TEST & EVALUATION A PROCESS

By
Jo Alamares

A MASTER OF ENGINEERING REPORT

Submitted to the College of Engineering Texas Tech University in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING

Approved

____
Dr J Smith

____
Dr A Ertas

____
Dr T. T. Maxwell

____
Dr M. M. Tanik

June 29, 2001

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Tom Kollman for his assistance in walking through the Systems Engineering process used in this paper. Tom has provided the opportunity to work on a project directly related to my field of interest that would be of great value.

The author also wishes to express sincere appreciation to Rick Key who has provided support and guidance on this topic.

TABLE OF CONTENTS

ABSTRACT

This paper discusses the overall Systems Engineering Process and then narrows focus to Integration, Verification & Validation (IV&V). After providing a background for the need to test software, various techniques are discussed and Systems Engineering along with some generic design science principles are applied to the development of IV & V course material in an attempt to correct deficiencies in the presentation. Thus attaining the goal of creating a well-rounded course and principles can be applied in the workplace.

LIST OF TABLES

LIST OF FIGURES

Number *Page*

NOMENCLATURE

Acceptance Testing – Verifying the system meets the specified requirements.

Analysis – Verification by technical or mathematical evaluation using mathematical representations (models, algorithms, and equations), charts, graphs circuit diagrams, and representative data or evaluation of previously qualified equipment.

Component Testing – Verifying and / or validating that a unit, module or sub-element or the system performs as required by the system design.

Demonstration – Verification through operation, movement, and / or adjustment or the item to show it's functionality. Compliance with qualitative standards for performance and go/no-go functionality shall be shown.

Inspection – Verification by visual examination of the item, reviewing descriptive documentation, and comparing the appropriate characteristics to a predetermined standard to determine conformance to requirements without use of special laboratory equipment or procedures.

IMP – Integrated Master Plan

IMS – Integrated Master Schedule

Integration – Identifying, defining, establishing, implementing and controlling interfaces, as well as verifying system functions that require multiple system elements. The purpose and objective of integration is to ensure that system elements will function as a whole.

ITV&V– Integration Test Verification and Validation

Integration Testing – Verifying the interfaces and / or system parts (modules, components, and subsystems) and their functions as lesser elements are integrated into the system.

System Testing – Verifying that the system meets its objectives.

Test – Verification through systematic exercising of the item with sufficient instrumentation to show operational performance. Collection, analysis, and evaluation of test data shall show compliance with specified quantitative criteria.

Testing – Exercising the system with the intention or goal of finding errors and proving that all system capabilities are satisfied.

Verification – Confirming that all system requirement are met by exercising the system in a test or simulated environment.

Validation – Assuring all user needs are met by the system in an operational environment.

C H A P T E R 1
INTRODUCTION

1 Background

Today, numerous technological advances surround our lives. We encounter at least one form of software driven interfaces each morning. For many, these interfaces are electronic hand held Personal Digital Assistants (PDA), mobile phones, calculators and even cars. Many people rely on computerized devices to conduct daily business and activities. In the most sever cases, failure of computerized gadgets not only costs money but in many cases cause deaths.

Machines are only as smart as the one who creates them – man. Humans have proven time and time again that they are not incapable of error. These errors cost companies thousands sometimes millions of dollars in recall costs to replace defective units.

These defects have uncovered the need to emphasize testing throughout the development and manufacturing of many products. It is easy to paint a mental picture of product testing for small replaceable hardware like a digital clock radio, a mobile phone or a television. But think about a large-scale system like a commercial airplane, a complex system that is also part of a bigger air traffic system. In this case, a failure in flight costs lives in addition to the future growth of the airline. Also, there is a potential to crash with another airplane either from the same airline or a different one costing even more damage.

Testing of large-scale systems such as commercial airplanes calls for many engineering disciplines to work together. To summarize testing principles, Hardware, Software and Systems engineers work together to ensure that the system requirements have been adequately satisfied. Although there are many forms and disciplines of testing, this paper will discuss software-testing techniques and related software testing statistics.

This project describes the systems engineering process and applied to an engineering analysis of ones Master's Project. In addition, this project consists of formulating and documenting a set of

Project requirements and verification methods. Finally, one must develop a master plan and schedule to define, organize, development and compile a Final Master's Report.

Test and integration plays an important part of the Systems Engineering Process. Testing helps to ensure that the product being delivered performs to its requirements and meets the customer specified needs. In addition, testing ensures that future development or add-on functionality does not negatively impact existing technology.

The Integration and Verification (I&V) group at Raytheon does not utilize a test process. Although Raytheon claims to follow the IPDS, this process does not fit well in the Software development environment. There are many factors out of our control that cause the process to fail. Hence, the process being used is not documented and accommodates any changes in an ad-hoc manner.

This unwritten process is detrimental to the overall success of the program. First, any knowledge and expertise gained is lost when experienced test engineers leave the company or retire. Once the experience is lost, all new employees do not have the benefit of learning from experienced personnel and no applicable references available.

The Raytheon Learning Center provides a course titled "SE IV&V". The aim of this course is to train future test engineers. Unfortunately, the scope of this course is so narrow that it only gives an overview of test engineering and provides no additional value aside from understanding the process as a whole without application. Updating this course, to include reference information and apply techniques used in the OO Concurrent Engineering to identify a sound testing process would prove to be a great resource for the current test engineering group and additions in the future.

With the help of Ron Townsend and Tom Kollman I have been given the opportunity to work with Rick Key (SE IV&V Instructor) to update the course material to include support materials such as but not limited to the following:

Web sites

Standards

Processes

Special Concerns (COTS/GOTS)

Test plans/procedures

Exercises (to apply the techniques learned)

Metrics

Pitfalls to avoid

The ultimate goal of this project will be to provide a course that allows a student to comprehend and apply the process taught in addition to providing recommendations for implementing the process identified within Raytheon.

C H A P T E R 2
STATISTICS IN TESTING

2 Why Test Software?

Human designers no matter how intelligent are fallible. Software development is abstract and complex by nature and must be accompanied by quality assurance activities. It is not unusual for developers to spend 40% of the total project time on testing. For life-critical software (e.g., flight control, reactor monitoring), testing can cost three to five times as much as all other activities combined. The destructive nature of testing requires that a developer discard preconceived notions of the correctness of his/her developed software. [Kaner]

Software Testing is the process of executing a program or system with the intent of finding errors. Often these processes involve activities aimed at evaluating any attribute or functionality of the system and determine that is meets its required results. Many physical processes accept inputs process the information and outputs are produced. Although many processes fail, the failures associated with physical process are often predictable. Software on the other hand fails in many different manners and often in bizarre ways. Detecting all of the different types of failures is generally costly and improbable. [Kaner]

Most defects found in software are design errors, not manufacturing defects. Software does not suffer from exposure to the elements and generally will not change until upgrades or until the software is obsolete. So, once the software is shipped, the bugs that were not fixed will be buried within and remain dormant until activation.

Software defects will always exist in any software module with moderate size. This is not because programmers are careless or irresponsible, but because along with the complexity of the software comes bugs that are difficult to fix and humans have only a limited ability to deal with complexity.

Software and digital systems are not continuous, testing boundary values are not sufficient to guarantee correctness. All the possible values need to be tested and verified but complete testing is impossible.

Further complication has to do with the dynamic nature of programs. If a failure occurs during initial testing and the code is changed, the software may now work for the test case that caused the initial defect. However, its behavior on the pre-error test case that passed before can no longer be guaranteed and contributing to the cost associated with regression testing.

2.1 Software Testing Techniques

There are two categories of software testing techniques. These are white box testing and black box testing. White box testing centers around the design of test cases to:

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false states.
3. Execute all loops at the extremities of their boundaries and within their operational bounds.
4. Exercise internal data structures to ensure their validity.

[UNISA]

In white box testing, although there are many techniques, the majority of testing falls into the category of basis path testing. Using this method, the designer determines the logical complexity of a procedural design and uses it as an aid to define a basis set of execution paths. The test cases that exercise the basis set are guaranteed to execute ever statement in the program at least once. [UNISA]

Testing for software defects can often be a difficult task. Unless we completely understand the usage of the system being produced, we often believe that the logical path is not likely to be executed. We focus testing efforts toward irregular patterns when the logical path may be executed on a regular basis. Our unconscious assumptions about control flow and data lead to design errors that can only be detected by path testing. [UNISA]

Basis path testing includes methods such as flow graphs, the basis set, deriving test cases and provides a foundation for automated testing. Test cases properly derived from the basis set are guaranteed to execute every statement in the program at least once during testing.

Flow graphs can be used to represent the flow of control in a program and can help in the derivation of the basis set. Each flow graph node (branch) represents one of more procedural statements. Below is an example of a flow graph.

The basis set consists of the set of paths to be explored during testing. Once the basis set has been determined, test cases are derived. These test cases will force execution of each path in the basis set.

On the other hand, testing is not complete only with white box testing. Black box testing is needed to derive sets of inputs that will fully exercise all functional requirements of a system. This type of testing focuses on finding errors in the following categories: [SQA-test]

1. Incorrect or missing functions
2. Incorrect interfaces
3. Errors in data structures or external database access
4. Errors in performance
5. Errors in initialization and termination of the system

Black box tests are designed to answer the following types of questions:

1. How is the function's validity tested?
2. What classes of input will make good test cases?
3. Is the system particularly sensitive to certain input values?
4. How are the boundaries of a data class isolated?
5. What data rates and data volume can the system tolerate?
6. What effect will specific combinations of data have on system operation?

Generally, software is not complete when testing begins. Testing occurs in multiple phases – the more the better. Hence, problems can be found early and fixed without detriment to the overall system and essentially saving money in the long run.

While white box testing should be performed early in the process, black box testing is generally used during later stages. For black box testing, test cases should be derived to reduce the number of additional test case that must be designed to achieve reasonable testing have been derived and reveal something about the presence or absence of classes of errors rather than an error associated with only a single test. [Kaner]

Equivalence partitioning is a black box method that divides the input domain of a program into classes of data from which test cases can be derived. Equivalence partitioning strives to define a test case that uncovers classes of errors and thereby reduces the number of test cases needed. These classes define a set of valid states for input conditions otherwise known as equivalence classes. [UNISA]

Equivalence classes can be defined using the following guidelines:

1. If an input condition specifies a range, one valid and two invalid.

2. If an input condition requires a specific value, then one valid and two invalid.
3. If an input condition specifies a member of a set, then one valid and one invalid.
4. If an input condition is boolean, then once valid and one invalid.

Second, the boundary value analysis method leads to a selection of test cases that exercise boundary values. This method complements equivalence partitioning since it selects test cases at the edges of a class. Test cases are not only derived from input conditions but output conditions as well. Some of the guidelines include:

1. For input ranges bounded by a and b
   - i. Include values a and b
   - ii. Include values just above a and just below b
2. If an input condition specifies a number of values, test cases should be developed to
   - i. Exercise the minimum and maximum numbers
   - ii. Exercise the values just above and just below these limits
3. Apply guidelines 1 and 2 to the output
4. If internal data structures have prescribed boundaries, a test case should be designed to exercise the data structure at its boundary.

Finally, the cause-effect graphing is a technique that provides a concise representation of logical conditions and corresponding actions. The steps for cause-effect graphing are:

1. List input conditions (causes) and actions (effects) are for a given module.
2. Assign an identifier to each.
3. Develop cause-effect graph.
4. Convert the graph in to a decision table.
5. Convert decision table rules into test cases.

2.2 Automated Testing

Simply put, "Automated Testing" means automating the manual testing process currently in use. This requires that a formalized "manual testing process" currently exist in your company or organization. [Kaner]

At a minimum, the formalized process includes:

1. Detailed test cases including repeatable results (development of test cases is described in section 3.0)
2. A standalone Test Environment, including a database that is capable of restoring the system to a known state so tests may be repeated each time there are modifications made to the application. (Restoring the test environment to a known state can also be referred to as the system initialization)

If the current test process does not include these points, it will never be possible to make any effective use of an automated test tool. It can also be stated that test methodologies excluding test cases and a standalone test environment as state above simply turn over software releases to a "testing group" who bang on their keyboards in an ad hoc fashion. Again, because no real process exists, it will not be possible to utilize automation techniques. The first step to automation in this case is to establish an effective testing process.

The case for automating the Software Testing Process has been made repeatedly and convincingly by numerous testing professionals. Most people involved in the testing of software will agree that the automation of the testing process is not only desirable, but in fact is a necessity given the demands of the current market. To be more precise, automating software testing saves time and money in the long run to ensure that previous functionality is not broken when continuous improvement is made (i.e., change to the software does not produce undesirable consequences).

2.3 Viable Automated Testing Methodologies

Every test too vendor will tell you that their tool is "easy to use" and non-technical users will be able to automate all of their tests by simply recording their actions and then playing back a recorded script. This statement alone is responsible for almost all of the false hopes that automated testing will solve all the problems.

The record/playback method does not work because of the following: [Kaner]

1. Scripts resulting from this method contain hard-coded values, which must be changed if any changes in the application are made.
2. Costs associated with maintaining such scripts are astronomical and unacceptable.

