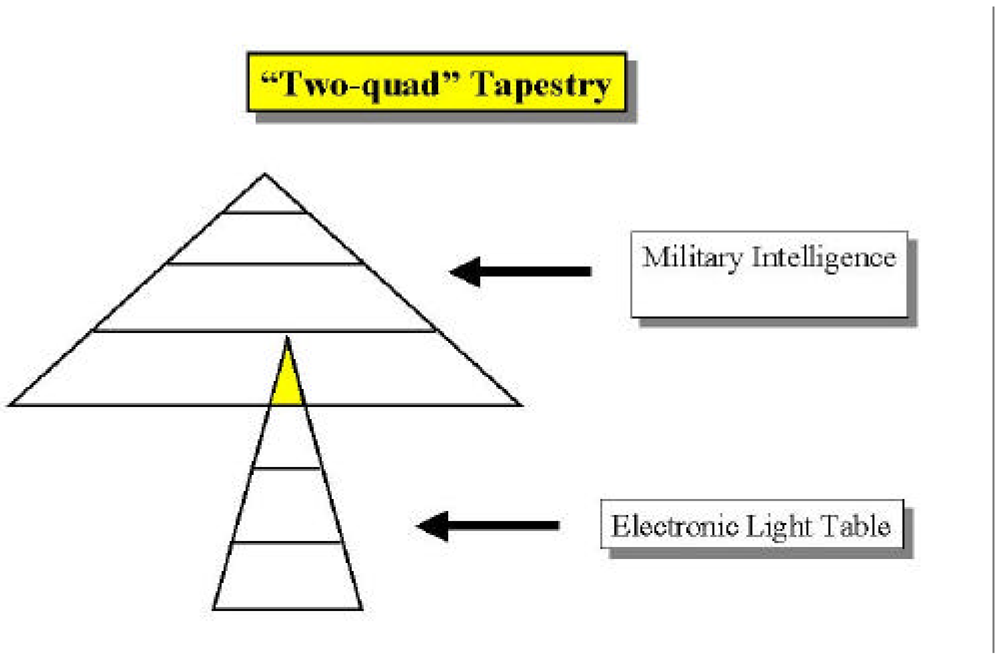


Figure 8 – Two-Quad-Tapestry

The general format for an "Option Field" Design Layout is depicted in Figure 9. [Warfield, J. 1994]

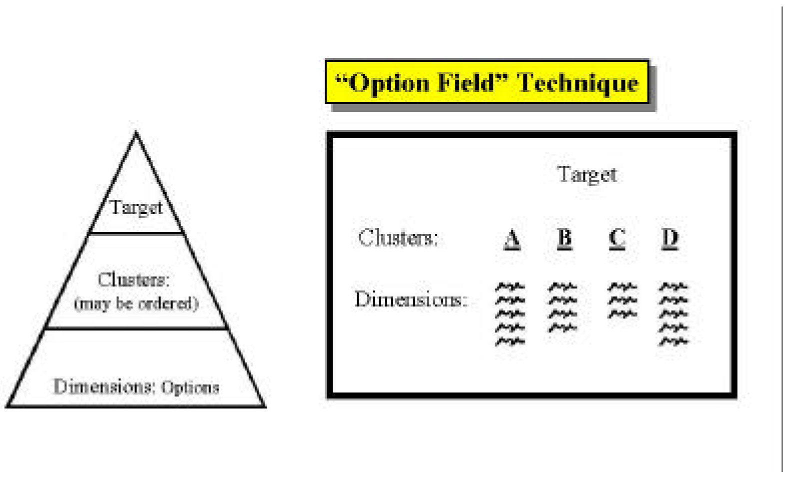


Figure 9 – Option Field Technique

The tables bebw are an “Option Field Representation” of the target “Electronic Light Table”. Due to space constraints, the layout will not be as in Figure 9, but will be broken up. Should the reader desire, Tables 1-4 could be constructed to assume the format of Figure 9.

CLUSTERS	SOFTWARE	HARDWARE	SUPPORT

Table 1 - Target: Complete Electronic Light Table System (Clusters)

SOFTWARE CLUSTER					
DIMENSION	Pixel Enhancement	Picture Enhancement	Targeting	Annotation	Intel Tools
OPTIONS	1. Brightness/ Contrast 2. Color Enhancement 3. Color Shading 4. Sharpen 5. Smooth	1. Zoom 2. Rotate 3. Roam 4. Crop	1. Lat/Long 2. MGRS 3. UTM	1. Text 2. North Arrow 3. Circles 4. Lines 5. Squares 6. Other Shapes	1. Mosaic 2. NITF Levels 3. Mensuration

Table 2 - Complete Electronic Light Table System (Individual Dimensions and Options - Software)

HARDWARE CLUSTER			
DIMENSION	Physical Table	Camera	Peripherals
OPTIONS	1. Auto 2. Manual 3. Fluorescent 4. Incandescent 5. Backlit	1. Digital 2. 35 mm 3. Auto 4. Manual	1. Maps 2. Ruler 3. Pictures

Table 3 - Complete Electronic Light Table System (Individual Dimensions and Options - Hardware)

SUPPORT CLUSTER			
DIMENSION	On Site	Phone	Informational
OPTIONS	1. Warranty 2. FSR 3. Software Help	1. FSR 2. Help Desk 3. Automated FAQ 4. Warranty	1. User Guide 2. Software Help 3. Warranty

Table 4 - Complete Electronic Light Table System (Individual Dimensions and Options - Support)

3.4.3 RELATIONSHIP ANALYSIS

Pictorial representations can be developed to further organize the brainstorming or "brain-writing" process. The Figure 10 shows the clusters within the Physical Design dimension. Organizing using the "Option Field Method" can lead to interactions the group may or may not have thought of without the graphical representation. (Kaake 2001)

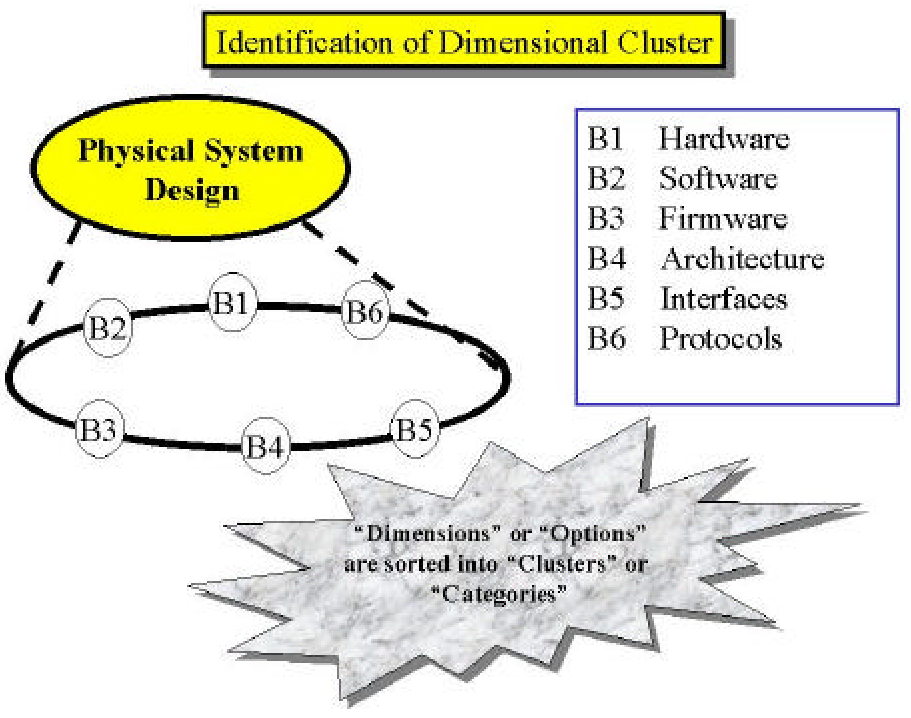


Figure 10 - Identification of Dimensional Cluster

Interactions within the Electronic Light Table, as previously defined, were developed using the ISM (Interpretive Structural Modeling) process. Results are depicted in the following figure. Each dimension within the cluster interacts or interfaces with every other dimension or component within the cluster. Flexibility within this design allows for all components to interact simultaneously or with selected components or a single component running independently. Effective and efficient design processes will improve situational awareness, Intelligence Preparation of the Battlefield (IPB) process, and R&S (Reconnaissance and Surveillance) planning. Further, the definition of success for the ELT will be defined through CONOPS (Concept of Operations) and system Mission Profile (MP) development.

3.4.4 EXAMPLE CONCLUSIONS

Utilization of the Option Field Method will allow for the effective "partnering" of all applications to the ELT program. The interrelationships between the software, hardware and support clusters as well as the infrastructure of the ELT are made sounder by the understanding of the interactions between the clusters. This understanding will be useful when the ELT is incorporated for use in another program. The Generic Design Process will allow for a Generic Process Flow when the ELT itself is utilized. (Kaake and Whittington, 2001) The conclusion gained through this work is that the Generic Design Process allows an understanding of the final product such that interactions of the product are designed to allow flexibility to accommodate change prior to the completion of the product; and, once completed, easier modification should conditions warrant.

3.5 DESIGN CONCLUSIONS

When designing the system of systems, the LSE must keep in mind how complex the system will become. Obviously, there are varying degrees of complexity, however, how it is managed is of utmost importance. The complexities involved in engineering a system of systems are so great, that without a template for design and implementation, the chances of system success are limited. It is in the understanding of the relationships between the lower level systems that system integration may successfully occur. This is done by any number of methods, specific to this chapter is the Option Field Method. When the system's complexity has been managed and its relationships understood from its lowest level to its highest (through an option field) the design of the system will be complete.

CHAPTER IV
MANAGEMENT CONCERNS FOR THE SYSTEMS ENGINEER

There are times that every Engineer will shy away from Program Management tasks. For the most part, this repulsion is due to a dislike for the business side of a contract versus the technical side of the contract. However, a Systems Engineer, particularly a Lead Systems Engineer, will inevitably find themselves performing some subset of these tasks. In this chapter, we will first look at the intersection of the professions of Systems Engineering and Program Management to determine what areas are whose responsibility and then we will examine specific concerns that, although are normally considered Program Management responsibility, drastically affect how a Systems Engineer manages the design and development of the system of systems.

4.1 SYSTEM ENGINEERING AND PROGRAM MANAGEMENT: ROLES AND RESPONSIBILITIES

Long before the age of digital cameras, scanners and high speed Internet access, there were Electrical Engineers. It did not take Electrical Engineers long to discover that by manipulating relays and switches and the like, they could alter/adjust the output of the piece of equipment that they were working on. As this practice became more and more prevalent, it became quite clear that there was a distinct need for Engineers that could trace electrons around a circuit or system and Engineers that could manipulate or "program" the electrons. Thus the field of Electrical Engineering split and gave way to the field of Software Engineering. Now, if someone says that they are a Software Engineer we feel quite comfortable in pigeonholing their job function. We may not be aware if they are a C++ programmer or a UNIX programmer or a JAVA programmer, but we have a good grasp as to their job function. However, if someone says that they are a Program Manager or a Systems Engineer, an immediate job description does not come to mind. If someone claims to be a Systems Engineer we could wonder: "Are you a Requirements Engineer?" "Are you an Integration, Test and Verification (IT&V) Engineer?" " Are you a Field Engineer?" The questions could go on and on. The list of questions that one could raise for a Program Manager is just as large. When it comes to the job description of Program Managers and Systems Engineers, particularly Lead Systems Engineers, the lists are so diverse that overlap between the lists is unavoidable. The overlap between the two job functions creates a very large gray area of responsibility. This portion of the chapter will present guidelines to help the project stakeholders decide on a synergistic approach to addressing the intersection of the professions.

4.1.1 COMPARISON OF SYSTEMS ENGINEERING TO PROJECT MANAGEMENT

A Guide to the Project Management Body of Knowledge (PMI 1996) (PMBOK), has been used by the Project Management Institute as a basis for certifying over 6,800 Project Management Professionals worldwide. Information gained from this book will represent the Project Management body of knowledge referenced in this section. The PMBOK recognizes that the Project Management body of knowledge intersects with two other bodies of knowledge as shown in Figure 11, those bodies being Application Knowledge and Practice and General Management Knowledge and Practice. It is the intersection of Project Management and the application area, Systems Engineering, which is the focus of the comparison. It is recognized that General Management Knowledge and Practice impact both Project Management and Systems Engineering, but that intersection is not addressed in this paper. [Kauffman, 1998]

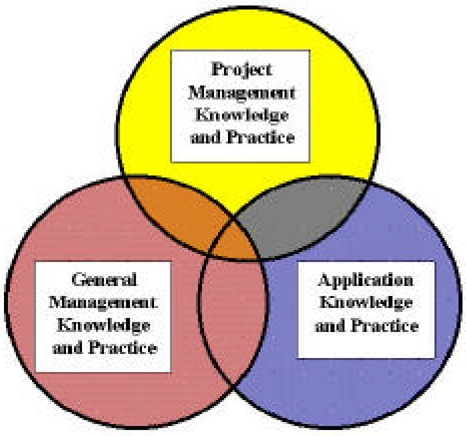

Project Management Knowledge and Practice
General Management Knowledge and Practice
Application Knowledge and Practice

Figure 11 - Relationship of Project Management to Other Management Disciplines

Table 5 compares the nine Project Management knowledge areas defined in the PMBOK to the major processes defined in Each of the following Systems Engineering standards: *IEEE Std 1220-1994, IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process*

(IEEE 1995) *and EIA Interim Standard, Systems Engineering, EIA/IS-632* (EIA 1994). The filled-in blocks indicate where there is a strong correlation between activities identified in the PMBOK and those in IEEE 1220 and EIA 632. [Kauffman, 1998]

PMBOK **IEEE 1220**	**Project Integration**	**Project Scope Management**	**Project Time Management**	**Project Cost Management**	**Project Quality Management**	**Project Human Resource Management**	**Project Communications Management**	**Project Risk Management**	**Project Procurement Management**
Requirements Analysis		🟦							
Requirements Baseline Validation					🟦				
Functional Analysis		🟦							
Functional Verification					🟦				
Synthesis		🟦							
Physical Verification		🟦			🟦				
Systems Analysis		🟦						🟦	
Control	🟦		🟦		🟦		🟦	🟦	
EIA 632									
Requirements Analysis		🟦			🟦				
Functional Analysis/Allocation		🟦			🟦				
Synthesis		🟦			🟦				
Systems Analysis and Control	🟦	🟦	🟦		🟦		🟦	🟦	

Table 5 - Comparison of Project Management and Systems Engineering Bodies of Knowledge

Clearly several of the IEEE 1220 and EIA 632 processes correlate to the scope, quality and risk knowledge areas of the PMBOK at this level. We further see that there is correlation, at some lower level, in the integration, time and communications management areas. The Systems Engineering manuals show almost no correlation with the cost, human resource and procurement management. Based solely on this analysis it is obvious that the Project Manager has the primary responsibility for these areas. However, the Systems Engineer, especially the Lead Systems Engineer, should not completely disregard these areas. It is not difficult to understand how the Lead Systems Engineer could have a valuable input to the hiring of a new Engineer, even though the human resources management is in the domain of the Program Manager. Even though the Systems Engineer is not responsible for these areas or other unmentioned areas, there is no doubt that they should understand these areas as they will undoubtedly affect other areas that the Systems Engineer is responsible for.

4.1.2 THE INTERSECTION OF SYSTEMS ENGINEERING AND PROGRAM MANAGEMENT

The previous section identified several high-level areas where Program Management and Systems Engineering intersect. In this section we will take a closer look at each area that is a supposed intersection between the two fields and determine if each section truly is an overlap, i.e. each group could manage/perform the section equally; or which of the two professions dominates when the lower level processes are examined.

4.1.2.1 INTEGRATION MANAGEMENT

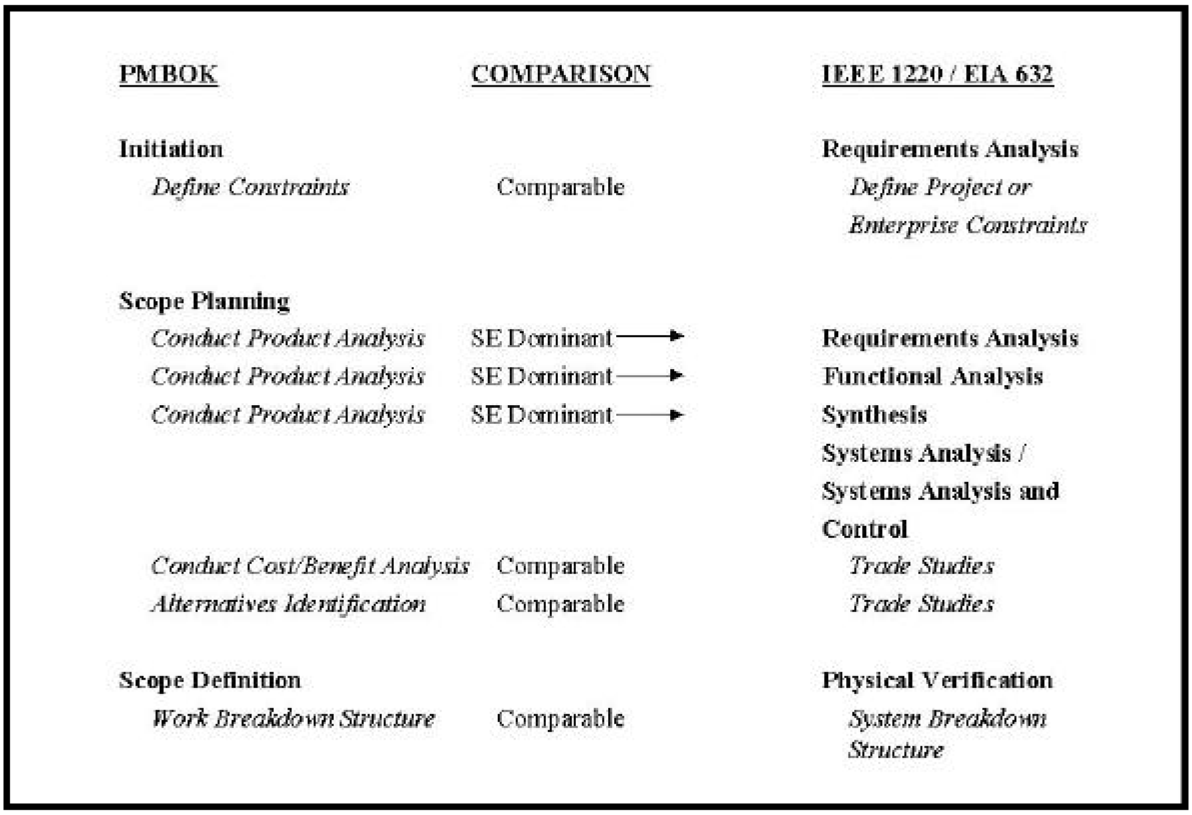
The PMBOK defines three processes under Integration Management: project plan development, project plan execution and overall change control. The only one of these three that directly correlates to a Systems Engineering process, Configuration Management (CM), is overall change control as defined in IEEE 1220 Control process and the EIA 632 Systems Analysis and Control Process. “Both the Project Management and Systems Engineering knowledge areas are concerned with ensuring that changes are beneficial and properly reflected in all other activities. The tools and techniques to carryout these activities appear to be comparable in both professions; therefore, neither can claim to dominate change control or configuration management.” (Kauffman, 1998) The point alluded to by Mr. Kauffman is well taken, either profession may take the responsibility of configuration management and, as long as this responsibility is agreed to, the configuration of the system (i.e. system of systems) will be managed. However, due to the importance of maintaining the system configuration from a technical standpoint, especially in the realm of software CM, it is in the Systems Engineer’s best interest to take a *very* active role in this process. If the role in question is that of a

Lead Systems Engineer, it is strongly advised that he/she assume the management responsibility role for system configuration management (see Chapter 4).

The remaining areas of Integration Management, project plan development and project plan execution, have no corollaries in the System Engineering processes and are, therefore, viewed as being exclusive to the Program Management profession.

