A Practical Approach for Knowledge System Incorporation into a Systems Integration and Test Process

By
Susan L. Armitage

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

_____

Dr. J. Borelli

_____

Dr. A. Ertas

_____

Dr. T.T. Maxwell

_____

Dr. M.M. Tanik

October 12, 2002

ACKNOWLEDGEMENTS

I would first like to thank Texas Tech University professors for their wisdom and insights with special thanks to Dr. Atila Ertas for developing this program and providing guidance throughout the year. Raytheon Learning Institute personnel at Raytheon Spring Creek and Garland, Texas and especially at Tucson, Arizona facilitated access to the program. Raytheon study groups were a welcomed and integral part of the program.

Mentors and colleagues that I have had the privilege of working with at Kaman Aerospace contributed to my development as an Integration and Test Engineer. At Raytheon Missile Systems my supervisors have aided my on-the-job continuing education on large pre-production systems, offered ideas for my Project, while providing unfaltering support to me toward completion of this Masters Program.

Last, but certainly not least, a huge thank you to my family and close friends for their many words of encouragement and escapes, especially to my mother Nancy Kephart for her glowing pride with each triumph. Finally, I would like to particularly recognize my partner, Dr. E.J. Langseth, who proof read, supported me and cheered me on through many long nights and weekends during my Masters work, and who has, unknowingly, taught me many lessons over the years about Integration and the importance of changing viewpoints to find solutions to seemingly impossible problems.

TABLE OF CONTENTS

DISCLAIMER

This report presents a generic methodology for performing the task of Systems Integration and Test engineering. The opinions and observations expressed in this paper are strictly those of the author and are not necessarily those of Raytheon, Texas Tech University, nor any U.S. Government Agency.

ABSTRACT

Engineering and Manufacturing Development (E&MD) programs are the cutting edge technology products of the defense industry whose creation is the experiment before beginning multiple system production. E&MD program output products are systems built from components that are developed, tested and integrated expectantly following standard processes; the final program phase is typically orchestrated by a Systems Integration and Test (I&T) team. Throughout program development, lessons will be learned in the form of errors and mistakes as well as accomplishments, all of which form the basis of program and engineering knowledge that needs to be captured, shared and available for reuse. However, the inherent nature of E&MD programs is chaotic and lends itself to the potential for company, department and program Knowledge Loss during the rapid and constantly changing development process, if the knowledge is not properly managed.

An organization's clearly defined processes are the key to minimizing chaos, provide repeatability and should viewed as vital to the capability of continuing to produce high-quality results. While the idea of process improvement can and should be applied to the entire project team, this report focuses on the Systems I&T Process. The improvement of the Systems I&T Process for an E&MD organization is initiated with the presentation of a model from program conception to completion. A Knowledge System capturing Lessons Learned is introduced and incorporated into said process. To demonstrate the benefits of the improved Systems I&T Process, an analysis is performed to substantiate interaction of the processes main elements and assess process reliability.

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Acceptance Test – Final suite of tests and inspections before accepting an item from a vendor or functional group.

ANOVA – Analysis of Variance, a procedure for resolving the total variance of a set component variances. [Webster's, 1989]

ASME – American Society of Mechanical Engineers

Axiomatic Design – The use of domain mapping and defined Axiom's satisfaction in the process of engineering design developed by Suh [1990].

BIT – Built In Test

C&R – Continuity & Resistance test to ensure that an electrical component is mechanically correct.

Chaotic System – A system with that has complete interdependence among all components within the system.

CMM© - Capability Maturity Model developed by Software Engineering Institute.

CN – Customer Need as an element of the Customer Domain in the Axiomatic Design framework.

Component – One of the lowest levels of elements of a System.

Design – A description of how a system and its components are to be built.

DP – Design Parameters are elements of the Physical Domain in the Axiomatic Design framework.

E&MD – Engineering and Manufacturing Development, a cutting edge technology program to build a unique system.

EC – Expected Cost (of Failure Analysis)

FR – Functional Requirements are elements in the Functional Domain in the Axiomatic framework.

Functional Reliability – "…the likelihood of successfully providing necessary functions that a system or component is intended to deliver" [Trewn & Yang, 2000]

Functional Test – A test or set of test designed to test the functionality of the unit under test or system.

I&T – Integration and Test

IEEE – International Electrical and Electronics Engineers

IMP – Integrated Master Plan

IMS – Integrated Master Schedule

INCOSE – International Council on Systems Engineering

Integration – Bringing together the units of a system through proper interfacing and ensuring that the system operates as designed and built.

ITLog – Integration and Test Log tool developed by Raytheon to track daily events.

KGD – Knowledge Gained Database, a shared repository of recorded E&MD program information and Lessons Learned.

Knowledge Loss – a product of neglecting to track pertinent information.

Lessons Learned – data and information discovered during E&MD program development.

LOC – Level of Confidence

MLH – Multi Level Hierarchical Model [Hubka & Eder, 1998] developed to describe a technical system.

OA – Orthogonal Array used in the ANOVA analysis

Process – set of practices performed to achieve a given purpose [Phillips, 2002] which may include tools, methods, materials, and/or people.

PV – Process Variables in the Process Domain of the Axiomatic Design framework.

Qualification Testing – A test or set of tests to verify functionality of a component or system in the operational environmental conditions.

RFP – Request for Proposal

RMS – Raytheon Missile Systems

SEI – Software Engineering Institute

System Reliability – Refers to the reliability of the system to function upon delivery.

TAFT – Test, Analyze, Fix, Test. The classic definition of I&T tasks.

UUT – Unit Under Test

Validation Testing – Testing that ensures that the system operates as designed.

Verification Testing – Testing that ensures that the system functionally performs as required.

CHAPTER I

INTRODUCTION

Knowledge is a complex concept in the world of technology and emergent discoveries, commonly defined as “acquaintance of familiarity gained by sight, experience, or report” [Webster’s, 1989]. The Knowledge Management Forum website has many contributors that have added their perspective including Denham Grey who asserts that “For knowledge to be of value it must be focused, current, tested and shared”. Some knowledge of the task at hand is necessary to get started, but especially in pioneering technology there will be knowledge discoveries during the evolution. Knowledge comes in the form of triumphs as well as failures and miscues and is collected in reports, test results, charts, and notes which can be termed components of a system. The Knowledge System is the core of what is widely acknowledged as Knowledge-based Management. In this report, the management of the Knowledge System is not included beyond a high level discussion and introduction of tool prototypes, only the internal system is broadly addressed.

This report addresses process improvement through incorporating a Knowledge System in the form of Lessons Learned into an organization’s Systems level Integration and Test (I&T) process in order to evolve the organization from one essentially process oriented into one “knowledge based”. The organizations’ processes don’t disappear, but are improved upon using knowledge gained at each step of the process; thereby improving the process itself. How Systems I&T knowledge is recognized, retained and re-used is the focus in this paper.

The process of incorporating a Knowledge System to facilitate process improvement can be related to any industry including medical, research, business, and financial. The focus of discussion will be the defense industry and cutting edge technical programs known as Engineering and Manufacturing Development (E&MD). The E&MD programs are a necessary step before full rate production commences, sometimes even before a formal customer request for the technology is issued.

1.1 The Focus System – E&MD

Chaos: state of utter confusion or disorder. [Webster's, 1989]

E&MD programs are typically highly chaotic. Unique E&MD development programs involve and require changing and constantly maturing requirements, quick design and re-design, rapid prototyping, with high customer visibility. E&MD programs are typically directed to solve new problems or enhance older legacy systems, usually employing a combination of old and new technology. The program development style both attracts and requires non-conventional methods and personnel since the ideas under development are almost certainly revolutionary. E&MD programs uncover surprises as systems are built one-at-a-time, whereas production programs produce many systems in parallel requiring that all the previous "bugs" are worked out so that the production line runs smoothly.

The new "cutting edge" programs generate interest in the system from all disciplines and management levels coupled with high expectations from the customer. Since there is no specific history for the new E&MD system, semi-unrealistic schedules get created and promoted, typically biased toward a production mentality creating high-stress environment. The pressure to maintain schedule promotes the temptation to short cut standard processes

thinking it will save time, but in fact short cuts usually lead to costly overruns and schedule slips as errors are repeated and work must be redone.

1.2 The Problem – Knowledge Loss

The fast paced, constantly changing nature of E&MD program characteristics allows for potential Knowledge Loss. Knowledge Loss might occur both intra-program and/or inter-program, within a company or department or with many categories where loss may derail a program or organization.

Preceding every step of E&MD system development are difficulties in convincing engineers to adhere to prescribed processes, and then ensuring that they have conformed to the process. Within any program, several generations of hardware and software units are created and re-engineered as problems are uncovered and resolved. The units are combined or integrated into sub-assemblies and the re-engineering cycle begins again at a new level of development. This cycle is followed for each stage of development until the final system is completed. The first product of an E&MD program is more of a science experiment, but a very important one as it sets the stage for the other systems to follow. E&MD programs create systems on a one-by-one basis, and there are many pressures and stresses on the engineering team during the Integration and Test phase of the system. That is, when more time is needed at the front end of the program, the back end of the program will suffer with even tighter schedules. The pressure of completing testing on-schedule can adversely affect engineers in the way the job is completed. If shortcuts are taken to preserve schedule, the process is compromised. Proper reporting and tracking may not be as complete as it should, which then contributes to Knowledge Loss. At each development level during the E&MD

program, there are opportunities for Knowledge Loss as problems are found and dispositioned. If problems are perceived to be of a smaller scale during the heat of Integration and Test, the engineers may not deem them important enough for tracking.

Rapid prototyping during Integration can also contribute to Knowledge Loss: if engineers charge ahead solving one problem after another, and get wrapped up in problem solving without tracking, the path that the system followed will be a mystery and can lead to additional Knowledge Loss compounded by unnecessary re-work. The program will suffer because there may be minimal repeatability with the next article. This type of loss can also carry over to other programs using similar technology.

During development, some programs elect to use off-the-shelf units for some system components. At the time the unit was first developed, it may have been on the leading edge of technology and have all the latest versions of internal components. Without proper tracking of the more technologically challenging components, updates and revisions will be ignored, lost and forgotten. The unit may go on to the next program, creating what would have been a preventable problem.

All forms of Knowledge Loss adversely affect the next step in the program. There is a high potential for wasted time re-discovering problems that were previously resolved. The next team in the program process will be burdened without the first hand knowledge of the previous team. The problem becomes magnified when transitioning from an E&MD program to a production program of the same system. Similar problems may arise, but without a tracking system, the production team will be confronted with identical problems.

1.3 Scope

Many engineering disciplines are involved in the development of an E&MD including Software, Quality, Safety, Mechanical, Electrical and Integration and Test. Systems I&T is the science of combining multiple units through well designed interfaces into a working system, and then testing the final product to prove compliance with given requirements. Testing is completed on several levels including unit, sub-assembly, system validation and system verification. Unit and Sub-Assembly testing usually consists of low-level electrical and mechanical checkouts. System Validation Testing involves functionality and operation testing. System Verification Testing is the final test stage which determines if the validated system performs its required task as per the customer requirements.

The Systems I&T Process for E&MD programs begins at program concept development and continues until the system's final product or "unit" is delivered to the customer. The term "customer" may assume many meanings including: field test, purchaser, production or other engineering organization. Post delivery may or may not involve further I&T support. The following report addresses the Systems I&T Process from program start-up to customer delivery only, no support beyond that is considered. Benefits of the improved process are presented through analysis.

1.4 Solution

During traditional education and training, most development engineers follow a version of the Standard Waterfall Process depicted in Figure 1.

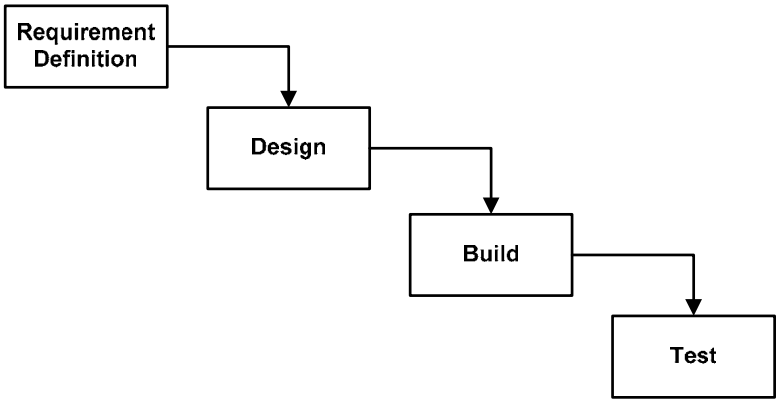


Figure 1 - Standard Waterfall Development Process

In practice this is standard process, Test is more of an after thought and is not in procedural focus until reaching the final step. In reality the process needs to be re-defined, and extended to include Integration in the test activities with appropriate inputs and outputs to the steps in the new process. Improvement of the Integration and Test process is then realized through application of and interaction with a Knowledge System database.

Useable, intuitive tools are needed to assist the engineer in Knowledge tracking in order to create a Knowledge System database. Separate tools can be part of the Knowledge Gained Database: a daily tracking tool to hold a repository of daily activities for all programs, and a second tool to handle more complex problem discovery and resolution.

Finally, the Systems I&T Process must be recognized as a benefit and become part of the engineers' culture. Through mathematical analysis, engineers and managers can recognize the benefits, and then accept, embrace and champion the process. The Knowledge based process must become the way to perform necessary tasks, in order to become automatic for the engineer to employ on a daily basis.

CHAPTER II

BACKGROUND INFORMATION

2.1 Literature Review

There are two main foci which are combined for discussion in this report: (1) Systems Integration and Test, and (2) Lessons Learned. Interweaving these two topics, as will be discussed herein, provides a practical framework for improved performance of Integration and Test with the incorporated Lessons Learned. Countless examples of both individual topics by numerous engineers and managers from many fields of technology may be found by performing a simple internet search which provides an extensive list of specific and related issues; however, the search for combined topics is practically non-existent. Most published reports involving Lessons Learned agree on its reuse, but fall short of providing a realistic and practical method of incorporation. The most relevant articles and presentations are discussed below.

A paper published on a DARPA website for a Synthetic Aperture Radar program [1997] includes a detailed discussion of the rapid prototyping methods used to develop the system and the importance of Process Improvement to complete the program. Lessons Learned as presented in a narrative in each of the main chapters covering System, Architecture, Detailed Design, and Integration and Test focused on the discovered issues and problem resolutions.

Sage and Lynch [1998], in “Systems Integration and Architecting: An Overview of Principles, Practices, and Perspectives”, take a broad look at the Systems Integration process including sections covering Standards, Analysis, Interfaces, and a lengthy section on CMMI.

Systems Integration is introduced as part of the larger Systems Engineering process which encompasses all aspects of System Development, which provides another perspective on the Integration process with references on Integration history. The authors illustrate, delineate and describe an industry integration approach, and will be discussed as one example of an Integration process.

An Integration and Test plan prepared by Aslam [1998] for the Moderate Resolution Imaging Spectroradiometer (MODIS) Emergency Back-Up System is another reference for a method of performing System Integration as it follows a sound approach to system development, Integration and test for the project.

Sheard, Lykins, and Armstrong [1999] published an article for the Software Productivity Consortium titled "Overcoming Barriers to Systems Engineering Process Improvement" discussing the latest trend in process improvement, Capability Maturity Models (CMM), and its importance to the Systems Engineering Process. The how-to on starting the process of improving processes is covered, including some of the associated problems such as lack of resources, lack of management support, and lack of interest.

An article published in the Evans Engineering website [1989] discussed results of a general survey on Lessons Learned from a Space Engineering flight hardware and software development project, was designed to provide the starting point for a much broader database of information soliciting additional information from the readers. The article consists of a generic checklist of repeated problems during the development phase of the program as well as substandard material, noting other deficiencies.

Dolores Wallace [2000] published an article on the Data and Analysis Center for Software website, maintained by the Department of Defense, entitled "Using Failure History to Improve Reliability in Information Technology". Two case studies in the use of "history data" are discussed therein.

Sheinman [1998] of the Mechanical Engineering Branch of a NASA agency published a paper titled "Lessons Learned, X-Ray Timing Explorer" where an overall discussion and checklist of Lessons Learned during the development of the project are listed. The information is categorized for Design, Fabrication, Management, and I&T.

In Dr Jane Marshall's [1999] presentation "Reliability Enhancement Methodology and Management" for TRW, addressed the creation of a methodology and model for integration of a Knowledge Base and Expert System with reliability prediction. Of particular interest is her Introduction on expert knowledge with inclusion of the types of information that should be added to any knowledge base.

Salomon [2002] presented a slide presentation to an Army organization about a proposed Knowledge Center, addressing questions to consider regarding knowledge mapping to a database and asking pertinent questions related to knowledge preservation.

One of the more pertinent articles discussing an applied use of a Lessons Learned database entitled "A Lesson in Lessons Learned" by Dennis Lee [2002] written for The NASA Academy of Program and Project Leadership. Included are observations on time and cost saving aspects of using the database as well as the usefulness of information retrieved from the system.

2.2 Background

Having worked seventeen years in engineering from development through Integration and Test, this author has experienced and observed the best and worst practices in the field of I&T. Whether within large or small companies or programs, the first challenge facing such a grounded development engineer is how to interpret what integration entails, as it is an acquired on-the-job skill. Integration is not an exact science, and as a result, part of that on-the-job lesson an engineer quickly learns, is the necessity of tracking all steps taken during Integration whether resulting in success or failure. This accumulation of successes and failures becomes the "Lessons Learned" concept discussed later in this report.

Many organizations and categories of engineering disciplines have discussed the importance of Lessons Learned. Typically, at the conclusion of a program or task, an "out-briefing" or concluding review will be conducted and, depending on the technical level and audience, either Program Management or Technical Leads will include a "Lessons Learned" topic in their presentation. In the best case scenario, the lessons will be captured during the ups and downs of the daily efforts of the involved engineers as well as include the major program efforts during ground-breaking discovery and the expected missteps along the way. But in actuality, even the presentation audience, while agreeing with the idea of "Learned Lessons", will not transfer those lessons into subsequent phases or programs. This typical shortfall results from the absence of a follow-on discussion of how to incorporate Lessons Learned, because the team has already moved on to the next challenge and has little patience for recording such experiences into a database -- assuming a database exists. The absence of review may also be driven by program conclusion budget issues.

CHAPTER III

SYSTEMS I&T MODEL

3.1 Introduction

The solution to the Systems I&T Knowledge Loss problem begins with characterization of a model including definitions of three key elements of the issue: Knowledge, Processes, and System Integration and Test planning. The Knowledge Loss problem is examined and the consequences that Knowledge Loss can have on any organization are discussed in section 3.2. Process introduction then follows with a high level discussion of Systems Integration and planning aspects. A broader more in-depth study follows of the merging of the three main topics through presentation of a proposed pathway for risk reduction of the Systems I&T Process.

Beginning with the System Requirements phase, proceeding through Integration, Test and Failure Analysis phases, Integration process flows will be presented and clarified. The broader will be dissected, explained, and discussion focused on Integration and Test's daily activities. Improvement of the Systems I&T process will then be presented with the introduction of a Knowledge "gained" system as a necessary part of the engineers' culture. This set of process flows is presented as one possible method and toolset for planning and performing Systems Integration.

3.2 Knowledge Loss Realities

Knowledge Loss within the engineering world and, in particular the area of Integration and Test, is an expensive problem. Starting at the program level and continuing through to product delivery, Knowledge Loss may be avoided by tracking and capturing

Lessons Learned. Ideally, managers and engineers catch issues at the design phase before extensive system build or Integration has begun.

Knowledge Loss will be even more costly if problems are not uncovered until the final steps of Integration, causing a restart at the design phase and redundant efforts and wasted dollars. For example, unnecessary costs resulting from Knowledge Loss would be months of lost time spent on a hardware problem. As observed by this author, an electronics unit was not functioning properly, so a "tiger team" was appointed to problem solve. Following several months and many needless overtime hours spent in a failed attempt to resolve the issue, a Senior engineer who happened to be sitting in on a status meeting, suggested a particular engineer in another part of the company who might be able to help troubleshoot. Within minutes the engineer determined that the particular hardware problem was due to a processor upgrade that had not been published.

A second lost time example discussed by Dennis Lee [2002] in a Project Management article involved a failed vibration test on a NASA program. Although the program managed to recover from the failure, the system under test was severely damaged and final testing was significantly delayed. Upon searching a NASA database, Lee found several articles that, had the team been aware of, would have helped avoid the original failure saving time and budget.