3. Scripts are not reliable, even if the application has not changed, and often fail on replay (pop-up windows, messages, and other things can occur that did no happen when the test was initially recorded).
4. Scripts generated are not always error free. If the tester makes an error entering the data or script step and the test must be recorded again.
5. Application changes require the test to be recorded again.
6. Tests generated using this method only exercise functionality that already works. Areas that have errors are encountered in the recording process (which is manual testing). These bugs are reported, but a script cannot be recorded until the software is corrected.

As you can see, this method has a lot of chances for error and generates quite an impact on budget and schedule. Two methods that have been proven to be effective are Functional Decomposition and Key Word Driven testing.

Functional Decomposition script development is aimed toward reducing all test cases to their simplest form. By this we mean organizing tests into functional tasks areas. Generally, these functional areas include: [Kaner]

1. Navigation
2. Specific function
3. Data verification
4. Return navigation

Using this scheme, it is necessary to separate function from data. Then an automated test script can be written exercising system functions using data files to provide both the input and expected results verification. Utilizing different levels to accommodate for the functional components of the system, this hierarchical scheme lends itself to the development of test scripts in a modular design.

Generally, there are five different types of test scripts: [Kaner]

1. Driver scripts perform initialization and then execute test case scripts desired in order.
2. Test case scripts perform application test logic using business function scripts.
3. Business function scripts perform specific business functions within the application.

4. Subroutine scripts perform application specific tasks required by two or more business scripts.
5. User-defined functions are functions that are generally application specific and incorporate screen-access functions and can be called from any script type above.

The highest level is the driver script. This level is the engine of the test. The driver script contains a series of calls to one or more test case scripts. Each subsequent level is lower and accomplishes a accommodates for a different need within the system.

Example: ATM (driver)

1. Initialize all objects
2. Open main menu
3. Execute test case ATM payment (Note: This is only one possibility of the numerous ATM functions that could be executed)
4. Request for next transaction
5. If yes, execute test case for next transaction
6. If no, return to main menu

Example: ATM payment (test case)

1. Access mortgage payment screen from main menu.
2. Post payment
3. Verify payment updates current balance
4. Return to main menu
5. Access account summary screen from main menu
6. Verify account summary updates
7. Access transaction history screen from account summary
8. Verify transaction history updates

9. Return to main menu

There are many advantages to using functional decomposition method. First, utilizing a modular design and using files to both input and verify data reduces redundancy and duplication of effort in creating automation. Another advantage of this method is that, scripts can be developed while application development is still in progress. If functionality changes, only the specific module of the script that was affected by the change needs to be updated. Also, since scripts are written in a modular form, and isolate system functions, it is possible to combine these into a higher-level test script in order to create complex test scenarios. Another advantage when using the functional decomposition method is that input and output data is stored easily and allowing for maintainable text record. The text record also functions as a testing log documenting the verification – a requirement for system testing. Finally, functions returning "true" or "false" values to the calling script, instead of aborting, allowing for more effective error handling, and increasing the resilience of the test scripts. Added robustness along with a well-designed recovery routine, enables unattended execution of test scripts

Although there are many advantages, there are an equal number of disadvantages. First, and probably the most difficult is the development of test scripts requires proficiency in the scripting language being used in the tool. Second, multiple data-files are required for each test case. There may be any number of inputs required. If these inputs can not be combined into a single file, there must be multiple input data files. The same is true for output data-files. At a minimum, the simplest test case will require one input file and one output file. Since multiple data files are used each change in the code results in changes to these files. Once code change can result in many data file changes. Finally, if a simple text editor is used to create and maintain the data-files, careful attention must be paid to the format required by the scripts/functions that process the files, or script-processing errors will occur due to data-file format and/or content being incorrect.

A second method used in automated testing is called the Key-Word Driven method. Key-Word Driven testing uses the actual test case document developed by the tester using a spreadsheet containing special key words. This method preserves most of the advantages of the functional decomposition method while eliminating most of the disadvantages. [Logica]

This method works by putting all the key words, input and verification data into a spreadsheet. The spreadsheet is saved as a tab-delimited file. The file is read by a controller script for the application and processed. When a key word is encountered, a list is created using data from the remaining columns of data. This continues until a blank field is encountered. The list is then passed as in input parameter to the utility script of the application. Each script associated with a given key word is then executed.

The scheme of this method is much like functional decomposition but have distinct differences. [Logica]

1. Driver script
   - i. Performs initialization
   - ii. Calls application specific controller script passing to it the file-names of the test cases generated and saved.
2. The “Controller”
   - i. Reads and processes the file-name received from the driver
   - ii. Matches on key words contained in the input-file
   - iii. Builds a parameter-list from the records that follow
   - iv. Calls utility scripts associated with the key words passing the created parameter list
3. Utility scripts
   - i. Processes input parameter list received from the controller script
   - ii. Performs specific tasks and user defined functions
   - iii. Reports any errors to a test report for the test case
   - iv. Returns to the controller script
4. User defined functions
   - i. General and application-specific functions may be called by any of the above script-types in order to perform specific tasks.

As with the functional decomposition there are many advantages. Additional advantages are as follows:

1. Test plan can be written in spreadsheet format containing all input and verification data. So, the test engineer need only produce one test document instead of supporting multiple formats.

2. Test plan can be written using any software package so long as the file can be saved in the tab-delimited form. If the test plan already exists, it is not very difficult to convert to a tab-delimited format.

3. After a number of generic scripts have been created for the testing application, re-use of these functions is very easy.

Unlike functional decomposition where the disadvantages are many, this method only has two. They are the following:

1. Development of customized functions and utilities requires a significant level of proficiency in the language being used.

2. Development of more than a few customized utilities will require a test engineer to learn many key words and formats which can be both time consuming and a substantial up front impact to test development.

As you can see automated testing has many advantages and disadvantages. If developed correctly, automation gives you the capability to perform many repeated operations in half the time that it takes a human to execute the same test. In our fast paced technological world, system stability and reliability are key.

2.4 Software Reliability

Naturally, all companies want to use and produce error free products. However, because of our human nature to err, often our mechanical and software products are only as good as the humans who create them. Since we can not expect everything to be completely error free forever, we can predict when errors occur under certain conditions.

This is otherwise known as software reliability. More specifically, software reliability is the probably of failure-free operation for a specific period of time in a specified environment. Software reliability plays an important role in system reliability. It differs from any other type of reliability where the construction or manufacturing is in question. Software reliability tests the perfection of the design.

Like automated testing, the realm of software reliability is still rather mysterious and not as well defined as hardware reliability. Reliability is a by-product of quality, and software quality can be measured. These quality metrics assist in the evaluation of software reliability. The appropriate model must be selected to suit the situation before applying the technique. Since software reliability as a discipline is still in its infancy, no good quantitative measures have been developed to represent software reliability without excessive limitation.

There is a distinct difference between hardware failure rate and software failure rate. For hardware, as shown in Figure 1, when the component is first manufactured, the initial number of faults is high but then decreases as the defective components are replaced or stabilize. The component then enters the useful life phase, where few, if any faults are found. As the component physically wears out, the fault rate starts to increase until the unit is replaced. [Dacsdtic]

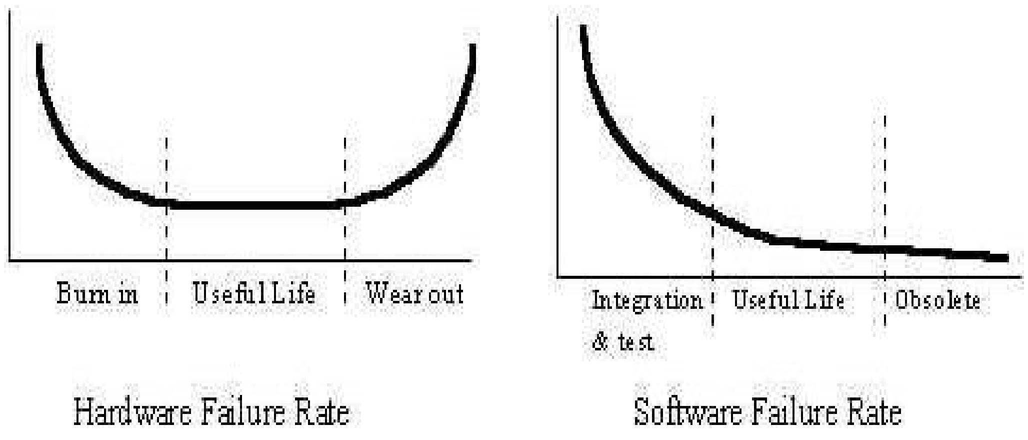


Figure 1Failure Rates

For software, the error rate is at the highest level at integration and test. Throughout all test phases, errors are identified and removed. This removal continues at a slower rate during its operational use; the number of errors continually decreasing, assuming no new errors are introduced. Software does not have moving parts and does not physically wear out as hardware, but is does outlive its usefulness and needs to upgraded or becomes obsolete.

This rate of decreasing defects can be described by the exponential distribution is

$$f(x) = l e^{l x} \qquad x \geq 0$$

where $l > 0$ is a constant. The mean and variance of the exponential distribution are

$m = \frac{1}{l}$ and $s^2 = \frac{1}{l^2}$ respectively.

In the example of software errors, the parameter l is called the failure rate of the system, and the mean of the distribution m is called the mean time to failure. Suppose that a given software component has a failure rate of .01. Then, the number of failures after 100 trials becomes very small as graphed in figure 1.

To increase the reliability by preventing software errors, the focus must be on comprehensive requirements and a comprehensive testing plan, ensuring all requirements are tested. Focus also must be on the maintainability of the software since there will be a "useful life" phase where sustaining engineering will be needed. Therefore, to prevent software errors, we must: [Dacsdtic]

1. Start with the requirements, ensuring the product developed is the one specified, that all requirements clearly and accurately specify the final product functionality.
2. Ensure the code can easily support sustaining engineering without infusing additional errors.
3. A comprehensive test program that verifies all functionality stated in the requirements is included.

Although automation is the way of the future it is clear that reliability has no relationship to automated testing.

One reliability test method is Reliability Sequential Qualification Testing. It is conducted to provide an evaluation of system development progress as well as assurance that specific requirements have been met prior to proceeding to the next phase. First, a Mean Time Between Failure (MTBF) is established for the system. Then design criteria is defined and allocated. After completion of the system design, reliability analyses and predications are made to evaluate the design configuration regarding compliance with system requirements. If the system proves to be compliant, then the design team proceeds in the construction of a prototype.

In a reliability sequential test, there are three possible decisions: 1) Accept the system, 2) Reject the system, or 3) continue to test. Figure 2 below shows the typical sequential test plan. [Montgomery]

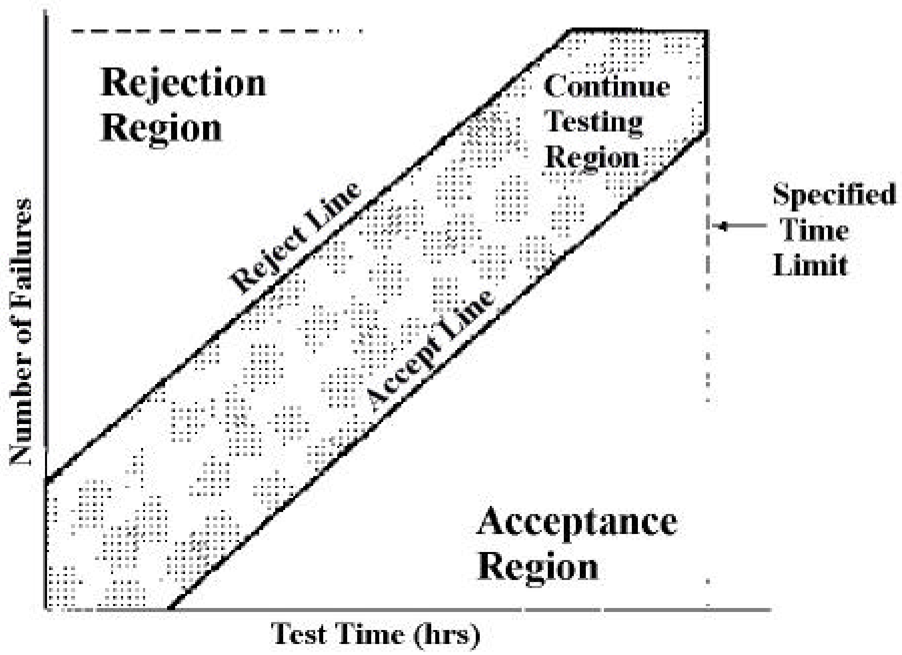


Figure 2 Sequential Test Plan

Sequential test plans are highly influenced by the risks that both the producer and consumer are willing to accept and link these risks with decisions made resulting from testing. The risks are defined as: [Montgomery]

Producer's Risk (a**):** The probability of rejecting a system when the measured MTBF is equal or better than the specified MTBF. In other words, this is the probably of rejecting a system when it should really be accepted. (Type I error)

User's or Consumer's Risk (b**):** The probability of accepting a system when the measured MTBF is less than the specified MTBF. In other words, the probability of accepting a system that actually should be rejected. (Type II error)

Sequential testing is conducted in a manner similar to hypothesis testing. An assumption is made and a test is done to support or disprove that assumption. A null hypothesis (H_0) is a statement or conjecture about a parameter such as the true MTBF is equal to 100. The alternative hypothesis (H_1) is the opposite "MTBF is not equal to 100". When testing an item, the desired result is to accept when the null hypothesis is true and reject it when it is false. These relationships are described in the table below. [Montgomery]

Table 1 Sample Testing Risks

True State	Accept H_0 and Reject H_i	Reject H_0 and Accept H_i
H_0 is TRUE	High Probability 1-α (i.e., 0.90)	Low Probability Error, α (i.e., 0.10)
H_0 is FALSE and H_I is TRUE	Low Probability Error, β (i.e., 0.10)	High Probability 1-β (i.e., 0.90)

It is necessary to specify two values of MTBF.