4.1.2.2 SCOPE MANAGEMENT

The PMBOK recognizes that there are two aspects to Scope Management: product scope and project scope. The PMBOK recognizes that a special applications area such as Systems Engineering should manage product scope. The primary focus of the PMBOK is therefore on project scope. However, project scope is to multi dimensioned to attribute solely to Project Management; there is sufficient overlap in this area to examine its individual constituents. Figure 12 illustrates how certain Systems Engineering processes relate to three processes defined in the PMBOK Scope Management knowledge area: initiation, scope planning and scope definition.



PMBOK	COMPARISON	IEEE 1220 / EIA 632
Initiation		**Requirements Analysis**
Define Constraints	Comparable	*Define Project or Enterprise Constraints*
Scope Planning		
Conduct Product Analysis	SE Dominant ⟶	**Requirements Analysis**
Conduct Product Analysis	SE Dominant ⟶	**Functional Analysis**
Conduct Product Analysis	SE Dominant ⟶	**Synthesis**
		Systems Analysis / Systems Analysis and Control
Conduct Cost/Benefit Analysis	Comparable	*Trade Studies*
Alternatives Identification	Comparable	*Trade Studies*
Scope Definition		**Physical Verification**
Work Breakdown Structure	Comparable	*System Breakdown Structure*

Figure 12 - Scope Management Process Overlap with Systems Engineering Processes

The first area of overlap is between the initiation process of the PMBOK Scope Management and the Requirements Analysis process of IEEE 1220 and EIA 632. These processes all require the definition of project or enterprise constraints that limit options of the project or system. [Kauffman, 1998] Between all of these manuals, there are no decisive conclusions as to who the dominant profession should be when dealing with the initiation process. As alluded to before, the Systems Engineer and the Program manager should strive to work together at the program outset to assure that there is not a duplication of efforts.

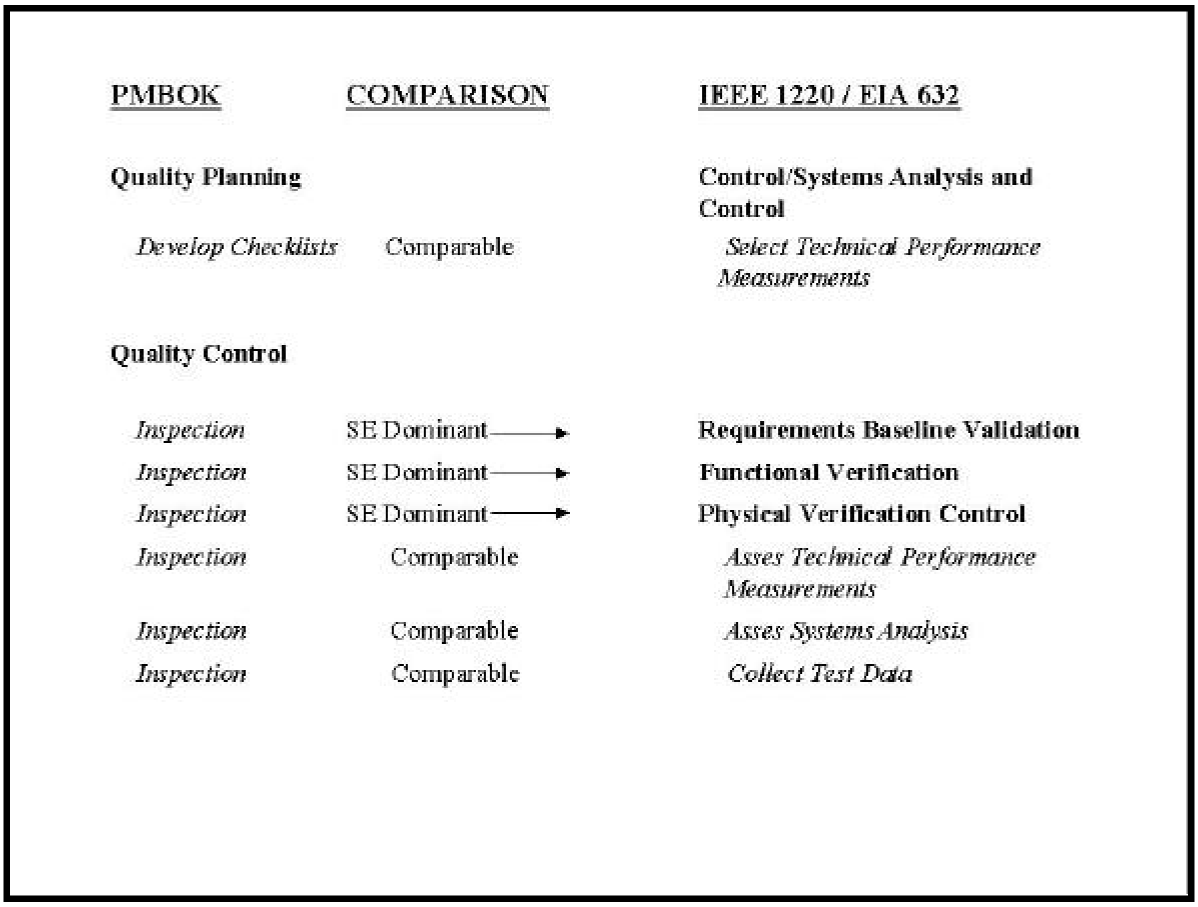
The next area of overlap is the product analysis subprocess of the PMBOK scope planning process with several major processes of Systems Engineering Requirements Analysis, Functional Analysis, and Synthesis. The PMBOK touches only lightly on product analysis when discussing scope planning while the Systems Engineering standards go to great depth in defining the processes and subprocesses to perform these tasks. Obviously this area of knowledge is dominated by the Systems Engineering profession.

Another area of overlap within scope planning is between the two PMBOK subprocesses, cost/benefit analysis and alternatives identification, and IEEE 1220 trade studies within the Systems Analysis and Control process. These processes are all similar, with neither profession dominating the area. Although the processes are similar, if the alternative in question is technical in nature, Systems Engineering should take the lead, whereas Program Management would likely be the best candidate for the lead in a cost/benefit analysis action.

The final area of intersection in Scope Management is the PMBOK's use of the Work Breakdown Structure (WBS) as a tool for scope definition. A strong correlation exists between the WBS and the IEEE 1220 System Breakdown Structure (SBS) developed in the Physical Verification process. The primary difference is that the PMBOK defines several WBS structures, including product-related, as well as, contractual and organizational structures. However, it would appear that, at a minimum, the SBS should serve as an input to the WBS to avoid duplication of work or the use of different product structures that may only serve to confuse or complicate the efforts of the project team. [Kauffman, 1998]

4.1.2.3 QUALITY MANAGEMENT

According to the PMBOK, there are three aspects to quality management: quality assurance, quality planning and quality control. If the PMBOK and IEEE 1220 and EIA 632 are examined closely, as in Figure 13 it is clear that the Systems Engineering standards deal with aspects of quality planning and quality control but do not address quality assurance as defined by the PMBOK. Due to this, we will assume that quality assurance is in the domain of Program Management.



PMBOK	COMPARISON	IEEE 1220 / EIA 632
Quality Planning		**Control/Systems Analysis and Control**
Develop Checklists	Comparable	*Select Technical Performance Measurements*
Quality Control		
Inspection	SE Dominant →	**Requirements Baseline Validation**
Inspection	SE Dominant →	**Functional Verification**
Inspection	SE Dominant →	**Physical Verification Control**
Inspection	Comparable	*Asses Technical Performance Measurements*
Inspection	Comparable	*Asses Systems Analysis*
Inspection	Comparable	*Collect Test Data*

Figure 13 - Quality Management Process Overlap with Systems Engineering Processes

When the subject of Quality Planning is broached, the discussion centers on the utilization of tools and techniques such as benefit/cost analysis, benchmarking, flowcharting and design of experiments. The Systems Engineering standards reference the use of Technical Performance Measurement (TPM) as a technique to plan and track the evolving performance of the system design against key performance requirements. Both professions have the tools and techniques in place to perform Quality Planning. However, due to the diversity of the tools and techniques, it is imperative that the project have early coordination and planning to develop the most effective plan for the system (of systems). This does not necessarily mean that the Systems Engineering method is chosen or the PMBOK method is chosen; it is possible, moreover probable, that the system will reap the benefits from a tailored approach between the two methods. Once again, this tailoring must be done and agreed to by all parties early in the development of the system. Both product results and process results of a project are dealt with in the quality control process of the PMBOK Quality Management knowledge area. The PMBOK covers a wide range of quality topics including: inspection techniques, rework decisions, control charts, Pareto diagrams and statistical sampling to drive process adjustments. Inspection techniques are handled at the process level, however, by the Systems Engineering standards. Several subprocesses within the IEEE 1220 Control process, systems analysis and the collection of test data, also address inspection techniques. “One aspect of Systems Engineering related to quality control, the assessment of product performance against Technical Performance Measurements, can be viewed as a complimentary technique that should be used within the PMBOK quality control process.” [Kauffman, 1998] The Systems Engineering standards provide a tremendous level of detail regarding the processes of Requirements, Baseline Validation and Functional Verification. The PMBOK merely defines these tasks as inspection tasks. Obviously the detail that the Systems Engineering manuals provide points to a decided domination of this particular aspect of inspection by Systems Engineering.

Within the Quality Control arena, Systems Engineering and Program Management combine to offer various levels of description and/or instruction. As demonstrated, the expertise varies from area to area. Further, as mentioned before, it is advisable to avoid duplication of efforts with open dialog at the outset of the program. This being said, management and production of a technical product requires a technical knowledge that Program Management may or may not have. The Lead Systems Engineer of the system *must* assume the technical management of the quality of the product (see Chapter 5).

4.1.2.4 RISK MANAGEMENT

There are four major subprocesses defined in the PMBOK under the definition of Risk Management: identification, quantification, response development and response control. It is clear that the Systems Engineering standards (IEEE 1220 and EIA 632) addresses these concerns, although they are referred to by a different name as seen in Figure 15. The Systems Engineering standards do not, however, go to the detail that the PMBOK does in describing how to handle these processes. Therefore, as seen in Figure 15, these processes are Program Management dominant, though the Systems Engineering processes should provide significant input when planning system Risk Management. The following is a pictorial description (Figure 14) of the Risk Management cycle as it pertains to all aspects of a system.

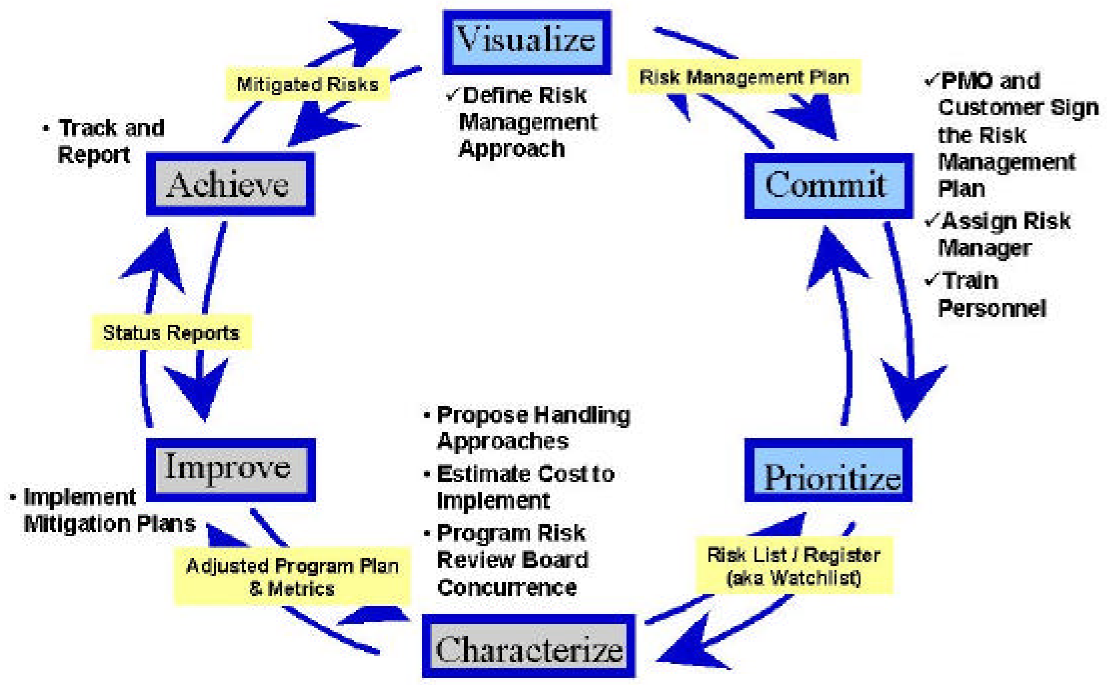
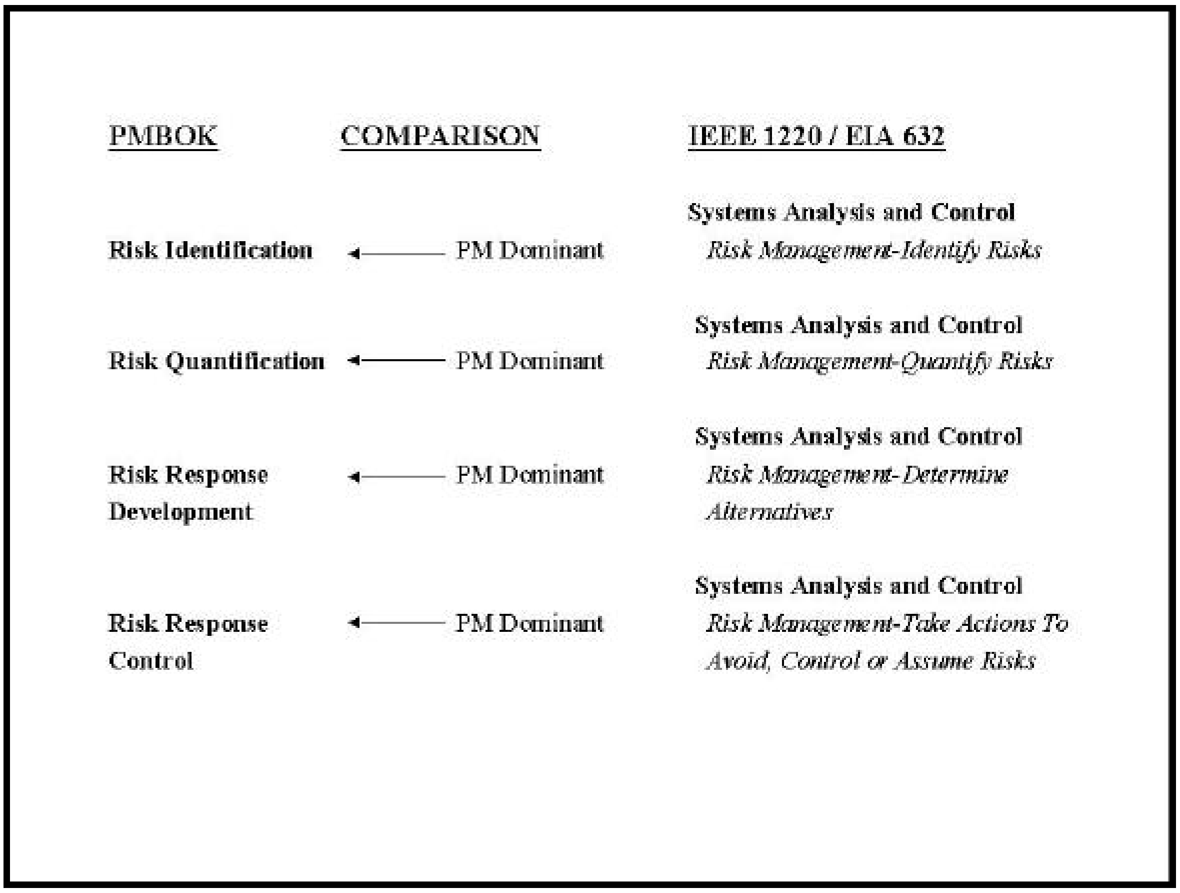

Visualize
✓Define Risk Management Approach
Risk Management Plan
✓PMO and Customer Sign the Risk Management Plan
✓Assign Risk Manager
✓Train Personnel
Commit
Prioritize
Risk List / Register (aka Watchlist)
Characterize
• Propose Handling Approaches
• Estimate Cost to Implement
• Program Risk Review Board Concurrence
Adjusted Program Plan & Metrics
Improve
• Implement Mitigation Plans
Status Reports
Achieve
• Track and Report
Mitigated Risks

Figure 14 - Risk Management "Life Cycle"

4.1.2.5 TIME MANAGEMENT

The Systems Engineering processes overlap with the PMBOK Time Management in three areas. First, the PMBOK process of activity sequencing overlaps the Master Schedule process as defined in the Systems Engineering standards. Secondly, the calendar based Systems Engineering Detailed

Schedule overlaps with the PMBOK schedule development processes and finally, the PMBOK's schedule control section overlaps/corresponds to the IEEE 1220 progress measurement processes. In all three areas, the PMBOK provides far more detailed tools and techniques than the System Engineering standards. The activities covered in the Systems Engineering processes represent only a small portion of the activities defined in the PMBOK. Two further processes defined in the PMBOK: activity definition and activity duration estimating do not have corresponding processes in the Systems Engineering standards. Clearly, Program Management dominates the Time Management body of knowledge.



PMBOK	COMPARISON	IEEE 1220 / EIA 632
Risk Identification	← PM Dominant	**Systems Analysis and Control** *Risk Management-Identify Risks*
Risk Quantification	← PM Dominant	**Systems Analysis and Control** *Risk Management-Quantify Risks*
Risk Response Development	← PM Dominant	**Systems Analysis and Control** *Risk Management-Determine Alternatives*
Risk Response Control	← PM Dominant	**Systems Analysis and Control** *Risk Management-Take Actions To Avoid, Control or Assume Risks*

Figure 15 - Risk Management Process Overlap with Systems Engineering Processes

4.1.2.6 COMMUNICATIONS MANAGEMENT

One aspect of PMBOK Communications Management, information distribution, overlaps with the Systems Engineering Control process, data management. These processes are nearly identical; therefore neither profession can claim dominance. The three remaining processes under the PMBOK Communications Management: communications planning, performance reporting and administrative closure are without comparable processes in the Systems Engineering processes. [Kauffman, 1998]

4.1.2.7 SUMMARY OF KNOWLEDGE AREAS

In summary of this section, there are three areas of knowledge overlap between Systems Engineering and Program Management: those where one profession dominates the other, those where the knowledge areas compliment each other and finally, those where there is a duplication of effort. Table 6 summarizes the distribution of knowledge areas.

Project Management	**Complementary (C) or Duplicative (D)**	**Systems Engineering**
• Project Plan Development • Project Plan Execution	• Overall Change Control / Configuration Management (D)	
• Project Scope Management	• Define Project or Enterprise Constraints (D) • Conduct Cost / Benefit Analysis and Alternatives Identification / Trade Studies (D)	• Requirements Analysis • Functional Analysis • Synthesis • Technical Scope
• Time Management		
• Cost Management		
• Quality Assurance • Process Quality Control	• Checklists / Select Technical Performance Measurements (C) • Inspection / Assess TPMs (C) • Inspection / Asses Systems Analysis and Collect Test Data (D)	• Requirements Baseline Validation • Functional Verification • Physical Verification
• Human Resource Management		
• Communications Planning • Performance Reporting • Administrative Closure	• Information Distribution / Data Management (D)	
• Risk Management		
• Procurement Management		

Table 6 - Summary of Knowledge Area Dominance by Profession

It may appear on the surface that these two groups, although different, share many of the same qualities, enough so that one might argue why there are two groups at all. Closer examination, however, reveals that there is a need for the two distinct groups. Project stakeholders can not afford to use anyone other than the professionals best equipped with the proper body of knowledge to handle the job. When dealing with or being part of one of these two groups, the most effective use of knowledge is simple -plan! If responsibilities are delegated early so that the appropriate professionals are utilized in a way to maximize their body of knowledge, the system will surely flourish! "Proper planning by project stakeholders can create a beneficial synergy between Project Management and Systems Engineering rather than conflict and duplicative efforts." [Kauffman, 1998]

4.2 TRAINING THE TECHNICAL KNOWLEDGEBASE

The second "management" issue that we choose to discuss is that of training the Systems Engineers. Doubtless, this is one of the most overlooked aspect of Systems Engineering. Contrary to popular belief (especially that of graduating college seniors), a college or advanced degree, while important and impressive, is merely the beginning of the professional training required for a fully groomed Systems Engineer.