Compounding Knowledge Loss is the failure to recognize Knowledge Gain. At the heart of engineering development is the dilemma of knowing when to document newly discovered knowledge while in the midst of technological discovery. The excitement of solving an issue after long days of calculations, re-configuration and re-testing may derail attention necessary to document the effort. Another missed opportunity of documenting

Knowledge Gained occurs when teams or individuals deeply engrossed in System Integration efforts fail to make note of an obvious solution. Standardized guidelines, tailored processes, external reviews and management participation are keys to capturing and documenting issues before they are abandoned or forgotten.

3.3 Process Definition

Processes are necessary for departments and companies to survive, to build on their ability to repeat successes, and gain customer confidence. Processes include steps, methods, techniques, procedures, routines, instructions, and practices used by an organization to provide standards and consistency. Processes may be developed and utilized by all company departments: Engineering, Financial, Factory, Documentation, Management and Staff. A company's set of standard processes should be easily available to and must be understandable by the employees. Training in process usage and their applicability is vital to the health of the company and empowers the employees.

Company internal engineering disciplines such as Systems, Mechanical, Electrical, and Software should each have their own standard processes that are shared within the organization to assist in providing a consistent and contiguous development flow. Engineering processes supply guidelines for engineering teams to effectively develop new technologies while providing engineers with creative latitude to tailor the processes as needed.

An important component of any process is periodic or task completion reviews. Process reviews may be attended by not only the team performing the work, but also by outside observers who will serve as the conscience for the group. Sub-processes created for

the task level provide a starting point for engineers to tackle an assignment, and may include a set of checklists that, when finished, signal the completion of the task. While providing visibility to other departments, engineering processes also provide tracking metrics for management.

The process must be managed -- written, tracked, improved upon and tailored to meet the needs of an organization. When the need is discovered, a new process should be documented, reviewed and integrated into the existing process collection. Deficient or obsolete processes are revised or deleted. At the company level, processes and sub-processes from all functional areas should be reviewed and merged when possible, to reduce redundancy and streamline the overall company process. Guidance for improving an organization's processes is provided by the Capability Maturity Module Integration (CMMI), a project sponsored by the U.S. Department of Defense and the National Defense Industrial Association.

3.4 Systems and Systems Integration Defined

A Request for Proposal (RFP), contractor bid, and customer contract award characterize the starting steps of an E&MD program, defining the system to be developed. In the defense world, the RFP is a published announcement of possible funding available for developing a product which contains high level customer needs. The contractor's bid responds to the customer's needs with major task and activity scope definitions and associated costs. Finally, contract award signals the official program start: the customer's needs are transformed into a collection of system specifications. Requirements are derived from the system specifications, and the system begins to take shape from the top-level view.

System requirements define “what” the system needs to do and are used by design teams to define “how” the system is going to operate. Through several iterations, the system design is broken down into the lowest level units or elements of the system. The system is then developed from the ground-up, utilizing the basic building blocks of interrelated elements: software, hardware, and electrical. Software elements include low-level burned-in firmware, system application or executable software, test software, and possible training or pre-planning system operation software. Hardware elements include mechanical parts, such as nuts and bolts, brackets, component housings, and gearing -- all of which are combined and built into sub-structures of the final system structure. Electrical elements include wiring, capacitors, resistors, printed circuit boards, processors, and power subsystems that are built into sub-assemblies fitting into the hardware structures. The electrical sub-assemblies provide the platform for software, via interface cabling, to execute, signal and control sub-assemblies and lower level elements, as required. Other hardware may include product packaging and support equipment for system delivery, and customer training.

Systems Integration and Test is the process of developing interfaces and bringing all elements together to form an operational system. To insure proper autonomous operation, elements of the system are built and tested at the individual unit level, and then combined into subsystems that are functionally tested. Ultimately the subsystems are integrated creating the final system to be tested for function and performance.

Figure 2 shows the classic high-level abstraction of the Systems Integration process as detailed in “Systems Integration and Architecting” [Sage and Lynch, 1998, pg 184].

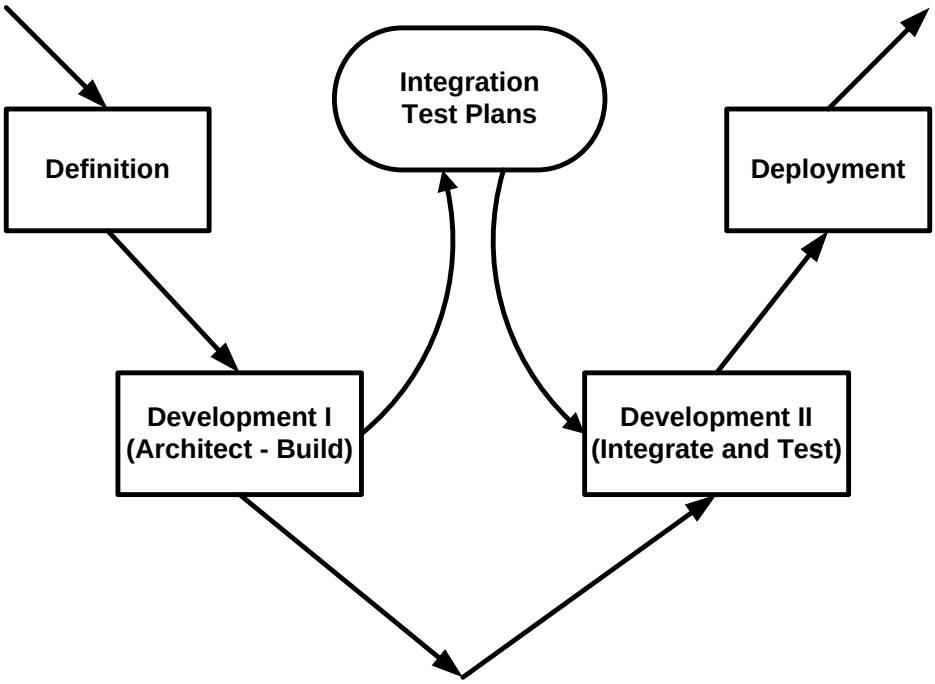


Figure 2 - System Integration Classic "V" Development Model

The route in the "V" model is followed to create the System Definition, Architect and Build, on what is commonly referred to as the "down-stroke", while the Integrate and Test, and Deployment reside on the "up-stroke". Rarely do any Systems I&T processes tread lower than this viewpoint, or provide insight into how the integrators initiate their tasks. Many organizations want to provide Integration teams the freedom to develop the system as they progress, and are reluctant to fully define processes. However, if this situation gets out of control, the lack of documentation fails to provide any foundation for an organization.

3.5 Pathway to Systems Integration Model

The Space Engineering Lessons Learned article [Evans, 1989] addresses the most often repeated problems during sub-system Integration and Test: (1) "…inadequate testing at the subsystem level…is often due to poor early test planning, including lack of test design

in parallel with subsystem design"; and (2) "…lack of good process (i.e., fabrication/ assembly/Integration/test) planning". Systems Integration team involvement should start well before the actual hardware and software elements and sub-assemblies are available for Integration, beginning with planning, and preparation at the program level system requirements' definition phase.

Pre-planning is non-existent in the classic "V" diagram; this approach is very limited for providing enough direction during the integration process for what is actually needed. A more functional model of the process is essential for planning, providing a roadmap for engineers to follow, and create repeatability in an organization's successes. Figure 3, an original adaptation of the "V" System Integration model, commences the process well before the System Integration phase. By altering the user's viewpoint and route direction, this model extends the classic Integration definition, depicting two interrelated paths -- Program Path for Hardware & Software, and System Integration Path -- which converge at System Integration.

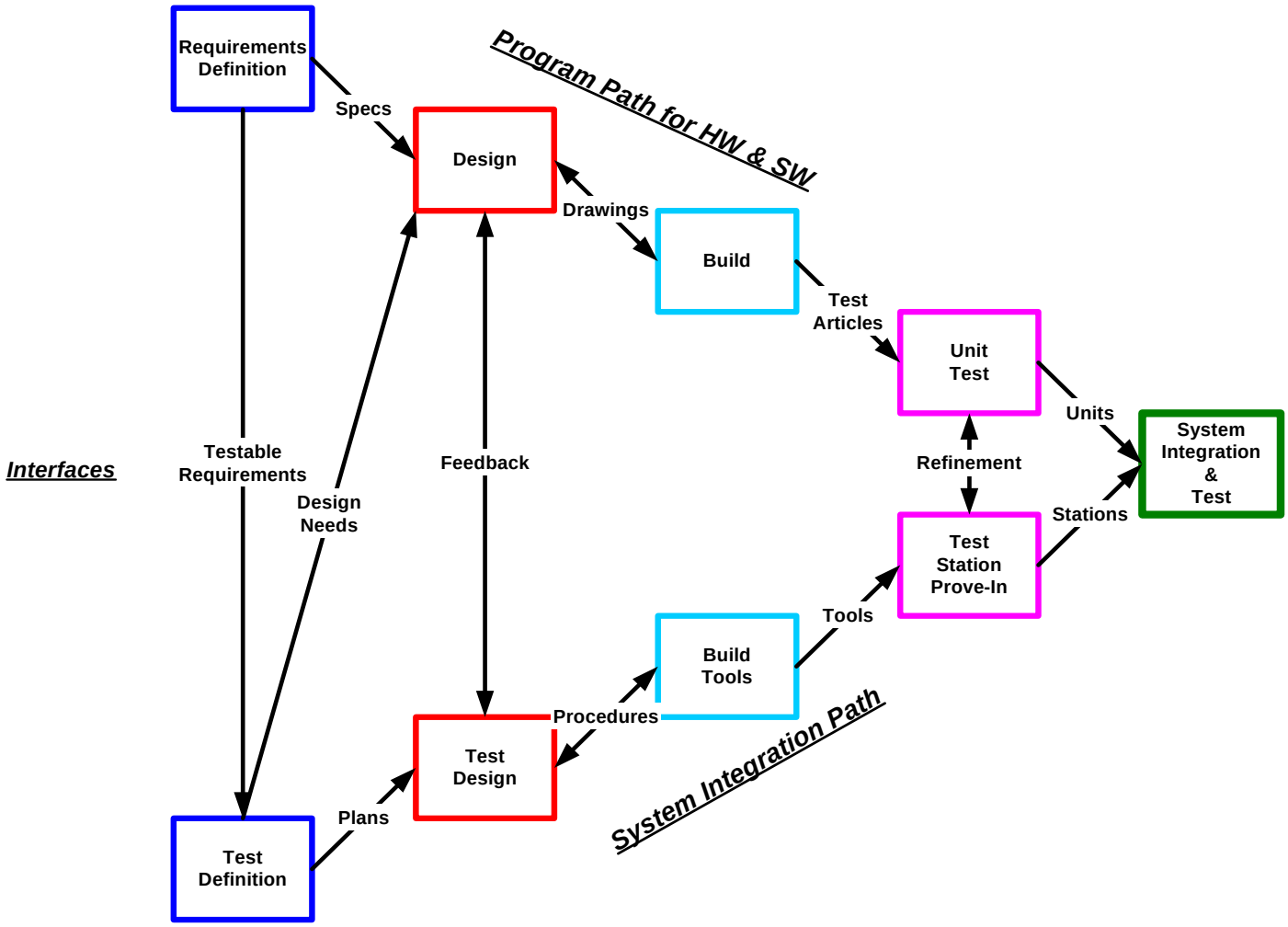


Figure 3 - Pathway to System Integration & Test Model

The model's multifunction pathway more realistically depicts the steps taken during planning phases to the culmination of the Systems I&T Process. Requirements Definition jump starts the Integration Test Definition by deriving and supplying the Testable Requirements. The Integration team starts by defining the required tests and Integration flow, based on system and contractual requirements; this is the point where the paths are furthest apart. At this point, the integration team is concerned with how the requirements affect later tasks, and hardware/software teams are concerned with commencing unit development.

As program time progresses and the two paths converge; at some process steps, ideas and task needs are shared. But the process Model is nonetheless considered chaotic due to the necessity for highly coordinated inter-group and inter-task communication, for without a database repository that can be accessed, personnel must remain in constant contact to keep the flow of information intact. The chaotic model is not optimal due to many human factors which may disrupt the information flow. The system Specifications from the Requirements Definition phase, and Design Needs from the Test Definition phase feed the Design step of the process. Plans output from the Test Definition phase are used by the Test Design step. At this point there is Feedback between the two pathways which further establish interfaces necessary for testing the system. Drawings and Procedures flow from both Design phases, and are used to build the system and test tools. The Design phase outputs needed in the Build phase are also used to feed back into the Design phase for possible re-work or re-design as issues arise. Test Articles and Tools are output from the Build phases, and Unit Test begins on the system while test Stations and Tools are "proven-in" using available hardware. The final step in the system build phase is Unit Test, performed by the developer

of the unit and involves some level of interaction with the Integration team in preparation for the ensuing Integration phase. In the final step before System Integration maximum interaction occurs between teams to insure knowledge transfer, and smooth transition to the System Integration & Test phase.

3.5.1 Knowledge System -- Introduction to the Model

A Knowledge Center as introduced by Saloman [2002] discusses Knowledge Management containing many types of database entries including administrative, meeting minutes, trip reports, publications, and Lessons Learned; it is a system of inter-related components. Recognition of the possible knowledge components within the Systems I&T Process is a critical first step for Knowledge System incorporation. There are many Integration and system build outputs from the pre-Integration phases of the Systems I&T Process including: Specifications, Plans (Integration Flow, Master Test Plan, and Integration Level Test Plans), Drawings, Test Procedures, and Tools. The main outputs from the process are typically customer "deliverable" objects, shared between the program teams, and available from one phase to the next. The program deliverables are generally in pristine form, include all the proper and finalized issues, and may be required for progression to the next step.

Questions arise regarding disposition of the "mistakes", "errors", "redesigns", lab notes, research, and "successes" discovered by the team during program development. The focus must shift from losing information (Knowledge Loss) to tracking and retaining data (Knowledge Gained). If the information in whatever form is tracked, the issues and events

become the foundation of the Lessons Learned or Knowledge Gained Database for the current program.

Sheard, Lykins, and Armstrong [1999] suggest that to overcome a documentation problem in process improvement, the team should "determine what processes are being implemented now, map these to the model, and find the best places to insert additional needed practices". In order to improve the entire System Integration process, Knowledge criterion must be defined and then incorporated at appropriate junctures along the Integration path, as well as within the System Integration phase, not only for the current program but also for future programs. During periods of discovery, Knowledge capture and utilization is imperative in a Knowledge Gained Database (KGD). The KGD and necessary tools for its input should be set-up for inclusion into a company wide repository. A KGD account should be established for the program and an appropriate team assembled and empowered to create guidelines for event inclusion in the KGD.

Engineers and managers, in order to make the KGD effective and useful, must incorporate it into their daily tasks. At each new phase in system development, the KGD must be accessed for any pertinent information. When major decisions are made or trade studies completed, a perfect opportunity arises for Knowledge input to the database. KGD users must realize and accept that inputting information from one program phase may not benefit the current program phase, but may greatly impact the next program or phase.

Figure 4 depicts an improved System Integration Pathway Model incorporating the KGD at significant phases.

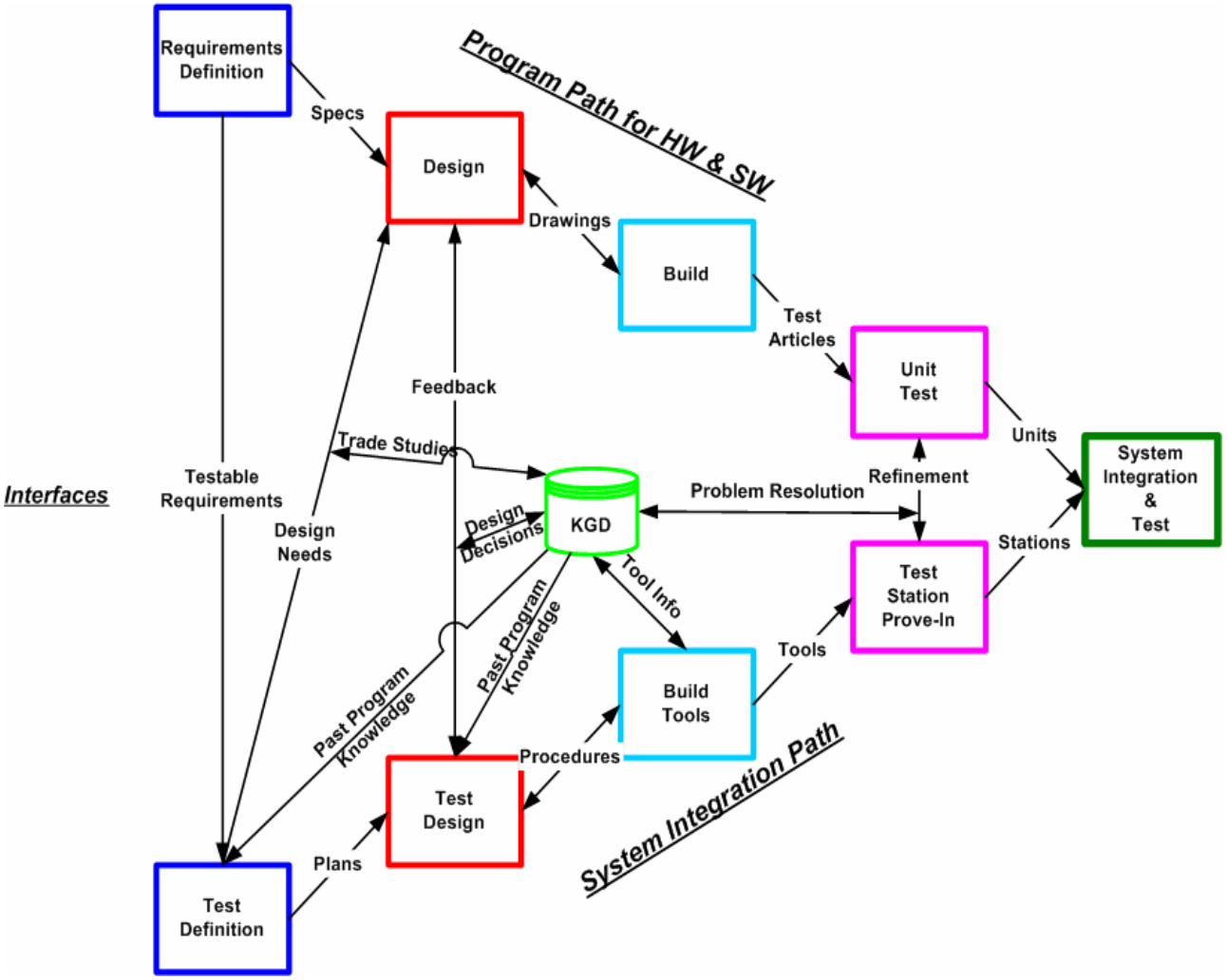


Figure 4 - Integration Path Model with Knowledge Gained Database

The KGD is the repository of inputs and outputs along the System Integration Path, including each step's information contribution. Notice the frequency with which the KGD is accessed, and realize that without the KGD, considerable information can be easily lost. Past Program Knowledge has great impact during the Definition and Design phases; Design Decisions are shared between the database and current programmatic issues. Design Decisions may be related to unique hardware discoveries made on other programs by a different team within the same organization. Discoveries would not normally be known and shared, especially in larger organizations, if not for a KGD. Trade Studies, Problem Resolution, and Tool Info are other types of knowledge that are both captured and shared within the development phases. There are many categories of knowledge that can be captured in the KGD, but it is dependent on the needs of a program, company or department which will define and delineate the knowledge categories.

3.6 Systems Integration and Test

The following subsections introduce a set of flows which describe the fundamentals of a Systems I&T Process. Proper planning allows the Integration team the opportunity to look for voids in the Integration process, while providing direction for its implementation. Beginning with a program level Integration and Test plan, the planning process is explained through Validation Testing including Failure Analysis of a unit within the system. The process flow diagrams are intended as tool templates for metrics and progress tracking during the System I&T evolution, and can be tailored to a program or organization. The flow diagrams presented herein have been used by the author and others with great success to help clarify and add visibility to the Integration and Test process. The diagrams detail the

Systems I&T process and are in themselves sub-processes, insertion of the KGD is included where appropriate.