1. Specified MTBF (θ_0) represents the system requirement.
2. Minimum MTBF (θ_1) this is considered to be acceptable based on the results of testing.

Given θ_0 and $\theta_1$the values for α and β must be decided. Most test plans use accept risk values between 5 and 25%. These values are normally negotiated in the test planning phase.

Software testing is extremely different from any other testing discipline because of its unpredictable nature and the increasing complexity. Many software-testing techniques exist for both manual and automated testing. General testing methods are categorized into white box testing and black box testing. These two categories complement each other throughout the testing process – white box testing at the front end and black box testing at the back end.

Automated testing on the other hand is still quite new and generally misunderstood. Provided that there is a sound testing process in place and a great deal of time is invested in learning the language of a given automated testing tool, automated testing can be used in regression testing. Automated test scripts, if developed correctly can be very useful in regression testing and other categories of test such as stress testing of a system.

Software reliability is an important role of software quality together with functionality, performance and documentation to name a few. As the test engineering discipline grows and matures meeting the quality and reliability needs of our highly technological environment becomes easier and more efficient. Many improvements can be made to increase reliability but all improvement points back to a sound testing process.

C H A P T E R 3
DOMAIN ANALYSIS IN SOFTWARE TESTING

3 Domain Overview

Domain analysis is the process of identifying, collecting, organizing, and representing the relevant information in a domain. This analysis is executed based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory and cutting edge technology within a domain. Domain analysis should carefully define the bounds of the domain under consideration. In addition, domain analysis should also consider the similarities and differences of the systems in the domain, organize an understanding of he relationships between the various elements in the domain, and represent this understanding in a useful way.

This chapter, attempts to analyze the domain of Software Testing. This area of the super-domain Test Engineering as a discipline has not had much exposure and its concepts misunderstood. Software testing is the domain that consists of the processes, databases and procedures that receives, document, track and reports the functional verification of a piece of software. Figure 1 shows the relationship between each of the domains. Notice that the parent domain overall is Systems Engineering and within Systems Engineering there is Test Engineering. Peeling away another layer for more detailed activity, we arrive at Software Testing. Although there have been clear boundaries drawn between the different layers as depicted in the figure below, since software testing is a sub-domain, there are many overlapping activities. This fuels the need for cross-training of individuals rather than single discipline experts.

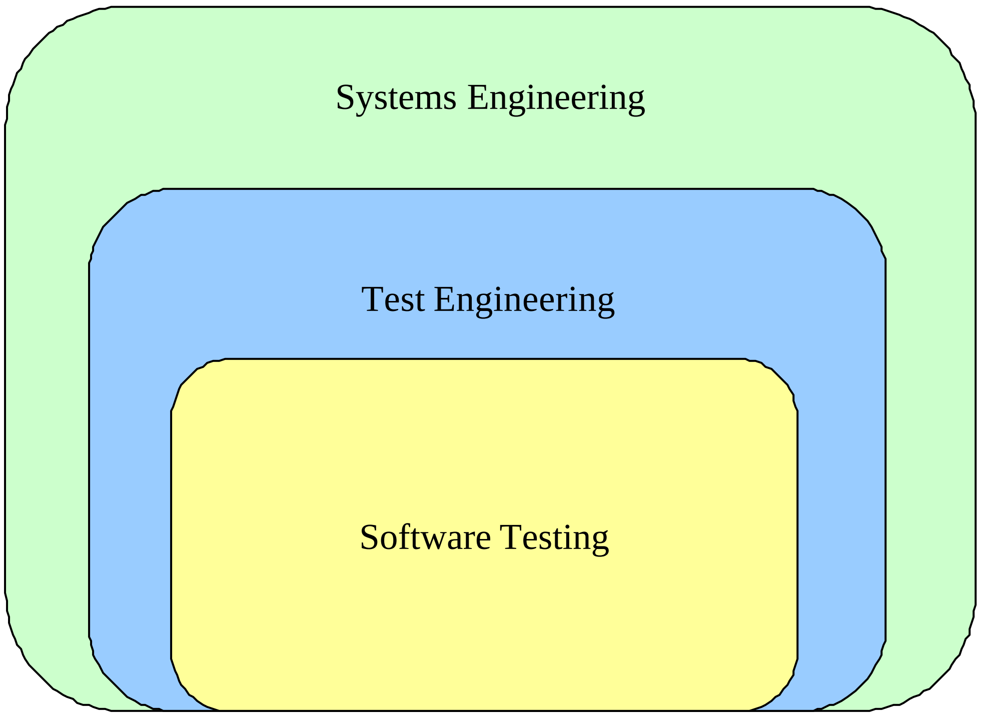


Figure 3 Software Testing Domain Relationship

3.1 Domain Specific Analysis and Modeling Approach

Simply stated, the approach for the software testing domain specific analysis and modeling consists of the following steps:

- Define the software testing domain
- Define the components that make up the software testing domain
- Describe the possible protocol for the components that could guide new component development
- Describe a simple methodology for locating, adapting, and integrating components to build applications in the domain.

Executing these steps demonstrates the basic principles of domain specific analysis and modeling.

3.2 Defining Software Test Engineering

Software testing is an activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas. Software testing is a trade-off between budget, time and quality.

Software testing is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonable small) set of ways. However, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible.

Regardless of the limitations, testing is an integral part of software development. It is broadly deployed in every phase in the software development cycle. It is not unusual to spend more than 50% of the development time on testing. Test usually is performed for the following purposes:

- Improve quality
- Verification and Validation
- Reliability
- Quality
- Functional (exterior quality)
- Engineering (interior quality)
- Adaptability (future quality)

There are many challenges in software testing. It is impossible to test for all possible errors. If everyone wanted to test for all possible errors, customers would never get new software because software developers would take so much time testing their product.

Although no methodology could solve the problem of complete testing, selection of the appropriate components and utilizing them effectively would be of great benefit and also provide timely delivery of a validated product. This new domain is defined as follows:

1. Test Planning becomes the information collector

2. Test Cases become the center of the test process correlating the collected information with the capabilities required of the system.
3. Test Procedures along with the Verification Trace Matrix become the method associated with information analysis and sharing information through collaborative communication.

Software testing promotes information sharing. This methodology can then be expanded to all types of testing (hardware, reliability, maintainability, etc.).

3.2.1 Test Planning

Test planning is an important part of any test process and is virtually the same for all types of testing. The purpose of a test plan is to describe the Integration, Verification and Validation (IV&V) concepts, associated logistics, and the integration approach. The test plan defines objectives and success criteria for each test and each requirement within each test. Depending on the size and complexity of the program, the test plan may be a combination of IV&V or may be split into separate plans for each component of Integration, Verification, and Validation. A typical test plan would contain the following:

> **Purpose:** This section identifies the purpose and scope of the test to be conducted. Also, this section summarizes the overall testing methodology.
>
> **Reference Documents:** This section lists applicable documents. These documents may include system specifications, requirements specifications, system and document specification standards.
>
> **IV&V Concepts:** This section describes in detail the IV&V concepts to be used during the test phase(s). Test concepts include the test methods, levels and organizational roles to be implemented during testing.
>
> **Schedules and Locations:** This section describes in detail the logistics behind the IV&V process as related to scheduling and test locations at all levels and test phases to be conducted.
>
> **Resources and Preparations:** This section identifies personnel, facility, and equipment requirements. In addition, this section normally identifies the test environment and conduct of testing.

Verification Requirements and Criteria: This section is the major focus of test planning. It links together test execution and requirements. This section details objectives for each test along with acceptance criteria for each requirement

Standards: This section contains standards for configuration management, hardware and software standards.

Notes and Appendices: This section contains any additional information regarding testing.

The Test Planning domain can be decomposed into the various levels of testing Sub-unit test, Unit test, Subsystem test and System. The components within these levels are Objective, Concepts, Resources, and Verification. In the new domain definition, sub-unit test is specific to the level and granularity of testing and can be tied to multiple sub-units. A similar relationship can be developed for subsequent levels of test planning allowing for a collection of necessary data to perform software testing.

3.2.2 Test Cases

Test cases are used to focus the type of testing to be performed. Test cases are broken into two categories (white box testing and black box testing).
White box testing test cases ensure:

1. All independent paths are exercised at least once.

   All logical decisions are exercised for both true and false paths.

   All loops are executed at their boundaries and within operational bounds.

   All internal data structures are exercised to ensure validity.

Although white box testing is not used to test conformance to requirements, it does add value. Some of the errors caught by white box testing are:

1. Logic errors and incorrect assumptions most likely to be made when coding for “special cases”.
2. Assumption errors about execution paths that lead to design errors.
3. Typographical errors that could likely be on a mainstream path as well as on an obscure logical path.

On the other hand, black box testing is focused on requirements verification. Testing in general weather it is focused on hardware, software or any other type of testing, is the most common form of testing. It attempts to find:

1. Incorrect or missing functions
2. Errors related to the interface(s)
3. Errors in data structures or external database access
4. Errors that affect performance
5. Errors that affect initialization or termination

For Domain Analysis, test cases (black box and white box) can be divided in the following way:

Equivalence Partitioning: Equivalence partitioning divides the data into two classes representing valid or invalid states. Classes can be defined if an input condition specifies a range or specific value. This case produces one valid and two invalid equivalence classes. Classes can also be defined if an input condition specifies a boolean or a member or a set. This case produces one valid and one invalid equivalence class.

- **Boundary Value Analysis:** Boundary value analysis identifies the errors that tend to occur at boundaries (extreme limits) of the input domain.
- **Cause Effect Graphing:** Cause effect graphing attempts to provide a concise representation of logical combinations and corresponding actions.
- **Basis Path:** Basis path, one form of control structure, is aimed toward deriving a logical complexity measure of a procedural design and use this a guide for defining a basic set of execution paths. Often this is an exhaustive measure of conditions, data flows, and loops.

3.2.3 Test Procedures

For all formal testing, test procedures are used to document the steps performed and detailed verification. Ideally, test procedures are generated to accommodate the test cases. Early drafting of test procedures has many advantages. These advantages include:

1. Provides a structure for later production of test procedures
2. Identifies potential areas for later development
3. Reduces lead time for production of final versions
4. Permits early customer comments

Test procedures contain step-by-step sequences of operations. This detail allows for repeatability for error conditions and establishes a baseline for validating discrepancies. Most important, a detailed test procedure ensures each system/subsystem interface element is verified against its corresponding interface definition.

After the formal execution of test procedures, results of test procedures are evaluated to:

1. Ensure that all steps have been performed and recorded
2. Confirm all discrepancies have been recorded
3. Close out test log
4. Prepare for a post test briefing

The outcome of these results determines if a test was successful. Test Procedures along with the Verification Trace Matrix become the method associated with information analysis and sharing information through collaborative communication. The test procedures and VTM can be broken down into data analysis, system validation and knowledge sharing. Data analysis provides data manipulation; system validation provides the verification and knowledge sharing permits the transfer of information to the customer. These components together facilitate efficient transfer of information and results to all involved participants. Table 1 shows a sample of a VTM.

3.2.4 Verification Trace Matrix (VTM)

The Verification Trace Matrix (VTM) is the glue that binds together test cases, requirements and test procedures. A VTM is most often presented in a table format. This table identifies requirements for each component of the system, a method for verifying the requirement and a test procedure that demonstrates the functionality associated with the requirement. See Table 1 below for an example of a verification trace matrix.

Table 2 Sample Verification Trace Matrix

Requirement ID	Requirement Text	Verification Method	Verification Level	Verification Location	Test Event
A001	The Temperature Control Subsystem (TCS) shall provide the capability to select the AC or Heat program via the touch	Demo	FAT	STF	TE01
A002	The Temperature Control Subsystem (TCS) shall initiate the AC program when the temperature in the room is more	Analysis	FAT	STF	TE01

3.2.5 Software Test Component Protocol

The new Software Test domain is similar to multi-user collaboration with various agents (information, learning, technology etc.). One possible protocol to guide new component development inter and intra Software Test domain/component communications and data sharing is JAVA, Enhanced JAVA Beans (EJB), Extensible Hypertext Markup Language (XML), and XML query language (XQL). In addition, utilizing collaboration software such as NetMeeting and CVW (Collaborative Virtual Workspace) integrate human and binary technologies.

3.2.6 Component Management Methodology

With a vast amount of knowledge, the issue of management is always at the top of the list. One methodology concept for locating, adapting, and integrating components into the new domain is to set up a component library. Accompanying this library would be a dynamic table expanding to accommodate the addition or deletion of components in the domain. Each component in the library is a record/object and includes how and why that component was used in a given application. The component log would also be updated each time it was used (check-in / check-out) to monitor any extensions and allow future owners to incorporate extended capabilities for ease of use.

The component breakdown of the Software Test domain consists can be very elaborate. Simply stated, applying domain analysis to the software test domain and its components are depicted into the following Figure 2 of the new Software Test Domain.