Jerry Watts and Brian Mar performed a study in 1996 asking professionals in the Systems Engineering field what further training was required for a person to be come a "complete" Systems Engineer. Surveys were sent to between March 19, 1996 to May 15, 1996 to various industry leaders with a response rate of 47.2%. The importance of 23 types of skills and knowledge for Systems Engineers was categorized in a range from "unnecessary" to "critical." Responses to the survey were provided on a Likert-like scale with 1 corresponding to "unnecessary," 2 corresponding to "useful," 3 corresponding to "important," and 4 corresponding to "critical." Table 7 displays the results of their survey.

Further, their survey asked the respondents how many hours of training their organizations offered for various types of skill and knowledge. To calculate the mean and standard deviation for the results, the answer code for each answer category was replaced by the center point for the answer category so that 4 was used for "8 hours or less;" 24 used for "between 8 and 40 hours;" 120 was used for "between 80 and 160 hours;" and 200 was used for "more than 160 hours." Table 8 displays these results.

Watts and Mar reached an interesting conclusion. "The survey showed no significant relationship between the hours of training offered and the emphasis placed on that type of skills and knowledge. This lack of a relationship is curious. There are several possible explanations. On possible explanation is that the Systems Engineers being hired already posses the desired skills and knowledge. A second possible explanation is that companies do not adequately assess the importance of various types of skills and knowledge in planning their training programs.

Skills and Knowledge	Mean	Standard Deviation
Basic Problem Solving	3.720	0.458
Development and Management of Requirements	3.640	0.638
Teamwork and Communication	3.600	0.577
System Optimization (Trade Studies and Decision Making)	3.520	0.714
System Interface Definition	3.480	0.510
Mission Analysis and Design	3.400	0.816
System and Component Integration	3.360	0.638
Architecture Development	3.320	0.690
Risk Analysis and Management	3.320	0.852
Systems Engineering Processes	3.320	0.852
Breadth of Experience with Different Systems	3.080	0.640
System Simulation and Modeling Skills	3.040	0.676
Design Techniques	3.000	0.707
Test and Verification Design and Management	3.000	0.764

Table 7 - Types of Skills and Knowledge Ranked by Decreasing Importance

A third possible explanation is that some types of skills and knowledge are easier to acquire, thereby requiring less training time. A fourth possible explanation is that the training is intended for a broader audience than Systems Engineers; hence, the amount of training offered for any category should correlate with the with the overall emphasis placed on the category by all interested parties. This is reasonable for general engineering skills and knowledge such as general problem solving and product design, but does not explain why there is no relationship for skills and knowledge that are typically reserved for Systems Engineering, such as the definition and management of requirements and definition of interfaces.” [Watts and Mar, 1996]

These statistics lead the reader to believe that companies do not adequately assess the importance of various types of skills and knowledge planning in their training programs. It is of further interest to note that some programs that companies do have to train are programs that are overlapping responsibility with Program Management as discussed previously in this chapter (Risk Assessment, Project Management Processes, etc.).

Skills and Knowledge	Mean	Standard Deviation
Tools and Automation	66.1	82.8
Project Management Processes	63.3	16.0
Mission Analysis and Design	46.6	68.6
Design Techniques	46.4	66.3
Teamwork and Communication	44.5	56.0
Systems Engineering Processes	43.5	53.7
Engineering Specialties (Logistics, Maintainability, Safety, etc.)	40.3	66.0
Architecture Development	31.3	48.7
Development and Management of Requirements	27.7	41.9
System Optimization (Trade Studies and Decision Making)	26.1	41.329.2
Test and Verification Design and Management	24.9	47.6
Human to Machine and Human to Human Interface Design	24.6	41.8
Basic Problem Solving	24.4	46.7
Present and Predicted Technology	23.5	41.0
System Simulation and Modeling Skills	22.8	40.1
Risk Analysis and Management	22.1	47.0
Breadth of Experience With Different Systems	21.7	41.7
System Interface Definition	19.3	18.8
Depth of Knowledge in a Specific System	19.1	41.8
Engineering Economics	17.6	26.4
Capture of the Design Data Base	16.7	26.4
Commercial and Military Standards	16.7	14.0
System and Component Integration	14.3	

Table 8 - Hours of Training Offered for Each Type of Skills and Knowledge, Ranked by Mean

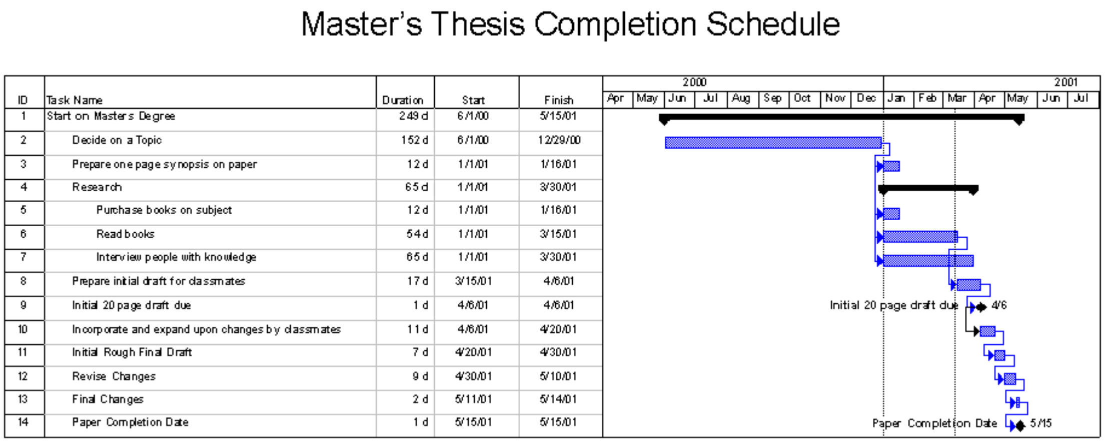
As a side note, this would further emphasize the fact that Program Management and Systems Engineering Management are not sitting down at the outset of a program to discuss roles and responsibilities. As a Systems Engineer in a System of Systems, current understanding of the technology, interface definition, network architecture, etc. are all subjects that have the ability to be quite dynamic. Keeping the technical knowledgebase trained is synonymous with success whether the

Systems Engineer in question is the Lead SE in charge of the system and other Systems Engineers (Technical Knowledgebase) or the SE new to the program that is "low man on the totem pole."

4.3 SCHEDULING

When dealing with schedule, most engineers recognize it as a necessary evil that is best left to Program Management. However, as Lead Systems Engineer and to a lesser extent, Systems Engineering for the system, we must understand that we are ultimately responsible for the schedule. Now, when we say that Systems Engineering is responsible for the schedule, we are not talking about the day-in day-out maintenance of the schedule; we are discussing the responsibility for the performance of the work that causes the schedule to be what it is. As such, a Systems Engineer must be aware of how technical decisions that are made will have a "ripple down" effect on the schedule. John Donne the famous Elizabethan poet once remarked, "No man is an island, entire of itself…" The same can be said of Systems Engineers (particularly Lead SEs) who only worry about the technical aspects of a program, i.e. there is more to the system than the technological side. It is granted that the main thrust of a System Engineer should be the technical performance of the system (of systems) however, a Systems Engineer *must* understand how any and everything affects the system. Schedules sometimes cause the system to be dynamic and at other times may be used to understand the flow of how the system will come together. Regardless of the application, the Systems Engineer should use schedules as a tool to help them accomplish their goal.

As such, there are several versions of electronic scheduling programs available. Most large companies use *Microsoft Project*® to do their scheduling. Whatever software the Systems Engineer's company uses, he/she should become very familiar with it in order to utilize it as a tool as mentioned above. Figure 16 shows an example of *Microsoft Project*® utilized by the author to help facilitate the writing of this paper.


Master's Thesis Completion Schedule

ID	Task Name	Duration	Start	Finish
1	Start on Masters Degree	249 d	6/1/00	5/15/01
2	Decide on a Topic	152 d	6/1/00	12/29/00
3	Prepare one page synopsis on paper	12 d	1/1/01	1/16/01
4	Research	65 d	1/1/01	3/30/01
5	Purchase books on subject	12 d	1/1/01	1/16/01
6	Read books	54 d	1/1/01	3/15/01
7	Interview people with knowledge	65 d	1/1/01	3/30/01
8	Prepare initial draft for classmates	17 d	3/15/01	4/6/01
9	Initial 20 page draft due	1 d	4/6/01	4/6/01
10	Incorporate and expand upon changes by classmates	11 d	4/6/01	4/20/01
11	Initial Rough Final Draft	7 d	4/20/01	4/30/01
12	Revise Changes	9 d	4/30/01	5/10/01
13	Final Changes	2 d	5/11/01	5/14/01
14	Paper Completion Date	1 d	5/15/01	5/15/01

Figure 16 - An Example of *Microsoft Project* â Scheduling Tool

4.4 BUDGET

Another tool available for the System Engineer to use is also thought of as a Program Manager's responsibility – budget. Although responsibility for the budget falls to the Program Management Body of Knowledge, the Systems Engineer should not hesitate to utilize it to his/her advantage. Most companies have a type of budgeting/accounting system that they utilize. It is beyond the scope of this paper to discuss all of these different programs, however, the Systems Engineer should become familiar with the reporting process that his/her company uses as it will ultimately help them with their job.

Occasionally, Program Management will ask a Systems Engineer (usually the Lead SE) for help in estimating budgets for the program. How the Systems Engineer will bid these efforts will vary wildly from system to system. It is the author's opinion that, as a general rule, the Systems Engineer should do three things after they have estimated the scope of the work and the amount of effort required; 1) discuss the results with the people doing the work. This will almost always influence the outcome. It is not uncommon for the people who are doing the actual work (programming, hardware, specialty engineering, etc.) to have a though different from the Systems Engineer that may cost more time or save more time. 2) When the discussion with the people performing the work is complete, talk to their supervisor. Usually, this will be a person in a lead role, i.e. software team lead, etc. Verify what was discussed with the people that work for them in step one. Finally 3) apply an engineering "safety factor" of 20%. The number that is derived should be as good an estimate as is possible.

When this number is returned to Program Management, it should be firmly stood on as the Engineering bid. Most Program Managers take the Systems Engineer’s input and incorporate it into their final bid. However, sometimes Management or the Customer will “challenge” Engineering to reduce cost. Depending on the system, the Systems Engineer may be able to help or, they may not. The key is to understand what the job can be done in such that a quality, final product is delivered to the customer. If asked to go below the “comfort zone”, the Systems Engineer should stand firm. Delivering a quality product is a top priority of the Systems Engineer.

The bitterness of poor quality still remains after the sweetness of the low cost is forgotten.

-Unknown

4.5 CONCLUSIONS ABOUT PROGRAM MANAGEMENT RESPONSIBILITIES FOR THE SYSTEMS ENGINEER

As discussed in this chapter, the Systems Engineer has responsibilities that overlap with Program Management. These gray areas may change from program to program. The ability of the Systems Engineering and Program Management to embrace this overlap and turn it into an advantage will largely influence the success or failure of the program. Key planning meetings at the outset of the program is the best place and time to discuss these overlaps and agree on who has what responsibility.

Further, even in areas such as budgeting and scheduling, where Program Management is the responsible party, the Systems Engineer should understand and utilize these tools to his/her advantage and, ultimately, the program’s advantage.

Training continues to be the chance that Systems Engineering has to stay “on top of the game.” With rapid technology development occurring today, staying up to date will not only improve the Systems Engineer, it will improve their ability to make a product. It should be noted that companies differ on the roles that they wish for their Systems Engineers to play. Training as to what is expected from the SE as well as what company protocol to follow may be as important to the company as the degree earned by the Systems Engineer!

Of course, this chapter has not been all-inclusive as to what Program Management responsibilities the Systems Engineer *could* assume/utilize. However, hopefully it brings awareness to the fact that most engineering but especially Systems Engineering can not be a technical island unto itself. A Systems Engineer is responsible for just that – the entire system, technically. He/she should utilize everything within their grasp to perform this role more effectively. As for the gray areas of overlap, plan wisely, early!

CHAPTER V
QUAILTY ASSURANCE AND THE SYSTEMS ENGINEER

Quality can not be inspected into the system.

-Robert Morland

5.1 SOME THOUGHTS ON QUALITY ASSURANCE

"A lot of people seem to think that God was the first quality professional, and those who have followed Him in the business of judging what is good and what is not good do so by divine right. They know that there are two types of people, the good and the bad, and that the good get to decide which is which. The purpose of having an independent quality department is to get away from this kind of thinking – to limit such choices to those who have nothing to gain from the decision." [Crosby, 1980] Quality Assurance is not just any facet of building a system, it is the only facet. All actions taken with the development of a system of systems *require* Quality Assurance planning in the initial stages and presence during all stages. For some reason, people that work and develop systems have come to see Quality Assurance as the "bad guy," that they are out to get them on all manners of technicalities. Throughout this chapter, we will discuss how very important Quality Assurance personnel are, not only to the final product but also to the bottom line. There is no excuse for having errors or mistakes in the system (of systems). Hard-working, dedicated professionals committed to exercising personal integrity make having a goal of zero defects possible. "You [a company] can get rich by preventing defects. You [a company] can never make much by simply "assuring" or "controlling." Police officers try to keep things under control. Lawyers often work at prevention. You have never seen a rich policeman. There are a lot of rich lawyers." [Crosby, 1980] In a system of systems, quality assurance becomes doubly important. Not only is the Systems Engineer concerned about the quality of the entire system, they must be concerned with the quality of the individual components of the system. In addition to the ethical requirements to create the system to the best of their technical ability, there is a potential monetary benefit for the Systems Engineer's company as well!

5.1.1 QUALITY DEPARTMENT FUNCTIONS

5.1.1.1 INSPECTION

At the heart of every quality program is data collection through mechanical and/or visual inspection, which permits the evaluation of a product or service status. Phillip Crosby came up with three things that all measurements should have in common. First, they are planned evaluations conducted in a planned manner for a planned reason; second, they are conducted by professionally trained inspection personnel; and third, the inspection personnel are organizationally separate from those organizations they are inspecting. Inspection is truly the first line of defense in a quality program. Remember that an inspector is not truly an inspector unless the inspection is independent and last. Changes made after the inspection void the inspection. Any change, even if due to the result of an inspection, requires a re-examination by the appropriate quality personnel. Remember that quality is everyone's duty. It is a constant effort by all to create the best possible product. Without doubt, an inspector will miss an occasional error with a system. It is up to the entire team to ensure that the number of errors that an inspector has to find is minimal to minimize the number of final errors (hopefully to zero). Remember the quote at the beginning of the chapter-"Quality can not be inspected into the system." Inspection is the final step in the Quality chain not the first.

5.1.1.2 TESTING

Today, most quality decisions are made based on the results of some type of functional testing of the product in some stage of development. In addition to the people who perform the actual testing, Test Engineers are employed to assist in developing test equipment concepts and establishing testing procedures; dependent upon the size of the program, these Test Engineers are usually Systems Engineers with specific knowledge. The only purpose of testing is to determine if the system will perform to the basic requirements document laid out in the contract. As a word of caution to Systems Engineers who might be performing the role of Test Engineer for the first time, there is an important pitfall to avoid. If, as the Test Engineer, the system passes its tests, the result will be a resounding "Ship it!" However, should the report be negative, the result will be "Test it again." Whenever the product passes, it is acceptable. For that reason, Test Engineers must write test procedures that require exact test results.

The biggest problem in testing today is software. "The computerized test equipment naturally requires computer programs that direct, and for the most part, conduct the test. There is no reason why these programs themselves should not be considered products and should not be required to meet the qualification test requirements. Testing equipment and software now represent a major financial investment, sometimes larger than the development cost itself. It is wise to make certain that this important function is well controlled." [Crosby, 1980] We have tested hardware for quite some time and, although equally important, we have developed processes that we are comfortable with. We are now going to discuss different ways of testing software, however, it should be noted that all of these tests are derivatives from some type of hardware testing. The result is a rather nice system of testing that is applicable for all aspects of the system. The Systems Engineer merely needs to tailor these tests to the system and/or testing at hand. The implementation of these tests may vary depending on what you are developing and what change management system you are using. The tests can even vary depending on the customer you are trying to satisfy! Some of these tests are performed with Quality Assurance on hand, the others are performed to assure that when Quality Assurance is on hand, things go very smoothly.

5.1.1.2.1 UNIT (MODULE) TESTING

Unit testing is performed at the module level of the system (or software). This is the most basic unit of the system. In a system of systems, this testing not only goes down to the individual system level, but to the piece parts within the individual systems. Often, software developers refer to unit testing as module testing. As with anything problems tend to "snowball" becoming bigger and bigger the longer they go unheeded. Ideally, enough time is scheduled to perform unit testing prior to incorporating the modules into the whole. Too often, however, teams develop the module but fail to unit test it for any of several reasons – in the interest of saving time, because of too heavy of a workload or shortened project schedule, and so on. As a result, when later testing is to be done and a problem is found, what would have been a simple fix at the unit level is monumental at other levels. We come back to our seeming recurring theme that advanced planning saves time and money in the long term.

If the team develops parts and software modules and accurate test cases based on the Requirements Document, Project Scope Document, Functional Specifications, and Detailed Design Specifications for the project (which they certainly should be doing), then those unit tests provide the means to perform the following key measurements:

- Determining whether the module (mechanical or software) contains major design or functional defects as measured against the specification documents.
- Determining when the module (mechanical or software) is ready to move into integration testing, entrance testing and ultimately into system testing.

- Determining whether the module (mechanical or software) performs correctly on the test environments (such as network, hardware, operating system, relational database management system, and so on) that are necessary for your potential customer base.

Usually, testing that is done at the unit level is quite informal, especially in software. The programmers devise whatever tests they deem necessary to fully exercise their code. At this stage, hardware may have been tested at the manufacturer, especially in this age of utilizing Commercial-Off-The-Shelf (COTS) products. In a perfect world, unit tests will be formally written and consistently executed by development for all modules, software and hardware alike.

5.1.1.2.2 INTEGRATION TESTING

Once all of the modules have passed their individual unit testing, we will need to begin putting all of the modules together. Once again, one can see how this is applicable at several levels. For a system of systems, it is applicable for each individual system that will make the whole and for the components of each individual system. Of course, perfectly sound modules can start acting in perfectly unsound ways after they are combined with other modules.

At integration time, the tests that will be performed will need to focus on two things. First, we want to assure that the module still functions as expected, i.e. when tested at integration time, the results of testing need to provide the same results as unit testing. The second aspect of this testing is called regression testing and is even more important. Regression testing is a battery of tests designed to uncover any effects (detrimental or otherwise) of additions to the unit tested modules. The type of testing that is typically done at the unit and integration level is known as glass-box testing. Most glass-box testing occurs at the unit level, however, it may occur at the integration level. The "level" at which a product or system is being tested is as follows:

- **Glass-Box Testing –** Glass-Box testing (or White Box testing) refers to the concept that the integrator (read Systems Engineer when dealing with the integration of the system of systems) has a good idea exactly what the input, processing and output of the tests is to involve and can provide specific tests of the data. Because the integrator knows so much about the inner workings of the system, they can view the results through the transparent glass box and immediately understand the meaning of the testing efforts.
- **Black-Box Testing –** Black-Box testing involves testing after unit and integration testing are complete. Black-Box testers have no knowledge of the inner working of the system as the previously mentioned Systems Engineer has. Black-Box testers typically only test the installation and functionality via the user interface. The inner workings of the system are a black-box to these testers. Black-box tests may uncover many problems that glass-box tests

miss because the black-box testers can not subconsciously tailor their tests to the known needs of the customer, contract, etc. Because the testers can't see in the box, they may test in a way that the integrator didn't think of.