3.6.1 Integration and Test Flow Plan

In reference to the System Integration Path as defined in section 3.5, Plans (one of which should be an Integration flow plan) are output from the Test Definition step. The overall integration plan for the elements comprising the system is the primary consideration during this time; but not calendar deadlines. Test contents are discussed in subsequent sections.

To show how elements are combined, the Integration flow diagram provides a map of tests to the appropriate Integration Levels. The Integration and Test occurring at the lower levels of the system are normally performed by the unit design and build level engineers. The System Integration team will not be involved in the day-to-day work at low levels of Integration, but should be cognizant of the overall Integration flow and resultant tests. Since lower level system elements are eventually "sold off" to the next level of Integration; the "buyer" at that level must understand what they are receiving. Figure 5 reads left to right and depicts a generic System Integration and Test flow progressing from the lowest level Units, to Components, to Subsystems and finally to the System.

The difficulty in generating this plan is determining what constitutes each element classification in the system. What differentiates a Unit, Component or Sub-assembly? The element delineation decisions should be based on program schedule, engineering disciplines, or an organization's administrative groupings.

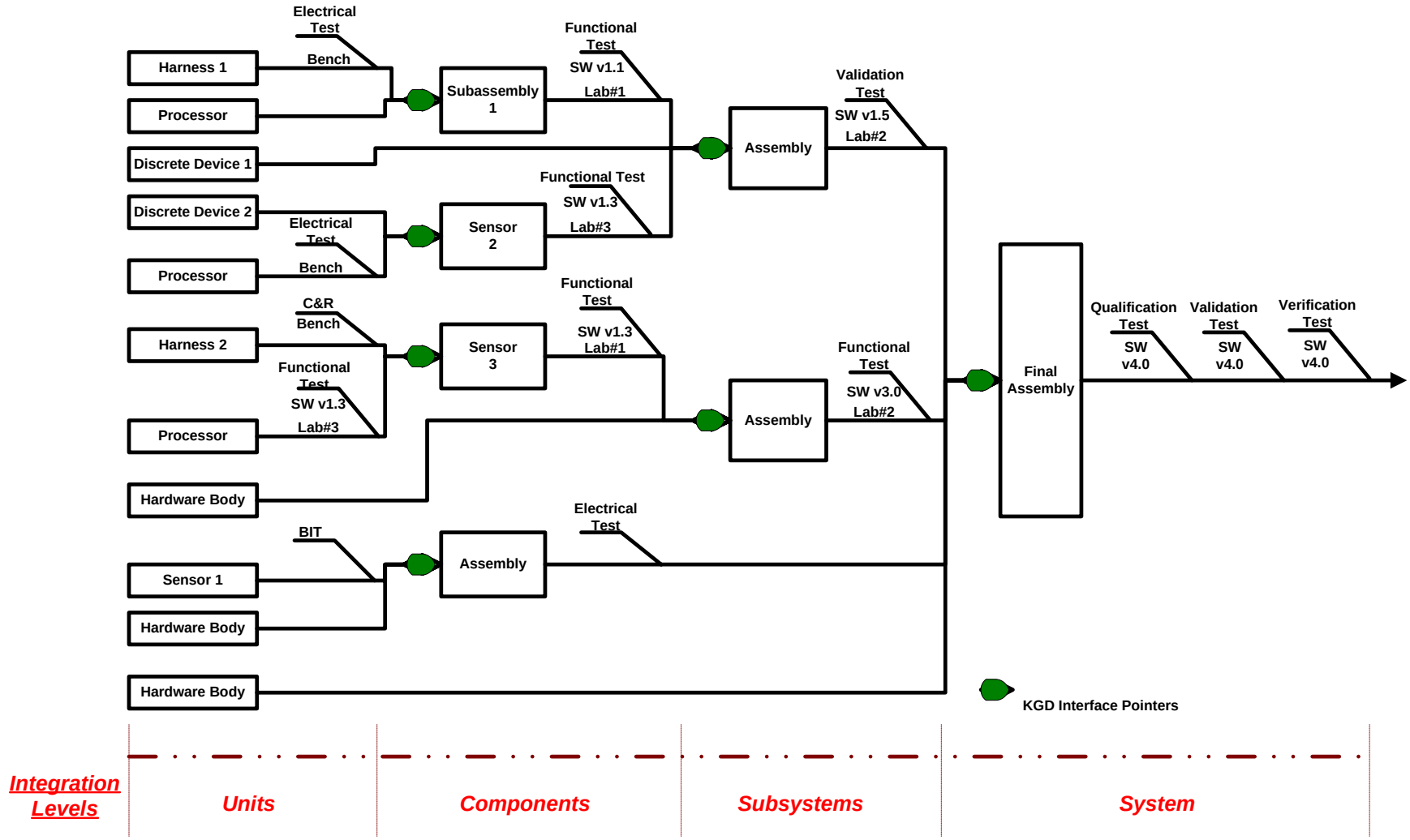


Figure 5 - Program Master Integration and Test Flow

The main elements depicted in this Integration Flow are hardware oriented. Software elements typically follow their own well-defined build processes, separate from the I&T process but interjected in the Integration flow for operating the hardware at required interaction points. The lowest level Units, such as special processor boards, low level sensors and other unique items, are tested autonomously via Built-In-Test (BIT), Continuity & Resistance (C&R) or low-level Functional Test. Units are brought together in the Component level as sub-assemblies and major system sensors, possibly requiring low-level firmware or special test software for Functional Tests. Components are combined to create major Assemblies at the Subsystems level usually requiring Validation Testing and lastly, the Final Assembly is integrated at the System level. Tests performed at the System level may include: Functional, Validation, Qualification and Verification, not all necessarily on every System. In the figure, the angled lines represent tests that are performed; the area under the line is for relevant test information such as lab assets, location or other special needs for the particular test. The Integration flow can be a useful tool to a system integrator to add visibility into the overall build of the system. A larger drawing can be generated and then divided into more workable sections by Integration Levels with more detail included. From the Integration Flow, schedules can be developed so that further planning can progress.

Green pointers are used in the Integration flow diagram at the start of each of the phases of Integration, and represent reminders to the team to perform a KGD search for critical and pertinent information, as well as to input entries when significant knowledge is gained.

A functional implementation of System Level I&T flow abstraction is a lower level, daily flow of the process. A generic example of a tracking daily Systems I&T flow is demonstrated in Figure 6.

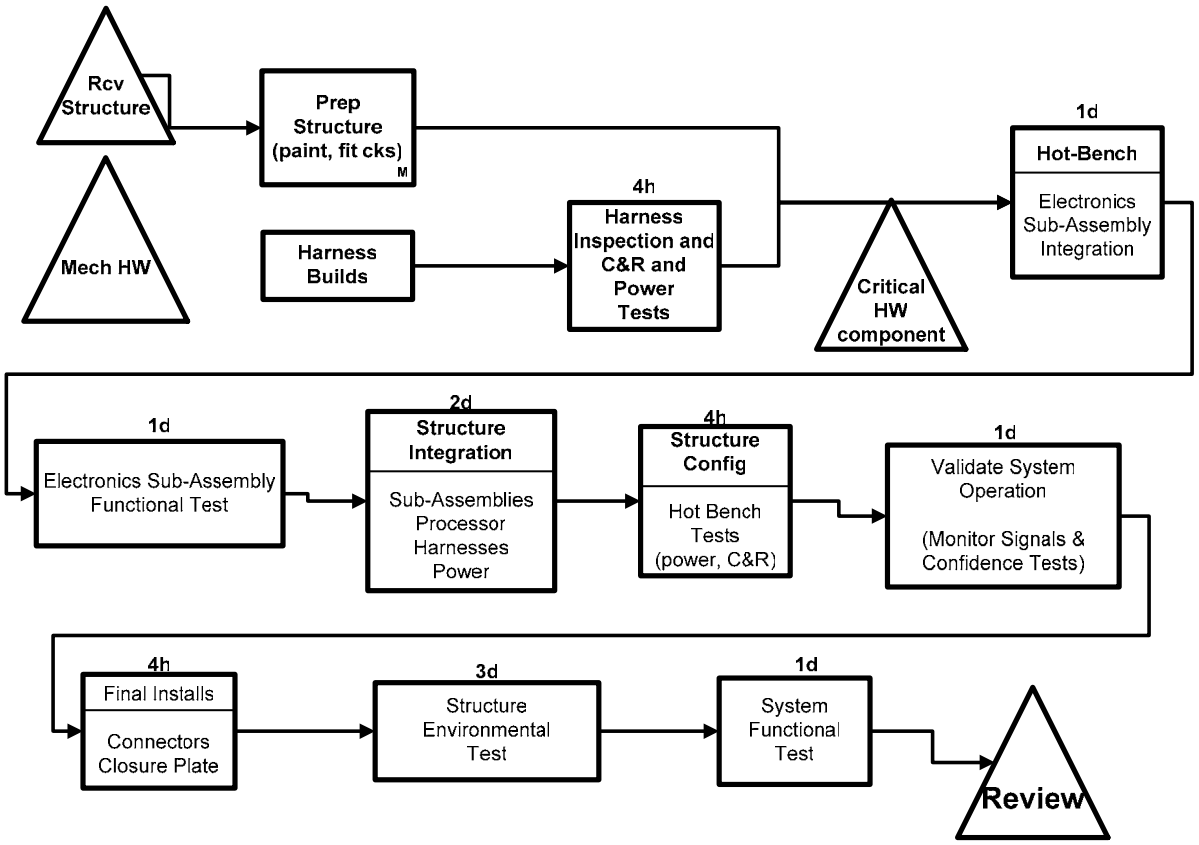


Figure 6 - Daily Integration and Test Flow Diagram

The Daily Flow diagram may be generated and tailored for any level of the process, is a subset of the overall system test flow for a testable unit, and provides detailed information about the Integration steps. Individual Daily Flow diagrams can follow one or more of the horizontal flow lines in the Program Master Integration and Test Flow (Figure 5), indicating Integration of hardware units and test of the new assembly. Diagramming the I&T flow for day-to-day activities related to the system or to lower level elements of the system, provides

visibility of daily schedules, serving as an interface to management and other involved engineering disciplines. Expected completion dates and task durations may be included to allow tracking, and provide a target completion date to the customer. Customizing activity boxes in the Daily Flow diagram can indicate completion.

3.6.2 Validation Test Flow

Validation Testing determines whether or not a testable unit meets program requirements and design. A single test or series of tests is performed on outputs of each level of the integration process as depicted in the Master Integration Flow in Figure 5, certifying operability and functionality; more rigorous testing is performed at higher levels of integration during Validation Test. In Figure 7 a baseline series of tests is presented in a Validation Test Flow which includes a success oriented Unit Under Test (UUT) sequential procedure, along with failure and design review routes, to be retested as required. The failures discovered during this flow are usually of significant impact, requiring analysis and possible re-design, thereby triggering KGD activity. Failure isolation and analysis is covered in more detail in the section 3.6.4.

Reports output from each test and analysis decision point are interspersed at significant steps in the process; copies are included in the UUT delivery and also routed to the KGD. The Validation process can be tailored by program or organization management for use with sub-assembly or component level functional testing as necessary. The test flow can be modified for the integration level of the UUT; whether a low level processor board not requiring a software release to operate, or other system elements incapable of performing tests such as BIT, which may have to be completed at higher levels of integration.

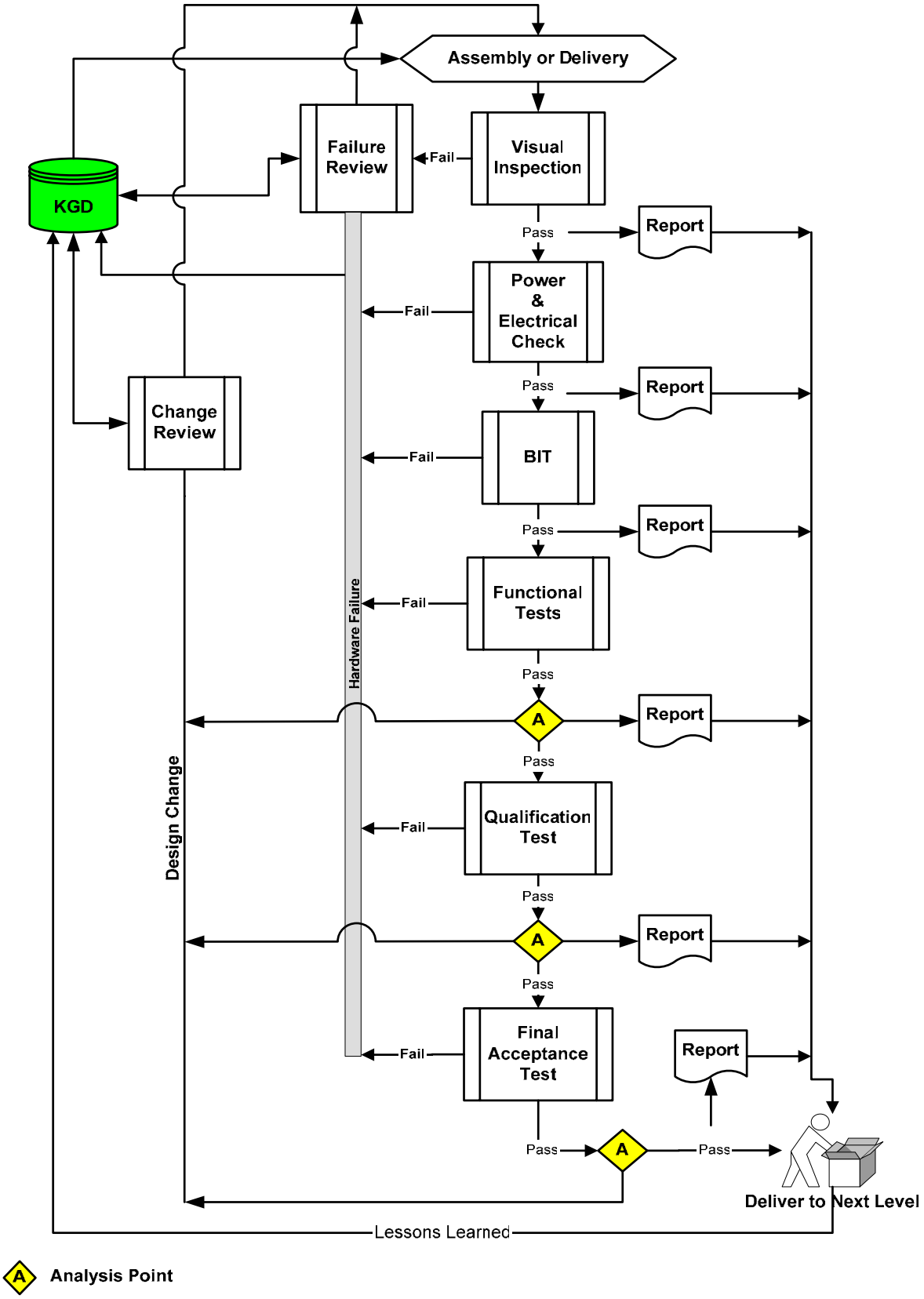


Figure 7 – Unit Under Test (UUT) Validation Test Flow

The Validation Test Flow process incorporates KGD inputs at the beginning of the process, and during any Failure or Change Review. Incorporation of the KGD in the process is a significant aspect of the Validation Test Flow. Candidates for KGD inclusion are: Lessons Learned during the implemented test flow, hardware failure notes, and outputs from any failure reviews. In the test flow process, Review teams benefit from searching an existing Knowledge System for any information pertinent to the Review.

Each test procedure in the Validation Test Flow is its own Process. First, a Visual Inspection is performed on the UUT in order to discover problems that may have occurred during handling and shipping, such as bent connector pins or loose wires. The next Validation step is the Power and Electrical Check performed on any electrical component: wire harnesses used for interfacing between components, processor boards, and power subsystems. Electrical checks include: Continuity and Resistance (C&R) tests, and a power form test prior to applying power to the UUT to assure there will be no damage to the UUT. Typical problems uncovered at the electrical test step are wiring errors and failed circuit board electrical components.

Depending on the UUT capabilities, qualified software may be required for the BIT Validation step. BIT conducts a self test to check individual components within the UUT, and reports status. The Functional Test checks basic functionality of the powered UUT; special test software may be required to execute low-level testing operations. The Functional Test is performed independently of the UUT's application within the final system; it is performed to test stand-alone functionality.

The next Validation Test step is a Qualification Test performed to demonstrate that the UUT is fit for use within the expected environmental conditions. Qualification testing may include a series of thermal and vibration tests to prove correct UUT functionality under extreme conditions, which should be tailored to the operational environment of the UUT. The Program Specifications, noted as a step output in Figure 4, typically include the customer's qualification requirements. To conclude Validation Testing, a Final Acceptance Test is performed which may include application-specific operational tests prior to the next level of integration or delivery to the customer.

Test failures resulting in a UUT modification or repair following the Change Review or Failure Review process step, trigger an entry to the KGD, then restart the Test Flow. Program Engineering Leads should be involved in Review process steps because, what seems like a simple problem resolution for one team, may adversely affect other teams. For example, a re-designed hardware component may solve a fit problem, but may no longer allow access to harness connectors for the Integration team.

Output data must be analyzed (Figure 7 yellow diamonds) as the test successfully advances beyond Functional Tests. Following the post-test Analysis decision, if the Design Change route is taken, a KGD entry should be generated; this presents an opportunity for engineers to re-design hardware/software, resulting in what may be some of the most valuable Lessons Learned during the I&T process.

3.6.3 Verification Testing

Verification Testing generally occurs at the System Level operation in a controllable environment. Following successful validation testing, the UUT is tested to verify

performance of the requirements within given tolerances. Special test stations, designed as closed-loop test systems, may be developed to provide a platform for operating the system, collecting system reaction, and real-time output. Computers, installation fixtures, and multi-media data collection devices are potential components of the test station. Repetitive testing is executed to substantiate correct system operation under semi-realistic conditions, and data analysis is performed on the results. This testing not only serves to verify the system, but can also save budget and time by shaking out system faults before final installation. Further more costly testing prior to final system customer acceptance may be required.

The Verification Test Flow is uncomplicated from the Integrator's viewpoint, because the system is operational at this point. The cycle of Test and Data Analysis is repeated a prescribed number of times to prove satisfaction of the performance requirements or, until an operational problem or system failure occur; either failure classification results in the UUT entering the Failure Analysis Flow (Figure 9). Data for Knowledge System input may include test result interpretation, as well as Verification Test failure descriptions.

3.6.4 Integration Failure Analysis

Perhaps the biggest challenge facing Integration engineers is the process of investigating failures; discovering the root cause of a failure might be compared to finding a lost puzzle piece. Failure Analysis is both the heart and heartache of Integration and Test activities. Basic failure process activities include: methodical steps; tracking events, no matter how small; dissecting problems to the smallest common denominator; and determining resolutions. A fundamental view of the Integration failure process and most

common approach of diagramming the process is the classic “Test, Analyze, Fix, Test” (TAFT) method as adapted in Figure 8.

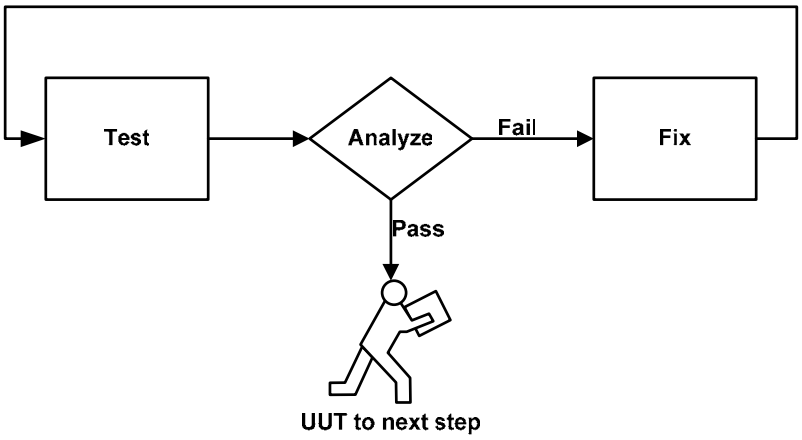


Figure 8 - TAFT Methodology

The TAFT method defines major steps that are followed to advance a UUT to the next level of Integration or customer delivery. One area of I&T process improvement requiring significant attention is the dissection of the Analyze step into more tangible segments. Due to its inherent ambiguity, the analysis step in the TAFT methodology creates difficulties for engineers. The process flow depicted in Figure 9 eliminates the ambiguity within the TAFT methodology’s Analyze step through delineating the I&T Failure Analysis Process. This Process is triggered whenever test failure occurs at any level in the Program Master Integration and Test Flow (Figure 5), or for example as detailed in the Validation Test Flow (Figure 7).