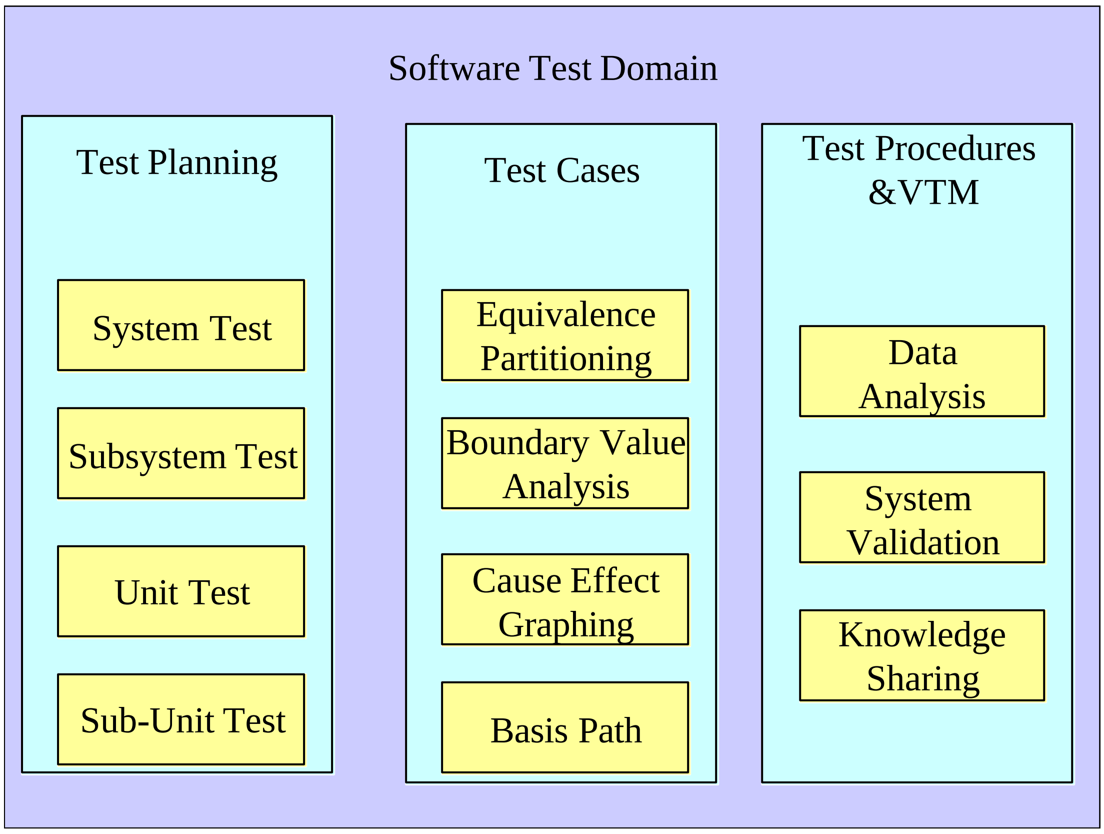


Figure 4 New Software Test Domain

These components can interface among one another and across to other domains via an intranet or internet connection, and utilize JAVA, XML along with collaborative software for information sharing.

C H A P T E R 4
SYSTEMS ENGINEERING PROCESS

4 Systems Engineering Process Description

Systems Engineering is a process employed in the evolution of systems from the point when a need is identified through production and/or construction and ultimately deployment of that system for consumer use. [Blanchard & Fabrycky, 1990] Overall, systems engineers bind many engineering specialists together to address the needs of all the elements of the system in a proper and timely manner.

Every project needs a process and an understanding of the activities within each stage. There are many variations in the applications of engineering functions to the system life cycle. These variations to the Systems Engineering process are made depending on the scope and complexity of the system and the extent of new design and development required.

The role of the systems engineer will usually be different from one situation to the next. However, in spite of these differences, it can be stated that engineering functions of one type or another are performed in each applicable phase in the system life cycle. [Blanchard & Fabrycky, 1990]

For this project, one will explore its application within the seven major stages. These processes can be applied to a variety of projects including the analysis of the Master's project. These stages from the TTU course materials, ENGR 5000, provided by Tom Kollman, are as follows:

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

For each of these stages there are major activities that need to be completed in order to proceed with the next stage, using the Raytheon IPDS 2.06 as a guide. The following sections describe each of these stages and identify applications of these stages to the Master's Project.

For the Master Project Analysis and Planning, I utilized the Raytheon IPDS 2.06 to follow the seven stages of Systems Engineering and tailored them to meet the needs of my Master's project. After reviewing these seven stages, since the topic of the Master's project has already been approved, Gates 1-4 can be bypassed. For completeness, a description of these gates are listed in Appendix 1. Submitting the proposal and obtaining an approved topic, is authorization to proceed to stage 2.

4.1 Stage 1 Business Strategy Planning and Execution

Acquisition of new business is the livelihood of every company. If systems engineers did not venture out to seek the latest and greatest technology, and acquire contracts to provide it, many companies would be out of business. The major activities in this stage are:

- Opportunity Identification
- Program Capture & Proposal Planning
- Competitive assessments
- Win Strategy Development
- Technology Development Planning
- Proposal Development & Submittal

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

Upon completion of this phase, the key outputs are:

- Win Strategy Package
- Decision Packages for Gates 1-4 (Interest/no interest, Pursue/ no pursue, Bid/no Bid, and Bid/proposal review, respectively)
- Key Trade-offs, Technical approaches
- Comprehensive proposal volumes package

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

This stage identifies the new business as a viable pursuit and at the end of this stage, the contract is awarded.

4.1.1 Statement of Need

Test and integration plays an important part of the Systems Engineering Process. Testing helps to ensure that the product being delivered performs to its requirements and meets the customer specified needs. In addition, testing ensures that future development or add-on functionality does not negatively impact existing technology.

One way that Integration and Verification (I&V) has been handled is by utilizing the IPDS 2.06 test process (see 3.9 for details). Although IPDS is a thorough and detailed process as a whole, which allows for complete documentation, often there are other factors which set back the efforts of I&V. Some of the factors out of our control that cause the process to fail include inefficient planning for forecasted work, staffing, and training. Hence, the process being used is not documented and accommodates any changes in an ad-hoc manner.

This unwritten process is detrimental to the overall success of the program. First, any knowledge and expertise gained is lost when experienced test engineers leave the company or retire. Once the experience is lost, all new employees do not have the benefit of learning from experienced personnel and few applicable references available.

4.1.2 Project Scope

The Raytheon Learning Center provides a course titled “SE IV&V”. The aim of this course is to train future test engineers. The scope of this course currently gives an overview of test engineering and provides little additional value aside from understanding the process as a whole without application. Updating this course, to include reference information and apply techniques used in the OO Concurrent Engineering to identify a sound testing process would prove to be a great resource for the current test engineering group and additions in the future.

Some key additions to the updated course material include support materials such as but not limited to the following:

- Web sites
- Standards
- Processes
- Special Concerns (COTS/GOTS)
- Test plans/procedures

- Exercises (to apply the techniques learned)
- Metrics
- Pitfalls to avoid

The ultimate goal of this project will be to provide a course that allows a student to comprehend and apply the process taught in addition to providing recommendations for implementing the process identified within Raytheon.

4.2 Stage 2 Project Planning, Management and Control

Once the contract is awarded, activities associated with the project startup and initial planning begin. Some of these include:

- Project Start-up, Initial & Detail Planning
- Integrated Master Plan (IMP) & Schedule (IMS) Development
- Project Performance Assessment Planning
- Project, Technical, Manufacturing, Operations and Support Management
- Planning Updates & Transition to Next Phase

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

The objective of this stage is to successfully plan, direct, and control projects to satisfy customer requirements. These activities are refined during course of a project. The stages that follow flow back to Project Planning, Management and Control until the shutdown of the program.

Upon completion of this stage, some key outputs are:

- Integrated Master Plan (IMP) & Schedule (IMS)
- Project Six Sigma Strategy
- Integrated Product Team (IPT) Structure
- Work/System Breakdown Structures
- All Project Detailed Plans

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

4.2.1 Outline of the Master's Project

Overall, project planning and scheduling is very important to ensure the success. This master view integrates all the elements of planning into one view giving us the opportunity to exceed customer needs and expectations.

For stage 2 it is necessary to identify major elements (e.g., chapters) of the solution to the problem statement. As applied to the Master's Project, an outline of the content in addition to a Master Plan and Master Schedule is sufficient. The Master Plan and Master Schedule can be found in sections 2.2.2 and 2.2.3, respectively.

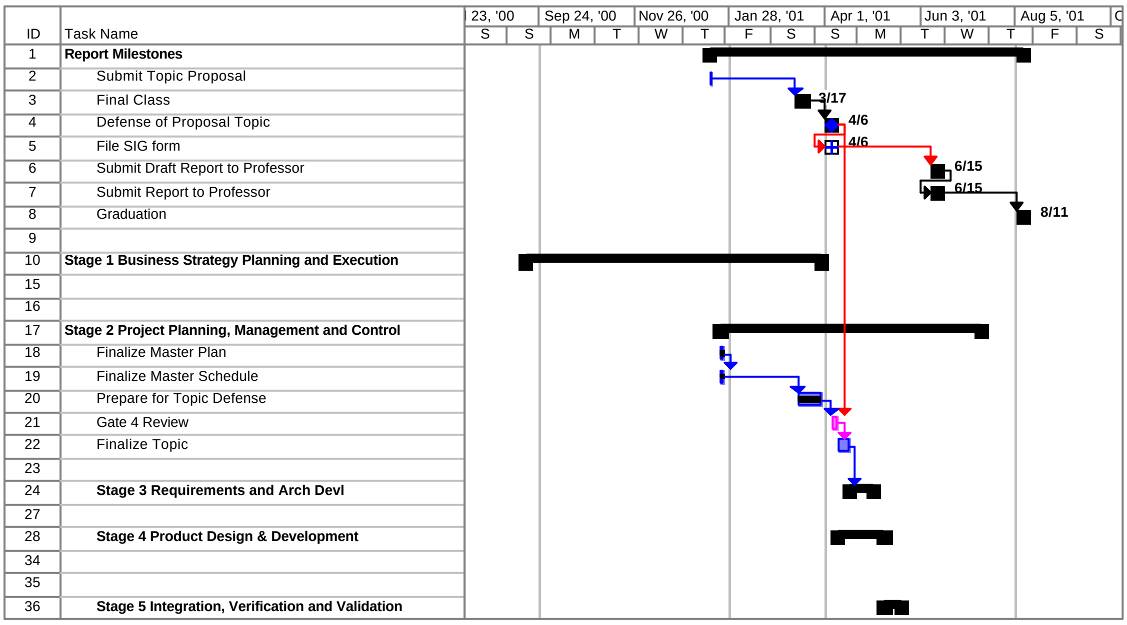
4.2.2 Master Plan

For the Master's project, Jo Alamares in the role of the student, assumes the same responsibilities as a Systems Engineer. The duties of this individual and others involved are shown below in the Master Plan.

Responsible Individual	Role	Responsibility/Plans
Jo Alamares	Student	Plans and coordinates inputs to Master's project
Rick Key	IV & V Course instructor	Provide guidance for improvements to IV&V course, maintain master copy of course materials and coordinate document updates etc.
Professor(s)	Customer	Approve/disapprove work Master's project activities

4.2.3 Master Schedule

Figure 1 below is the top level master schedule followed by the Detailed Master Schedule Table 1, which contains the lower level activities associated with each stage.



ID	Task Name
1	**Report Milestones**
2	Submit Topic Proposal
3	Final Class
4	Defense of Proposal Topic
5	File SIG form
6	Submit Draft Report to Professor
7	Submit Report to Professor
8	Graduation
9	
10	**Stage 1 Business Strategy Planning and Execution**
15	
16	
17	**Stage 2 Project Planning, Management and Control**
18	Finalize Master Plan
19	Finalize Master Schedule
20	Prepare for Topic Defense
21	Gate 4 Review
22	Finalize Topic
23	
24	**Stage 3 Requirements and Arch Devl**
27	
28	**Stage 4 Product Design & Development**
34	
35	
36	**Stage 5 Integration, Verification and Validation**

Figure 5 Master Schedule

4.3 Stage 3 Requirements and Architecture Development

The purpose of Stage 3, Requirements and Architecture Development is to define and analyze customer requirements for the system. Most important, this phase is to define the valid requirements baseline for the system. Once the baseline requirements have been established, proceed to develop a functional architecture for the system. Below are some activities that summarize stage 3.

- System Functional & Physical Architecting
- Product Architecting & Requirement
- Requirements Development & Validation
- Technical Analyses
- Prototyping
- System ITV&V Approach

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

These activities yield the following outputs:

- System, Products & Components Requirements Definition
- System & Product Physical & Functional Architectures
- Validated requirement Baselines
- System & Product Preliminary Designs
- Six Sigma Design Integration

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

4.3.1 Summary of Master Project Requirements

This project will demonstrate the use of the Systems Engineering Process to write the master's project. To get us started, the core requirements of the final project are as follows:

1. The project shall be 80-100 pages in length.
2. The project shall conform to the TTU Master's project format provided by Dr. Ertas.
3. The project topic shall be submitted to and approved by Dr. Ertas. [this is otherwise known as the problem statement that was submitted last month]

4. The final project shall be submitted to Dr. Ertas by the end of June.
5. The student shall provide two copies of the final report. [one copy each for grading and ABET accreditation boa rd]

From these core requirements, derived requirements shall be generated to further specify the requirement and permit testability.