Consider the following example given by Greg Mandanis: Suppose that the system logic contains the Effective Date value to be a date no more than one year in the past or the future. If this condition is not met, then the following error message should appear:

```
Insert failed – Case Effective Date too far in the past/future.
Enter a different Effective Date…
```

A glass-box test of the program's data validation algorithms on the New Case form may include testing a boundary of the Effective Date field by entering a date way in the future or way in the past. If the incorrect date is accepted by the system, then the glass-box unit test failed, and the programmer [in our case integrator] should open a defect report in the change management system and correct the code before performing the integration test.

A black-box test of the program's [in our case system's] data-validation algorithm could also identify such an error, but the black-box tester would only know that the incorrect data was accepted for some reason as opposed to the actual logical error in the programming.

Conducting numerous glass-box tests early on against the various program modules (such as the user interface and engine) and testing the various design document specifications (such as the data model and data-flow diagram) is a good practice for developers to employ during the development of a system. Remember, the Systems Engineer should run these tests both before and after integration of a combination of modules:

- To verify that each module works as designed as a stand-alone component.
- To verify that two or more modules get along together, without breaking the other's designed functionality.

5.1.1.2.3 ACTIVITIES PRIOR TO SYSTEM TESTING

As continuously emphasized in this paper, planning will "cure what ails you." As final system testing nears, there are certain items that should be checked and double-checked prior to the final testing. Of course, that list will vary from project to project, however, here is a generic list:

- **The demo data (test data that the demo uses) is current.** For example, the data would have to test for a ten-digit zip code to be current today. We don't want the data to present any issues in preventing customers' acceptance testing.
- **The Technical Publications Team has finished the software's Release Notes.** This will ensure that the customer can install the software properly.

- **Unit testing is complete.** First things first. If unit testing is not complete, then the system testing should not begin.
- **Integration testing is complete on a formal build configured by the configuration manager (CM).** If the integration team hasn't finished the configuration testing, then the system testing should not begin.

5.1.1.2.4 SYSTEM TESTING

System testing refers to a detailed effort to test every single aspect of functionality of the system in an attempt to anticipate as many user errors as possible. System testing may begin only after the successful completion of unit and integration testing. System testing is most beneficial if it is done in-house under "live-fire" conditions. The testers will try to rip apart the system and make it fail. Earlier tests try to make sure that the system lives up to the design expectations, but during the system tests, the testers are literally trying to break the system. Now, this is not a malicious attempt to break the system. If the system is going to break, it is best that it break with the appropriate people there to take action. It is best if this testing can even be done in-house, however, that is rarely possible with a system of systems.

Beta testing is a term that is usually applied to software, however, we will choose to loosely apply it to our system. Generally, systems testing is done in house while beta testing is done out of house. The biggest difference between the two types of testing is that systems testing tends to be extremely structured while beta testing is very unstructured. The difference between these two types of testing is important because systems testers and beta testers focus on different criteria. In systems testing, we want a structured, methodical approach to testing that exercises the entire system. Beta testing undergoes testing in a haphazard way that is different from any preconceived notions or constraints initially placed on the system.

5.1.1.2.4.1 DEVELOPING THE SYSTEM TEST PLAN

The system test plan should pick up where the unit testing and the integration testing left off, with the objective to be the complete testing of all modules and components as a fully integrated system. When creating the system test, it is imperative that it be done to t plan that can be traced back to the contractual requirements document. The following are some general steps to consider when creating the system tests:

1. **Specify the source of testing that is needed.** This step is usually pretty easy, as systems tests should cover every single facet of the system. If the project is very large, the work may be broken up among several people such that the scope is limited.
2. **Define the test environment.** Include all hardware and software that is needed for the test equipment. Indicate any test data requirements if large data files are required.
3. **Break down System testing effort into major categories.** Here is where the various types of tests to be run are indicated. Below is a subset of possible tests:
   - **Stress testing.** This system test involves multiple testers and/or automated test machines that issue an inordinate amount of transactions in order to simulate a heavy use environment. The objective is to determine if the system can handle peak-volume activity. This testing is often run concurrently with performance testing.
   - **Functionality testing.** This system test exercises the various functions of the system when testers process various transactions.
   - **Integration and compatibility testing.** These tests exercise the combination of modules to determine if they can work together or not.
   - **Recovery testing.** In this testing, the system is forced to fail and tested to determine how well it recovers from the failure. Key to this test is the system's ability to trap error messages necessary for recovery from the failed condition.
   - **Performance testing.** The objective of this system test is to measure the system's performance at run time.
   - **Security testing.** The objective of this test is to verify that the system provides sufficient safety to thwart illegal access and potential harm to the system. Of course, the security test will need to be done in such a manner as to accommodate the final security level of the system. Further, in a system of systems, the security levels of the lower level systems will need to be taken into account when deciding the final security level.
   - **Boundary testing.** These tests check the minimum and maximum data input ranges for the system.
   - **Concurrency testing.** This system test assesses how well the system handles multitasking, communication, and synchronization between tasks. Some questions to be answered during this testing are: Which of several users will have first dibs, and which users will be locked out, etc.

Obviously, most of the above testing is software-related in nature. It is assumed that hardware testing such as environmental testing was accomplished at the unit test level or before.

4. **List all of the test cases that will be tested.** This is an excellent crosscheck for the contractual requirements document.
5. **Write the individual test cases.**

5.1.1.3 QUALITY ENGINEERING

Quality Engineers are the Systems Engineers of the quality world. Broadly, Quality Engineering is supposed to be responsible for determining and planning the work for the rest of the quality department. They should be responsible for the overall quality concerns of the company by ensuring that all the loose ends are tied up such that the system as a whole will achieve its desired results. This means that they must decide who inspects and tests what, where data is to be collected and who should supply information to the system.

"Quality Engineers should collaborate with Design Engineering concerning a new product's performance characteristics and meet with Manufacturing Engineers concerning the way the product will be manufactured. In that way Quality Engineers can determine how the product should be inspected, tested, and controlled during its life both in and outside of the company. Detailing these requirements, training people to accomplish them, and measuring results are what Quality Engineering is all about." [Crosby, 1980]

5.1.1.3.1 CORRECTIVE ACTION

"Today, most nonconformance problems, with the exception of unknown phenomena, are preventable. All that is required are some organizational discipline and professional guidance." [Crosby, 1980] The old adage of "slow and steady wins the race" is never truer than when dealing with the complex. A system of systems is certainly complex. Unknown phenomena will happen, however with early planning and professionalism, a zero fault system is possible.

When problems are found, the key determinant to the system is not necessarily what is found, but what is done with what is found. When dealing with the quality organization, all of the planning, inspection, testing, measuring and other activities that quality performs is a complete waste of time if they do not lead to the prevention of future problems. The saying "Those who do not learn from history are doomed to repeat it" is quite to the point in this situation. The best sources of information about situations requiring corrective action are observation of actual rejections and analysis of trends. Because the evidence is so obvious, actual rejections are the easiest. Trends are less obvious; they require detective work that may require lab analysis and other kinds of sleuthing.

A corrective action database is highly recommended. Usually, when dealing with a system of systems, the components of the system whole will not be co-located. Because of this, a database allows for a “shared” knowledge that can allow operators at one site to expedite the resolution of any problems that they have by learning from the mistakes of others. This is corrective action in its most pro-active form.

5.1.1.3.2 AUDITS

Very little is spoken about more and understood less than the auditing process. “Auditing is the last refuge of those who really don’t know how to run a prevention-oriented life. Audit is the Bat Masterson of business. When you get into trouble, just call old Bat. He’ll find all of the bad guys and drag them all to justice.” [Crosby, 1980]

When used properly, an audit can be an invaluable tool. It is a planned examination of a function, carried out either by determining conformance to procedures in process or by critical analysis of the product or service that is the result of the process. That is it. There is no method that yields better results in exposing shoddy, inattentive or misguided work. However, this is all that the audit will catch; undedicated, bored or careless activity.

Philip Crosby gives these few basic rules for an audit to follow. These rules are applicable at all stages of development. As the Systems Engineer in a system of systems, effective use of the audit tool will be rewarded in spades in the future.

- Be specific about what you want to be audited and against what criteria the audit is to be accomplished.
- Select individuals to conduct the audit that couldn’t possibly be interested in the outcome one way or the other.
- Brief the audit team carefully and give them time to write a proper report.
- Do not tell them in any manner what kind of results you expect to find.
- Remember that the findings will point only to the front line troops. The real cause of the problems lies behind the findings.

5.1.1.3.3 QUALITY EDUCATION

As discussed in Chapter 4, continuing education is imperative to a product line’s success. Specific to quality, education must be paramount to all involved in this process (quality). How will an inspector verify the newest functionality if they do not understand what it is? Further, the latest quality standards themselves require refresher courses. According to Philip Crosby, quality education takes three basic forms

1. Orientation to the concepts and procedures of quality; the problems that have a harmful effect on the product; and the expectations of the customer.

2. Direct skill improvement in such specific things as soldering, computer programming, procedure writing, etc.
3. A continual low-level but concentrated barrage of quality idea communications to serve as reminders and conditioning, to make quality a thought always in everyone's mind. Nothing flashy, just positive ideas that are in good taste and current.

5.1.1.3.4 PRODUCT SAFETY

There are a few thoughts to convey on product safety. How can we keep our products from hurting our customer or others? The stories in the newspaper are certainly unnerving with large jury awards and the like. When handled maturely, a safety problem is no different than any other problem. If you are wrong, admit it, correct it and smile the whole time. People only want to be treated fairly until their rights are trampled on; then they want revenge. Let me be very pointed in this fact. This issue gets back to being a concerned professional and doing what is right by everyone. Product safety is not a legal problem, it is an ethical one.

5.1.2 HANDLING DEFECTS

Inevitably, there will be problems in putting the system of systems together. If there weren't, Systems Engineers wouldn't be necessary! How these defects are handled is the number one determinant in the success or failure of the overall system. All of these problems will be reported in the Change Management System (CMS). Changes to the baseline and problems with the technical baseline can occur at any time. The following list of common causes of defects was modified from Mandanis [2000] to apply to a system of systems:

- **Insufficient or Erroneous Requirements:** Did the customer *clearly* state what they wanted you to build? Do you *understand* what the customer wants built? Deficiencies in the Technical Requirements Document (TRD) or Project Scope Document (PSD), functional specifications and detailed design specifications can lead to defects.
- **Deviation in Coding from Design Specifications:** Occasionally a programmer develops code that deviates from the original design specifications. If done as an enhancement, make certain that the customer is in agreement. If the programmer intentionally deviated from the specification, warn once and keep a sharp eye on their work; terminate the employee on the project (if not the company) the second time. Intentional disregard for the customer's wishes is poor business and can lead to safety issues.

- **User Documentation Defects:** When creating the end-user documentation, including operational instructions, software loading procedures, etc. there is incomplete, erroneous or misleading information.
- **Insufficient or Erroneous Testing:** This one strikes close to home. This is directly under the perue of the Lead Systems Engineer. Defects may arise due to lack of or insufficient module, integration or system testing. Further errors may arise if the customer requires a Formal Acceptance Test (FAT) and the error survives through it.
- **Hardware Damage:** Insufficient vendor testing or module testing of the hardware can lead to a defect. Care should be taken in these cases because hardware damage can lead to a safety problem.
- **Incorrect Configuration and Reconfiguration of the Hardware or Software:** Changes to the baseline ***MUST ALWAYS*** be recorded in the change management system. Defects can arise from incorrect configuration and/or reconfiguration of the module, integration or system build candidate. Software developers can sometimes fail to check in or reconfigure their code changes into the configuration management system, resulting in fixes missing in the next software build.

The above list is by no means unabridged in its account of defects possible. When one occurs one of these (or another) type of defect, they must be recorded in the change management system. Usually, in the CMS, there are subsets of issues to be reported. The two main subsets will be 1) a process to identify change to the baseline of the system (i.e. configuration management which we will discuss in detail in Chapter 6) and 2) discrepancies to the technical requirements of the project. When discrepancies are reported, they need to be flagged with a priority. Priority will usually be similar to the following:

- **Priority 1 – FATAL:** The entire system is adversely affected and completely inaccessible. This error can be caused by anything including software, hardware or network configuration.
- **Priority 2 – CRITICAL:** A major part of the system or critical functionality is affected and no reasonable workaround is available. In the system of systems, this would equate to an entire module of the overall system going down.
- **Priority 3 – MEDIUM:** A moderate part of the system or critical functionality is affected. A reasonable workaround is available, but requires tremendous effort on the part of the *user*. The solution may not be postponed forever.
- **Priority 4 – MINOR:** System functionality is not adversely affected and users are able to complete their transactions despite the problem. Often, this issue may be transparent to the user.

- **Priority 5 – ENHANCEMENT:** A request to modify the baseline with a change that has not been previously identified in the Technical Requirements Document or the Project Scope Document.

It is foolish and perhaps unethical to keep these defects from your customer. The customer will understand that there will be problems as the system is built. They may not like the fact that their schedule may slip or that it may cost them more money, however, they would always rather know than not know.

When a problem is encountered, typically the following steps are followed depending upon the organization in charge's processes and procedures:

- **Perform a valid test against the most up-to-date version of the system.** Before entering a defect into the CMS, verify all of the facts concerning the error. Determine the level (module, integration or system) that the problem affects. Be certain that once the level is determined, there is no "waterfall" affect, i.e. a system problem may affect the integration and module levels.
- **Isolate where the problem is.** During any test cycle (Module, Integration, System or FAT), if the tester uncovers an error, the problem needs to be isolated whether it is in the delivered software, a user-error, a data error or a customer customization error.
- **Enter only a legitimate Discrepancy Report or enhancement into the Change Management System.** Once the problem is isolated and determined to truly be an error, we have a legitimate DR that has been screened for entry into the CMS.
- **Submit the DR or Enhancement to the Change Control Board (CCB).** Not every DR or enhancement is automatically targeted for future release. This board is comprised of Engineering Management (specifically the LSE), Program Management, Configuration Management, Quality Assurance and the customer at a minimum depending on the issues. The CCB will discuss the merits of the action in question, taking all aspects into account and render one of the following decisions:

Approval. The change should be adopted in the project. This approval may or may not result in a change in project scope. For instance, the change may only result in a change of how a particular project goal is implemented in software.

Deferral. The change has merit, but will adversely affect the current project scope, schedule, or resources. The change is tabled until some future date or software version.

Field Fix. The change has merit, but only limited applicability. It is approved to be made in a particular customer implementation of the software, but not in the general release.

Refusal. The change is rejected, and does not warrant future consideration. The basis of the refusal can either be made public or not, based on common practice in your organization.

The CCB's decision should be required to be unanimous. Obvious problems can occur if there is dissention in the decision.

- **Write Documentation.** This usually is included in a software error. Documentation would include installation notes and release notes that include a list of defects and fixes included in the release.
- **Unit and Integration test the fix or enhancement.** This is done to verify the fix and insure that no new errors have been introduced as a result of the fix.
- **Perform regression tests on the fix or enhancement and the documentation.** This test all areas of the system's functionality that are determined to be at risk from the particular fixes. By fixing even one area of functionality, the risk is run of breaking a number of other areas. These tests are to be run by quality assurance with strong help from the Systems Engineers.
- **Deliver the fix or enhancement to the customers.** After the fix or enhancement passes the aforementioned QA testing, certify it for release. If the issue was a software fix, FTP and Web-based software download sites are a common delivery medium of choice in the brave new world of e-Commerce. Obviously, if the software is classified or proprietary to the outside world, other arrangements will need to be made.

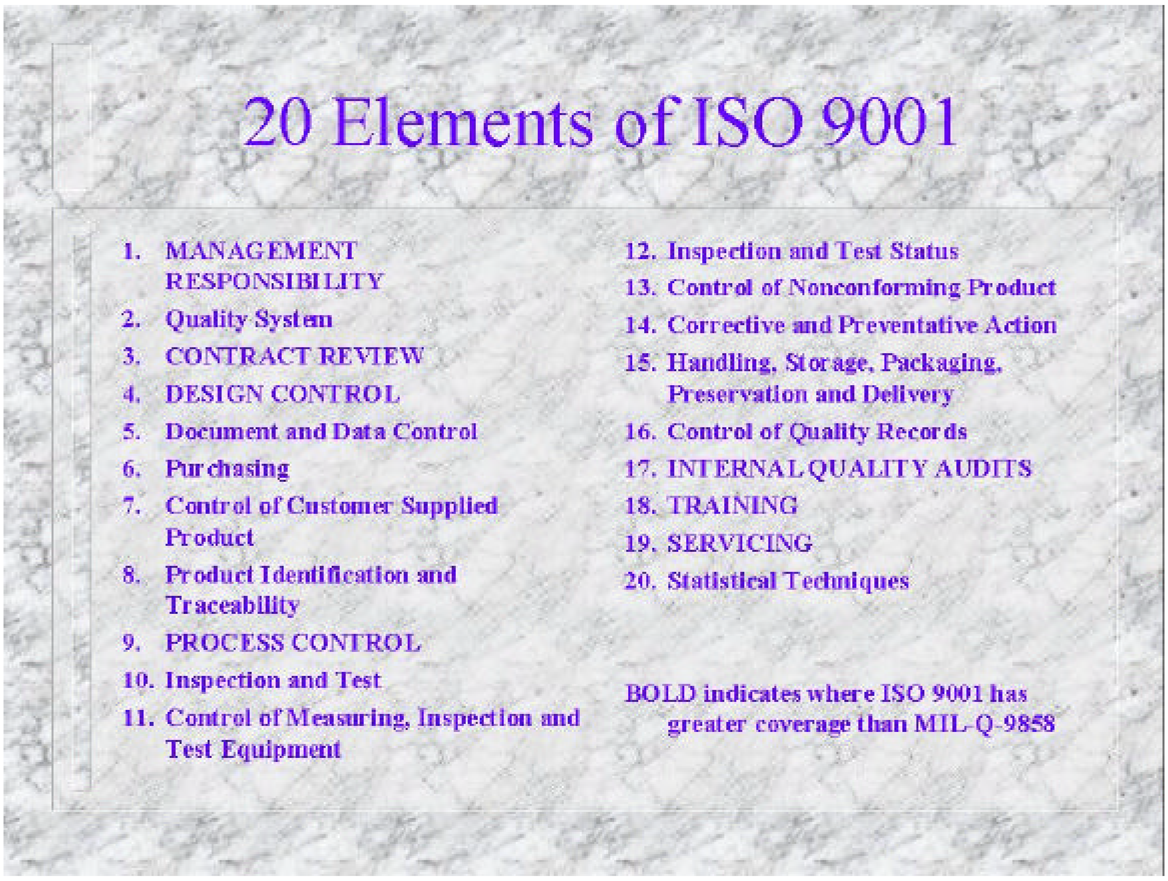
5.1.3 ISO 9001

Several people are intimidated when they see this phrase. They are unsure how this organization is going to change their business. ISO is from the Greek work "isos" meaning "equal." It is an international standard for assessing and registering management systems. It was developed by the International Organization for Standards (ISO) in 1987 and has been adopted by over 76 nations. One might be curious as to what a management system is. Briefly stated, the ISO 9001International Standard describes and defines the basic elements of a management system needed to ensure that our company's products and services meet or exceed our customer's needs or expectations. ISO only requires a few commonsensical efforts:

- An acceptable, documented management system
- Proof that employees actually follow the system and procedures
- Periodic audits of process quality
- A corrective/preventative action process that ensures that you fix areas that fail to comply with the management system.

In short, ISO 9001 says: 1) Document what you do, 2) Do what you document and 3) Have evidence to prove it.

Basically, there are 20 Elements of ISO 9001 as shown in Figure 17.



20 Elements of ISO 9001

1. MANAGEMENT RESPONSIBILITY
2. Quality System
3. CONTRACT REVIEW
4. DESIGN CONTROL
5. Document and Data Control
6. Purchasing
7. Control of Customer Supplied Product
8. Product Identification and Traceability
9. PROCESS CONTROL
10. Inspection and Test
11. Control of Measuring, Inspection and Test Equipment
12. Inspection and Test Status
13. Control of Nonconforming Product
14. Corrective and Preventative Action
15. Handling, Storage, Packaging, Preservation and Delivery
16. Control of Quality Records
17. INTERNAL QUALITY AUDITS
18. TRAINING
19. SERVICING
20. Statistical Techniques

BOLD indicates where ISO 9001 has greater coverage than MIL-Q-9858

Figure 17 – 20 Elements of ISO 9001

As demonstrated above, the key to achieving and maintaining ISO certification is documenting and following a process. Documenting a work process not only qualifies for ISO certification, it helps lay the foundation for the work to be done. In a system of systems, the intermingling of components and processes can be overwhelming if the processes are not followed.