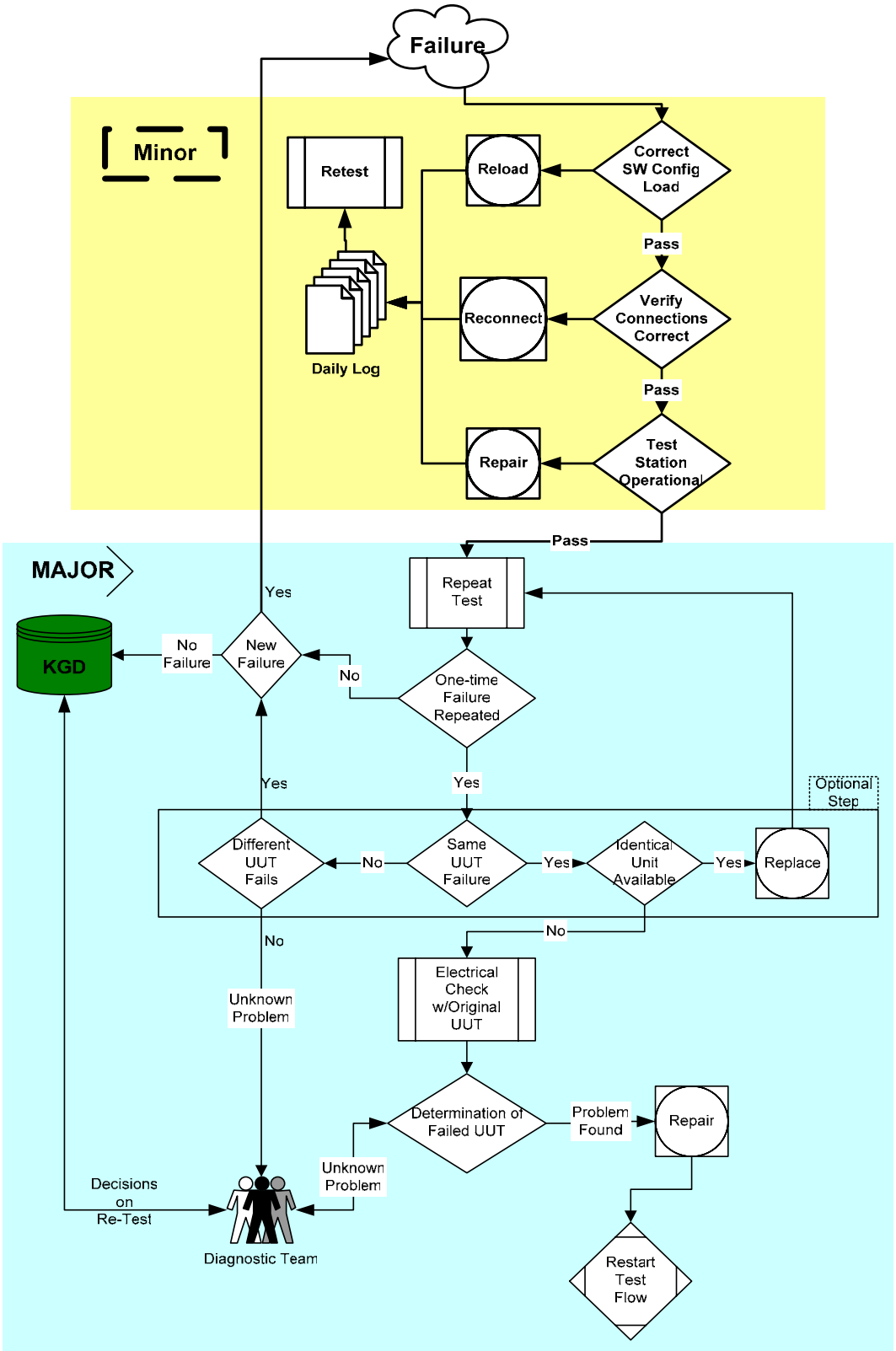


Figure 9 - I&T Failure Analysis Process

The Failure Analysis Process begins with test Failure, and is used in conjunction with the Validation Test Flow discussed in section 3.6.2. If a UUT fails during test, the failure process is set in motion; if the process determines root cause, Validation testing restarts. The Integration team starts "peeling back the onion" to find the source of the failure, prompting the low-level analysis process: minor steps, represented in yellow; and major activities (green background), should the failure remain unresolved. Inclusion of a KGD greatly enhances the Failure Analysis Process, by providing opportunities to track information for future database searches. Pre-existing KGD elements providing related failure's histories may be useful for analysis at this low-level integration task.

During Correct SW Config Load verification, the test engineer confirms whether or not the loaded software is corrupted; and if the correct or most recent released configuration is loaded. Within the Verify the Connections step, UUT connections, and any special test equipment or station connections, are examined to assure completion per latest available inter-connect drawings. Checking connections is sometimes referred to as a "wiggle test" -- verifying that correct connections are made and are secure. Test setup configuration and individual operation of each piece of test equipment is validated for the Test Station Operational check. If the cause of failure is found to be within any of the Minor Process steps, an immediate fix is effected, an entry into a KGD daily log is generated, and UUT testing is resumed upon exit from the Failure Process. Standard practice, however, often bypasses the Minor Process steps resulting in obvious Knowledge Loss.

If none of the Minor steps is determined to be the cause of test failure, the process is bumped to the Major category, the first step of which is a Repeat Test of the UUT, a step not

always recommended or possible if there has been a hard (complete) failure of the UUT. As with any experiment, repeatability more easily enables solutions to the problem. If a New Failure occurs, the whole failure process is restarted. If the UUT failure is repeated, an Electrical Check is performed, such as an inspection followed by a continuity and resistance test, to try and determine if there is an electrical root cause. If UUT failure is determined, Repair is made and the appropriate test Restarted. If at this process juncture, there is not a New Failure which may indicate there is No Failure remaining to investigate or there is an unrepeatable phantom problem; the process ends with an entry in the KGD and the UUT is returned to the test flow. The No Failure results are the most difficult to diagnose due to non-repeatability.

If when nearing the end of the analysis process, UUT Failure is yet to be determined, it's labeled an Unknown Problem requiring review by a Diagnostic Team, minimally composed of involved program experts to determine resolution. The failure investigation may need to be raised a level to a more formal failure review process, involving upper level management and outside program engineers to provide a fresh perspective.

In the Failure Analysis Process is an Optional Step provided there is an identical functioning UUT or sub-assembly of the UUT available for diagnostic purposes; the "known good" for comparison against failures, saving time and budget for the program. The failed test may then be repeated using the Identical Unit or interchange one UUT component, if the I&T team determines it is safe. If a Different UUT passes, it indicates to the Diagnostic Team an Unknown Problem exists in the original unit. If the replacement component or

UUT fails it is considered to be a New Failure, and the analysis process restarts for both units.

This somewhat circuitous process can become extremely confusing. Process steps can be repeated many times with new components, using different software versions, or replacing test station elements, as in the previous narrative. Methodical tracking of the exact test configuration that was in place at each test iteration is a crucial step and helps avoid repeating tests, organizes direction, and aides in determining failure root cause. As the team works through the process, it is imperative that a test conductor carefully, stringently, and succinctly tracks the Failure Analysis configurations and findings. Tracking team progress and analysis results may be accomplished through a computer based interactive tool, spreadsheet, or lab notebook, just as long as the process is tracked.

CHAPTER IV

TOOL PROTOTYPES FOR THE KNOWLEDGE SYSTEM

4.1 Introduction

For engineers under time and schedule pressures, nothing is more frustrating than to be required to use a tool that is impedes rather than helps progress. So that the tools may be effective in helping improve engineering processes, and complete an organization's transition to becoming Knowledge Based, it is critical that they be fast, intuitive, and readily available with installed user-friendly applications. Poorly managed or badly designed computer tools, will be ignored by frustrated users, and Process Improvement will fail. Networked computer stations, with installed applications, should be available at every possible engineering testing location; the server must be large enough to handle enormous data loads and still provide fast user interface.

The Systems I&T Process Improvement detailed in the preceding chapter flow diagrams, introduced the need for two separate tools interfacing to the KGD: Minor and Major event logs. A Minor event log should be geared toward engineers' daily tasks, and have a good graphic interface preventing excessive manual data entry, thereby reducing the amount of time spent entering information, quickly selecting from prepared lists and pre-defined choices. In contrast, the Major event log should not be a detailed delineation, since the information may be more complex and requiring freedom to enter all relevant data in a narrative format.

4.2 Minor Event Entry Tool

An example of a Minor Event log tool is Raytheon Missile Systems' (RMS) ITLog [used by permission of RMS administrator]; the tool supports I&T processes for recording problems, documenting issues, and tracking corrective actions. A separate engineering department maintains the database and tool, provides training, and performs metrics tracking for factors such as cycle time and problem causes. In addition to the I&T team, other teams have access and visibility into ITLog's database, including: Quality, Mechanical, Configuration Management, and Program Management. Figure 10 is the main screen of the most recently released version of ITLog.

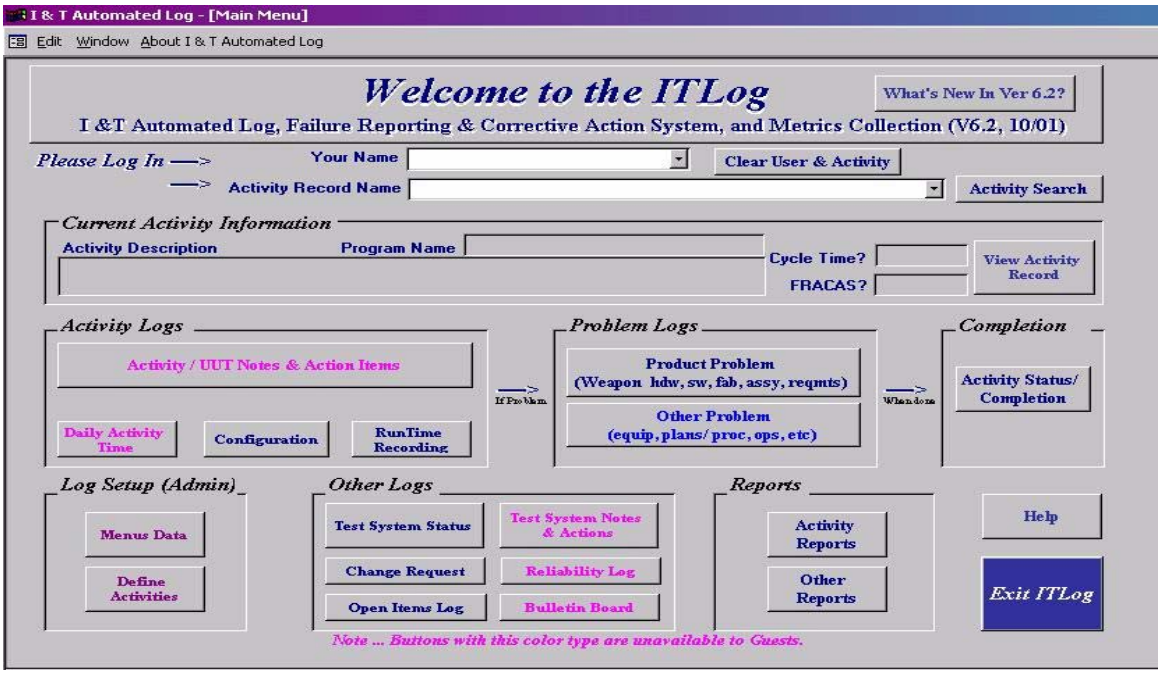


Figure 10 - RMS ITLog Main Screen

Pull down menus include: personal names for logon to the Tool and activity selection, including such items as "Unit # Vibration Test", "Assembly # Integration", and "System # Validation Test", where "#" is unique identification for a specific UUT. Selection

of activity automatically displays the program name and a basic activity description in the Current Activity Information window.

The user selects from the Problem Logs or Reports windows, depending on the type of information to be entered or retrieved. The Product Problem Log screen (Figure 11) is utilized when a problem is discovered during Test that requires tracking. The report entry is automatically assigned a number, and using the logon, an Originator and Observed Date, which may be modified to reflect the actual event's date if necessary. Optional pull down menus incorporate most problem aspects including: Environment, Location, Cause Process Code, Phase Found, and Test System. In this particular tool there are two pages of entries: the problem information as shown, and a second page for corrective action and disposition information.

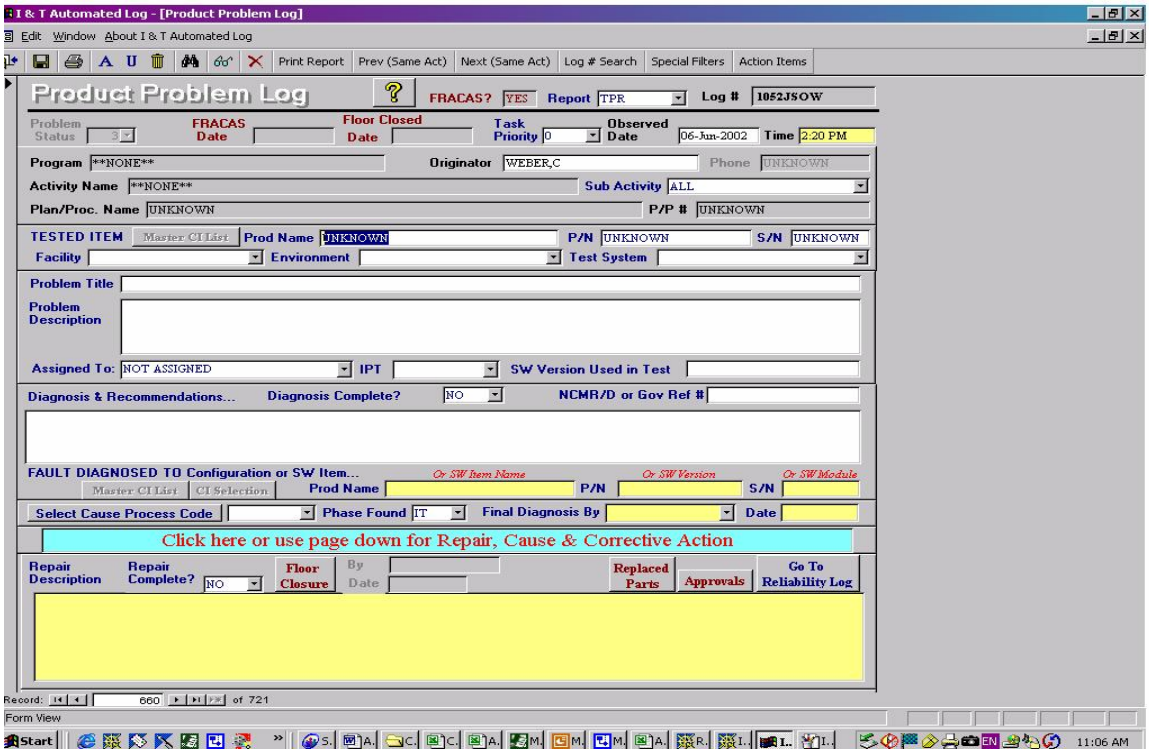


Figure 11 - ITLog Product Problem Screen

Pertinent details are entered by the originator including a Problem Title and Description. Two important fields are Assigned To and Cause Problem Code: the person who discovers or enters the problem is not necessarily appropriate to assignment for follow up and problem resolution. Selection of the problem code is important to management for metrics tracking. Finally, Diagnosis & Recommendations are entered and gives a first impression of the root cause, and suggestions for repair or investigation.

Activity Reports Main Menu (Figure 12), as selected from the main screen (Figure 10), enables the user to format and generate many different report styles from available data. For instance, the Output Report from the I&T Problems selection provides a simple report, in a word processor, or in a spreadsheet application for data analysis.

Figure 12 - ITLog Activity Report Screen

4.3 Major Event Entry Tool

A separate Knowledge System tool should be utilized for capturing Major events, and require a simple management signoff flow for database entry. A typical one page format may include: date, title, keywords and issue description. Other related events may be incorporated, such as formal review board proceedings, as part of the issue narrative.

Xerox DocuShare® is currently available for use within Raytheon, as one example of a Major event capture tool [used by permission of RMS administrator]. Following logon to the Raytheon Intranet site, the user interfaces with the intuitive tool in a standard internet browser window. The Intranet's home page screen gives general information about the tool, provides templates for documentation submittal, and includes a complete list of Raytheon user-sites with individual document categories. A DocuShare Introduction and Overview window (Figure 13), or Raytheon home page (Figure 14) are both selectable from the Intranet home page.

Figure 13 - DocuShare Introduction Window

Figure 14 - Raytheon DocuShare Home Page

The Raytheon home page is comprised of: folder options for selecting a particular functional business structure, tutorial information, and account access instructions. Once a folder is chosen, the user enters one of the business units, and a new page appears as below in Figure 15, which includes a search option. From this folder option screen, any number of queries can be performed on the current folder.

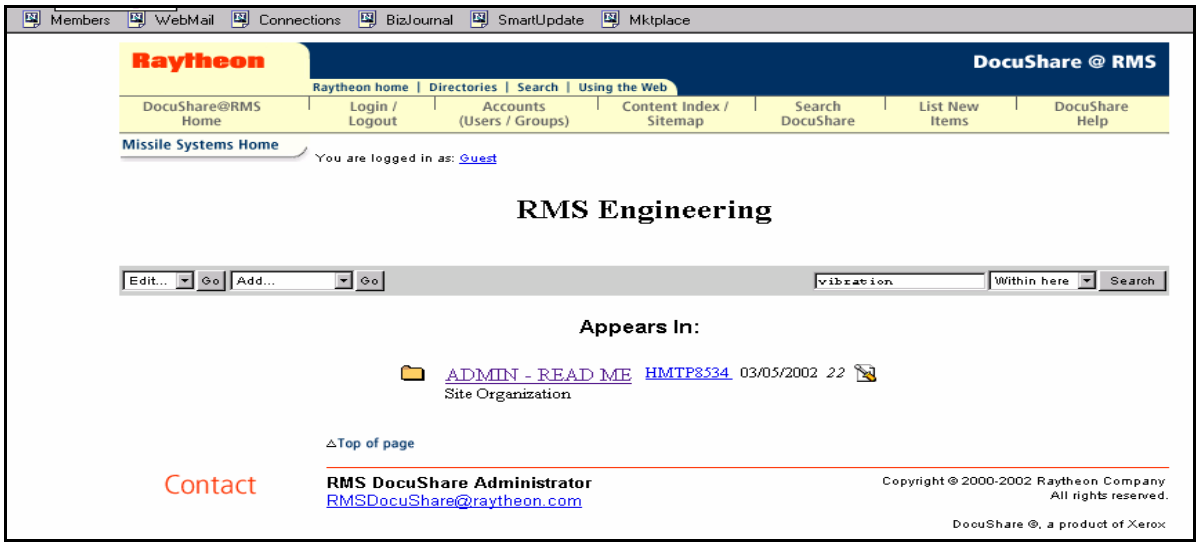


Figure 15 - Engineering Folder Option

From the top line of the browser the Search DocuShare button is selected if a more encompassing search is needed, and Figure 16 is displayed. The inputs in the Major Search Screen allow the user to search either selective portions of or the company's entire database. Query results including documents and document folders are displayed in a new window.