4.3.2 Requirements Derivation

The core requirements alone are not sufficient to characterize the complete final report. For each core requirement stated above, where possible, it is necessary to derive detailed requirements to further clarify the scope and intent of the final project.

Writing a *"good"* requirement can be very difficult. At times, it seems that one needs to be a lawyer to write requirements engineers will agree with. In addition, once approved by the customer, requirements become part of specification documents. From this point, all program design and development must abide by the specifications. A good requireme nt has the foll0owing attributes:

- Clear, concise and unambiguously states a statement of need (only one interpretation)
- Faithfully reflects the source documents
- Adds value for the user
- Verifiable

4.3.3 Verification Trace Matrix (VTM)

For the final project, the following list of derived requirements was generated. These detailed requirements are the basis for the Verification Trace Matrix (VTM). This matrix associates a test method with requirement.

In the VTM, each requirement posses the following components:

- **Requirement ID** is a unique identifier for each requirement. There are many different conventions to assigning requirement Ids. Generally, this consists of a combination of alpha and numeric characters. The best choices are unique identifiers for each requirement.
- **Requirement Text** contains the requirement text for a given requirement.

- **Requirement Group** contains the grouping of a requirement. For large systems grouping of requirements allows for easy breakdown and logical component organization. This makes all phases of the Systems Engineering process easier.
- **Verification** identifies the method and of verification. There are five main methods used. These methods are Analysis, Inspection, Demonstration, Test and Regression. These methods are described below.

  ? **Inspection** consists of a visual verification of a requirement – hence Inspection. (e.g., verifying that the color of the paint shall be white)

  ? **Demonstration** consists of exercising a function to yield a pass/fail result. (e.g., demonstrate that the electric wiring is complete by turning on a light switch)

  ? **Analysis** consists of using statistical techniques to determine compliance with a given requirement. (e.g., the structural integrity of the foundation can be determined by analyzing the materials used)

  ? **Test** consists of exercising functionality using a specific input and yielding a specific output. (e.g., depressing the ON switch to activate the security system triggers the subroutine that requests the needed information to the security system) For our house project, this level of testing does not apply since we neither have the time or the expertise to test such low level details.

  ? **Regression** consists of exercising functionality that is already existing using the same method. Regression should always be tied to another test method. (e.g., Regression / Inspection, Regression / Demonstration, Regression / Analysis).

For the purposes of this project, test methods are limited to inspection due to the subjective nature of generating a training curriculum.

Table 3 Verification Tract Matrix

REQUIREMENT ID	REQUIREMENT TEXT	Verification
D1	The Master's Project shall contain the following sections as in the TTU Master's project format provided by Dr. Ertas.	Inspection of the Master's project content

REQUIREMENT ID	REQUIREMENT TEXT	Verification
	▪ Title Page ▪ Acknowledgements ▪ Table of Contents ▪ Abstract ▪ List of Figures ▪ List of Tables ▪ Nomenclature ▪ Introduction ▪ Application ▪ Results ▪ Conclusion and Recommendation ▪ References ▪ Appendices (optional)	
D2	The Acknowledgements page shall recognize any individuals who contributed to the project at the discretion of the author.	Inspection of the Master's project content
D3	The Table of Contents shall list the contents of the project.	Inspection of the Master's project content
D4	The Abstract shall provide a brief description of the project	Inspection of the Master's project content
D5	The List of Figures shall provide a list of all figures in the project	Inspection of the Master's project content
D6	The List of Tables shall provide a list of all tables in the project	Inspection of the Master's project

REQUIREMENT ID	REQUIREMENT TEXT	Verification
	project	Master's project content
D7	The Nomenclature shall define all terms in the project	Inspection of the Master's project content
D8	The Introduction chapter shall describe the problem and identify the scope of the project.	Inspection of the Master's project content
D9	The Application chapter shall describe the methods used and applied to the generation of the SE IV&V course material.	Inspection of the Master's project content
D10	The Results chapter shall describe the outcome after applying the methods in the application chapter.	Inspection of the Master's project content
D11	The Conclusion and Recommendations chapter shall describe any observations and recommendations.	Inspection of the Master's project content
D12	The References chapter shall list all references used to produce the project.	Inspection of the Master's project content
D13	The Appendices shall contain detailed information to supplement the content of the project.	Inspection of the Master's project content
D14	The final project shall provide resources for I&V processes and standards such as web sites, templates for, test plans and procedures	Inspection of Master's project content

Although there are four major verification methods, (i.e., Analysis, Demonstration, Inspection, and Test), the Master's project requirements will be verified by inspection since content and format are verified visually.

4.4 Stage 4 Project Design and Development

Now that the baseline customer requirements have been defined, this stage of the process decomposes those system-level requirements into an affordable system design. This includes the definition of operational concepts, hardware, software, test and support requirements that can be achieved within the cost and schedule defined in the initial proposal.

- Component Architecting
- Technical Validation & Technical Performance Measurement (TPM)
- Simulation/Model Development & Validation
- Component Design & Build through IPDP
- Embedded Engineering Design Reviews

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

Outputs of this stage include:

- Requirements Maturation
- Component Detailed Design
- Six Sigma Design Integration
- Design Simulation, Modeling &Analysis Data
- System ITV&V preparation, e.g., procedures, analysis software & support test equipment

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

4.4.1 Master's Project Content

After clearly defining the project requirements, the next stage encompasses research and compilation of pertinent information to support the development of the Master's Project content. These activities are executed in an iterative manner to expand the elements listed in stage 2. At this point, based on the research, modifications to the Master Plan and Master Schedule may occur.

4.5 Stage 5 System Integration, Verification and Validation

At this stage, the system requirements must be baselined for each build and system functional and allocation baselines are established. Now the components / units are ready to be integrated into one system for testing and verification of design. This stage integrates hardware and software products in a building block manner. The system is validated to ensure that it performs the functions stated in the design. The major activities in this stage include:

- System Demonstration & Builds Integration
- Conduct System Integration, Test and Verification
- System Validation
- Technical Audits

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

Outputs of this stage include:

- Product Verification Data
- Six Sigma Build Plans
- Facilities Drawing Package
- Product/System Demonstration Test data
- First Article/Customer Acceptance Test Reports

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

4.5.1 Master's Project Verification and Validation

For the Master's Project, this stage collects all the expanded chapters identified in stage 2. This stage also ensures that the requirements in stage 3 are satisfied according to the VTM. Finally, arriving at a solution effectively validates the Master's Project.

4.6 Stage 6 Production and Deployment

This stage manufactures the products that were designed and qualified during the preceding stages. Specification documents drawings and other design documentation provides a stable design used in production. Due to the nature of the project and topic of software testing, only the deployment portion of this stage is implemented.

Major Activities include:

- Factory Planning
- Order Mobilization
- Ongoing System production
- Unit & Acceptance Testing
- Supply Chain Management
- System deployment

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

Outputs of this stage include:

- Product Fabrication & Assembly Plans
- Six Sigma Production Requirements & Data for Continuous Improvement
- Product Quality Assurance Verification Data
- Product/System Delivery

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

4.6.1 Master's Project Delivery and Defense

In the context of the Master's Project, Stage 6 Production and Deployment, encompasses the delivery defense of the Master's Project to the graduate committee. The defense of the Master's Project serves as a forum for acceptance testing. Upon successful review, any minor changes or adjustments are identified by the graduate committee and lead into Stage 7 of the Systems Engineering Process.

4.7 Stage 7 Operations and Support

The objective of stage 7 is to provide ongoing support (modifications and improvements) as necessary throughout the life of the system. This ongoing support includes providing the fielded system with items such as tools, spare parts, technical documents, and repair and maintenance.

Some of the major activities in this stage include:

- Operations & Support Planning
- System & Faculty Operations
- Warranty Service & Support

- Support of Supply, Test Training Publications and Facilities
- Obsolescence & Disposal Management

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

Outputs of this stage include:

- Logistics Support Data
- Field Service/Support Data
- System, Support and User Documentation Currency
- Plans for Next Evolutionary Cycle

[Kollman, TTU Master's Intro to Systems Engineering course ENGR 5000]

4.7.1 Master's Project Clean-up

This stage focuses on incorporating any suggestions made by the graduate committee. After the required changes are made and approved, the completed Master's Project is submitted to the college. At this point the student receives authorization to proceed to the graduation ceremony.

4.8 Use of the 7 Stages for Integration Verification and Validation

As you can see the Systems Engineering process can be used to describe any number of processes. Typically the seven stages break down into components. IPDS 2.06 portrays this as somewhat independent processes. Rather, these process viewed from a different perspective, are more like a subset of embedded processes.

One way that I have observed the implementation of the IV&V process is the following:

[Raytheon IPDS 2.06]

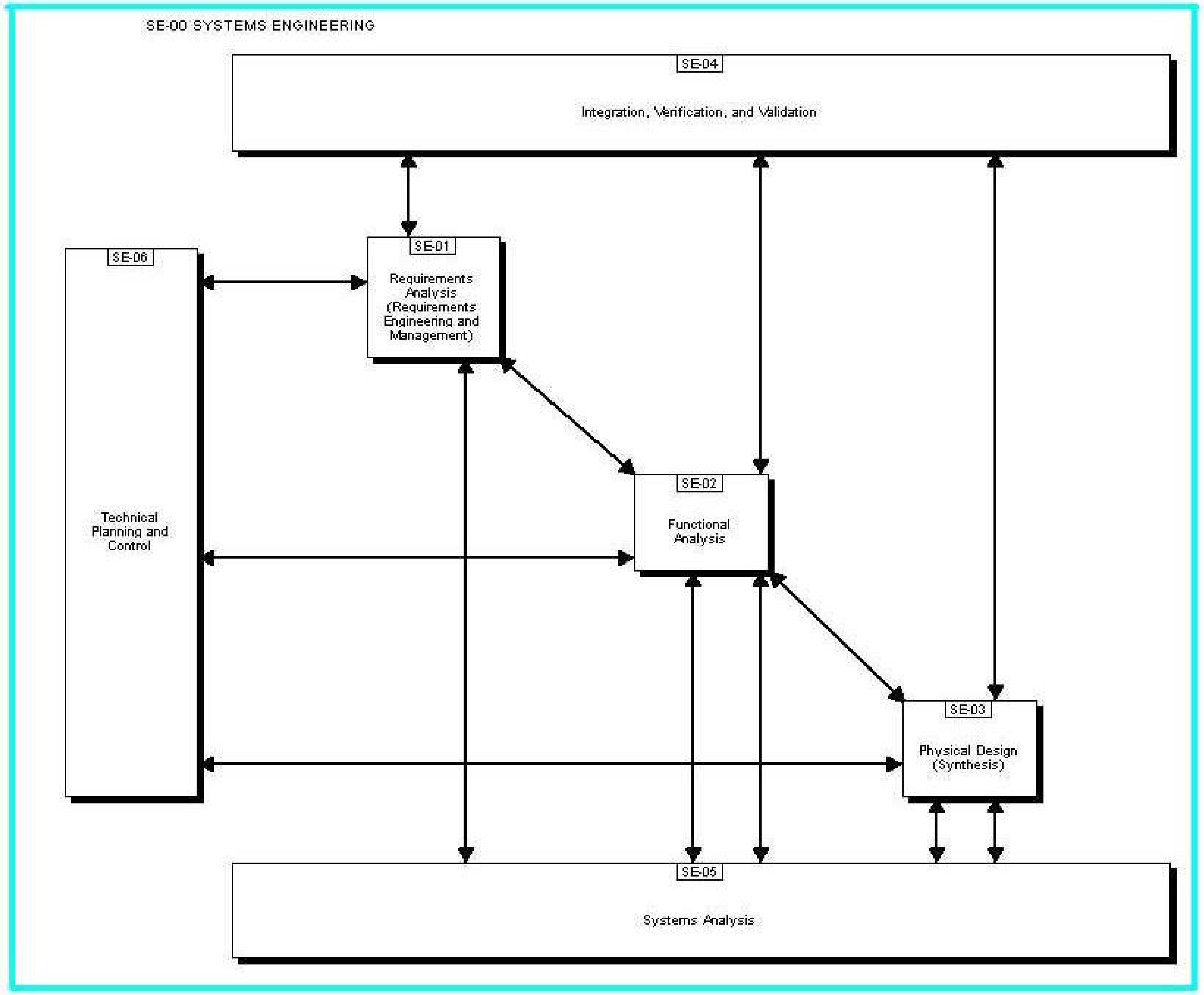

SE-00 SYSTEMS ENGINEERING
SE-04
Integration, Verification, and Validation
SE-06
Technical Planning and Control
SE-01
Requirements Analysis (Requirements Engineering and Management)
SE-02
Functional Analysis
SE-03
Physical Design (Synthesis)
SE-05
Systems Analysis

Figure 6 Level 2 Systems Engineering Process [IPDS 2.06]