Figure 18 shows an example of a high level product/system process. With apologies to the reader for the type being so small, several other "descriptive" figures will be shown that give greater detail to this figure.

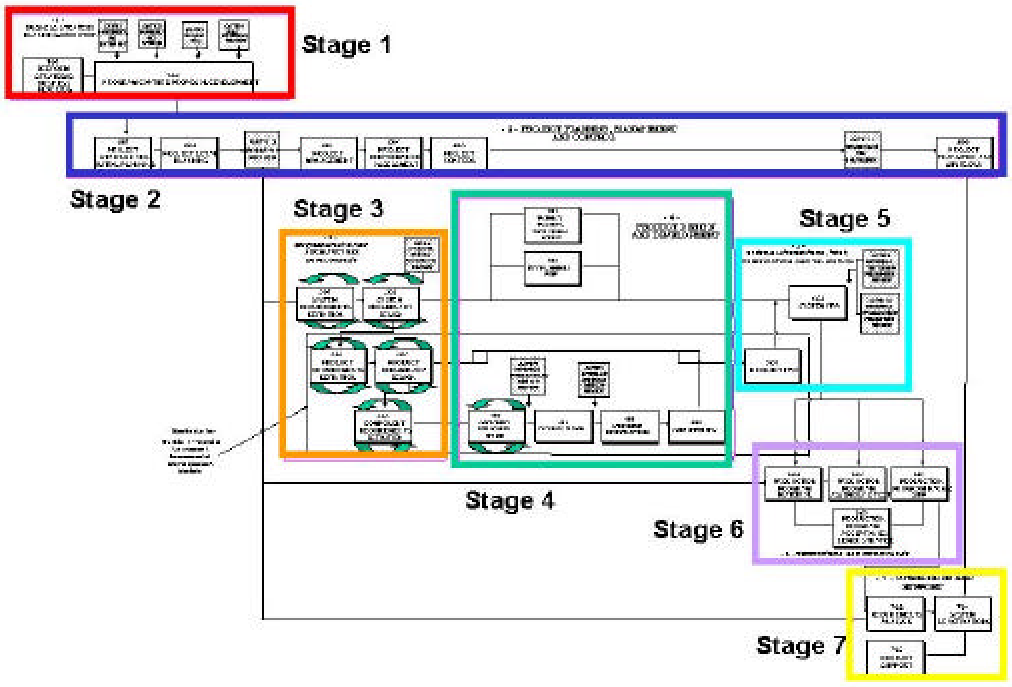


Figure 18 – Top Level Product Process

As shown in Figure 19, the Product Process is handled through a "gating" process. This process allows for decisions to be made by the appropriate level of management (Engineering, Program management or otherwise) as to whether or not the process is ready to proceed. Notice that the various levels of management are responsible for this process; Business Development for gates 1-4, Program Management for gates 5 and 11 and Engineering for gates 6-10. Whereas this process may be fine for some systems to follow, others may require a different approach. Any approach is fine as long as it is a documented process. Further, the process should be tailorable so that it may be applied at levels of the system under the top level (i.e. module or component level).

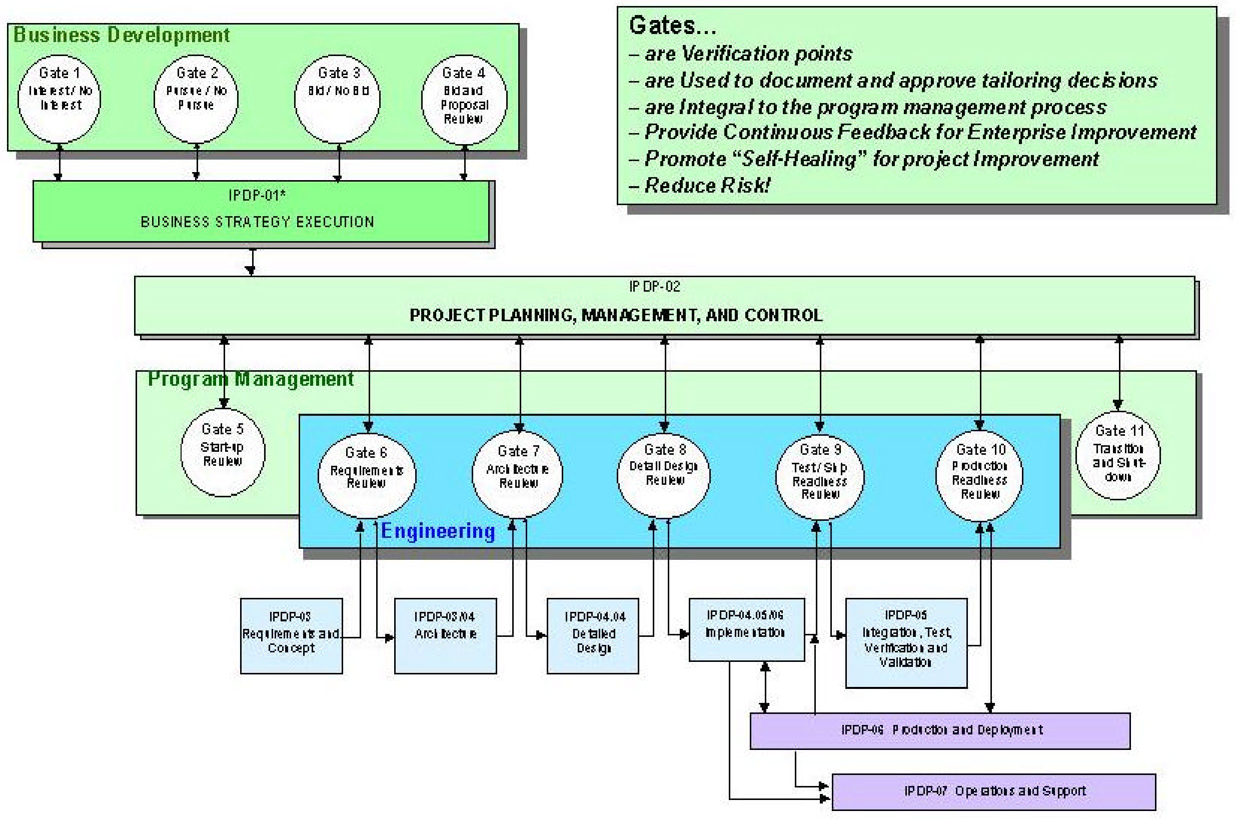

Business Development
Gate 1 Interest / No Interest
Gate 2 Pursue / No Pursue
Gate 3 Bid / No Bid
Gate 4 Bid and Proposal Review
IPDP-01* BUSINESS STRATEGY EXECUTION
Gates...
– are Verification points
– are Used to document and approve tailoring decisions
– are Integral to the program management process
– Provide Continuous Feedback for Enterprise Improvement
– Promote "Self-Healing" for project Improvement
– Reduce Risk!
IPDP-02 PROJECT PLANNING, MANAGEMENT, AND CONTROL
Program Management
Gate 5 Start-up Review
Gate 6 Requirements Review
Gate 7 Architecture Review
Gate 8 Detail Design Review
Gate 9 Test / Ship Readiness Review
Gate 10 Production Readiness Review
Gate 11 Transition and Shut-down
Engineering
IPDP-03 Requirements and Concept
IPDP-03/04 Architecture
IPDP-04.04 Detailed Design
IPDP-04.05/06 Implementation
IPDP-05 Integration, Test, Verification and Validation
IPDP-06 Production and Deployment
IPDP-07 Operations and Support

Figure 19 – Process Definition Through Gating Process

As the Systems Engineer evaluates the process that they are following, he/she will notice various levels of maturity in the process. Figure 20 shows pictorially the evolution that the process will go through. Obviously, level 5 is the desired level for the overall system, however, one will notice that with various groups of people involved, Program Management, Business Development, Engineering, etc., the individual components shown in Figure 19, may be at different levels of maturation. Evolving this process so that all groups are on the same page is one of the greatest challenges that the SE will have to face. Complicate this with the fact that in a system of systems, the SE will probably not only have to deal with different disciplines as mentioned above, they will probably also have to deal with different companies and their philosophies.

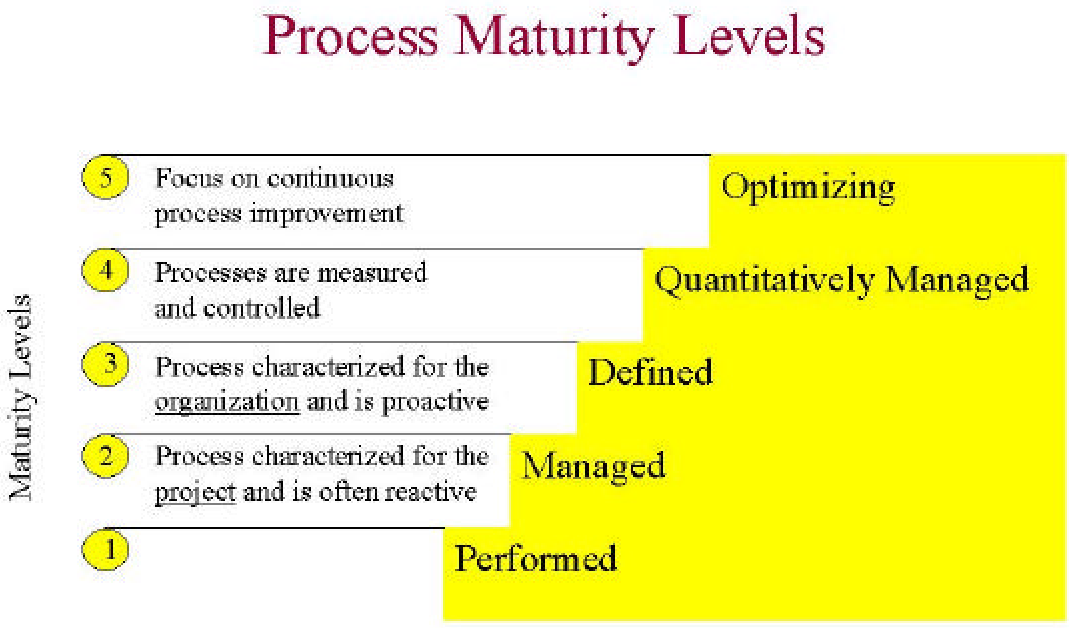

Process Maturity Levels
Maturity Levels
5
Focus on continuous process improvement
Optimizing
4
Processes are measured and controlled
Quantitatively Managed
3
Process characterized for the organization and is proactive
Defined
2
Process characterized for the project and is often reactive
Managed
1
Performed

Figure 20 – Process Maturity Levels

5.2 CONCLUSIONS ABOUT QUALITY

In keeping with the theme of the paper, quality of the product is enhanced with early planning. Whether the early planning has to do with the testing, safety concerns, training or process development, clearly defined roles at the outset will prevent most serious quality assurance issues from coming up. One can complete the project with no final errors. The Ford Motor Company has a motto that "quality is job one." As engineers, our commitment to a quality product and the resulting output are a calling card that will always be associated with us. We must always strive to ensure that quality is at the top of our priority list.

CHAPTER VI
CONFIGURATION MANAGEMENT AND THE SYSTEMS ENGINEER

The goals of using CM are to ensure the integrity of a product and to make its evolution more manageable. Although there is overhead involved in using CM, it is generally agreed that the consequences of not using CM can lead to many problems and inefficiencies. The overhead of using CM relates to time, resources, and the effects on other aspects of the software lifecycle.

-Susan Dart

6.1 SOME THOUGHTS ON CONFIGURATION MANAGEMENT

Controlling the ever-creeping scope and other risks affecting the system is one of the biggest challenges that the Lead Systems Engineer must face. As soon as the marketing department presents the Requirements Document at the project kick-off meeting, the team can respond with the Project Scope Document (PSD). In the PSD, the project team may identify which technical features that may be included-or excluded-from the project's scope. The mission of the technical team in these early stages is to negotiate the final terms of the technical scope with Program Management and the customer to keep the project scope from becoming a constantly moving target. During this initial stage, the LSE will need to identify and prioritize any potential risks that can jeopardize the project's success.

Some may wonder why we can't just rule with an iron fist and make zero changes. In an ideal world, refusing to make changes is great, in the real world, however, things aren't always as easy as this.

There are two types of changes that will need to be dealt with in the system, necessary changes and "nice" changes. Putting off "nice" changes is easy – they may be shrugged off or implemented at a later time. Necessary changes, however, are those that crop up during the development stages of the project. For example, it may be discovered that two modules do not work together properly, the situation may be due to poor planning, poor programming, poor networking, etc. but something needs to change. How the LSE (and others as the case may require) decides to make the change and what that actions means to the whole of the system.

6.1.1 DEVELOPING A CHANGE MANAGEMENT PLAN

Once again, the common theme of this paper pops up: plan early. However, this time it comes with a different twist; although we may plan and go to great lengths to nail down every conceivable facet of the project including creating a detailed WBS, the project can still fall victim to change (Notice, however, that with good initial planning, these changes are held to a minimum). Due to the fact that change is inevitable, the LSE (in conjunction with Program management) will need to devise a good Change Management Plan (CMP). A good CMP will encompass (at a minimum) the facets listed in Table 9 [This is a modified version of a table from Mandanis, 2000].

Identification	Change can come from many different places, both internal and external. A project can experience "scope creep" of the requirements, objectives, design, technologies, markets, staffing levels and other resources.
Assessment	After the scope change has been identified, the key is to identify the effects that these changes will have on the project. These changes can affect critical path, schedule and budget.
Aversion	Unfortunately, Murphy has many siblings, i.e. changes and problems rarely come singularly, they come in bunches. It is exceptionally important for the Systems Engineer to, after identifying and assessing the initial change, to attempt to learn from this change so that other changes and issues from the same and other sources may be averted.
Monitoring	The LSE must monitor the project to assure that changes do not "sneak in the back door."

Table 9 – Facets for a CMP to Have

The CMP that the LSE (or their company implements) will need to describe (at a minimum) the items in Table 10 [This is a modified version of a table from Mandanis, 2000].

6.1.2 HANDLING PROJECT CHANGES

In dealing with project changes, the name of the game is "proactive." Greg Mandanis [2000] identifies four steps that may be taken to handle project risks and changes; these will be modified to be more applicable to Systems Engineering and the system of systems.

1. Find out the system/project status from the technical team during weekly status meetings
2. Evaluate the cause and effect of the change

Key Project Documents	These documents include the Requirements Document, Statement Of Scope (SOS) and detailed design specifications. These documents will be used as the measuring rod to determine whether or not to make a particular change.
Software	Software is a key portion of the CMP as software will change more radically and more often than any other portion of the CMP. How these changes will be addressed should be included in the CMP
Baseline Project Plan (BPP)	This portion of the CMP will largely be used by Program Management. This plan is used to determine how the changes will affect the schedule. Chapter 3 has a more detailed discussion on how a Systems Engineer should deal with schedule changes.
Change Control Board (CCB) Composition	As mentioned previously in Chapter 5, the CCB should have, as a minimum: Engineering Management (specifically the LSE), Program Management, Configuration Management, Quality Assurance and the customer. This diversity will insure that all aspects of the system (not only the technical concerns of the LSE) will be handled.
Change Control Board Processes	This portion of the CMP is imperative to plan early. Within the CCB, there will be disagreements as to how to handle various aspects of change. Without predetermined protocol, i.e. determined prior to the first negative encounter, the CCB will have little chance to perform its job function.

Table 10 – Items for the CMP

3. Modify the plan and anticipate schedule changes to help Program Management and the customer plan for the change
4. Notify all parties involved of any project changes and allocation of resources

When changes are to be made, the CCB must approve the change regardless of the initiator. As shown in Figure 21, however, when a problem is reported and a CMDB is in place, often the problem has been reported before and is resolved more quickly. The Change Management plan should address problems such as these to allow for easy disposition.

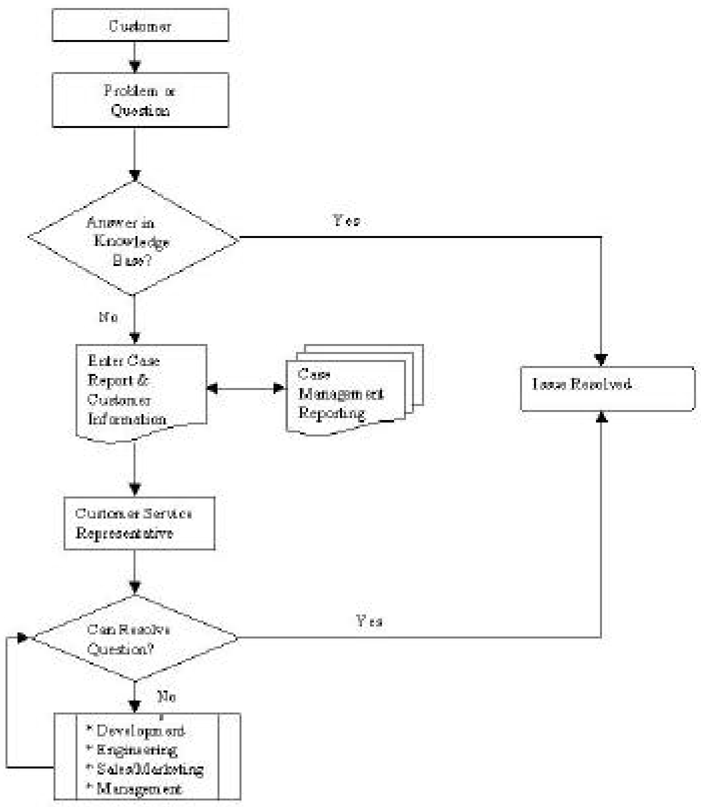


Figure 21 – Problem Resolution with a CMDB

6.1.2.1 GETTING PROJECT STATUS

It is imperative that, as the Lead Systems Engineer, we maintain a constant vigil over the status of the project. It is recommended that status meetings be held once a week. If meetings are more often (except in special and emergency circumstances), no work will get done because everyone will constantly be going to meetings. However, waiting more than a week will bear on the ability of the LSE to manage the technical team. These weekly meetings should include all members of the technical team (when practical) and should last for no more than an hour. Prior to the report, team leads should present the LSE with written reports of activity, problems, changes, etc. that have occurred since the last meeting. General issues that should be included in these weekly meetings are (at a minimum):

- Is the team following the prescribed development processes and procedures?

- Are any issues impeding the work in progress from meeting the planned technical goals and objectives?
- Is the team completing the project tasks on schedule?
- Has module testing reported any high-urgency or severe problems?
- Are there any training issues to discuss

As the building of the system continues, the formal meetings and status reports are the vehicles by which the LSE instills a sense of responsibility in the technical team.

6.1.2.2 EVALUATION OF THE CAUSE AND EFFECT

There may be, during the course of the project, a change that is waiting in the wings to jump up and cause the project to fail. As an example, a competitor introduces a new feature that your product does not include. A sure way to introduce problems such as release date slip, resource overallocation of resources or shoddy workmanship is to blindly accept huge scope creep changes. It is fine to say no to changes that can affect the outcome of the product, in fact, it is your job's duty to say no if the fate of the project is on the line!

Large changes inevitably end up in front of the CCB (which the LSE is part of). Two of the major responsibilities of the CCB is figuring out the cause of a change and determining how the change will affect the project as a whole. "In some cases, examining changes to past projects that [the LSE's organization] has undertaken may prove instructive. By understanding such past problems, [the LSE] can better respond to similar challenges in the current project." [Mandanis, 2000]

6.1.2.3 MODIFYING THE SCOPE, SCHEDULE OR RESOURCES

Let us suppose that after one of the LSE's weekly meeting, it is determined that the project is considerably over budget and more than 500 hours in excess of the scheduled budget. The answer is to go through the project cycle that initiated the project. Hopefully, this will help identify what area in the project has some "wiggle room" to help get the project back on track. Unfortunately, we may only change ONE aspect of the project's three aspects: scope, schedule or resources. It will need to be determined which needs to give so that the project can stay on course. Talk to the Program Manager and possibly the CCB to determine the most reasonable solution to the problem. If the changes that require modification lie outside the initial project scope, the responsibility may be transferred to Program Management/Marketing for later upgrade.