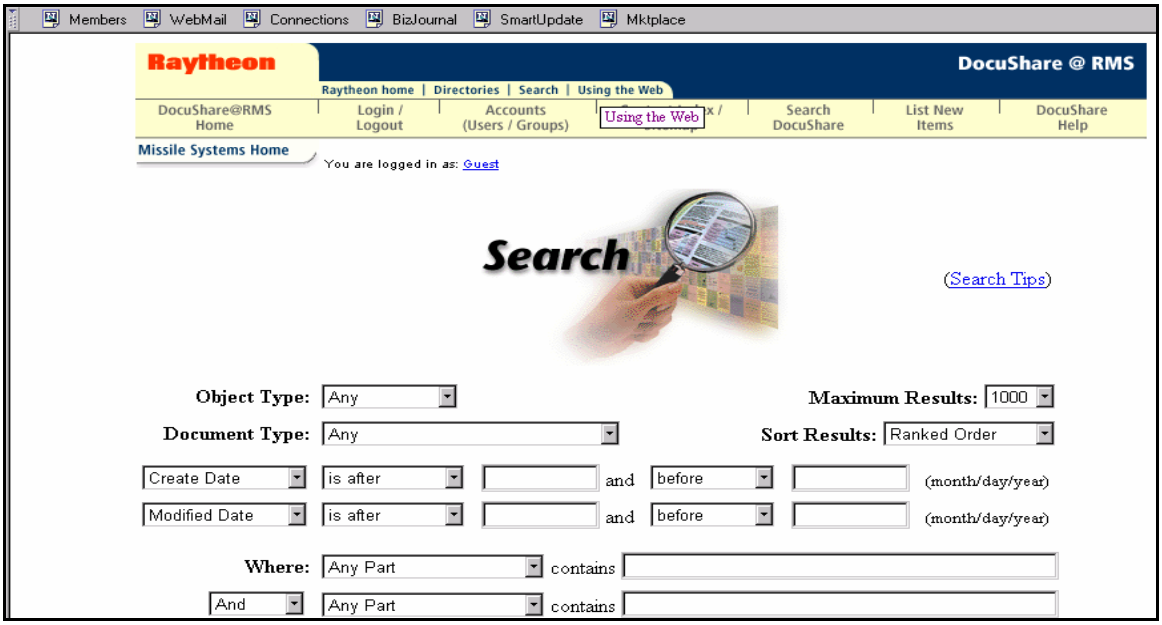


Figure 16 - Major Search Screen

Tool options include a subscription service for managers to perform a signoff. An alert is generated when a document is entered in selected folders, and events are accepted into the database following approval.

4.4 Industry Experiences

In every aspect of modern industry, numerous other tools are used. One such tool is NASA's Lessons Learned Information System (LLIS), as referenced in a project management article about a Lessons Learned experience [Lee, 2002]. The author relates several database entry examples, stating that LLIS is "surprisingly agile and easy to use".

A second tool example is the Navy Warfare Development Command's Navy Lessons Learned System (NLLS) [NWDC, 2001] which was discussed in an internal briefing presentation regarding development of the tool.

The tools to perform the task of tracking and managing a Knowledge System are widely available; the difficulty of tool selection and Knowledge System process implementation is a topic for separate study.

CHAPTER V

ANALYSIS OF THE SYSTEMS I&T MODEL

5.1 Discussion

Most nationally recognized engineering disciplines have developed, utilized, and accepted standards, such as the American Society of Mechanical Engineering (ASME), and Institute of Electrical and Electronics Engineers (IEEE) which maintain build standards and testing procedures. Additionally, the Software Engineering Institute (SEI) has developed Capability Maturity Model (CMM®) ratings providing a nationally recognized standard for software assessment of an organization's standards and processes. A younger organization, the International Council on Systems Engineering (INCOSE), is working on development and promotion of standards. Integration and Test engineering typically falls under the jurisdiction of Systems, but national standards are nearly non-existent. While major engineering disciplines follow structured practices and are held to higher standards, Integration and Test engineering must become more completely defined, and adhere to common standards to be respected within the engineering community. Integration and Test remains somewhat of a mystery and, at times is considered to be performing "black magic" engineering; the decree of Integrators is "just make it work".

The building blocks of standards are processes, an example of which is the Improved Systems I&T Process discussed earlier in section 3.3; the focus of improving the process is the incorporation of a Knowledge System, which includes a Lessons Learned database. A presentation on Reliability Enhancement [Marshall, 1999] offers one rationale for promoting process improvement due to "Poor Feedback Therefore No Formal Use of Lessons Learned".

Many engineers and managers proclaim that understanding and incorporating Lessons Learned will improve not only the next task or program, but the overall organization as well. An organization's successful transition to becoming Knowledge Based requires obtaining management buy-in. Lack of characterization of the importance of incorporating Lessons Learned can inhibit gaining full management support, which can then be a major obstacle for communication to engineers of the necessity for Lessons Learned feedback. How do engineers and managers appreciate and understand the importance of the proposed process improvement? Studies, proclamations, presentations, suggestions, and assertions initiate organization transition ideas; an analysis goes one step further, providing a solid foundation to argue for process improvement solutions. The benefits of an improved I&T process as presented in Chapter III are examined in the following sections utilizing variance and functional reliability analyses.

5.2 Analysis Introduction & Scope

Typically, analysis is performed on tangible objects using scientific experiment and observation techniques to qualify, quantify, or substantiate objects and their components. After measurements are taken and information is collected, data analysis of the object under investigation is performed, which provides output results that are difficult to dispute. Basic tenets of experimentation and analysis methodologies can be extended to different categories of systems with elements. A process, after further examination and expansion of analysis definitions, is discovered to have system components, thereby providing the elements needed for performing an analysis.

5.3 Variance Analysis

A critical aspect of system operation is its component's interactions and inter-relationships. Analysis may reveal that one of the components is unimportant, or that a relationship among components is not useful, consequently affecting the system's design. In traditional experimentation, one parameter is held constant while others are varied over a selected range of values and observations are collected; the procedure is repeated for all possible combinations which result in a very lengthy experiment.

In the Taguchi Technique (Orthogonal or Factorial Experiment) [Ertas & Jones, 1996], the Analysis of Variance (ANOVA) statistical method includes analysis of parameters and their interactions to more quickly arrive at an equivalent result. An original adaptation of the Taguchi Technique methodology is the basis for a variance analysis on the Improved Systems I&T Process (Chapter III) in order to examine the process elements and their interrelationships. ANOVA parameters are called "factors", and a range of values, referred to as "levels", for the factors with definitions, is established. Since the experiment subject is a process within the context of this report and not a material object, analysis parameters of the methodology need to be customized to better align to the subject under investigation.

5.3.1 Variance Analysis (ANOVA) Foundation

Key elements, or factors, of the Improved Systems I&T Process (Section 3.1) are: Knowledge Database, Planning, and Processes. To determine importance and dependency, an assessment of these three factors provides rationale for completing the following analysis; in context of this report, of highest importance are the Processes. The Knowledge Database factor is an input and output vehicle for both Planning and Processes factors, thereby holding the next highest importance. Because of its minor dependency and association, Planning, an

output of the Processes, holds the lowest factor importance. The system's factors, by their inherent definition, can have one of two values or "levels" that can be described by several combinations of the same binary pattern: good or bad, used or misused, complete or incomplete, helpful or harmful, valued or ignored.

Therefore, the system under analysis is defined as a two-level Orthogonal Array (OA) with three factors, as just described. The data point characterization of a two-level OA, is 2^k total points, where k is the number of factors, which in this case equates to 2^3 or 8 data points. The data points define the experiment as L_8, depicted in a linear graph in Figure 17 [Ertas & Jones, 1996].

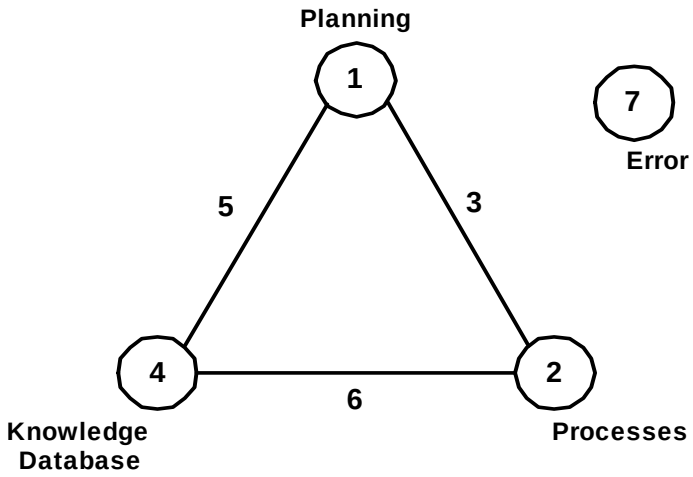


Figure 17 - Process Improvement Linear Graph

Graph lines connecting the factors represent an interaction between each of the two factor combinations; Error accumulation, the final system element, is considered to have the smallest effect among the factors.

5.3.2 ANOVA Development

A binary Level of Confidence (LOC) of the factors replaces a possible range-of-values for the analysis. Factors having a high LOC are assigned a value of "1", indicating that it is well thought out, completely defined, and correctly utilized. Factor values of "2" represent a low LOC signaling that it is incomplete, incorrect, or not utilized properly. Equating the levels to the scale of percentages, a high LOC falls in the 70% range, a low level is in the 30% range. The low percentage would translate to a 30% chance that the factor or interaction is useful, and a 70% chance that something is wrong. The error LOC representation reflects the system LOC; an error confidence level assignment of "1" indicates a high LOC in the system, but a low occurrence of error. Conversely, an error assignment of "2" indicates a low LOC in the system with a higher occurrence of error.

The three system interactions are also evaluated using LOC values. The interaction between the Process and Knowledge Database factors is defined as the most important of the three interactions due to the focus of the system, which is the incorporation of a Knowledge System into the Systems I&T Process. In order to be effective, the Knowledge System must be used for retrieval and deposit; a low LOC could indicate, for example, that the Knowledge System was not properly used within the Process, or that the database was not maintained.

The interaction priority between Planning and Processes is impacted by the fact that plans are an output of the process, and the Processes individually ranked as the most important factor. Planning outputs (Section 3.6) include: test plans, schedules and specifications which are noted as outputs from several steps in the process flow. A low LOC in the Planning/Process interaction may indicate that: the process does not completely

delineate the plans, or the Plans fail to reflect the process requirements. For the ensuing analysis, the Planning/Process interaction holds the second highest priority of the interactions.

The final factor interaction to consider is between Planning and Knowledge Database. In this scenario, a high interaction LOC relates to the Planning being properly archived and maintained; a low LOC may indicate that there is little or no data exchange. The Planning/Knowledge Database interaction holds the lowest level of importance of the three interactions.

The experiment's results, or "observations", are the final parameter necessary for the ANOVA. Since there are no measurable results as found in the classical definition of an experiment, observations of input groups have been replaced with, for the purpose of defining this Process, an evaluation termed "Final Rating". By first assigning each factor and interaction with a ranked value representing the seven System elements, the Final Rating of each input group was calculated. Using the above LOC discussions, each element is rated in importance from highest (7) to lowest (1), as in the table below.

Table 1 - Ratings of Elements in the Process Improvement System

Element	X1 Planning	X2 Process	X1/X2 Planning/ Process	X3 Database	X1/X3 Planning/ Database	X2/X3 Process/ Database	Error
Rating	5	7	3	6	2	4	1

"Final Rating" (Y) was calculated for each group of element inputs by using the following formula:

$$Y_j = 100 - \sum_{i=1}^{k} ConfidenceLevel_{i,j} * Rating_i \qquad \text{(Eq. 1)}$$

where: "i" is the element (column) counter,
"j" is each group (row) of inputs,
"k" is the number of elements (seven),
"ConfidenceLevel" is the assigned element LOC,
"Rating" is the evaluated rank from Table 1

To build the L_8 OA: "Groups" are created from all possible combinations of LOC values for each factor, totaling eight combinations of two levels for three factors; interaction LOC values are then varied; and "Final Ratings" are calculated for each group using Eq.1. Column elements are labeled to match the linear graph depiction of the system (Figure 17), in the resultant L_8 OA summarized in Table 2.

Table 2 - L_8 OA for I&T Process Improvement System

		ELEMENTS							
		X1	X2	X1X2	X3	X1X3	X2X3	Err	Y
		Plan 1	Process 2	3	DB 4	5	6	7	Final Rating
GROUPS	1	1	1	1	1	1	1	1	72
	2	1	1	1	2	2	2	2	59
	3	1	2	2	1	1	2	2	57
	4	1	2	2	2	2	1	1	54
	5	2	1	2	1	2	1	2	61
	6	2	1	2	2	1	2	1	54
	7	2	2	1	1	2	2	1	54
	8	2	2	1	2	1	1	2	53

To understand the applicability of the L_8 table, refer to row 3: the Planning element (X1) is assigned high LOC, the Processes element (X2) is assigned low LOC and the Database (X3) is assigned high LOC. The interaction between Planning and Processes (X1/X2) is assigned low LOC, indicating a problem such as poor information flow. The interaction between Planning and Database (X1/X3) factors is assigned a high LOC possibly indicating smooth and complete data flow. The low LOC assigned to the Processes/Database interaction (X2/X3) possibly translates as a problem, such as not having available process information in the database. The remaining groups can be similarly analyzed. "Final Ratings" are calculation results, using Eq. 1, given the LOC input group assignments.

Of special note is the first group in the analysis, which has all high confidence level ratings, yet has a Final Rating in the seventies. In this System Analysis, as in reality, there is no such thing as a "perfect" group. Every group will still have Errors, and all the "1's" for the input group mean statistically those elements are within the 70% level of confidence.

5.3.3 ANOVA of the Process

In the Process analysis, input groups are composed of combinations of LOC values for each of the factors and interactions. To ascertain the desired end result, "percentage of contribution", for each of the System's factors and interactions, Analysis of Variance (ANOVA) is calculated via an algorithm executed on the L_8 OA. The results are extrapolated to determine element importance in the particular system under analysis.

The algorithm's percentage of contribution, also termed Variance Ratio, is calculated from the statistical "F" test equation, based on the variability between and within groups. Variability values for the F ratio are obtained using the Mean Squares of each group and the

error. Calculations using the Sum of Squares for each group and error, and degrees of freedom, determine the Mean Squares. To calculate the Sum of Squares, incorporate the Main Effect of each group and error with the Total Sum of Squares calculation for that population. Finally, the Main Effect is calculated using the Final Ratings in the L_8 OA as shown in Table 2. Analysis of the results is completed through evaluation of variation in the data.

5.3.3.1 ANOVA Algorithm

The equations that comprise the resultant algorithm are found in *The Design Engineering Process* [Ertas & Jones, 1996, Chapter 7.9]. The ANOVA algorithm is initiated by calculating the Main Effect (MEF) averaging the Final Ratings of the groups and error, as follows:

$$MEF = \left(\frac{\sum (RatingsAtHighLevel) - \sum (RatingsAtLowLevel)}{NumberOfRatings} \right) \qquad \text{(Eq. 2)}$$

Referring to the L_8 OA in Table 2, each column of confidence levels is cross referenced with corresponding row labels to obtain the Ratings. Low level of confidence element's Final Rating values are added together and subtracted from high level of confidence Ratings values, and the mean is then calculated. For the Planning Factor (X1), the MEF equation is constructed as follows substituting "Y" for Final Ratings:

$$MEF_{X1} = \frac{(Y_1 + Y_2 + Y_3 + Y_4) - (Y_5 + Y_6 + Y_7 + Y_8)}{4} \qquad \text{(Eq. 3)}$$

The high level of confidence values for X1 are seen in rows 1, 2, 3, and 4 of Table 2 and correspond to "Y" or Ratings values, which are added together. Low level confidence values are in rows 5, 6, 7, and 8, and again the corresponding "Y" or Ratings values are

summed. Finally, the low level of confidence sum is subtracted from the high values sum, and the result is averaged. This process is similarly applied to each of the other factors, interactions, and error.

Next, the Total Sum of Squares (SS_T) variation is calculated using the Final Ratings in the following equation:

$$SS_T = \sum Y_i^2 - \frac{T^2}{N} \quad \text{(Eq. 4)}$$

where: N is the Total number of observations,
T is the sum of all observations,
Y_i are each of the Ratings

To determine the variability in-between the factors and interactions, the Sum of Squares (SS_i) is calculated using the MEF in the following equation:

$$SS_i = 2(MEF_i)^2 \quad \text{(Eq. 5)}$$

where: "i" is each factor and interaction

The Sum of Squares for the error (SS_E) due to extraneous variables or random variation is calculated using:

$$SS_E = SS_T - \sum_{i=1}^{k} SS_i \quad \text{(Eq. 6)}$$

where: "i" is each factor and interaction,
"k" is the number of factors and interactions,
SS_T is the Sums of Squares Total

The Degrees of Freedom (v_i) for each of the system's elements is determined as follows.

$$\upsilon_i = NumberOfLevels - 1 \quad \text{(Eq. 7)}$$

where: "i" is each factor or interaction,
"NumberOfLevels" = high & low in the subject system (2)

Degrees of freedom for the error (v_{error}) calculation, is as follows:

$$\upsilon_{error} = (N-1) - \sum_{i=1}^{k} \upsilon_i \qquad \text{(Eq. 8)}$$

where: "i" is each factor or interaction up to "k",
N is total number of data points (2^k: 2^3=8)

The Mean Squares (MS_i) calculation determines variance *between* the means in the groups for the factors and interactions, and is calculated:

$$MS_i = \frac{SS_i}{\nu_i} \qquad \text{(Eq. 9)}$$

where: SS_i are the Sum of Squares value for each factor and interaction,
v_i is the degree of freedom for the factor or interaction

Mean Squares (MS_e) determine the variance *within* the group or error is calculated:

$$MS_e = \frac{SS_e}{\upsilon_{error}} \qquad \text{(Eq. 10)}$$

where: SS_i are the Sum of Squares value for each factor and interaction,
v_{error} is the degree of freedom for the error

The "F" test can indicate the presence of a significant difference in the estimates of the Mean Squares, when there is a large variation, thereby demonstrating a significant contribution of an element in the system. The F-test (F_i) is calculated:

$$F_i = \frac{MS_i}{MS_E} \qquad \text{(Eq. 11)}$$

Finally, the Percent Contribution of each factor (PC_i), in the total system variance, is calculated, providing clarification of the F test results:

$$PC_i = \frac{100(MS_i - MS_E)}{SS_T} \qquad \text{(Eq. 12)}$$

and the Percent Contribution to error (PC_E) result:

$$PC_E = 100 - \left(\sum_{i=1}^{k} PC_i \right) \qquad \text{(Eq. 13)}$$

5.3.3.2 ANOVA Summary

The analysis of variance elements for the L_8 OA in Table 2 are inputs to the algorithm and the results are presented in the ANOVA Summary in Table 3.

Table 3 - ANOVA Summary Table for System I&T Process Improvement

Source	Degrees of Freedom (v)	Sum of Squares (SS_i)	Mean Squares (MS_i)	*F* Test	% of Variance
Plan	1	50	50	25	17.1
Process	1	98	98	49	35
Database	1	72	72	36	25.7
Plan/Process	1	18	18	9	6.4
Plan/Database	1	8	8	4	2.9
Proc/Database	1	32	32	16	11.4
Error	1	2	2		1.4

Given the three Process elements under discussion, Percent of Variance and “F” test results both demonstrate that the Processes have the highest effect on the system. The more significant results in this analysis are for the Interactions, highlighting the importance of the interaction between the Processes and the Database. The results substantiate the failure of the null hypothesis; there is, in fact, a strong contribution to the system of the interaction between Processes and the Knowledge Database.

5.4 Reliability Analysis

After System Variance Analysis, demonstrating the importance of the three main elements and their relationships in the highest abstraction of the Process, the next logical step is dissection of the system. A Reliability Analysis of the system’s functions determines

component reliability and expected failure costs of the system. The Functional Reliability algorithm calculates failure of both dependent and independent component relationships. The foundation for such an analysis is presented in Trewn & Yang's "*A Treatise on System Reliability and Design Complexity*" [2000] and Ertas' "*Decision Making Processes*" [2002] lecture.

First, the System is further described and delineated adding definition to the main elements of the system. The Axiomatic Design process provides a known and respected methodology for system characterizations; wherein the methodology involves domain mapping to show needs, requirements, and design relationships. A relatively recent methodology called Multi-Level Hierarchical modeling [Hubka & Eder, 1988] presents a practical next-step viewpoint. Mathematical design matrices are generated from the modeling, representing the relationships in the design, and providing the necessary framework for Analysis.

5.4.1 Axiomatic Design Domain Descriptions

In the Axiomatic Design framework, the problem is divided into four domains: Customer, Functional, Physical, and Process. Within each domain are associated design elements: Customer Needs, Functional Requirements, Design Parameters, and Process Variables. An adaptation of the Axiomatic Design process hierarchy diagram [Tate, 2002] is included in Figure 18. Orange lines reflect the mapping that occurs during the design process as domain elements are mapped to the next domain's elements. The designer develops the next domain's elements and then returns to the previous domain, represented by

the “zigzagging” blue lines, thus insuring that the elements are consistent. The design process is reiterative and continues until the design is finalized.