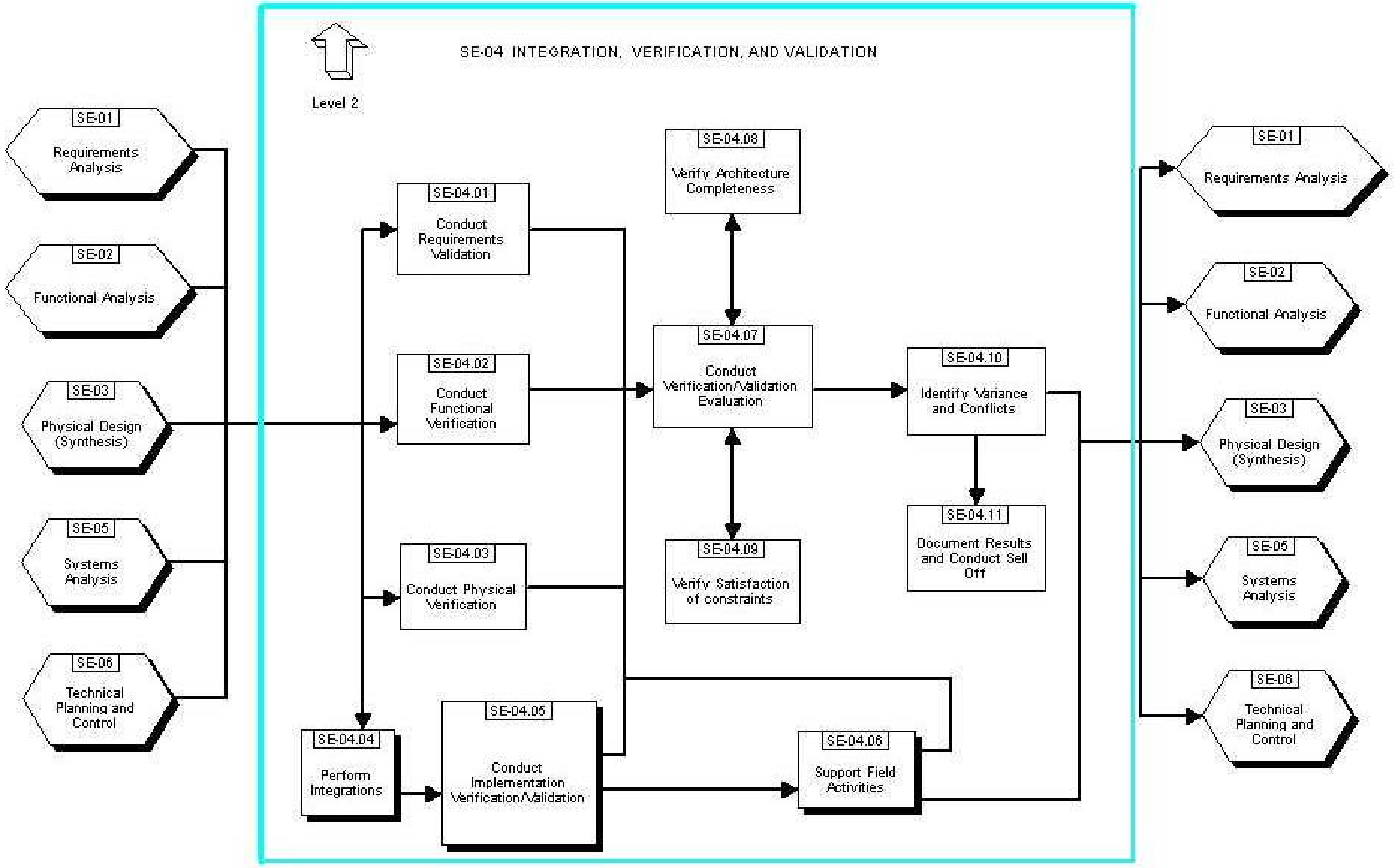

Level 2
SE-04 INTEGRATION, VERIFICATION, AND VALIDATION
SE-01 Requirements Analysis
SE-02 Functional Analysis
SE-03 Physical Design (Synthesis)
SE-05 Systems Analysis
SE-06 Technical Planning and Control
SE-04.01 Conduct Requirements Validation
SE-04.02 Conduct Functional Verification
SE-04.03 Conduct Physical Verification
SE-04.04 Perform Integrations
SE-04.05 Conduct Implementation Verification/Validation
SE-04.06 Support Field Activities
SE-04.07 Conduct Verification/Validation Evaluation
SE-04.08 Verify Architecture Completeness
SE-04.09 Verify Satisfaction of constraints
SE-04.10 Identify Variance and Conflicts
SE-04.11 Document Results and Conduct Sell Off
SE-01 Requirements Analysis
SE-02 Functional Analysis
SE-03 Physical Design (Synthesis)
SE-05 Systems Analysis
SE-06 Technical Planning and Control

Figure 7 Systems Engineering Integreation, Verification and Validation [IPDS 2.06]

The Systems Engineering Processes above carry with them detailed task descriptions that describe the activity that must be completed during that phase of the process. It can bee seen in Figure 4, there are interdependencies between each of the component processes. Peeling each layer of the onion called the Systems Engineering Process as descirbed by Raytheon IPDS 2.06, reveals more detail. It can also be observed that poor performance up stream in the process creates a wave that grows as we progress through the process.

In addition to the Systems Engineering Process, Raytheon IPDS 2.06 has another resource for Test Engineering. This resources is the Test Engineering Development processes. The purpose of these processes is to plan for, develop, and implement an optimum set of embedded test components and external test systems. These test components support all test activities to significantly reduce cost and cycle time while enhancing the prime system quality and customer satisfaction.

> “With prime systems becoming more complex and the competitive pressures of cost and cycle time becoming more pervasive, a disciplined integrated test engineering process that identifies, characterizes, and implements all of the required test activities throughout the prime system's life cycle is critical.” [Raytheon, IPDS 2.06]

These processes provide a method for the complete development of IV&V component processes and document tailorability. Below is the second level of the Test Engineering Development Process depicted in Raytheon’s IPDS 2.06. Each of the components are descibed in more detail with several levels of components. For simplicity, level 2 of each component has been expanded only one level lower.

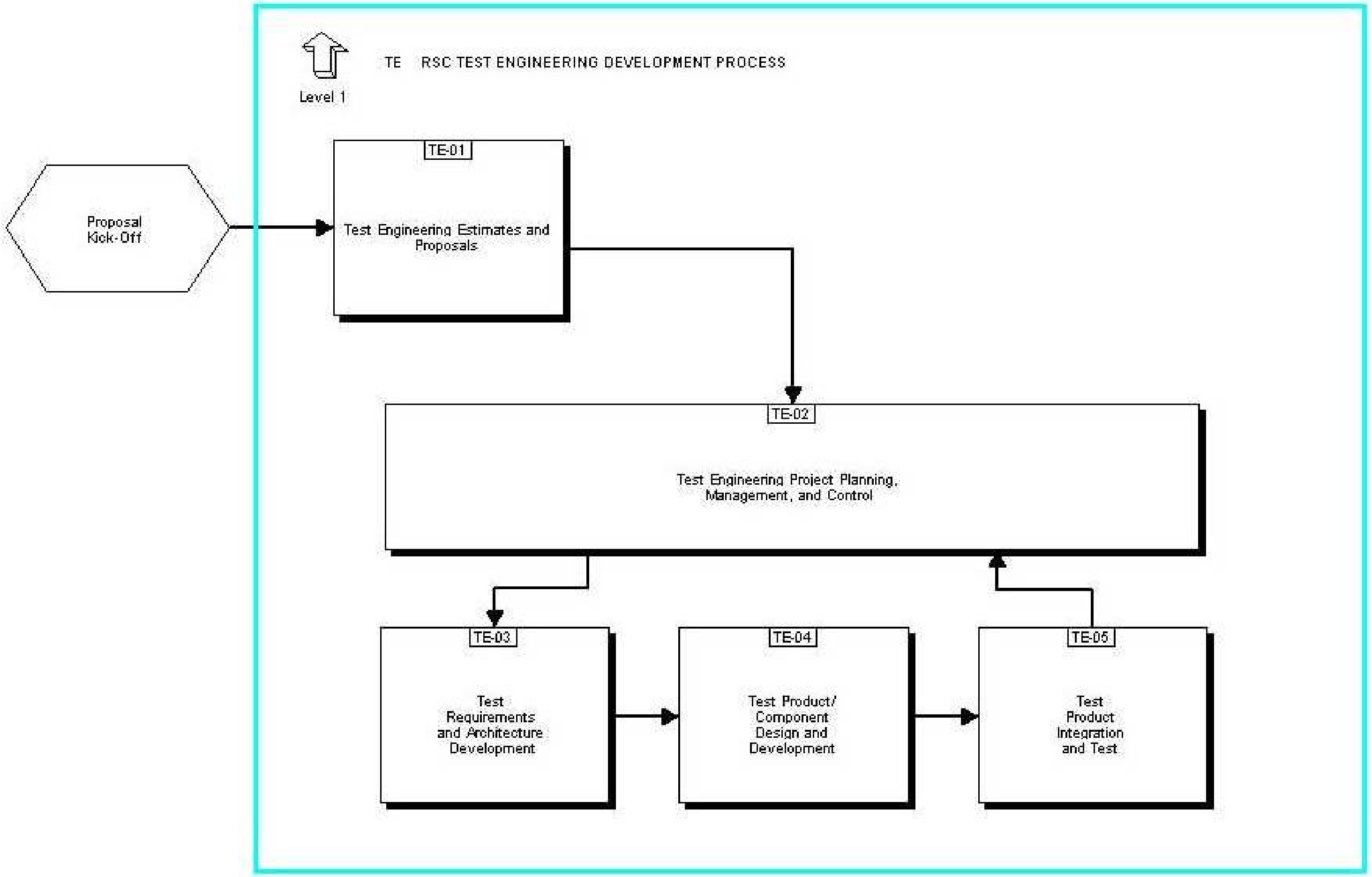

Level 1
TE RSC TEST ENGINEERING DEVELOPMENT PROCESS
Proposal Kick-Off
TE-01
Test Engineering Estimates and Proposals
TE-02
Test Engineering Project Planning, Management, and Control
TE-03
Test Requirements and Architecture Development
TE-04
Test Product/ Component Design and Development
TE-05
Test Product Integration and Test

Figure 8 RSC Test Engineering Development Process [IPDS 2.06]

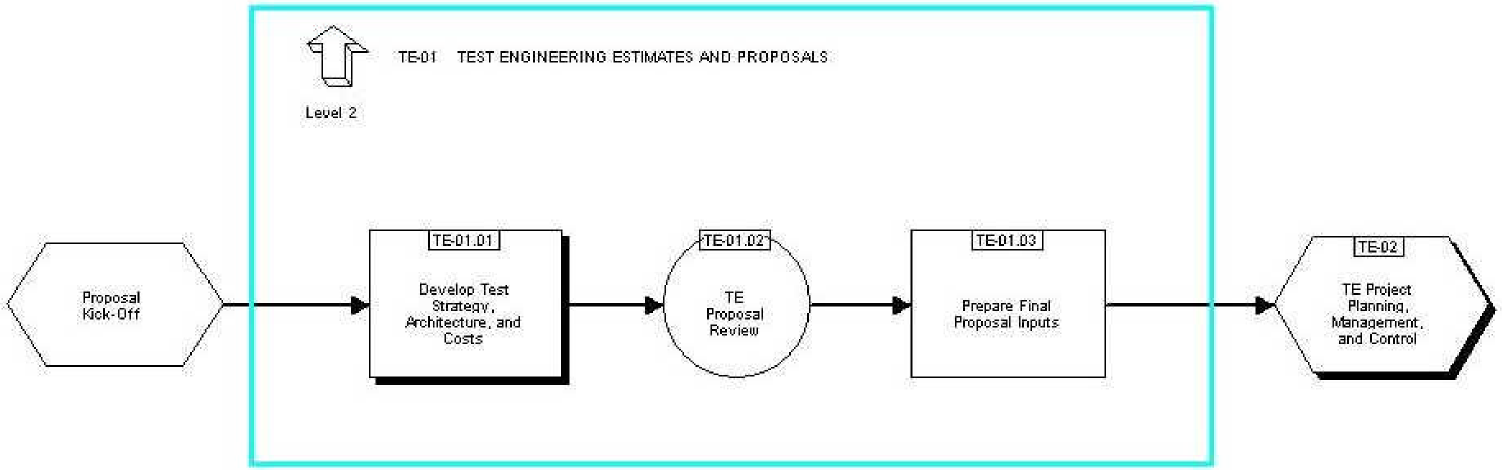

TE-01 TEST ENGINEERING ESTIMATES AND PROPOSALS
Level 2
Proposal Kick-Off
TE-01.01
Develop Test Strategy, Architecture, and Costs
TE-01.02
TE Proposal Review
TE-01.03
Prepare Final Proposal Inputs
TE-02
TE Project Planning, Management, and Control

Figure 9 Test Engineering Estimates and Proposal [IPDS 2.06]