6.1.2.4 NOTIFICATION OF CHANGE

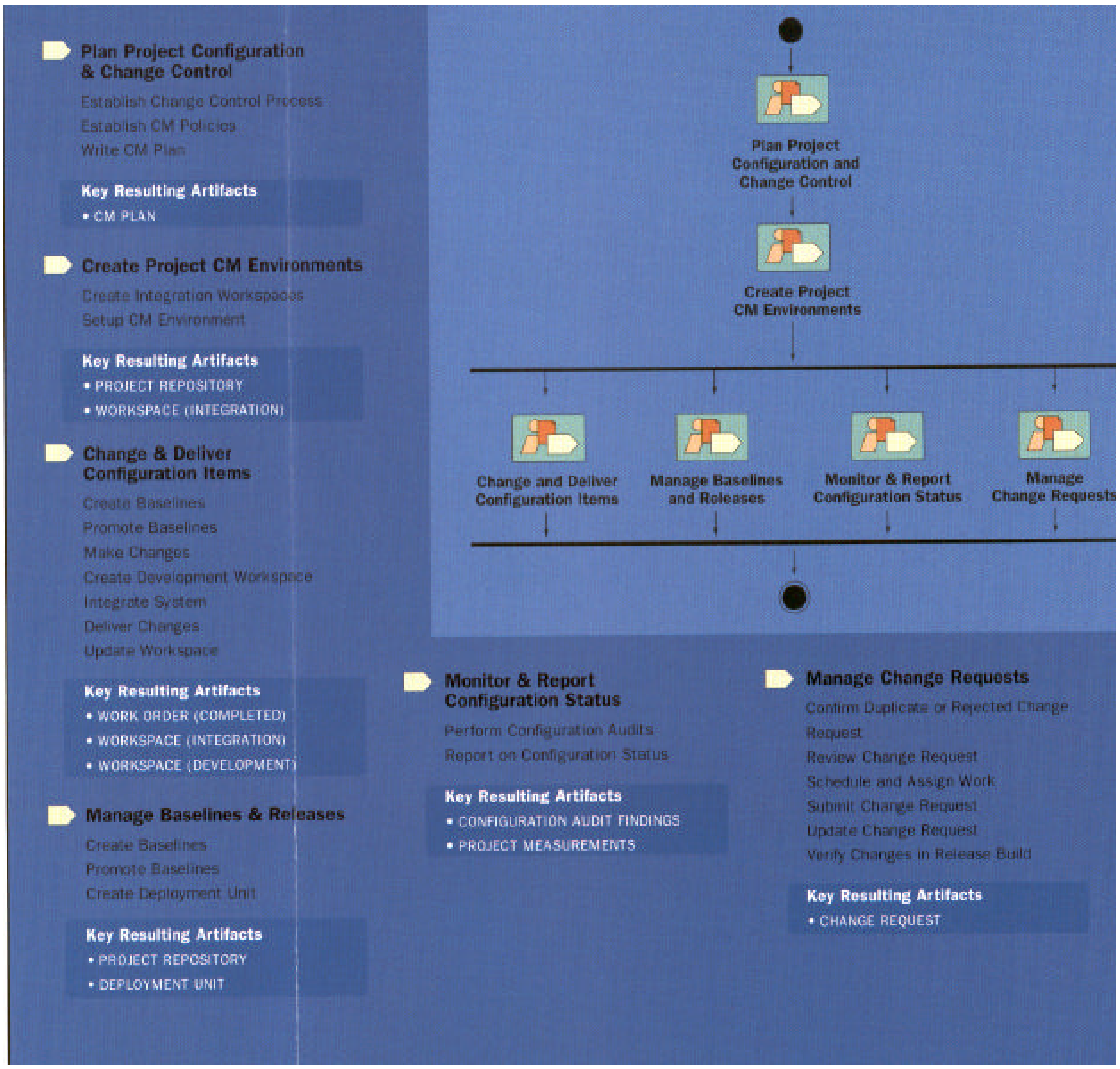
Whenever there is a change in the project's scope, time or resources, the LSE must absolutely communicate with Program Management, the technical team, the customer and all other affected parties concerning these changes. The notification needs to be formal in nature, sending all involved a memo and making appropriate changes to the Configuration Management Data Base (CMDB). If deemed necessary, the LSE may further delineate any repercussions of the change in the weekly status meeting.

6.1.3 CHOOSING A COURSE OF ACTION

After a change is approved, there are five alternative courses of action in dealing with the change so that the project doesn't slip too far off the yellow-brick road. First, the Work Breakdown Structure (WBS) can be reorganized so that tasks are completed simultaneously rather than in sequentially. This way, critical tasks are completed before any changes can delay their completion. Typically, this course of action is most effective when implemented early in the development process. A second technique is to alter the schedule by changing the start and finish times of non-mission-critical tasks. Of course, the customer must approve these schedule changes. The third approach is to throw more resources at critical activities that may be affected by the change. Using extra engineers doesn't always help. The trick to allocating extra resources is making certain that this doesn't create a problem in other areas, such as ignoring other tasks for the sake of finishing one task. A fourth approach is to diplomatically push for some of the features to be scheduled for a future release. Finally, if all else fails, the activity may be removed from the critical path. This would require the LSE to weigh the relative time and resources gained by removing a task versus potential losses in functionality and the resulting effect on the customer's wish list.

Table 11 shows the Risk Priority Matrix that allows the LSE to cross-reference the probability of a change or error occurring with the risk severity and come up with risk aversion options. The risk aversion option that is chosen will help to choose which of the five courses of action is most suited to the issue at hand.

As a further reference to a CM process, Figure 22 [Rational Software Group, 2000] demonstrates a generic approach that may be tailored to any system. It will be imperative that the LSE work with Configuration Management in the system of systems to ensure a high levelbeginning that translates down to a pinpoint functionality plan. If the LSE's organization already has a CMP that they utilize, fine. If not, create one!



Plan Project Configuration and Change Control
Create Project CM Environments
Change and Deliver Configuration Items
Manage Baselines and Releases
Monitor & Report Configuration Status
Manage Change Requests

Plan Project Configuration & Change Control

Establish Change Control Process
Establish CM Policies
Write CM Plan

Key Resulting Artifacts
- CM PLAN

Create Project CM Environments

Create Integration Workspaces
Setup CM Environment

Key Resulting Artifacts
- PROJECT REPOSITORY
- WORKSPACE (INTEGRATION)

Change & Deliver Configuration Items

Create Baselines
Promote Baselines
Make Changes
Create Development Workspace
Integrate System
Deliver Changes
Update Workspace

Key Resulting Artifacts
- WORK ORDER (COMPLETED)
- WORKSPACE (INTEGRATION)
- WORKSPACE (DEVELOPMENT)

Manage Baselines & Releases

Create Baselines
Promote Baselines
Create Deployment Unit

Key Resulting Artifacts
- PROJECT REPOSITORY
- DEPLOYMENT UNIT

Monitor & Report Configuration Status

Perform Configuration Audits
Report on Configuration Status

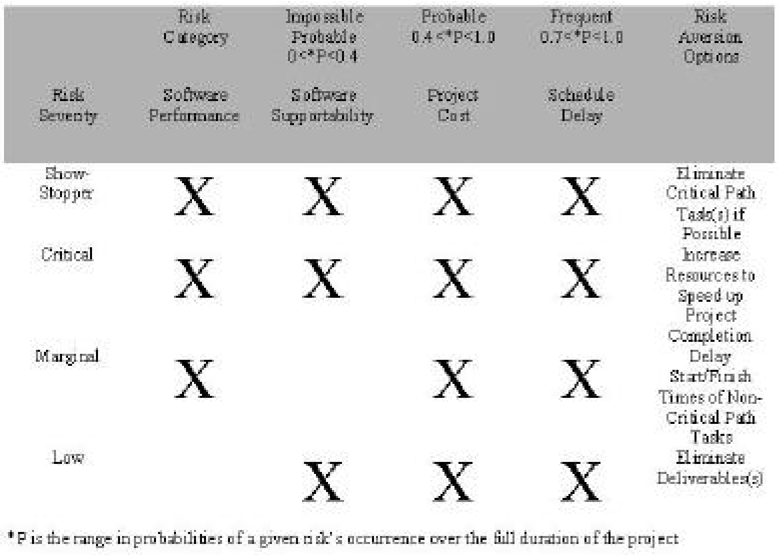
Key Resulting Artifacts
- CONFIGURATION AUDIT FINDINGS
- PROJECT MEASUREMENTS

Manage Change Requests

Confirm Duplicate or Rejected Change Request
Review Change Request
Schedule and Assign Work
Submit Change Request
Update Change Request
Verify Changes in Release Build

Key Resulting Artifacts
- CHANGE REQUEST

Figure 22 – A Generic Approach to a CM Process



	Risk Category	Impossible Probable 0<*P<0.4	Probable 0.4<*P<1.0	Frequent 0.7<*P<1.0	Risk Aversion Options
Risk Severity	Software Performance	Software Supportability	Project Cost	Schedule Delay	
Show-Stopper	X	X	X	X	Eliminate Critical Path Task(s) if Possible
Critical	X	X	X	X	Increase Resources to Speed up Project Completion
Marginal	X		X	X	Delay Start/Finish Times of Non-Critical Path Tasks
Low		X	X	X	Eliminate Deliverables(s)

*P is the range in probabilities of a given risk's occurrence over the full duration of the project

Table 11 – The Risk Priority Matrix

6.2 CONFIGURATION MANAGEMENT STANDARDS

Table 12, is a listing of some of the more common Configuration Management Standards (CMST)

Guidelines for Configuration Management	ISO 10007
Configuration Management	MIL-STD-973
Specification Practices	MIL-STD-490
Quality Systems – Model for Quality Assurance in Design/Development, Production, Installation and Servicing	ISO 9001
Document and Data Control Procedures Manual	DDC000

Table 12 – Some Common Configuration Management Standards

6.3 CONCLUSIONS ABOUT CONFIGURATION MANAGEMENT

As with all other issues for the Systems Engineer, the Configuration Management issue is best handled by planning early. It is imperative that a Configuration Management Plan (CMP) be implemented immediately after program award. The understanding of where the hardware and software are in the development cycle is of utmost importance to the LSE. Further, as a result of having a CMP, a change control board (CCB) should be implemented to assure that the changes implemented are what is best for the system as a whole. As a risk mitigator for change, the LSE should have status meetings once a week with the engineering leads to assure that all changes have been approved and the system is on the right track.

CHAPTER VII
SPECIAL ENGINEERING TOPICS

7.1 INTEROPERABILITY OF THE SYSTEM

When creating a system of systems, interoperability is obviously a number one priority of the engineering team. For example, are all of the operating systems the same? Are all of the applications the same? Getting the systems to function together as one system is the reason why we were hired in the first place. In getting this done, we apply some technical know-how, a little ingenuity and "viola!" a system of systems. However, when we speak of interoperability, there are a couple of layers below that we must discuss.

First, there may be political ramifications of the system. For instance, how will we deal with the scenario that one initial system belonged to the Navy and one initial system belonged to the Army. Obviously both groups will want as little change from their initial system to the final system as possible but something will have to give! A word to the wise, even though Systems Engineers are the technical leads of the system, leave the politics to the Program Manager! However, there still may be difficult problems that arise technically. These two systems may have been built under different standards and now are migrating to a third standard. The complexity of the interoperability has just increased by an order of magnitude!

One aspect of interoperability that I would like to focus on is Defense Information Infrastructure Common Operating Environment (DIICOE). This initiative, put forth by the United States of America Department of Defense basically wants its computer structure such that, no matter where an operator goes, whatever they do and no matter what they do with a computer, it appears completely the same. This is interoperability at its most minute detail! Of course, "appearing the same" is a relative term; different applications will appear different but the structure of the operating system and how the computer or network is set up to run is nearly the same. DII COE is mission-application independent. It is:

- -A basis for a system architecture
- -An approach
- -A collection of reusable software
- -A software infrastructure
- -A set of guidelines and standards

"The DII COE is *not* a system; it is a *foundation* for building a shared system." [Wheeler, 2001] Figure 23 [Wheeler, 2001] is a pictorial representation of the runtime structure of the common operating environment.

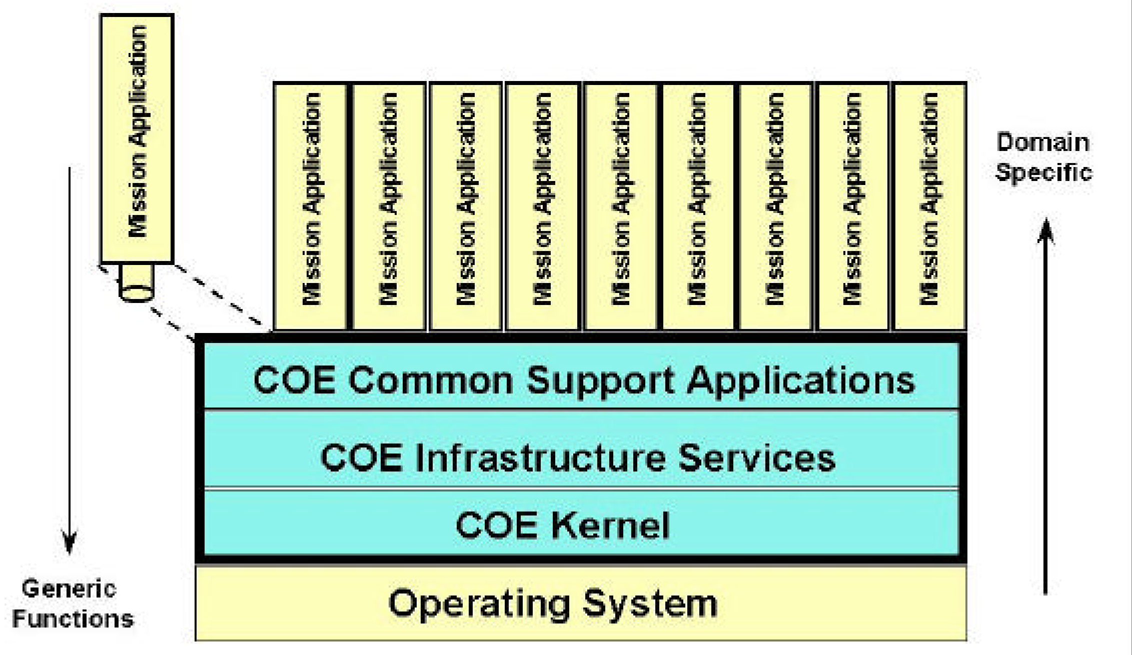


Figure 23 – DIICOE Runtime Structure

Notice that the less domain specific the structure is, the more generic the structure is. There is a similar structure to hosting applications as well. The end result is that most government contracts want the system to be DII COE Level 5 compliant. As this indicates, there are various levels at which a contract can comply. Figure 24 [Wheeler, 2001] is a pictorial representation of the levels of DII COE compliance. When one begins a DII COE process, they must begin a migration path. The purpose of this migration path is to take their "stovepiped" systems and create a DII COE "segmented" system. Segmentation is a process performed by software developers to convert their existing application software to a format that is capable of being installed by the COEInstaller and executed in the DII COE runtime environment.

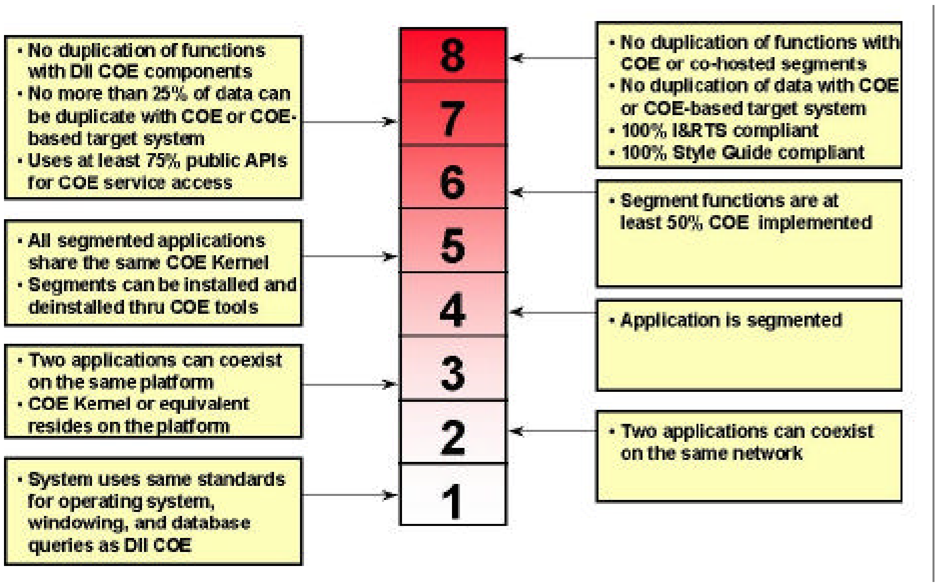
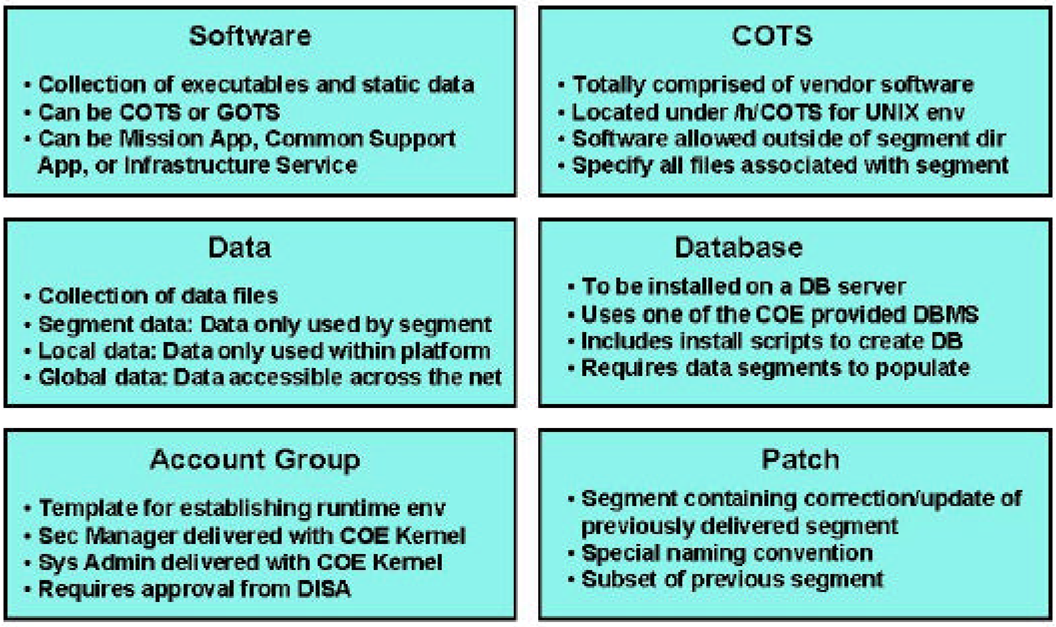

• No duplication of functions with DII COE components
• No more than 25% of data can be duplicate with COE or COE-based target system
• Uses at least 75% public APIs for COE service access
• All segmented applications share the same COE Kernel
• Segments can be installed and deinstalled thru COE tools
• Two applications can coexist on the same platform
• COE Kernel or equivalent resides on the platform
• System uses same standards for operating system, windowing, and database queries as DII COE
8
7
6
5
4
3
2
1
• No duplication of functions with COE or co-hosted segments
• No duplication of data with COE or COE-based target system
• 100% I&RTS compliant
• 100% Style Guide compliant
• Segment functions are at least 50% COE implemented
• Application is segmented
• Two applications can coexist on the same network

Figure 24 – Levels of DII COE Compliance

Obviously the next question is how is the environment expected to be segmented. Figure 25 [Wheeler, 2001] shows some of the different segment types found in a DII COE environment.