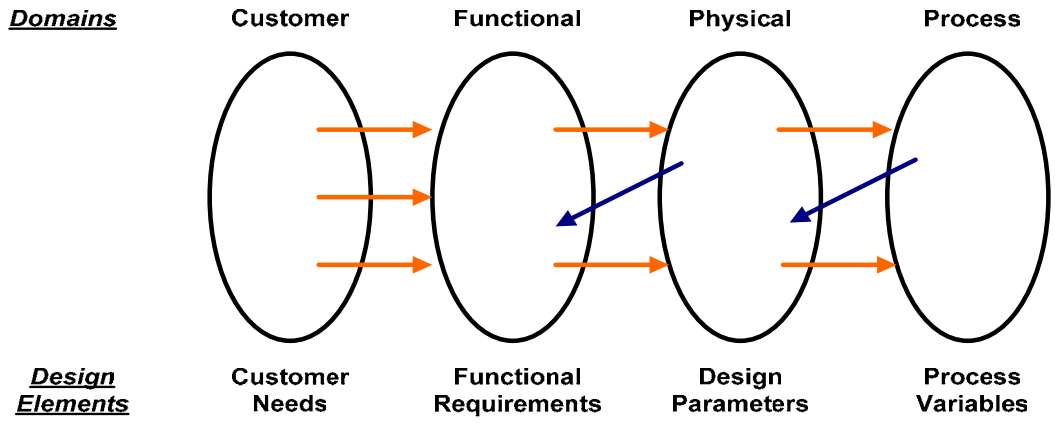


Figure 18 - Axiomatic Design Hierarchy Mapping

5.4.1.1 Customer Needs (CNs)

Using the Axiomatic Design model in Figure 18, the improved Systems I&T Process as described in section 3.5 becomes the analysis subject. Beginning with the Customer Domain, the Process is characterized and described in terms of Customer Needs. In this Process, Customer Domain is two-fold: program or administrative management; and end-user, Integration & Test Lead Engineer and engineering team members. The primary management Customer Need is a process providing for high-level progress tracking to enhance risk reduction, thereby saving time and money. The engineer’s Customer Need is an easy to implement and use process of assistance, rather than interference with daily tasks.

5.4.1.2 Functional Requirements (FRs)

Inputs and outputs of the Systems I&T Process, delineated in sections 3.5 and 3.6, are used to deconstruct Customer Needs and define the FRs. The sections describing the Process identify the following system functions: Test Sequencing, Event Scheduling, Lessons Learned Database, Processes and the need for Process Revision, and Failure Handling. The

Customer Domain maps into the following Functional Requirements, labeled for future reference.

FR1: The test sequence should be fully defined at the program level

FR2: A step-by-step timeline of events is needed for I&T activities

FR3: A Lessons Learned Database available for minor and major event entries

FR4: Existing System I&T sub-processes will be revised reflecting modification for Lessons Learned addition

FR5: Define System Integration and Test sub-processes for program activities

FR6: Course of action for test failures

5.4.1.3 Design Parameters (DPs)

Design Parameters (DPs) define boundaries and expectations for FRs. Each FR is analyzed for completeness, and DPs are developed to respond to and cover the needs of objects in the Functional Domain. For example, FR3 calls for an available Lessons Learned Database, and is defined more completely by DP3, DP4, and DP5, describing database design needs. The DPs are sequentially numbered for reference purposes.

DP1: Test planning should encompass all tests from start-up time to final test

DP2: A testing timeline will be tracked, starting at program start-up and ending at delivery

DP3: A daily Lessons Learned Database will handle infinite entries and will be updated as tests and activities are performed

DP4: Entries into a major event Lessons Learned database will include design decisions, failure resolutions, hardware studies and Lessons Learned

DP5: Entries into the major database will be reviewed by a panel before inclusion

DP6: Include review/checklist item for database inclusion to existing processes

DP7: The Integration and Test process will include all testing levels from unit through system and will outline preferred flows

DP8: A failure flow process will address generic steps followed to correct and document test failures

5.4.1.4 Process Variables (PVs)

Process Variables (PVs) implement the Design Parameters (DPs) which in turn satisfy Functional Requirements (FRs). The focus when designing PVs is a succinct description of the PV while still addressing needs of each DP. Another PV design goal is the development of a PV combining one or more DPs, thereby simplifying the overall system design. Objects in the Process Domain of the system under analysis prove to be sub-processes within the overall Systems I&T Process.

PV1: Program Test Plan

PV2: I&T Schedule

PV3: Knowledge Database

PV4: Current System I&T Sub-Process Modification

PV5: System Level Test Sub-Process

PV6: Failure Sub-Process

5.4.2 Domain Mapping

Domain mapping is a critical step in the design process, providing a framework for the analysis, capturing the designer's perception of relationships between elements. Using domain elements as outlined in 5.4.1, a domain map is developed and depicted in Figure 19.

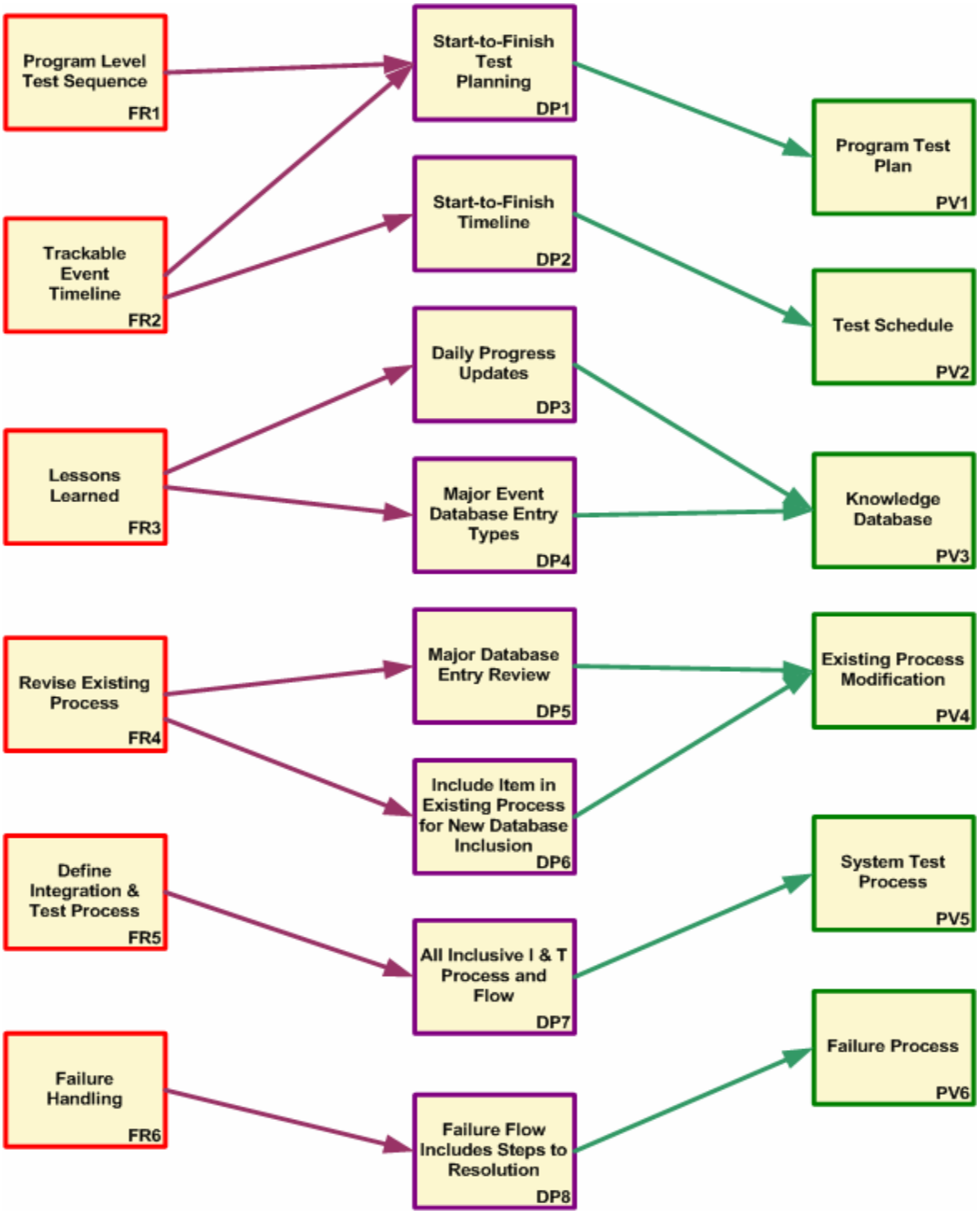


Figure 19 - Project Domain Mapping

This Domain Map flows from Functional Requirements (FRs) to Design Parameters (DPs) to Process Variables (PVs) for the improved Systems I&T Process. Intentional color

coding traces the design of the Process through Analysis. To translate the domain map, follow the arrows. For example, DP1 is “Start to Finish Test Planning” which maps to FR1, “Program Level Test Sequence” and to FR2, “Trackable Event Timeline”. Both FR elements relate to planning, and the DP, in the form of a duration expectation, adds a boundary to the two functional elements. The PV that satisfies the DP is “Program Test Plan”. The remaining mapping is similarly analyzed.

5.4.3 Hierarchical Modeling

Multi Level Hierarchical (MLH) modeling is a conceptual design process [Trewn & Yang, 2000] providing a more precise characterization of the relationship between the functional requirements, and components, the lowest level structure of the system. In MLH methodology, system levels, or spaces, include: Functional, Physical, and Component. An adaptation of the MLH model in Figure 20, maps from functional space to component space in two steps [Hubka & Eder, 1988] [Ertas, 1996].

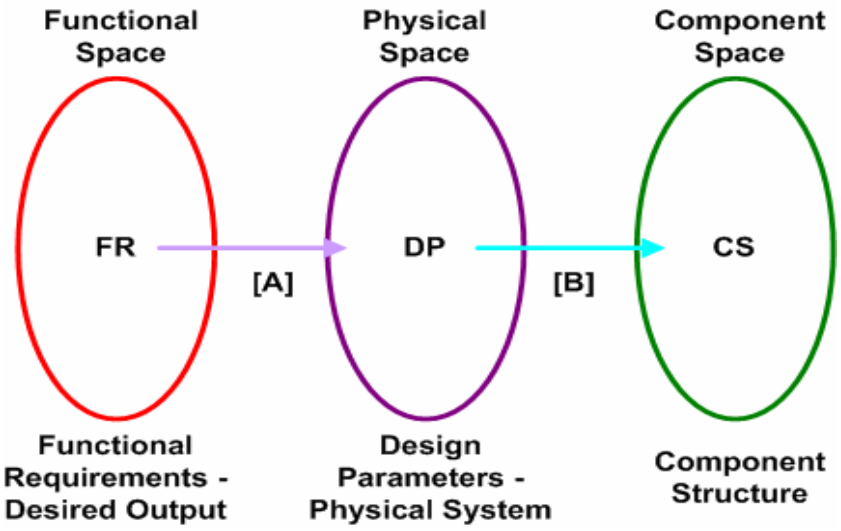


Figure 20 - Adapted High Level MLH Model

Matrix identifiers [A] and [B] represent relationships between elements, determined by mapping the domain elements. FRs mapped directly to the Cs help the system design process by graphically depicting the structure level design of the system. Matrix relations of [A] ° [B] determine a third matrix outcome that shows the relationship from the input of matrix [A] to the output of matrix [B]. The MLH model showing the FR to C mapping with a resultant relationship matrix labeled [D] is depicted in Figure 21.

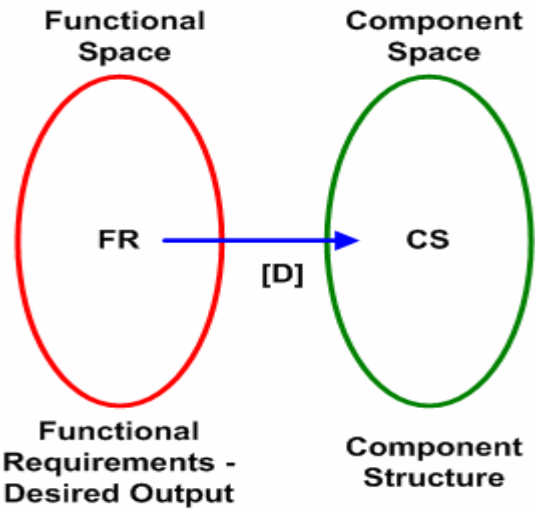


Figure 21 - MLH Model for FR to CS Mapping

Figure 22 is an Axiomatic Design diagram extrapolated from the model in Figure 21 of the Functional Requirements (FRs) to Components (Cs) mapping for the Improved I&T Process. Final versions of Domain Map diagrams are included herein, many versions were created by the iterative process of "zigzagging" through hierarchical depictions which are used to refine the Improved I&T Process' design.

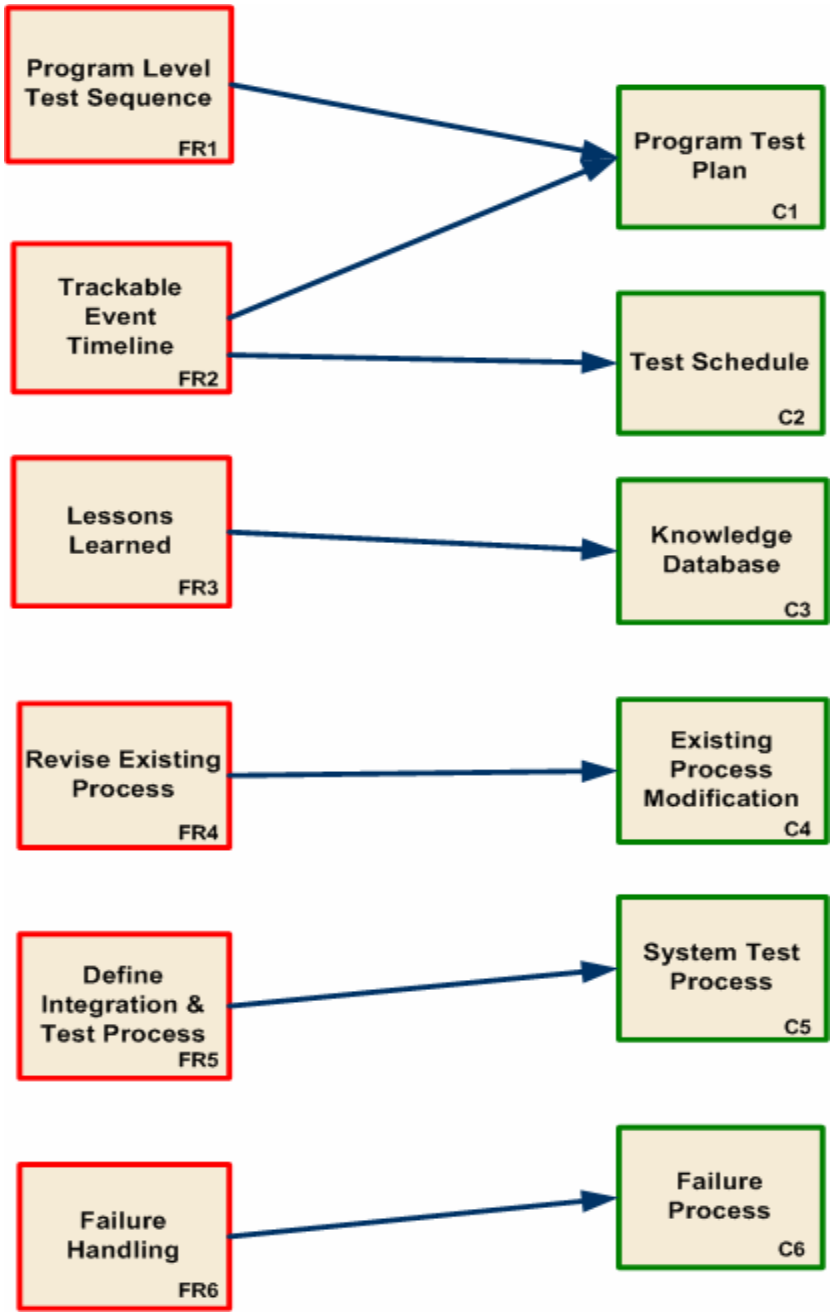


Figure 22 - Domain Space Mapping

5.4.4 Domain Matrices

The domain space relationship matrices illustrated in Figure 20 ([A] and [B]) and in Figure 21 ([D]), are applied to the Axiomatic Domain elements (5.4.1), to mathematically

represent design matrices. Matrices in this context test the Axiomatic Design principle's Axioms for the Process design. In particular, the "Independence Axiom" states that the design must "maintain the independence of the FRs", whereas the "Information Axiom" states that the design must "minimize the information content of the design" [Yang & Zhang, 2000].

Evaluation of element coupling tests the Design Matrices for independence, critical to keeping design elements from dependence on other elements. When element interdependence occurs, failure of one element would, in all likelihood, affect other dependent elements, possibly creating a domino effect resulting in a completely failed system. Developing an uncoupled design is a goal which will have a nearly diagonal line in the Design Matrix. Conversely, if a coupled design is created, the independence of the elements is too stringent, leading to a poor product.

Table 4 depicts Design Matrix [A] for the FR to DP map of the Improved I&T Process, and Table 5 depicts Design Matrix [B] for the DP to C map. Relationships from the Process's FRs are directly mapped to Cs, resulting in matrix [D] (Table 6). The matrix designs are interpreted as column elements that "satisfy" row elements.

Table 4 - FRs to DPs Design Matrix ([A])

	DP1	DP2	DP3	DP4	DP5	DP6	DP7	DP8
FR1	1							
FR2	1	1						
FR3			1	1				
FR4					1	1		
FR5							1	
FR6								1

The Project Domain Mapping (Figure 19) is the framework upon which these matrices are developed. For example, FR2 has two arrows, one connected to DP1 and another to DP2. In Table 4, the number 1 represents those relationships in the matrix. The remaining matrix elements are likewise entered for all three matrices.

Table 5 - DPs to Cs Design Matrix ([B])

	C1	C2	C3	C4	C5	C6
DP1	1					
DP2		1				
DP3			1			
DP4			1			
DP5				1		
DP6				1		
DP7					1	
DP8						1

Logically, the rationale behind Table 4 and Table 5 carries through to show the mapping from FRs directly to Cs in Table 6.

Table 6 - FRs to Cs Resultant Design Matrix ([D])

	C1	C2	C3	C4	C5	C6
FR1	1					
FR2	1	1				
FR3			1			
FR4				1		
FR5					1	
FR6						1

Dependency on one side of the diagonal line shows a triangular decoupled design, satisfying the Axioms. The Design Matrices and domain definitions are used in the following section on reliability and cost analysis.

5.4.5 Functional Reliability Analysis

There is a distinction between Functional Reliability and System Reliability, wherein the entire system is examined for operational reliability. The converse of a system's reliability is the system's failure vulnerability which can be determined through independent and dependent component failure analysis, translating into Expected Cost. Within any given system, components are the product of the functional requirements which define the system. Functional Reliability is defined as "the likelihood of successfully providing the functions that a system or component is designed to deliver" [Trewn & Yang, 2000]. In order to analyze the system, input parameters in Section 5.4.1 and 5.4.4 (functional requirements, design parameters, components, and domain matrices), deconstruct the improved Systems I&T Process elements. Functional Reliability Analysis involves probability of failure of components coupled with cost of loss and replacement of components as parameters.

A Knowledge System filled with information, provides metrics for analysis parameter inputs, however, such a Knowledge Database is currently not available with data for this type of analysis. Because of the type of system (Process) and lack of a metrics database, necessary analysis parameters must be extrapolated from experience. Solicited from a cross section of task lead engineers and managers at Raytheon Missile Systems, and external customers, responses to a questionnaire (Appendix A) had a response ratio of approximately 63%. Respondents were asked three questions relating to issues about each of the six Improved I&T Process components, when individual components have failed: impact to daily tasks, level of participation for repair, and probability of failure.

For purposes of this analysis, the Expected Cost (EC) analysis is initiated using the inverse of reliability, the probability of system component failure, as assessed by

questionnaire respondents at project start-up. Reliability then, is the percent confidence level of completion and correctness; the Probability of Failure is the probability that the component is ignored, incomplete or incorrect at the start of the program. In best case scenario, all the process's components would have a 100% confidence level, be perfect at program startup, have 0% failure, and never require updating. However, experience (which should be captured in a Knowledge Database) dictates that there will always be problems and the need to modify and adapt process components as the program progresses.

Table 7 shows a more complete description of each component with the tabulation of the failure probability from the engineering questionnaire.