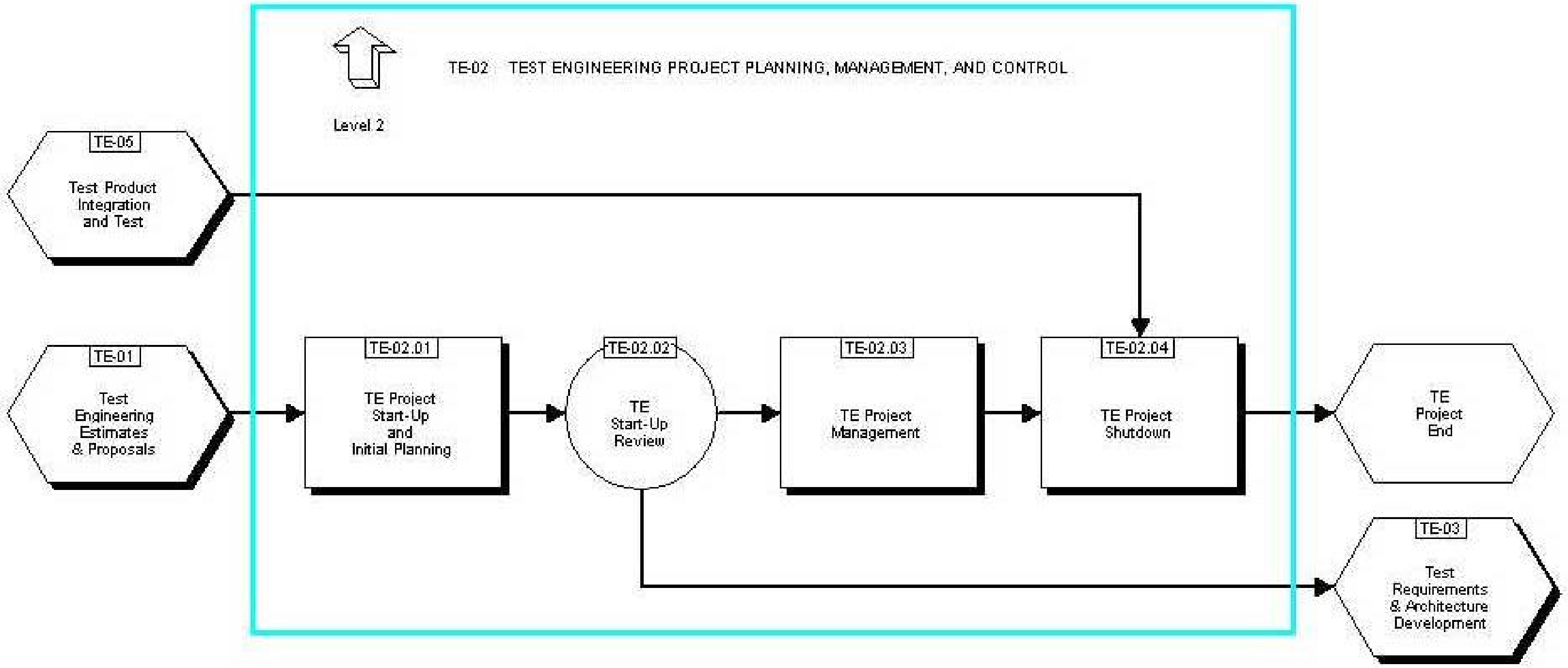
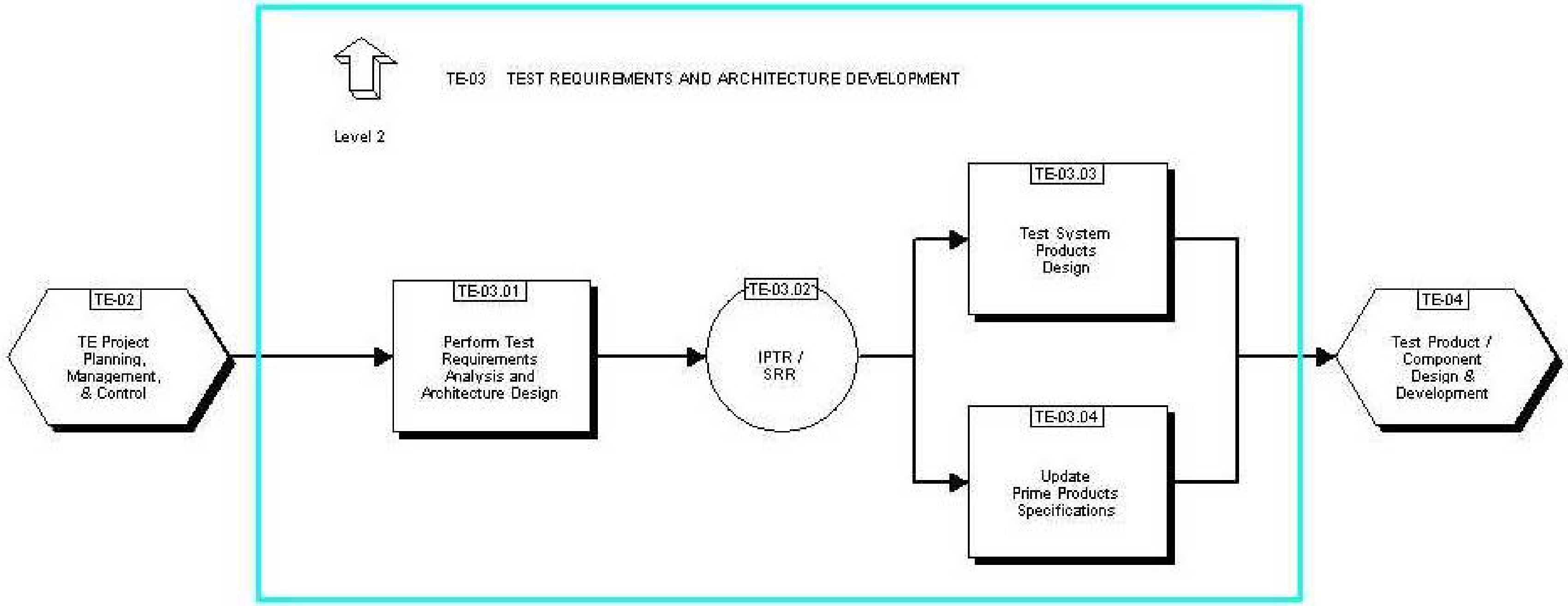

TE-02 TEST ENGINEERING PROJECT PLANNING, MANAGEMENT, AND CONTROL
Level 2
TE-05
Test Product Integration and Test
TE-01
Test Engineering Estimates & Proposals
TE-02.01
TE Project Start-Up and Initial Planning
TE-02.02
TE Start-Up Review
TE-02.03
TE Project Management
TE-02.04
TE Project Shutdown
TE Project End
TE-03
Test Requirements & Architecture Development

Figure 10 Test Engineering Project Planning, Management, and Control [IPDS 2.06]



TE-03 TEST REQUIREMENTS AND ARCHITECTURE DEVELOPMENT
Level 2
TE-02
TE Project Planning, Management, & Control
TE-03.01
Perform Test Requirements Analysis and Architecture Design
TE-03.02
IPTR / SRR
TE-03.03
Test System Products Design
TE-03.04
Update Prime Products Specifications
TE-04
Test Product / Component Design & Development

Figure 11Test Requirements and Architecture Development [IPDS 2.06]

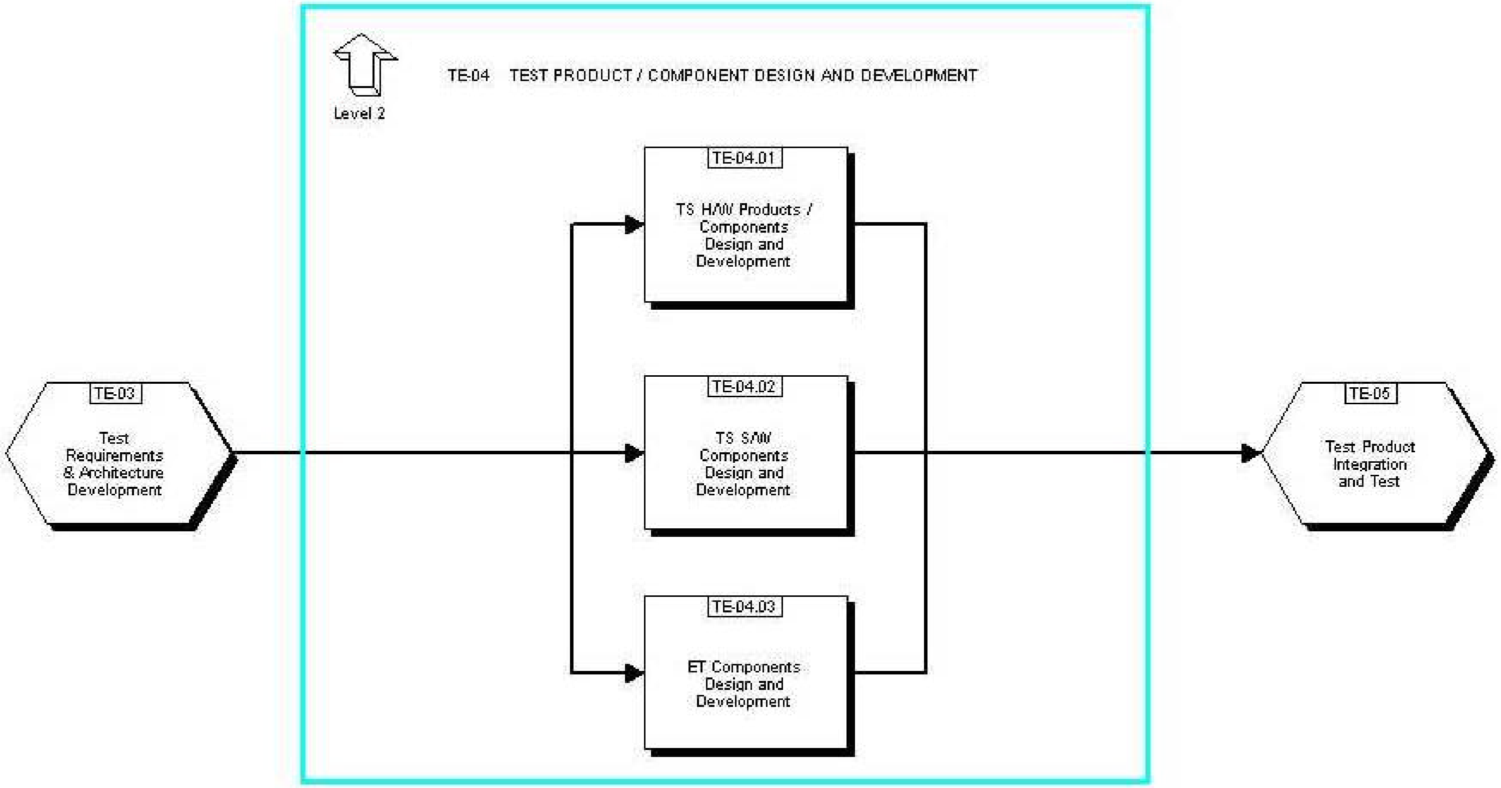

Level 2
TE-04 TEST PRODUCT / COMPONENT DESIGN AND DEVELOPMENT
TE-03
Test Requirements & Architecture Development
TE-04.01
TS H/W Products / Components Design and Development
TE-04.02
TS S/W Components Design and Development
TE-04.03
ET Components Design and Development
TE-05
Test Product Integration and Test

Figure 12 Test Product / Component Desig n and Development [IPDS 2.06]