Software

• Collection of executables and static data
• Can be COTS or GOTS
• Can be Mission App, Common Support App, or Infrastructure Service

COTS

• Totally comprised of vendor software
• Located under /h/COTS for UNIX env
• Software allowed outside of segment dir
• Specify all files associated with segment

Data

• Collection of data files
• Segment data: Data only used by segment
• Local data: Data only used within platform
• Global data: Data accessible across the net

Database

• To be installed on a DB server
• Uses one of the COE provided DBMS
• Includes install scripts to create DB
• Requires data segments to populate

Account Group

• Template for establishing runtime env
• Sec Manager delivered with COE Kernel
• Sys Admin delivered with COE Kernel
• Requires approval from DISA

Patch

• Segment containing correction/update of previously delivered segment
• Special naming convention
• Subset of previous segment

Figure 25 – Segment Types of the DII COE Environment

Implementing a DII COE system is time consuming and costly, however, it is becoming more and more frequently a requirement of military contracts. It is a good system and a good idea, and, no matter how hard military contractors fight the inertia of change, it is apparently here to stay.

7.2 SYSTEM LOGISTICS AND THE SYSTEM ENGINEER

I can build you the perfect product. Just don't ask me to ship it.

-Anonymous

When logistics is mentioned to most people, they immediately think of shipping a product or the effort leading up to the shipment of a product. As a general rule, shipping and asset tracking do play a large part of system logistics. However, for the Systems Engineer and the system of systems, there are several other aspects of logistics to consider!

First, when designing a system of systems, a LSE must maintain the support base of the system. Logistics will become a nightmare if there is a "chink in the armor" of the logistics chain of the system. As discussed before, most systems of systems will not be collocated. How will information, data resources, etc. get from one site to the other without a proper logistics chain. It is a well known military fact that the supply chain to an army is one of the, if not the key factor to maintaining that force's readiness. Logistics usually does not fall under the Lead Systems Engineer's perue, however, they must work closely together in order to assure that this operation runs very smoothly. Depending on the application, the logistics network connecting the lower level systems, may be as important as the electronic network to the system.

Second, the LSE must be aware of the infrastructure of the system. Obviously the LSE is well aware of the technical infrastructure to the system, however, there is another side of infrastructure that must be made aware: how the systems work when NOT performing their technical job. What I am referring to here is not the data flow from one system to the next throughout the system, we are talking about the way that system A handles thing versus System B or System C. This "human factor" will become very important in getting the job done. We would like to believe that all aspects of every logistics department is the same, but they will not be! Understanding this dichotomy will help the LSE expedite the system build as they attempt to fit into this "infrastructure."

The third area of logistics that must be understood is in fact shipping. We are very used to the fact that, if we want to mail a letter, we put the letter into an envelop, apply a stamp and put it into a mailbox and we are done! When shipping parts of a system, or the entire system, the responsibility grows. When dealing with government contracts, there are numerous forms to fill out that transfer responsibility to the government, obey security regulations for the government, etc. Regardless of the system, how it gets where it is going will be an issue. Another area of shipping to think about is what happens if the entire system is to be moved. The LSE then gets to verify that the next site has all of the appropriate connections (i.e. CAT 5, phone, Internet, power, water, etc) so that the system can perform in the way that it was designed to.

None of these tasks present monumental tasks to be overcome, however, in a complex system of systems, the logistics of assuring that everything is where it is supposed to be, when it is supposed to be can be very difficult or very hard for the LSE. The way to get on top of this issue and stay on top is to be certain that an initial logistics plan was created if the LSE's company doesn't already have one. Once again, the key is to plan ahead.

7.3 SYSTEM SECURITY

Security of the system of systems is an issue that will be very dynamic depending on the final function of the system. E-commerce systems have very different security requirements than an industrial system. Military systems possibly have the most complex security regulations of all. Without a doubt, however, within a system of systems, network security will be an issue. Most of the lower level systems that comprise the overall system will need to communicate with each other. They will do this over a network. This communication varies radically from company to company, so, as stated before, how it is protected varies.

"The increasing globalization of business – and the almost universal accessibility of U.S. companies in a worldwide e-commerce marketplace – have created new, often unknown, and serious global security threats. The have created new "cyber-security" requirements that are reshaping the competitive climate and driving new measures to protect proprietary information.

Despite the pervasive nature of these threats, the United States government is not positioned to defend its electronic/information borders. The burden rests on companies and organizations themselves. To date, the response has been largely defensive, consisting of strategies to counter known hostile elements. In reality, however, threat elements constantly identify new vulnerabilities, making configuration-dependent and passive measures ineffective." [Vigilinx, 2001]

In order to counter these issues, there is a need for the implementation of Security Intelligence. If we define Intelligence, we might say it is information needed by decision makers that provide insight into the capabilities, activities and intentions of hostile elements.

In the military, this definition is accurate as well, accurate intelligence allows the leaders to know how to user their resources to their best advantage.

Not too long ago, one could know reasonably well who their competitors were and what measures they could reasonably take to nullify any advantage that that competition had in the marketplace. Since every major corporation is now "networked" together via the Internet, "cyber-security" is reshaping the competitive climate and driving new measures to protect proprietary information.

A commission established to appraise U.S. Intelligence in 1996 found that "*…the scope of the actions required to deal with the problem, especially the protection of computer networks in the private sector, would necessarily exceed the roles and capabilities of U.S. Intelligence agencies."* If the government is not able to help defend information networks, reliable Security Intelligence must be fashioned to take care of the problem.

Hazard has been part of trade since the onset of time. The wares are different nowadays, but networks to the "outside world" are still threatened by:

- **Benign Intruders** – electronic explorers and hackers
- **Script Kiddies** – cyber-delinquents and crackers
- **Opportunists and Thieves** – criminals engaging in computer – assisted crime
- **Hactivists and Extremist Groups** – political, religious and social elements
- **Industrial Espionage and Sabotage** – competitors, labor elements and disgruntled employees
- **International Espionage** – state–sponsored intelligence gathering
- **Information Warfare** – direct attack or unintentional collateral damage

Network Security Intelligence (NSI) is a subset of an intelligence package. This subset should be owned and operated by every modern, computer-based enterprise in order to maintain an effective security posture. "SI is comprised of vital information concerning specific technology-linked vulnerabilities and threats and their impact on the security of electronic information. It enables organizations to effectively manage risk, minimize system security weaknesses, prepare rapid responses and gain the confidence necessary for implementing judicious corrective actions." [Vigilinx, 2001]

When dealing with a system of systems, network intelligence will be very important to the final customer. The network will need an intelligence product that will identify threats and apply preventative or remedial actions that will keep the system's information proprietary. There are several commercially available products that are multi-dimensional and address a wide range of IT and security concerns. These products will provide an "extra set of eyes" for network protection.

CHAPTER VIII
MAINTENANCE OF THE SYSTEM AND THE SYSTEM ENGINEER

When dealing with any system, but especially a system of systems, maintenance and parts obsolescence become an exceptionally important facet of the system. Since maintenance is usually thought of as something to be done after the system is deployed, usually very little up front planning is performed to take into account the Preventative Maintenance (PM) of the system. This way of thinking could not be more off base! Maintenance should always be one of the first things discussed due to the fact that it will drastically affect how the initial parts order for the system will be handled.

In today's world, Common-Off-The-Shelf (COTS) products are being used much more frequently. These products change rapidly and require special maintenance. The maintenance of the component parts is usually described in the literature provided with the COTS product; this individual literature can help the Systems Engineer create Preventative Maintenance requirements for the whole system. Obviously this job is made difficult by understanding how the individual jobs come together to make the whole, but therein lies the job of the Systems Engineer! Further, a plan is needed to on how to implement system upgrades, future requirements, etc. It should be obvious that planning for the maintenance of the system earlier rather than later benefits everyone and everything involved.

8.1 RELIABILITY CENTERED MAINTENANCE

8.1.1 BACKGROUND ON RELIABILITY CENTERED MAINTENANCE

Reliability Centered Maintenance (RCM) is a process for achieving equipment reliability by identifying the Preventative Maintenance tasks that are needed to prevent or mitigate equipment failure. The airline industry first defined the RCM process in the early 1960's. It gained acceptance due to its effectiveness and the fact that it provided a more economical solution than other fielded solutions. It is now the principal process for commercial and military aircraft and many nuclear power plants.

8.1.2 PRINCIPLES OF RELIABILITY CENTERED MAINTENANCE

"RCM is based on the realization that equipment reliability cannot be improved by putting a limit on operating age. It also recognizes that equipment overhauls are likely to cause more problems than they prevent." [Harrington, 2000]

When the reliability of a system or piece part is called into question, the model of reliability that most readily comes to people's minds is the "bathtub" model as is depicted in Figure 26.

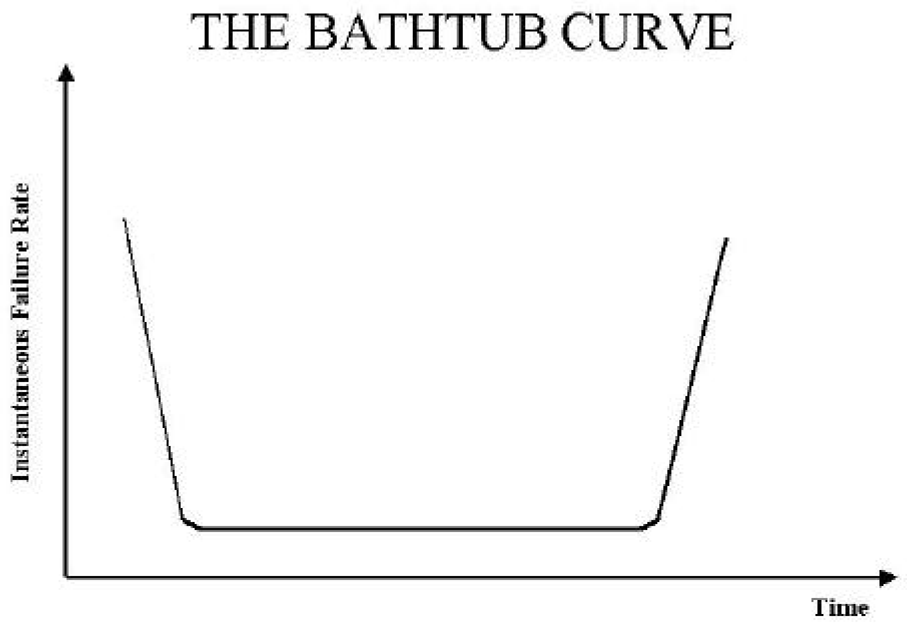


Figure 26 - The Bathtub Curve of Reliability

New equipment, or new components of equipment may suffer from “infant mortality” i.e. a small percentage of new equipment can be expected to fail in the first few days or weeks of service due to imperfections created in the manufacturing process or due to imperfections in the quality of the materials of construction. After this initial “wearing-in” time, the reliability of the product or components the failure rate can be expected to be relatively low and constant for the majority of the life of the product or part. As the product (or part) gets older, the rate of failure may be expected to rise dramatically in a short period of time.

Most Preventative Maintenance (PM) programs are designed around the bathtub curve. Most of a company’s time and effort is given to estimating when the “wearout” stage of the equipment is; and then replacing the part immediately prior. It is true that there could be a period of infant mortality with the new components, however, that is deemed less intrusive than a breakdown of the component.

In the 1960’s the Boeing Aircraft Company was promoting its new, large 747 airplane. United Airlines and other companies were very interested in having an aircraft in their fleet of this size but were very concerned about the PM costs to the company. The cost associated with the PM program for the plane gave United's management cause to try and improve the bathtub PM curve. Figure 27 [Harrington, 2000] shows how the results that they found completely changed their way of designing a PM program. "They discovered that only 11% of nonstructural aircraft equipment could benefit from putting a limit on operating age. A whopping 89% could not benefit from a limit on operating age. The study showed that the bathtub curve applied to only 4% of aircraft equipment." [Harrington, 2000] Realization of this fact caused a complete overhaul of the PM programs and the Reliability Centered Maintenance (RCM) programs were born.

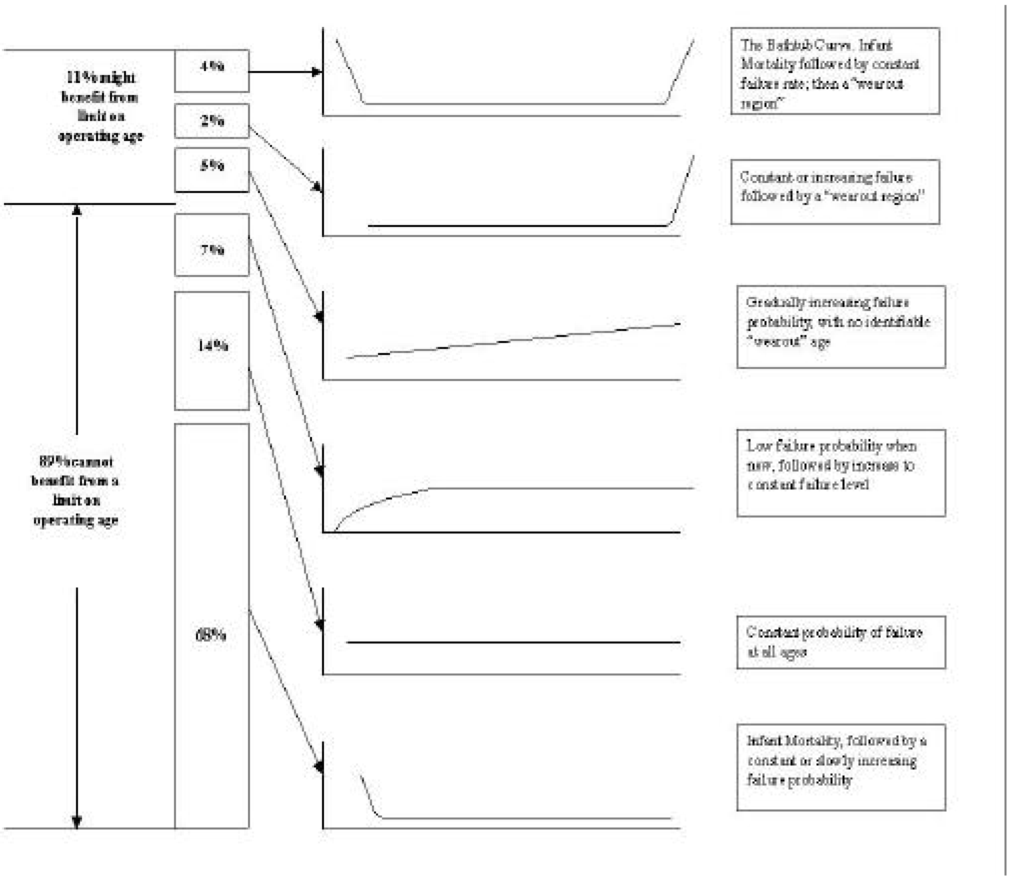

11% might benefit from limit on operating age
89% cannot benefit from a limit on operating age
4%
2%
5%
7%
14%
68%
The Bathtub Curve. Infant Mortality followed by constant failure rate; then a "wear out region"
Constant or increasing failure followed by a "wear out region"
Gradually increasing failure probability, with no identifiable "wear out" age
Low failure probability when new, followed by increase to constant failure level
Constant probability of failure at all ages
Infant Mortality, followed by a constant or slowly increasing failure probability

Figure 27 - Age Reliability Patterns

Within the RCM process, there is a clear distinction between the four categories of PM tasks; they are:

1. Time-Directed (TD) tasks aimed directly at failure prevention or retardation.
2. Condition-Directed (CD) tasks aimed at detecting the onset of a failure or failure symptoms.

3. Failure-Finding (FF) tasks aimed at discovering a hidden failure before an operational demand
4. Run-to-Failure (RTF) a deliberate decision to run to failure because the others are not possible or the economics are less favorable.

The RCM program may be divided into six steps. These steps are modified from Robert Harrington's RCM Program. [Harrington, 2000] These steps represent a "classic" RCM Program:

1. Sub-System Description
2. Components by Sub-System
3. Failure Mode by Component
4. Failure Cause and Proposed Tasks
5. RTF Review
6. PM Tasks

These steps can be described as follows:

Step 1— The purpose of Step 1 is to fully describe the sub-system that has been chosen for the analysis. Having done that, the functions performed by the sub-system are listed. The obvious problem here for a system of systems is what is considered a sub-system; i.e. an entire sub-system, a process within the sub-system, a component, etc. The correct answer is all of them! Of course, dependent upon the complexity of the system (of systems) this can lead to several layers of a RCM plan. If several layers are necessary, then we must observe all of the layers!

Step 2— In step 2, the components that make up the sub-system are listed. A component is defined as a division of the sub-system, which has an identifiable function in the sub-system. Some examples of components would be a Motor Driven Conveyer, a Motor Driven Hydraulic System, or a Punch Press. Do not make the mistake of assuming that this step is the same as step one! It is important to list the components of *each layer*. Only then may we be sure that all of the appropriate PMs for the various parts are taken into consideration.

Step 3— The purpose of Step 3 is to identify the failure mode associated with each component and the effect of the failure on the function of the sub-system. A failure mode answers the question, "How Can the Component Fail?" Step 3 is in essence a Failure Mode and Effects Analysis (FMEA) analysis. It identifies possible failure modes and the effect of the failure on the functions provided by the subsystem. The effects of each failure are determined by passing each failure mode through a Logic Tree Analysis (Figure 28). By answering the three questions in the logic tree the failure mode is categorized as a hidden failure, a safety problem, a minor or insignificant economic problem, or a line outage problem. As is becoming a recurring theme, this must be done for each level of the system that requires a RCM approach.

Step 4— In Step 4, the Failure Causes (why the failure occurred) are listed along with the PM tasks that will prevent or mitigate the failures. The RCM team members apply their knowledge and experience to the choice of the most appropriate tasks. The frequency for performing the task and the craft designated to perform the task are listed. This process is best understood with a standard format form. Most companies have a designated "Problem Report (PR) or Discrepancy Report (DR) that issues such as these may be categorized into and then learned from at a later time.

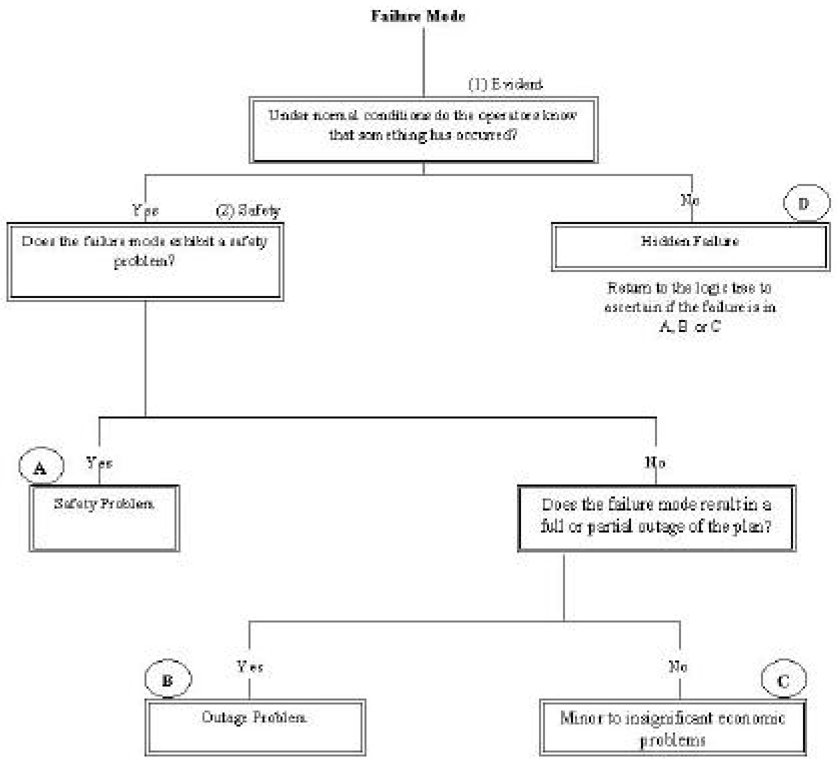

Failure Mode
(1) Evident
Under normal conditions do the operators know that something has occurred?
Yes
(2) Safety
No
D
Does the failure mode exhibit a safety problem?
Hidden Failure
Return to the logic tree to ascertain if the failure is in A, B or C
A
Yes
No
Safety Problem
Does the failure mode result in a full or partial outage of the plan?
B
Yes
No
C
Outage Problem
Minor to insignificant economic problems

Figure 28 - Logic Analysis (Decision) Tree

Step 5— This step gives the team a second chance to review their Run-to-Failure (RTF) decisions and change them if they feel so inclined. This is a so-called "sanity check." Only RTF decisions are considered in this step. If the team would like, a RTF decision can be rescinded and a PM task designated for the failure model in question.