Table 7 - Component Failure Analysis

C #	Definition / Rationale	Failure%
1	The *Test Plan* is a roadmap to major test steps. It should be program-wide and clearly show inputs and outputs to tests including plans and reports.	38
2	This *Schedule* should: (1) cover Systems I&T tasks at a high programmatic level, (2) relies on other inputs and is typically driven by events beyond the control of the Systems I&T team. There should be a separate more detailed schedule for the Systems I&T team to follow on a daily basis. For these reasons, the schedule will have lower reliability/confidence, resulting in a slightly higher failure probability.	41
3	A working *Database* should be readily available. Problems may arise in regards to the training and proper use of the database.	34
4	The existing S*ub-processes* can be *Modified* on-the-fly and used during the project. The assumption is that management supports and in fact requires the use of the sub-processes.	40
5	I&T *Test Sub-processes* should focus on the overall test plan for the program as well as the daily test flow, covering expectations to meet program requirements and goals. Problems arise in following these processes and flows, as schedules and costs tighten toward program completion.	39

C #	Definition / Rationale	Failure%
6	The *Failure Process* is one of the most difficult process flows to delineate, so it is typically considered at a higher level of abstraction. When is a failure hard enough to warrant a review or a database entry? If the failure is quickly/simply fixed is it worth spending the time to raise the visibility?	33

5.5 Expected Cost Analysis

When initiating the analysis, the Cost of Loss of Function (C_{FRi}) is calculated, indicating the effect that a "broken" or misinterpreted function has on the Process. "Cost", as translated by a representative cross-section of typical program product teams in the questionnaire, is a percentage of time rather than dollars. Engineering disciplines included in this questionnaire were presumed to have their own internal procedures and processes.

Table 8 - Cost of Loss of Function Summation

	System Reqts	Mech	Elec	SW	System I&T	Field Test	Program Office	Mean C_{Fri}
FR1-*Seq*	40	33	40	33	73	74	72	**52**
FR2-*Events*	52	53	43	40	69	60	82	**57**
FR3-*LL*	24	40	33	23	46	46	41	**36**
FR4-*Revise*	40	53	53	43	54	56	52	**50**
FR5-*Procs*	48	46	80	60	73	68	51	**61**
FR6-*Failure*	54	50	66	43	67	44	48	**53**

Each team's effect on Component cost analysis is considered from the perspective of the Systems I&T team. The Cost of individual function losses to each team was extracted from the engineering questionnaire from the respondent's use of historical team impact experience. Final Cost value means for each component are then calculated from the

sampled population, represented by values on a scale of “No Cost” (0%) to “Extremely High Cost” (100%).

The System Requirements (System Reqts in Table 8) team tracks program requirements and is responsible for program level documentation. Mechanical and Electrical (Mech, Elec) teams perform design, implementation and unit test of mechanical and electrical system aspects. The Software (SW) team develops and maintains all software for a given program. The System I&T team is one group responsible for overseeing final system integration and conducting program level testing. A Field Test team interfaces with the final customer to install, test, and provide training for the system upon delivery. The Program Office team provides program level administration and is the program policy decision making body. Cost of Loss of Function estimates tabulated from questionnaire output, plus the calculated means (C_{Fri}), are included in Table 8 for each FR described in section 5.4.1.2.

To better understand the focus of Table 8 and the ensuing analysis, refer to row “FR1”, defined as “The test sequence should be fully defined at the program level”. Loss of this function might be translated as a misinterpretation of the contract during system test definition when an aspect of the test sequence is vicariously omitted and unplanned. The question of how teams are potentially affected is answered by each team and the associated impact or concern regarding the “loss”. “System Reqts”, “Mech”, “Elec”, and “SW” would all be slightly impacted and probably only affected in schedule. “System I&T” has a major impact because of schedule and cost problems associated with adding a task. There would also be a larger impact to “Field Test” and “Program Office” teams due to a possible product delivery schedule adjustment. Cost of Loss may also be exemplified by examining “FR2”,

which is "… step-by-step timeline of events". There is a logical increase in Cost of Loss as time progresses along the team involvement timeline, reading the flow from left to right; obviously, the greatest Cost impact for schedule problems is on "Program Office".

Summarized in Table 9, the Cost of Component Replacement (C_{Ck}) was estimated using the same technique as in Cost of Loss estimation. As in the previous estimation, a cross-section of major teams within a project was incorporated calculating the mean cost for each component as a percentage of time. In the questionnaire, Replacement Cost is an estimation of percent of participation when having to contribute to or re-create the particular failed Systems I&T Component (C) defined in section 5.4.1.4.

Table 9 - Cost of Replacement of Components

	System Reqts	**Mech**	**Elec**	**SW**	**System I&T**	**Field Test**	**Program Office**	**Mean C_{Ck}**
C1-*Plan*	54	30	66	30	86	64	72	**57**
C2-*Sched*	44	53	63	43	77	54	76	**59**
C3-*DB*	26	40	43	23	51	58	36	**40**
C4-*Mods*	40	50	63	36	66	46	38	**48**
C5-*Tests*	46	46	63	50	75	62	36	**54**
C6-*Failure*	42	30	80	50	66	32	47	**50**

For instance, "C4" is the "Current System I&T Sub-Process Modification" component. The "System I&T" team would have the highest participation since the sub-processes are modified by that team. The remaining teams estimate their time-relative involvement when the System I&T sub-processes affect their own processes.

5.5.1 Assessing the Chaotic Process – Component Dependency

The chaotic process is examined without benefit of the key process improvement feature, the Knowledge System Database, prior to completing the improved Systems I&T Process analysis. Since the Database is essentially used as a focal point for sharing data and information within programs and organizations, a totally inter-dependent chaotic process would result without the database as depicted in the standard linear graph (Figure 23, [Ertas, 2002]). The linear graph is a reflection of the component dependency matrix created for the Expected Cost analysis in the next section.

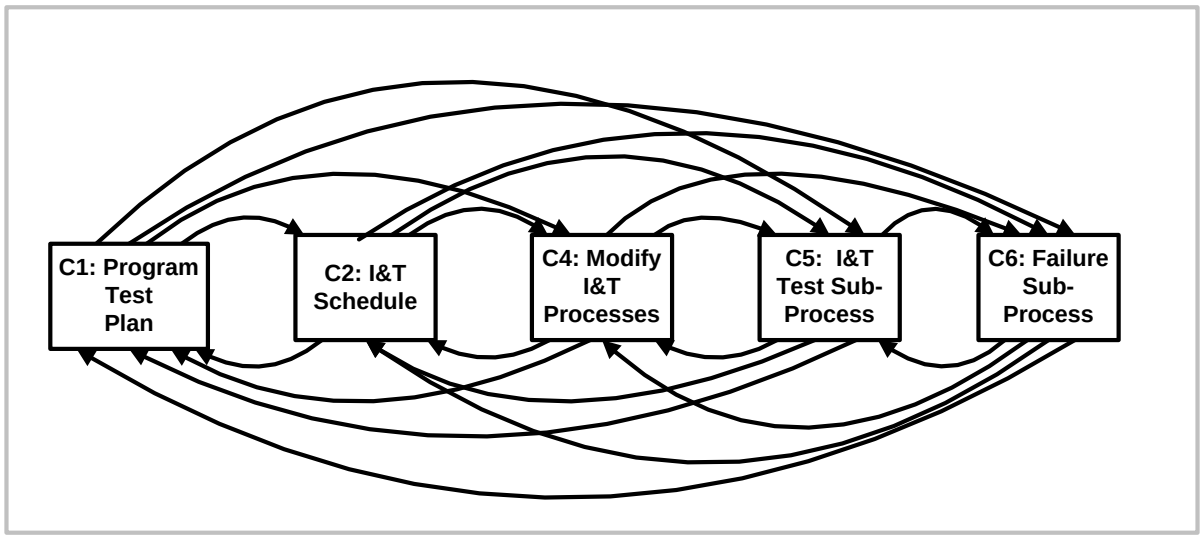


Figure 23 - Chaotic Process Dependency Linear Graph

Should one system component fail, the balance of the chaotic system has a high probability of failing functional as well as operational reliability, due to tight component coupling; this scenario typifies modern day programs and organizations that do *not* employ a Lessons Learned Database. For example, if the Failure Process is created without benefit of past experiences, relying only on other current Process components, the Process will most

certainly fail at some point. This theory applies equally to the remaining components and their inter-relationships and dependencies.

The component dependent failure block diagram for the chaotic process (Figure 24), derived from the linear graph in Figure 23, realistically illustrates the complex nature of the chaotic process, partially since the problem solving aspects of Systems I&T are not a linear process without a Knowledge System.

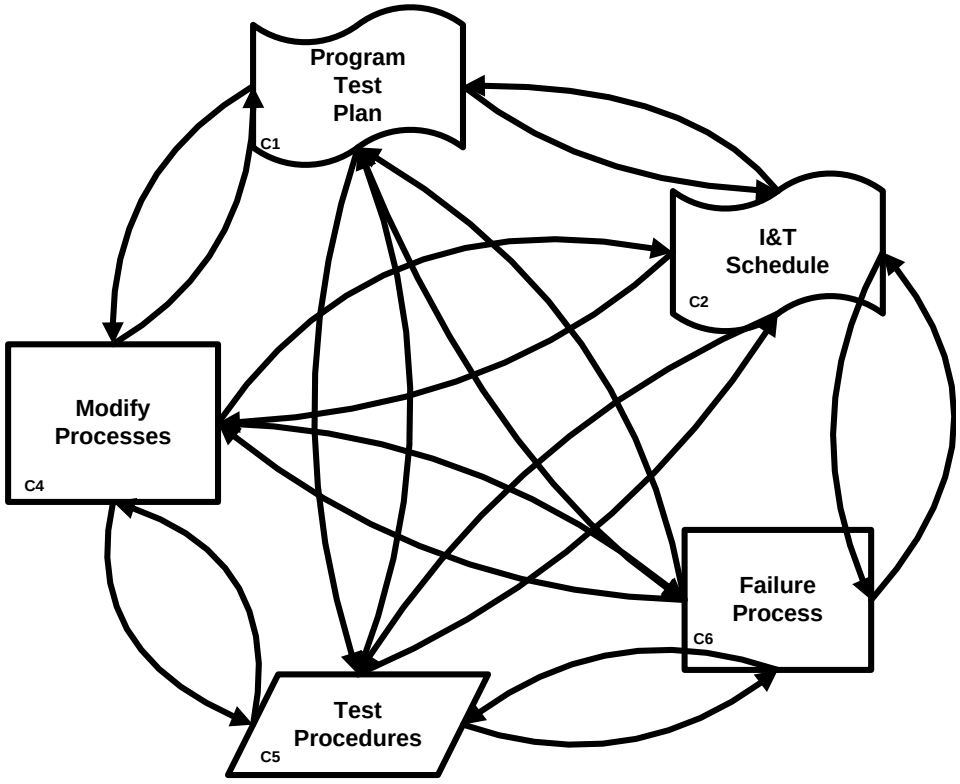


Figure 24 - Chaotic Process Interdependency Block Diagram

In this diagram, if a relationship exists that may result in similar behavior between two components, it creates dependency. Component dependency is stated in terms of failure, if component "x" fails, component "y" may also fail. Specifically arrow direction indicates failure: if the component at the source fails, the component at the destination may also fail.

5.5.2 Chaotic Process Analysis

Table 10* - Systems I&T Chaotic Process Analysis Design Parameters [*format adapted from Trewn & Yang, 2000 and Ertas 2002], organizes and summarizes data for the Expected Cost (EC) of Failure Analysis, and includes pertinent information from sections 5.4.4 and 5.5. Failure Probability (Table 7) and Cost of Replacement (Table 9) component data is summarized in the lower left quadrant. The FR to Component relationship Matrix "D" (Table 6) is replicated in the upper right quadrant.

The Parameter Table also incorporates a matrix of component to component dependencies in the lower right quadrant, reflecting probabilities of dependency failures, with four values defined for the matrix: low (25%), moderate (50%), high (75%), and total (100%). It is evident from Figure 23 and Figure 24, that the Chaotic System component dependency matrix will include some degree of dependence between every component. Implicitly, each component is dependent upon itself, so ones (100%) are entered on the diagonal to represent each of those relationships. Additionally, the matrix exhibits dependencies between every component at a moderate level of dependency (50%), and a null value (0) for the Database component.

Table 10* - Systems I&T Chaotic Process Analysis Design Parameters

FRs	Failure Probability (P_{FRi})	Cost of Loss of Function (C_{FRi})
FR1-*Seq*	0.38	52
FR2-*Events*	0.63	57
FR3-*LL*	0.34	36
FR4-*Revise*	0.4	50
FR5-*Procs*	0.39	61
FR6-*Failure*	0.33	53

Component vs. FR Relationship Matrix (D)

	C1	C2	C3	C4	C5	C6
FR1	1					
FR2	1	1				
FR3			1			
FR4				1		
FR5					1	
FR6						1

Components	Failure Probability (p_k)	Cost of Replacement of Component (C_{Ck})
C1-*Plan*	0.38	57
C2-*Sched*	0.41	59
C3-*DB*	0.34	40
C4-*Mods*	0.4	48
C5-*Tests*	0.39	54
C6-*Failure*	0.33	50

Component Dependency Matrix

	C1	C2	C3	C4	C5	C6
C1	1	0.5	0	0.5	0.5	0.5
C2	0.5	1	0	0.5	0.5	0.5
C3	0	0	0	0	0	0
C4	0.5	0.5	0	1	0.5	0.5
C5	0.5	0.5	0	0.5	1	0.5
C6	0.5	0.5	0	0.5	0.5	1

Functional Requirements (FRs) data, Failure Probability and Cost of Loss of Function (Table 8), are included in the table's upper left quadrant. To determine Failure Probability of FRs (P_{FRi}), Reliability of the Functional Requirements (R_{FRi}) is first calculated using Matrix "D" and FR probability of failure (p_k) [Trewn & Yang, 2000 and Ertas, 2002]:

$$R_{FR_i} = \prod_{k=1}^{n} (1 - p_k)^{d_{ik}} \qquad \text{(Eq. 14)}$$

where: "i" is the FR being calculated,
"d" the Matrix D entry,
"n" is the total number of FRs,
p_k is the probability of failure from Table 7

Failure Probability of each FR (P_{FRi}) is next calculated from Reliability values:

$$P_{FR_i} = 1 - R_{FR_i} \qquad \text{(Eq. 15)}$$

5.5.3 Assessing the Improved Process – Component Dependency

With Improved Systems I&T Process component dependency evaluation, the Knowledge System component must be re-inserted into the Process for EC Failure Analysis and comparison. As in the Chaotic Process Analysis, component dependency failures are analyzed using logical deduction. Determined through examination of each component, dependencies with every other component are evaluated of how failure may affect other system components. Interrelationships inherent in the Process described in Chapter III translate into component dependencies. As is evident from the block diagram in Figure 25, inserting the Knowledge Database into the core of the Process greatly simplifies the dependencies.

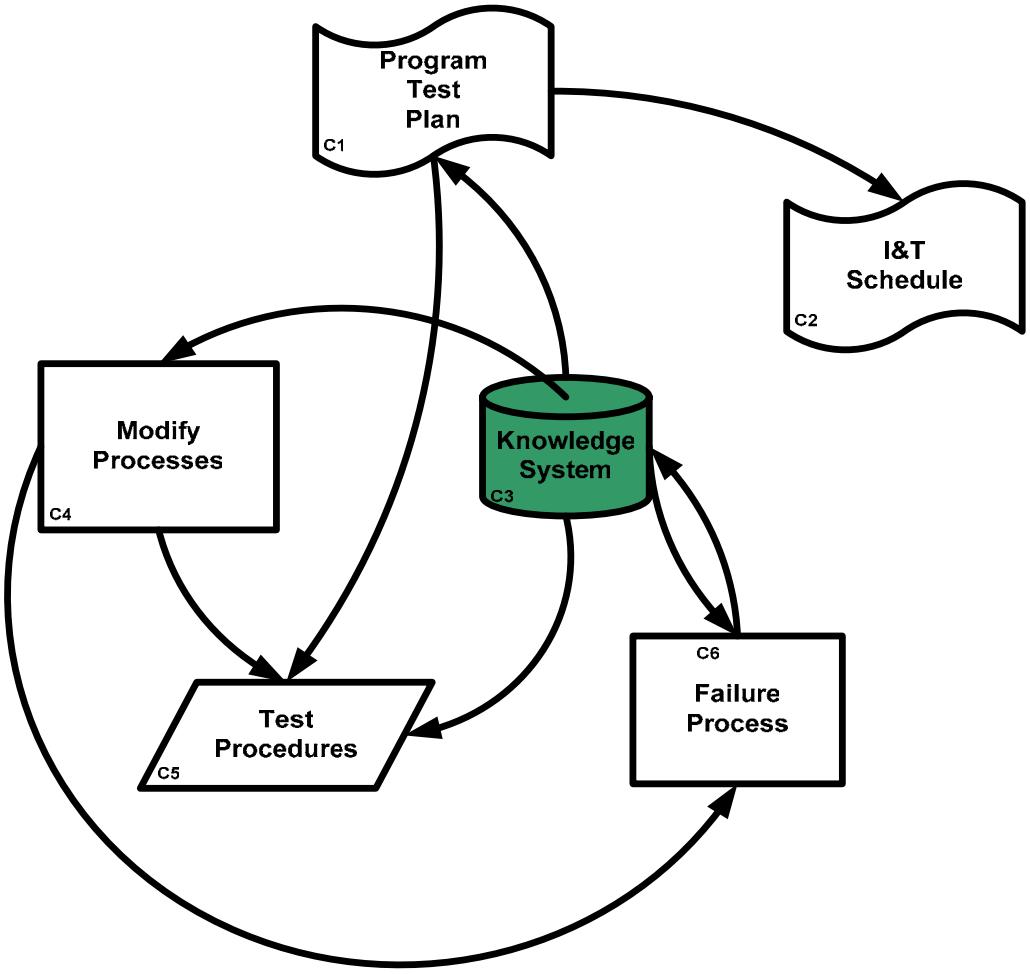


Figure 25 - Improved Systems I&T Process Block Diagram

Interpreting the above figure, if Program Test Plan (C1) fails, Systems I&T Schedule (C2) may also fail. Program Test Plan failure could occur if a test is erroneously overlooked during high level planning, which in turn would have a good probability of negatively affecting the Schedule. Another scenario might be a Failure Process (C6) error, causing a possible failure of the Knowledge System (C3). This error may cause important Failure Process entries to be omitted from the Database, the history and any available information relevant to that particular event are lost.

The standard linear graph (Figure 26, [Ertas, 2002]) of component dependencies echoes the Block Diagram, but the primary importance of the Knowledge System is not as easily visualized, nor does it have the central focus necessary for clarity of the Process.

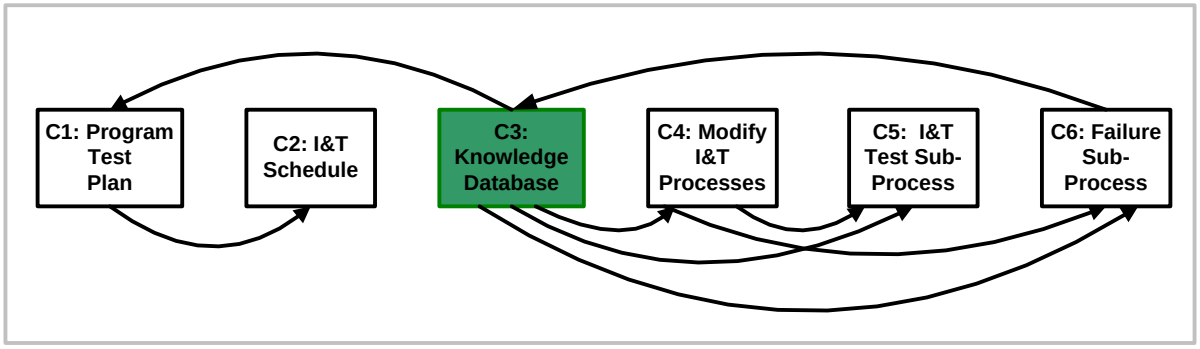


Figure 26 – Improved Process Component Linear Graph

5.5.4 Improved Process Analysis

Table 11* - System I&T Process Analysis Design Parameters [*format adapted from Trewn & Yang, 2000 and Ertas 2002], summarizes the analysis information for the Improved Process. As in the Chaotic Process Parameter Table, pertinent (identical) data from section 5.4.4 and 5.5 is summarized. Within this Parameter Table, a matrix of component to component dependencies in the lower right quadrant reflects probabilities of dependency failures in the Improved Process of Figure 25. In the matrix, probability entries of low (25%), to moderate (50%), high (75%), and total (100%) are based on logical deduction of dependency assessment, represented by directional arrows in the figure. As previously stated, component dependency is expressed in terms of failure, if the source component fails, the destination component may also fail.

Table 11* - System I&T Process Analysis Design Parameters

FRs	Failure Probability (P_{FRi})	Cost of Loss of Function (C_{FRi})
FR1-*Seq*	0.38	52
FR2-*Events*	0.63	57
FR3-*LL*	0.34	36
FR4-*Revise*	0.4	50
FR5-*Procs*	0.39	61
FR6-*Failure*	0.33	53

Component vs. FR Relationship Matrix (D)

	C1	C2	C3	C4	C5	C6
FR1	1					
FR2	1	1				
FR3			1			
FR4				1		
FR5					1	
FR6						1

Components	Failure Probability (p_k)	Cost of Replacement of Component (C_{Ck})
C1-*Plan*	0.38	57
C2-*Sched*	0.41	59
C3-*DB*	0.34	40
C4-*Mods*	0.4	48
C5-*Tests*	0.39	54
C6-*Failure*	0.33	50

Component Dependency Matrix

	C1	C2	C3	C4	C5	C6
C1	1		0.25			
C2	0.25	1				
C3			1			0.75
C4			0.75	1		
C5	0.25		0.25	0.25	1	
C6			0.75	0.5		1

The completed matrix is translated so that a row component *relies on* a column component, consequently implying that if a column component *fails*, it may cause row component failure. The remainder of the Component Dependency Matrix is similarly interpreted relative to the Process, not to any particular program. Looking at row C2, I&T Schedule component, and column C1, Program Test Plan component, because the schedule *relies on* plan elements to complete the scheduling activity, there is some dependency (0.25). Extrapolating from the component dependency matrix of row/column C6/C3, the dependency of Failure Process/Knowledge Database, is assessed as a 0.75 dependency value: if the Knowledge Database *fails* via errors or omissions, the Failure Process has a high probability of being adversely effected due to lack of information. Continuing in similar fashion, the matrix is completed for the Expected Cost Analysis.

5.5.5 Expected Cost of Failure Analysis

The final step of the Functional Reliability algorithm is to calculate the Expected Cost (EC) of component failure for independent and dependent relationships; completing the analysis and comparison of the Chaotic and Improved Processes. Inputs to the EC calculation are extracted from both component dependency matrices and estimated failure probabilities (Chaotic Process in Table 10, and Improved Process in Table 11). The expected costs are calculated with the following equations [Trewn & Yang, 2000and Ertas, 2002].

For the Independent case, the cost of each variable is considered autonomous of other variables in the system using:

$$E[C_{Ck}] = p_k * \left(C_{Ck} + \sum d_{ik} C_{FRi}\right) \qquad \text{(Eq. 16)}$$

where: "k" is the component,
"i" is the function,

"d" is the value from Design Matrix [D],
C_{Ck} is the Cost of Component Replacement,
p_k is the Failure Probability of the Component,
C_{FRi} is the Cost of Loss of the Function

For the Dependent case, the equation for Expected Cost of failure includes a term for the component dependency matrix entries:

$$E[C_{Ck}] = \sum \left[p_{\frac{k}{i}} * p_j \right] * \left(C_{Ck} + \sum d_{ik} C_{FRi} \right) \qquad \text{(Eq. 17)}$$

where additionally: "k" is the counter for the component,
$p_{k|i}$ is the component dependency probability value

Results of the analyses of the Independent and Dependent cases for both the Chaotic and Improved Process are presented in Table 12.

Table 12 - Expected Cost Analysis Summary

	$E[C_{C1}]$ Test Plan	$E[C_{C2}]$ Schedule	$E[C_{C3}]$ Database	$E[C_{C4}]$ Mod Procs	$E[C_{C5}]$ Test Procs	$E[C_{C6}]$ Failure Proc
Independent Failure	63.08	47.15	25.50	43.20	44.46	33.66
Chaotic Process Dependent Failure	127.53	198.73		116.62	109.61	117.30
Improved Process Dependent Failure	50.69	84.34	44.06	64.19	76.38	80.07

Considering the components independently, the C_{C1} Test Plan component is the most costly upon failure. High cost of Test Plan failure can be due to the fundamental reliance of the Systems I&T Schedule on the Test Plan. The dependency analysis shows that for both the Chaotic and Improved Process, the highest associated component failure cost is C_{C2}, the Systems

I&T Schedule. Logically, costly schedule failures would be an accurate result, given that in real practice, no matter which component breaks in the system, the schedule will be the most affected component. In both the Independent and Improved Process Dependent Failure analyses, C_{C3}, the Knowledge System, is the least costly component upon failure.

The most significant aspect of the Analysis is the noted improvement in every component from the Chaotic to the Improved Process through incorporation of a Knowledge System which proves to be the least costly of all component failures. The EC Failure Analysis is based on experiences of task lead engineers and managers who clearly rate the database with the lowest cost of loss and replacement, and when coupled with probability of failure, provides the basis for results in the above table. From a practical viewpoint alone, the Analysis results provide welcomed confirmation of the benefit, importance, and necessity of incorporating a Knowledge System into the organization's processes.

CHAPTER VI

CONCLUSION

Creativity is allowing yourself to make mistakes. Art is knowing which ones to keep.

- Scott Adams, 'The Dilbert Principle'

The lack of practical, well defined, processes for Systems Level Integration of E&MD programs, covering the totality of the integrators job, and the need for Process improvement, has been discussed. The Chaotic I&T Process, where process steps are totally interdependent, creates a nightmare for information sharing and timely program execution. The common misunderstanding of the Systems I&T Process is that is begins at product assembly. Because of this and the reality of daily engineering tasks being typically chaotic, as if to be constantly fighting fires, the traditional I&T process flows seem inadequate. The information gained while conducting Systems I&T tasks, is hopefully tracked in a notebook somewhere; the reality is that Knowledge is usually lost. These deficiencies inspired the creation of the Pathway to Systems I&T Model (Figure 4) that incorporates a Knowledge System or Lessons Learned Database, herein named the “Knowledge Gained Database”. In this Knowledge Gained Database, information is captured and readily available from program conception to product delivery. Headaches can be lessened through adherence to this proposed standard, as defined by the detailed flow diagrams presented in this report.

The Process Improvement cycle begins once management champions the Lessons Learned database incorporation, and requires their engineers to use the methodology’s sub-processes and available tools as presented. Using cost as a basis, proving the benefit of the

Improved Systems I&T Process, is the key to obtaining management buy-in. This is accomplished through statistical analysis that: (1) examines process component variability and interrelationship importance through ANOVA and, (2) evaluates Functional Reliability through system component dependency using Effective Cost of Failure Analysis. Results of the ANOVA show statistical significance of the key elements (Planning, Processes, and Knowledge Database) of the System and their interrelationships. Functional Reliability results clearly show substantial improvement from a Chaotic to an Improved Process. These two results provide the proof of benefit and validation for Knowledge System incorporation.

Systems I&T Process definition and analysis provides material for a discussion of the underlying issue of Knowledge System incorporation, what is it, and what does it cost. Every engineer intuitively knows that un-captured Knowledge is partially responsible for program cost and schedule overruns, since tasks end up being repeated. So the real question is what it costs *not* to have a Knowledge System.

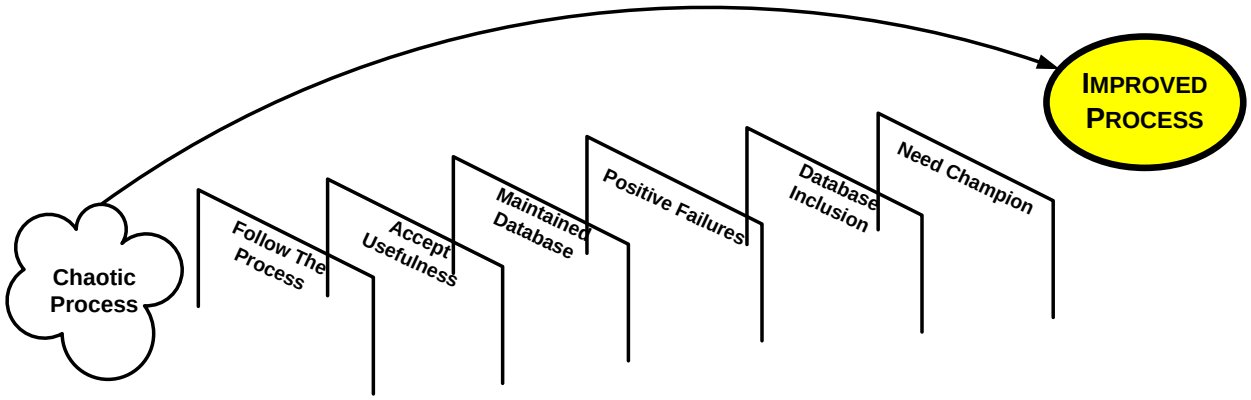
6.1 LEARNED LESSONS OF LESSONS LEARNED

This report would be incomplete without some commentary on Learned Lessons of the Lessons Learned for Knowledge System Incorporation: the lessons that sparked the idea for this report. The first and foremost lesson is following the Process, which can save untold amounts of time and repeated effort. During a very recent Integration activity, a team of engineers was checking rework of an electronics board. After dissipation of a cloud of smoke from a powered connection that was attempted in an awkward position, the checklist for testing the board was revisited and connections had been completed in the wrong order. A very simple process step was omitted that could have saved hours of further rework and a damaged board. This misstep was reminiscent of a line from the movie parody “The Adventures of Buckaroo Banzai”, when

Buckaroo pronounces during brain surgery, "…don't tug on that, you don't know what it's attached to". The current-day integration practices must change from "shoot from the hip" mentality to science.

Secondly is the issue of tracking information; E&MD Engineers, and in particular Integration engineers, are notorious for not writing anything down. Excuses run rampant, ranging from "I didn't have the time" to "I didn't think it was that important" to "nobody reads this stuff anyway". Development needs to be followed-up with recording and documentation of test findings and discovered anomalies.

The third major lesson is to find a "Champion", which is one of the hardest tasks to tackle. Implementation, follow-up, training, and database maintenance are all critical activities needing oversight which keeps the Process alive.



The above graphic illustrates the hurdles that must be overcome to progress from a Chaotic to an Improved Process. Many follow-up questions arise that ultimately help the implementation of the Improved Process: In what stage of development is the information most useful? How does an organization determine high enough importance to an issue to include it in

a database? Where and when during the process do employees make a search to retrieve information?

Murphy's Law is constantly in play in the fast-paced, high-stress world of E&MD Integration: if something can go wrong, it will, at the worst time. Engineers need to realize that there is no such thing as a bad integration mistake. All mistakes can be transformed into Lessons Learned for present and future tasks. The System is often delicate, high precision, highly integrated and scientifically intricate. Integrators need to understand the big picture, or be able to research the interconnections, history, and inter-system functionality to promote proper diagnosis of problems and either summon responsible engineering disciplines or recommend resolution.

There will always be problems and failures during development that must be viewed as discoveries rather than negative "mistakes". Failures can be attributed to incorrect upfront planning or design, errors in judgment, a slip of the hand, and still others result from the unexpected. Missteps, errors, oversights and omissions are unavoidable and all part of the creative process, the trick is minimizing their numbers through adherence to the Improved Process; as Scott Adams proclaims, "knowing which ones to keep" becomes the Lessons Learned.

REFERENCES

The Adventures of Buckaroo Banzai Across the 8th Dimension, W.D. Richter, director, 1984.

Aslam, Tanweer. *Integration and Test Plan for the Moderate Resolution Imaging Spectroradiometer (MODIS) Emergency Back-Up System*, 26 January, 1998. http://ltpwww.gsfc.nasa.gov/MODIS/SDST/MEBSI&T.pdf.

Ertas, A. & Jones, Jesse C., *The Engineering Decision Process*, 1996, John Wiley & Sons, Inc., New York, NY.

Ertas, A., *Decision Making Processes*, 2002, Raytheon MS Program, Engineering Department, Texas Tech University, Lubbock, Texas (Briefing Slides and Lecture Notes).

Evans, Dennis Charles, *Space Engineering Lessons Learned*, Nov 1989, http://users.erols.com/ee/sysengll.htm#4.0.

Hubka, V. and Eder, W.E., *Theory of Technical Systems*, 1998, Berlin, Springer-Verlag, 2nd Ed.

Lee, Dennis *A Lessons in Lessons Learned.* 2002, http://appl.nasa.gov/knowledge/issues/lessons_learned.html, NASA.

Macnaughton, Donald B. *Which Sums of Squares are Best In Unbalanced Analysis of Variance*? 10 May 1998, http://www.matstat.com/ss/easleaao.pdf.

The New American Heritage Dictionary of the English Language, 1976, Houghton Mifflin Company.

Phillips, Mike *CMMI V1.1 Tutorial, PowerPoint presentation.* 09 April 2002 http://www.sei.cmu.edu/cmmi/publications/euro-sepg-tutorial/, © Carnegie Mellon University.

Marshall, Jane *Reliability Enhancement methodology and Management*, 1999, (PowerPoint Presentation for TRW Aeronautical Systems).

Raytheon Missile Systems "ITLog" & DocuShare used by permission of Raytheon system administrators (D.Uhlken for ITLog and S.Hatch for DocuShare site).

Sage, Andrew P.& Lynch, Charles L. *Systems Integration And Architecting: An Overview of Principles, Practices, and Perspectives*, The Journal of the International Council on Systems Engineering, Volume 1 Number 3, pp. 176-227, John Wiley & Sons, Inc.© 1998.

Sheard, Sarah H. & Lykins, Howard & Armstrong, James R. *Overcoming Barriers to Systems Engineering Process Improvement*, 1999 © Software Productivity Consortium, NFP.

Sheinman, Oren R. *Lessons Learned, X-Ray Timing Explorer (XTE)*, 1998, http://mscweb.gsfc.nasa.gov/543web/pages/xte.html.

Trewn, Jayant & Yang, Kai, *A Treatise on System Reliability and Design Complexity*, Proceedings of ICAD2000, June 21-23, 2000)167.

Wallace, Dolores R. *Using Failure History to Improve Reliability in Information Technology*, 2000, National Institutes of Standards and Technology, Information and Technology Laboratory.

Webster's Encyclopedic Unabridged Dictionary of the English Language®, 1989, Portland House, New York.

Young, Michael Dr. *Notes.pdf*. 02 May 2002 http://www.siu.edu/departments/cola/psycho/faculty/young/ResMethodsStuff/Exam3%20Notes.PDF.

http://chublake.com/Psy%203020/ANOVA%20Factorial.pdf.

http://www.aic.nrl.navy.mil/~aha/lessons/

http://www.asme.org/codes/#.

http://www.atl.external.lmco.com/projects/rassp/RASSP_legacy/casestudies/SAR/CAS-SAR_INDEX.HTM

http://www.ieee.org/portal/index.jsp?pageID=home.

http://www.incose.org/stc/index.html

http://www.km-forum.org/what_is.htm

www.nwdc.navy.mil/NLLS/files/2001_NLLS_str_cmt_mtg_presentations/Program%20Manager%20Quicklook.ppt

http://www.software.org/pub/externalpapers/KnowledgeManagementIEEE.pdf.

APPENDIX A

MASTERS PROJECT QUESTIONNAIRE

The focus of my TTU Masters Project is Systems Integration and Test (I&T) process improvement for Engineering and Manufacturing Development (E&MD) programs. For purposes of discussion in my project and questionnaire, the Systems I&T responsibility includes such tasks as; the system interconnect design (major harness build), unit checkout of the major subsystems of a program, the Integration of those units into a system, and the final functional and operational tests of the system. The system tests also include program system checkout and verification (using 1 unit for each test) for such environmental tests as Electromagnetic Interference, Vibration and Thermal Qualification.

Systems I&T planning activities are included in my project as follows:

- **Master Test Sequence.** This is a program level test sequence. It is the highest level of planning and has very little technical content. This may be a document that describes the sequence and outlines the tests to be completed for the program. Individual test plans and working procedures are included elsewhere. Test types included in this discussion are EMI, HALT, Qualification, and Hardware In the Loop (HIL) performance prediction.

- **I&T IMP/IMS.** I&T portion of program level scope definition and schedule, referred to as an Integrated Master Plan (IMP) and Integrated Master Schedule (IMS). Create task definitions and system I&T event schedule for periodic tracking and reporting at the program level. Lower level day-to-day schedules are separate from this activity and created at the task leader's discretion.
- **Lessons Learned Database.** A process that uses a database to track issues, problems, events and dispositions during the course of the program for the system I&T team. There will be a tool such as RMS ITLog used for daily events with a separate tool for major events. The database should be shared among all departments and programs.
- **Tailor Existing I&T Processes**. There should be a library of company or department standard processes available that provide guidelines for completing tasks by the system I&T team. This activity is the revision of those standard processes to reflect modifications for the Lessons Learned and tailor to the specific needs of the task or program.
- **Test Flow and Procedures**. This activity provides more insight for the program leaders into the system I&T daily tasks. It includes the nominal I&T process flow of the test article for all system level test activities and test procedure development.
- **Failure Process.** This process starts with handling the minor failures or issues that are found on a daily basis during Integration, through major failures or errors discovered during critical testing. It should include in-lab daily guidelines as well as a more formal process such as the RMS Failure Reporting, Analysis, and Corrective Action System (FRACAS) for major issues. The process should include guidelines for the decision of what constitutes the need for a formal process, reviews, root cause analysis and a plan for a course of action.

Although these planning activities are delineated in this questionnaire for Systems I&T, they have an effect on other disciplines and functions within a program due to the nature of I&T. The inter-discipline effects are the focus of this questionnaire.

Please designate your current engineering function by selecting and changing color:

System Reqts	Quality	ME	EE	SW	Manuf.	System I&T	Field Test	Program Office	Cust.

Your responsibility Level: Engineer ( ), Task/IPT/CAM Lead ( ), Program/Lead/Chief ( )

Given each of the previously defined I&T activities, based on your experience and independent of the other activities and engineering disciplines, estimate each of the following:

"Broken" is defined for the following as failed, mis-interpreted, not used or ill-defined.

A. The **Impact** (on a 1-10 scale, 10 max.) on your daily tasks of a broken:

1. Master Test Sequence ( )
2. IMP/IMS (I&T portion) ( )
3. Lessons Learned Database Process (assume active and current) ( )
4. Program specific tailored I&T Process ( )
5. I&T Test Flow and Procedures ( )
6. Failure process ( )

B. How critical is your engineering function's **Level of Participation** on a relative time scale (1-10 scale, 10 max.) in helping to re-define, re-plan or re-scope each of these major program activities when discovered to be broken:

1. Master Test Sequence ( )
2. IMP/IMS (I&T portion) ( )
3. Lessons Learned Database Process (for current program/task) ( )
4. Tailored I&T Process ( )
5. I&T Test Flow or Procedures ( )
6. Failure process ( )

C. The **Probability** (1-100% scale) that the following problems are discovered at the <u>start of a program</u>:

1. Major Scope Added to Master Test Sequence (eg. risk reduction test) ( )
2. I&T portion of IMP/IMS is Incomplete ( )
3. Lessons Learned Database Process is Not Used ( )
4. Tailoring of I&T Processes is In-Complete ( )
5. I&T Test Flow and Procedures are In-Complete ( )
6. Failure Process is Ill-Defined for all major anomalies ( )

D. The **Probability** (1-100 scale) that the following problems are discovered <u>following a major customer design</u> review (such as CDR):

1. Major Scope Added to Master Test Sequence (eg. risk reduction test) ( )
2. I&T portion of IMP/IMS is Incomplete ( )
3. Lessons Learned Database Process is Not Used ( )
4. Tailoring of I&T Processes is In-Complete ( )

5. I&T Test Flow and Procedures are in-complete or have major errors ( )
6. Failure Process is Ill-Defined for all major anomalies ( )

E. The **Probability** (1-100 scale) that the following problems are discovered <u>following the first major test</u> of the program:

1. Major Scope Added to Master Test Sequence (eg. risk reduction test) ( )
2. I&T portion of IMP/IMS is Incomplete ( )
3. Lessons Learned Database Process is Not Used ( )
4. Tailoring of I&T Processes is In-Complete ( )
5. I&T Test Flow and Procedures are in-complete or have major errors ( )
6. Failure Process is Ill-Defined for all major anomalies ( )