Step 6— Step 6 is the output of the program. Its purpose is to sort the Step 4 PM tasks into desired formats for entry into a Configuration Management Maintenance System (CMMS) program. Thus, separate lists are prepared showing tasks by craft by uptime or downtime, by frequency, or by equipment number. The DR/PR-like process described in step 4 may accomplish this. A full explanation of how the PM task is to be performed is given in the step. The PM tasks are next transferred into PM work orders and implemented through an appropriate manual PM program or a CMMS program. Unfortunately, finding the appropriate CMMS software may be a problem. "Failure rates in implementing CMMS packages is high, with some companies installing two or three different CMMS packages before they find one they are comfortable with. These failures can occur for many different reasons. The most predominant of these are not developing and/or adhering to a timeline for installation and implementation that takes into account the company's goals and objectives, and the reluctance to implement the changes necessary that come with a good CMMS." [http://www.amsinc.com/cmt/OVERVIEW.HTM, 2001]

8.2 THE "BACKFIT" RCM PROCESS

The Center for Maintenance Technologies have discussed the benefits and disadvantages of utilizing a "classic" RCM program vs. a more tailored, "shorthand" program that they refer to as a "Backfit" program. The following has been taken from their website, http://www.amsinc.com/cmt/OVERVIEW.HTM:

The "Backfit" methodology is a "shorthand" version of the "Classic" RCM analysis methodology specifically designed for use where:

- existing PMS is satisfactory but probably excessive,
- considerable operating experience has been achieved with the item being analyzed,
- analysts applying the methodology have expert knowledge of the item being analyzed, and
- the activity responsible for the quality of results selects this methodology for use.

This process was originally outlined in the U.S. Navy's Reliability-Centered Maintenance Handbook, but can be applied to any equipment.

This form of RCM validates three fundamental assumptions made during development of the original PMS task requirements:

1.A specific dominant failure mode must be prevented to assure system or equipment operational reliability. This dominant failure mode corresponds to the output of the "Classic" methodology's Failure Modes and Effects Analysis (FMEA).

2.The task that is specified actually restores or maintains inherent reliability. This corresponds to the "applicability" rules in the "Classic" methodology's RCM Decision Logic.

3.The task that is specified costs less than the cost of the failure it is designed to prevent. This corresponds to the "effectiveness" rules in the "Classic" methodology's RCM Decision Logic.

Clearly, the "Classic" and "Backfit" methodologies are linked. One is simply a shorthand validation of the other, which applies when certain pre-conditions are met.

The actual "Backfit" process itself comprises six basic steps. The steps in this process are:

Step 1: Determine the specific failure mode the preventive maintenance task is designed to prevent. This would normally be a dominant failure mode identified in the original Failure Modes and Effects Analysis (FMEA). The task description should allow the analyst to determine what condition the task is designed to prevent.

Step 2: Determine whether this specific failure mode is likely to occur in service as a result of age degradation. Unlikely failures do not require preventive maintenance tasks.

Step 3: Classify the task as a time-directed, condition-directed, failure-finding, servicing or lubrication task. This is important; there are different rules for task applicability that are linked to specific types of tasks.

Step 4: Apply the rules for task applicability for the type of task specified. These rules are based on failure characteristics. If the applicability rules established for that type of task are not met, the task is not applicable. If they are satisfied, proceed to Step 5.

Step 5: Apply the rules for task effectiveness. These rules are based on failure consequences. If the effectiveness rules established for that type of task are not met, the task is not effective. If they are satisfied, proceed to Step 6.

Step 6: Examine whether further improvement can be made to the task as it is now specified. Even though the task may currently satisfy the rules for effectiveness, further improvement may still be possible.

The RCM "Backfit" Process is a straightforward approach to maintenance task validation. Successful application of this process requires sound operational experience with the hardware involved, as well as good maintenance history data for review to ensure the analyst has the proper basis for making good judgments. The extent of operational experience, required to provide such a basis, must be determined on a case basis by each analyst.

Use of the RCM "Backfit" process provides many capabilities and benefits for evaluation/improvement of maintenance requirements. First and foremost, the RCM "Backfit" Process identifies maintenance tasks, which are not applicable or effective, for improvement or removal.

The RCM "Backfit" Process provides a sound basis for making recommended changes. The rules for applicability that pertain to time-directed and condition-directed tasks allow the analyst to determine whether that type of task applies. If it does not, the rationale for revising or deleting the task can be presented clearly. The requirement to demonstrate evidence of age degradation makes a powerful argument for extending the intervals of maintenance tasks. In many cases, this can be done through empirical age exploration. Using both elements of risk (probability as well as severity of failure) helps one to see more clearly what value each preventive task actually provides. The logic conveyed by the RCM "Backfit" Process permits organizations to dispel the use of folklore, mythology, and anecdotal experience in justifying maintenance requirements, and to refine or eliminate those requirements using explicit RCM rules.

Finally, the RCM "Backfit" Process provides results much more quickly and less expensively than the application of the classic RCM analysis process (which remains available for use). The "Backfit" process helps analysts analyze existent PMS requirements far more efficiently than they can using classic RCM. This process does not waste resources by re-creating the original basis for PMS tasks; it takes the tasks as they are and determines whether they still make sense. Where they don't make sense, a basis for change has been established. [http://www.amsinc.com/cmt/OVERVIEW.HTM, 2001]

8.3 CONCLUSIONS ABOUT SYSTEM MAINTENANCE

We have seen a couple of examples of a PM process applied to a system. Of course, within these processes are the PM processes that apply to the sub-systems and even the component levels. When dealing with a system of systems, every layer of the overall system must be dissected to assure that no part is overlooked with respect to its PM process.

When maintenance is discussed, we tend to think of the system getting older and steps that we may take to prevent the onset of age-related problems. However, maintenance of a system also requires the Systems Engineer to be forward thinking about 1) future upgrades, 2) new technology, 3) future requirements, etc. One of the single most motivating factors in a customer's desire to create a system of systems is the fact that equipment that currently exists gets folded in. These "legacy" systems will require upgrades to maintain their place in the technological community.

As discussed earlier, initial planning for maintenance will solve most issues whereas last minute thinking may result in overbearing cost. As part of the initial planning process, spares for the system should be considered. With technology developing so rapidly, initial planning for the spares is imperative; waiting could lead to a parts obsolescence issue that could lead to a system incompatibility. Further, initial planning allows maintenance to become a viable part of the business structure instead of a "knee-jerk" reactionary issue too late in the process. The Center for Maintenance Technologies created Figure 29 below to illustrate this fact.

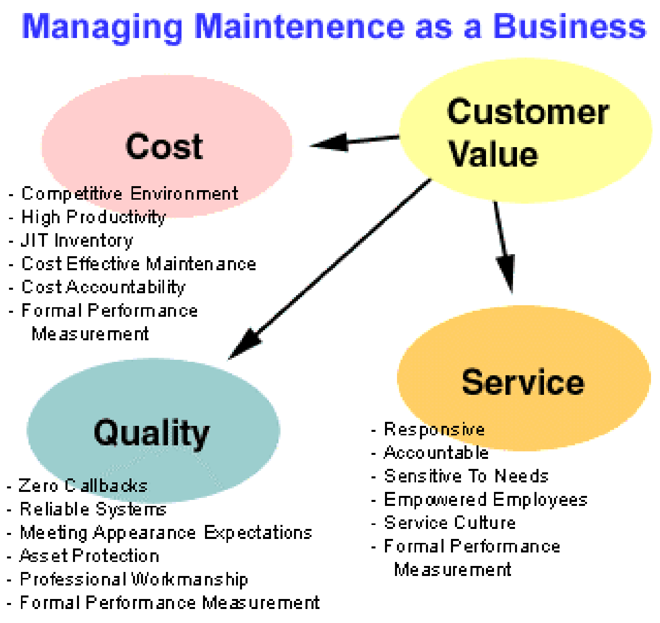

Managing Maintenence as a Business
Cost
- Competitive Environment
- High Productivity
- JIT Inventory
- Cost Effective Maintenance
- Cost Accountability
- Formal Performance Measurement
Customer Value
Service
- Responsive
- Accountable
- Sensitive To Needs
- Empowered Employees
- Service Culture
- Formal Performance Measurement
Quality
- Zero Callbacks
- Reliable Systems
- Meeting Appearance Expectations
- Asset Protection
- Professional Workmanship
- Formal Performance Measurement

Figure 29 - Managing Maintenance as a Business

CHAPTER IX
CONCLUSIONS AND RECOMMENDATIONS

The methods laid out in this book have been applied wholly to one system that the author was the LSE on. In keeping with the principles of this paper, the system was delivered one month ahead of schedule, at a greater profit than expected and one very happy customer. Secondly, the author applied these techniques to a program that he came in on about halfway through its life. By completion, the program was back on schedule and in good shape financially! These principles will always work well, even to dig a system out of a hole, however, they work best when applied from the beginning with exceptional planning! The individual portions of this paper have been applied to various programs as well with good results. From applying these principles, a list of priorities has been derived for the LSE as follows:

1. **Safety**
2. **Quality**
3. **Schedule**
4. **Cost**

As stated earlier, safety is not a legal issue, it is an ethical one. Quality is next on the list. Products delivered should represent the absolute best that the LSE has to offer.; anything less is not acceptable. Schedule and cost sometimes "flip-flop" depending on the company and the product line, but they always fall a very distant third and fourth place to the first two.

As a means of a quick summary of the paper, the following are some exerpts of the conclusion paragraphs contained within each chapter.

When designing the system of systems, the LSE must keep in mind how complex the system will become. Obviously, there are varying degrees of complexity, however, how it is managed is of utmost importance. The complexities involved in engineering a system of systems are so great, that without a template for design and implementation, the chances of system success are limited. It is in the understanding of the relationships between the lower level systems that system integration may successfully occur. This is done by any number of methods, specific to this chapter is the Option Field Method. When the system's complexity has been managed and its relationships understood from its lowest level to its highest (through an option field) the design of the system will be complete.

The Systems Engineer has responsibilities that overlap with Program Management. These gray areas may change from program to program. The ability of the Systems Engineering and Program Management to embrace this overlap and turn it into an advantage will largely influence the success or failure of the program. Key planning meetings at the outset of the program is the best place and time to discuss these overlaps and agree on who has what responsibility.

In keeping with the theme of the paper, quality of the product is enhanced with early planning. Whether the early planning has to do with the testing, safety concerns, training or process development, clearly defined roles at the outset will prevent most serious quality assurance issues from coming up. One can complete the project with no final errors. The Ford Motor Company has a motto that "quality is job one." As engineers, our commitment to a quality product and the resulting output are a calling card that will always be associated with us. We must always strive to ensure that quality is at the top of our priority list. When maintenance is discussed, we tend to think of the system getting older and steps that we may take to prevent the onset of age-related problems. However, maintenance of a system also requires the Systems Engineer to be forward thinking about 1) future upgrades, 2) new technology, 3) future requirements, etc. One of the single most motivating factors in a customer's desire to create a system of systems is the fact that equipment that currently exists gets folded in. These "legacy" systems will require upgrades to maintain their place in the technological community.

Finally, if this paper leaves the reader with only one thought, it should be to plan early. Be specific in the plans as to who is responsible for what and when. If this occurs, the system will be a success!

REFERENCES

Crosby, Phillip. *Quality is Free: The Art of Making Quality Certain.* New American Library. New York, New York. 1980.

Dogru, A. "Domain Specific Analysis and Modeling", Middle East Technology University, Ankara, Turkey, and Engineering Department, Texas Tech University, Lubbock, Texas, (Briefing slides and Lecture Notes: January 18-20, 2001).

Electronic Industries Association. *EIA Interim Standard, Systems Engineering,* EIA/IS-632. Washington DC: EIA, 1994

Flynn, M. *Intelligence Must Drive Operations: How Intelligence Can Clear THE FOG OF WAR,* **Author:** LTC Michael T. Flynn, G2, 82d Airborne Division, US Army, http://call.army.mil/call/nftf/julaug00/flynn.htm. Editor's Note: This article was previously published in the *Military Intelligence Professional Bulletin*, January-March 2000.

Ginac, Frank P. *Customer Oriented Software Quality Assurance.* Prentice Hall. New York, New York. 1997

http://www.amsinc.com/cmt/OVERVIEW.HTM

http://www.VITec.com

Harrington, Robert. *Reliability Centered Maintenance Program.* Society for Maintenance and Reliability Professionals. Fall 2000.

Institute of Electrical and Electronics Engineers, Inc. *IEEE Std. 1220-1994, IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process.* New York: IEEE, 1995.

Kaake, W. "Scout Mission Planning Tool Applique' Design" Master's Report for Texas Tech

Kaake, W. & Whittington, B. "Domain: 'Intelligence'", Engineering Department, Texas Tech University, Lubbock, Texas; Date: FEB 2001. Classification: (U) Unclassified. (Analysis and Briefing).

Kauffman, Donald. *Project Management and Systems Engineering: Where the Professions Intersect-Generate Synergy Not Conflict.* White Paper. Presented at the 8th Annual International Symposium of the International Council on Systems Engineering. Vancouver, British Columbia, Canada. July 26-30, 1998.

Mandanis, Greg. *Software Project Management Kit for Dummies.* IDG Books. Foster City, CA. 2000.

Maxwell, T. "Generic Design Science", Engineering Department, Texas Tech University, Lubbock, Texas, (Briefing slides and Lecture Notes: March, 2001).

Microsoft Corporation. *Microsoft Project ®* . 1998.

Plsek, Paul E. *Creativity, Innovation and Quality* ASQ Quality Press. Milwaukee, Wisconsin 1997 pg. 186ff.

Project Management Institute. *A Guide to a Project Management Body of Knowledge*. Sylvia: PMI, 1996.

Ranky, P. *Concurrent / Simultaneous Engineering (Methods, Tools & Case Studies)*, Biddles Limited, The Book Manufacturers, Woodbridge Park, Guildford, Surrey, UK. © 1994

Ranky, P. *An Introduction to Total Quality Management & Control, and the ISO 9001 Quality Standard*, **Media:** CD ROM/Microsoft Word Document. CIMware Ltd. © 1997 CIMware Ltd. And CIMware UK&USA. http://www.cimwareukandusa.com

Ranky, P. *An Introduction to Concurrent / Simultaneous Engineering*, **Media:** CD ROM/Microsoft Word Document. CIMware Ltd. © 1997 CIMware Ltd. And CIMware UK&USA. http://www.cimwareukandusa.com

Rational Software Corporation â. Process Chart. 2000

Tanik, M. & Ertas, A *Interdisciplinary Design and Process Science: A Discourse on Scientific Method for the Integration Age*, *Journal of Integrated Design and Process Science*, September, Vol. 1 No. 1: pp. 76-94. 1997.

Vigilinx Corporation. *The Power of Intelligence*. Parsippany, NJ 07054. 2001

Warfield, J *A Science of Generic Design: Managing Complexity Through Systems Design*, Second Edition, Iowa State University Press, © 1994.

Warfield, John N. *A Philosophy of Design*, Presentation supplement, First World Conference on Integrated Design and Process Technology. Society for Design and Process Science. Austin, TX. December, 1995.

Warfield, John N. *Understanding Design Science and Its Implementation*. Presentation supplement. First World Conference on Integrated Design and Process Technology. Society for Design and Process Science. Austin, TX. December, 1995.

Warfield, John N and A. Roxana Cardenas. *A Handbook of Interactive Management*. Second Edition. Iowa State University Press. 1994.

Watts, Jerry G. and Mar, Brian W. *Important Skills and Knowledge to Include in Corporate Systems Engineering Training Programs*. White Paper Study with Boeing Defense and Space Group. 1996.

Wheeler, Ken. *Defense Information Infrastructure Common Operating Environment Overview*. White Paper. Defense Information Systems Agency. 2001.

Whittington, B. "Electronic Light Table" Master's Report for Texas Tech

APPENDIX A

NOMENCLATURE

ADRG	ARC Digitized Raster Graphics
AKA	Also Known As
ATE	Automatic Terrain Extraction
BBT	Black-Box Testing
BMP	Bitmap
BPP	Baseline Project Plan
BT	Boundary Testing
CCB	Change Control Board
CD	Condition Directed Tasks
CFE	Customer Furnished Equipment
CFI	Customer Furnished Information
CM	Configuration Management
CMDB	Configuration Management Data Base
CMMS	Configuration Management Maintenance System
CMP	Change Management Plan
CMS	Change Management System
CMST	Configuration Management Standards
CT	Concurrency Testing
CONOPS	Concept Of Operations
COTS	Commercial Off The Shelf
DDC	Document and Data Control
DDS	Detailed Design Specifications
DIICOE	Defense Information Infrastructure Common Operating Environment
DPPDB	Digital Point Positioning Data Base
DR	Discrepancy Report
DTM	Digital Terrain Model
EIA	Electronic Industries Alliance
ELT	Electronic Light Table
FAT	Formal Acceptance Test
FF	Failure Finding Tasks

FMEA	Failure Mode and Effects Analysis
FS	Functional Specifications
GBT	Glass-Box Testing
GFE	Government Furnished Equipment
GFI	Government Furnished Information
GIF	Graphical Interchange Format
GOTS	Government Off The Shelf
HP	Hewlett Packard
HTML	Hyper Text Mark-up Language
ICT	Integration and Compatibility Testing
IEEE	Institute of Electrical and Electronics Engineers
IPB	Intelligence Preparation of the Battlefield
IR	Infrared
ISO	International Organization for Standards
IT&V	Integration, Test and Verification
JIT	Just In Time
JPEG	Joint Photographic Experts Group
LANDSAT	Land Satellite
LSE	Lead Systems Engineer
LTA	Logic Tree Analysis
MIG	Multi-Image Geopositioning
MIL-STD	Military Standard
MP	Mission Profile
NIMA	National Imagery and Mapping Agency
NITF	National Image Transmission Format
NSI	Network Security Intelligence
NTM	National Transmission Message
PDA	Personal Digital Assistant i.e. Palm Pilot, etc.
PM	Preventative Maintenance
PMBOK	Program Management Body of Knowledge
PMS	Preventative Maintenance System
PR	Problem Report
PSD	Project Scope Document
PT	Performance Testing

QA	Quality Assurance
QE	Quality Engineer
QM	Quality Management
RCM	Reliability Centered Maintenance
RD	Requirements Document
RFP	Request For Proposal
ROI	Region Of Interest
R&S	Reconnaissance and Surveillance
RT	Recovery Testing
RTF	Run To Failure
SAR	Synthetic Aperture Radar
SAT	System Acceptance Test
SBS	System Breakdown Structure
SE	Systems Engineer
SEGL	Systems Engineering Group Leads
SOS	Statement Of Scope
SPOT	Satellite Pour L'Observation de la Terre
SQA	Software Quality Assurance
ST	Security Testing
STT	Stress Testing
SYERS	Senior Year Electronic Reconnaissance System
TD	Time Directed Tasks
TERCAT	Terrain Categorization
TK	Technical Knowledgebase
TIFF	Tag(ged) Image File Format
TPM	Technical Performance Measurement
TRD	Technical Requirements Document
TSQ	Triply-Structured-Quad
UTM	Universal Transverse Mercator
WBS	Work Breakdown Structure
WBT	White Box Testing
XWD	Window System window dumping utility (XWindows)