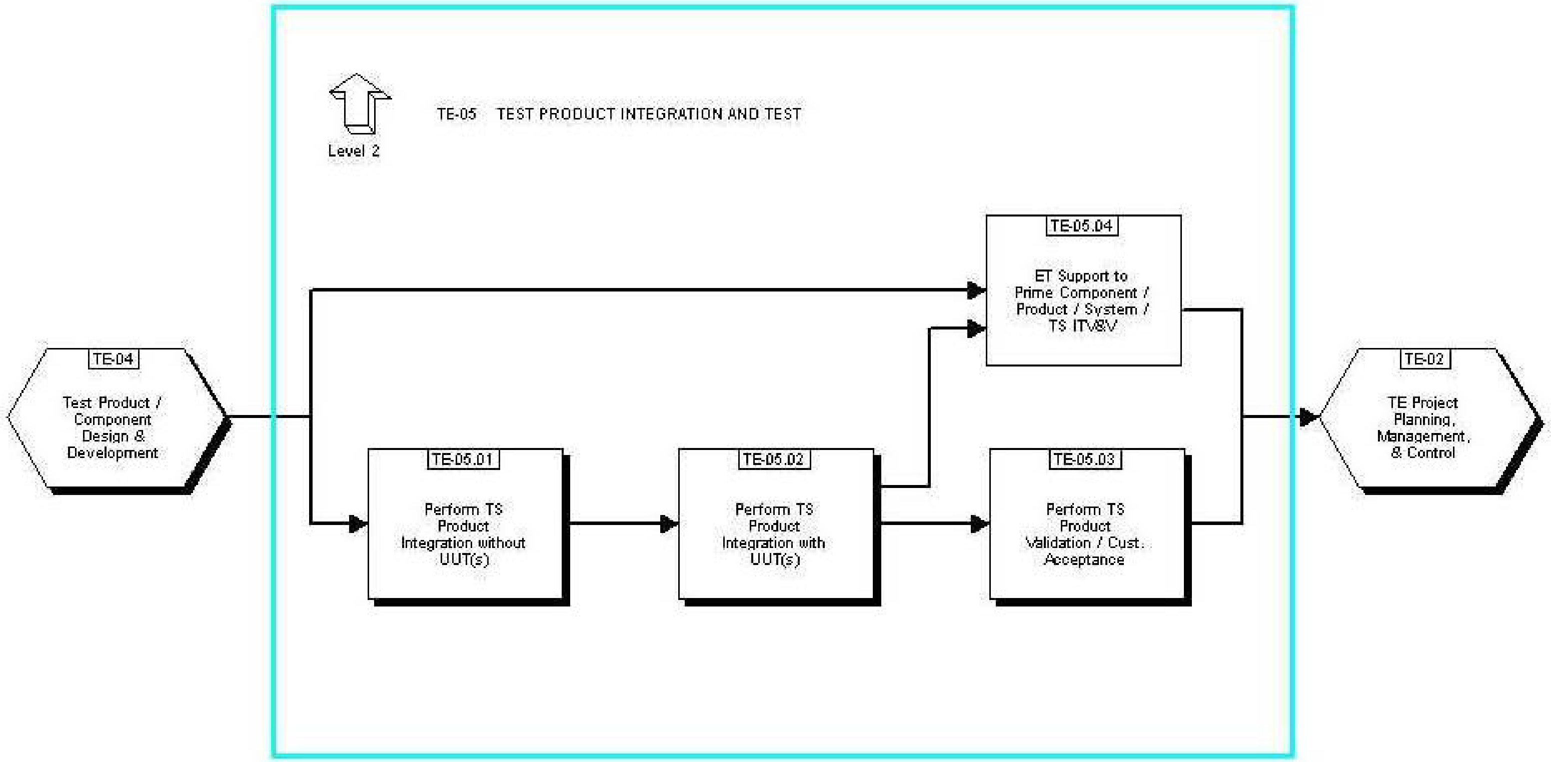

Level 2
TE-05 TEST PRODUCT INTEGRATION AND TEST
TE-04
Test Product / Component Design & Development
TE-05.04
ET Support to Prime Component / Product / System / TS ITV&V
TE-05.01
Perform TS Product Integration without UUT(s)
TE-05.02
Perform TS Product Integration with UUT(s)
TE-05.03
Perform TS Product Validation / Cust. Acceptance
TE-02
TE Project Planning, Management, & Control

Figure 13 Test Product Integration and Test [IPDS 2.06]

The above process is great and is virtually self-documenting given that stage 2 Project Planing, Management and Control have been properly completed. Some of my observations regarding the tailoring of the process include the following:

- Excessive detail in Test Engineering Process Development Process
- Test Engineering Process is clearly a subset of the Systems Engineering but is depicted as a separate set of processes in IPDS.
- Detailed Test Engineering Development Process requires patience to be effective.
- Identification of tailorable steps can be difficult with multiple layers.
- Navigation around IPDS can be difficult and time consuming.

As with any process, there are common errors that many make. Some of these include:

- Lack of planning in the proposal phase which contributes to ineffective forecasting of work load and personnel requirements.
- Lack of training and inability to quickly integrate new engineers.
- Inability to identify and mitigate risk early in development

No matter what tailoring is done within the Test Engineering realm, it is clear that the success of IV&V is dependent on output received from the Requirements development stage – a symptom of compartmentalization. In the past, compartmentalization of skills was common and necessary to address the growing needs of the defense industry with its cutting edge technology. However, now that the commercial industry has caught up an in some cases surpassed the technological advances, many engineers have migrated to the commercial industry leaving a void to fill. This is the major driver for cross training of individuals and de-compartmentalization.

CHAPTER 5
GENERIC DESIGN SCIENCE

5 Introduction

This chapter shall explore the use of methods covered in the Science of Generic Design to design the SE IV &V course material. In exploring the use of various Science of Generic Design principles, many activities will be conducted. These activities include:

- Identify a specific topic which is of Class B or Class C
- Identify the "significant" options, variables, items, components, etc.
- Identify structure of options
- Name categories
- Identify the design dimensions (structure options)
- Discover clusters (structure dimensions, interactions, interrelations)
- Choose sequence for clusters
- Choose sequence for dimensions in clusters
- Display results on an options Field

The intent of a Science of Generic Design is to accommodate very large class systems. These systems are called sociotechnical systems. A subset of these systems are technological systems which can be broken down into the following three classes:

> "The Class A component of this partition consists of members that are clearly founded in physical science. Among the examples of these are radio, television, laser and maser technology, semiconductor chips, electrical motors and generator, transmission lines, telephones, airplane wings and control systems, automated chemical plants, and internal combustion engines.
>
> The Class B component of this partition consists of members that are sometimes referred to as "intellectual technical" or products of "artificial intelligence". Examples of this Class include computer software, and textbooks about computer softw are, computer languages, and that portion of the physical layout of human living

> and working environments that has been designed on the basis of some postulated image of human behavior in that environment.
>
> The Class C component of this partition is comprised of a mix of members from Classes A and B, whose satisfactory performance depends on appropriate integration of these two classes into a synergistic units." [Warfield]

For this project, it has been determined that the Systems Engineering Integration Verification and Validation (SE IV&V) curriculum falls into the Class B component. The SE IV&V curriculum because of the textbook nature of technical material are directly related to Class B components.

5.1 Overview

"Science fills or contributes to the satisfaction of two major social needs: the need to know and the need to know now." [Warfield] A science of design satisfies our human need to know how. We can look to the world around us to see if we can detect any difficulties that might be remedied by an improvement in our know-how.

> "Areas in which major social difficulties have arisen, typically are characterized by one or more of the following attributes: loss of life, contamination of the environment due to "accidents" mammoth cost overruns on the projects costing billions of dollars, significant economic loss accruing to individuals because of criminal behavior in enterprises, transportation accidents, huge loans that cannot be repaid, and erosion of confidence in organizations to accomplish their ostensible purposes." [Warfield]

There are many techniques used in the generic design science. In general, the purpose of these techniques is to provoke:

- Generation of ideas
- Clarification (interpretation) of ideas
- Structuring of ideas
- Interpreting Structures of Ideas
- Amending of ideas

Through a combination of methodologies discussed in the course text, one can obtain their final goal. The seven methodologies are listed below and described in section 4 along with appropriate conditions for use.

1. DELTA Charts
2. Ideawriting
3. Nominal Group Technique (NGT)
4. DELPHI
5. Interpretive Structural Modeling (ISM)
6. Options Field Methodology (OFM)
7. Options Profile Methodology
8. Tradeoff Analysis Methodology (TAM)

5.2 Methodologies

A methodology is a sequenced set of process components, each of which is dedicated to attain specific products related to problem solving and design. Methodologies should be common to most humans. No matter what kinds of specific activities occupy their attention.

5.3 DELTA Charts

DELTA charts provide a common representation of Generic Design Science methodologies. Its general applicability is indicated by the acronym as shown below:

"**D**" stands for <u>D</u>ecision (A selection from two or more options represented on the chart.

"**E**" stands for <u>E</u>vent (A specific instant of time that corresponds to the initiation or conclusion of an activity)

"**L**" stands for <u>L</u>ogic Element (The logical operations "AND" and "OR")

"**T**" stands for <u>T</u>ime (A relationship of time precedence)

"**A**" stands for <u>A</u>ctivity (The beginning, duration and end of an activity)

5.4 Ideawriting (Brainwriting)

A design science provides many ways to generate ideas. A Science of Generic Design introduces two brainstorming methods, Ideawriting, and Nominal Group Technique (NGT). These techniques have different purposes and advantages for each. Ideawriting and NGT methodologies described in subsequent sections will be used for the development of the collaboration environment.

Using Ideawriting it is possible to generate many ideas in a short period of time. This methodology can be used where collective idea generation is expected to be valuable. Figure 12 shows the DELTA chart for Ideawriting.

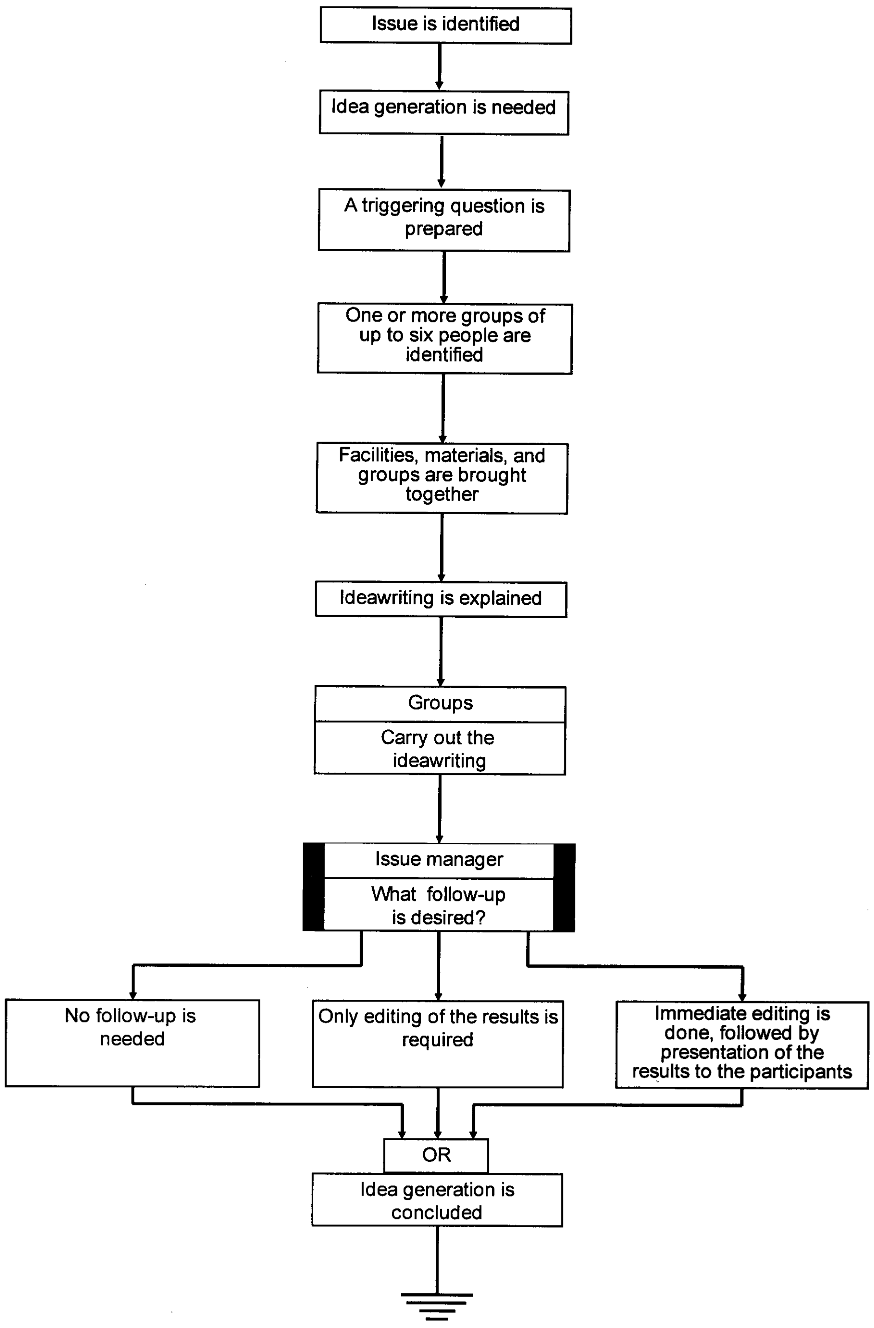

Issue is identified
Idea generation is needed
A triggering question is prepared
One or more groups of up to six people are identified
Facilities, materials, and groups are brought together
Ideawriting is explained
Groups
Carry out the ideawriting
Issue manager
What follow-up is desired?
No follow-up is needed
Only editing of the results is required
Immediate editing is done, followed by presentation of the results to the participants
OR
Idea generation is concluded

Figure 7.12 DELTA chart of ideawriting process. Copyright © 1982 SGSR.

Figure 14 Ideawriting Process DELTA Chart - [Warfield]

In general, this methodology is appropriate for all efforts where collective idea generation in a short period of time is needed and especially useful for issue formulation. Some of the issues include problem definition, and identification of objectives. Ideawriting is a less structured methodology than Nominal Group Technique that allows for greater idea generation. Because of the structure and focus of NGT, Idea writing will not be used in the application section of this project.

5.5 Nominal Group Technique (NGT)

The second type of brainstorming technique is Nominal Group Technique (NGT) has many facets and is more sophisticated than Ideawriting. Not only is it possible to generate ideas but due to its structured nature, it generally achieves more than Ideawriting. NGT is used to:

1. Generating ideas
2. Clarifying ideas
3. Perform a preliminary partitioning of the set of generated and clarified ideas
4. Build a spirit of participation and teamwork or group morale

NGT uses a trigger question to kick off the idea generation process. A good trigger question will encourage contributions from those who are normally quiet. Outcomes of the use of a trigger question include the spontaneous occurrence of ideas triggered by ideas from other contributors. Unlike the Ideawriting method, ideas are silently generated using the NGT process and are not exchanged among the working group members. The ideas are then addressed individually as the facilitator records the ideas on flip chart pages and post them on the wall. This process of addressing the ideas continues in order to clarify and consolidate ideas.

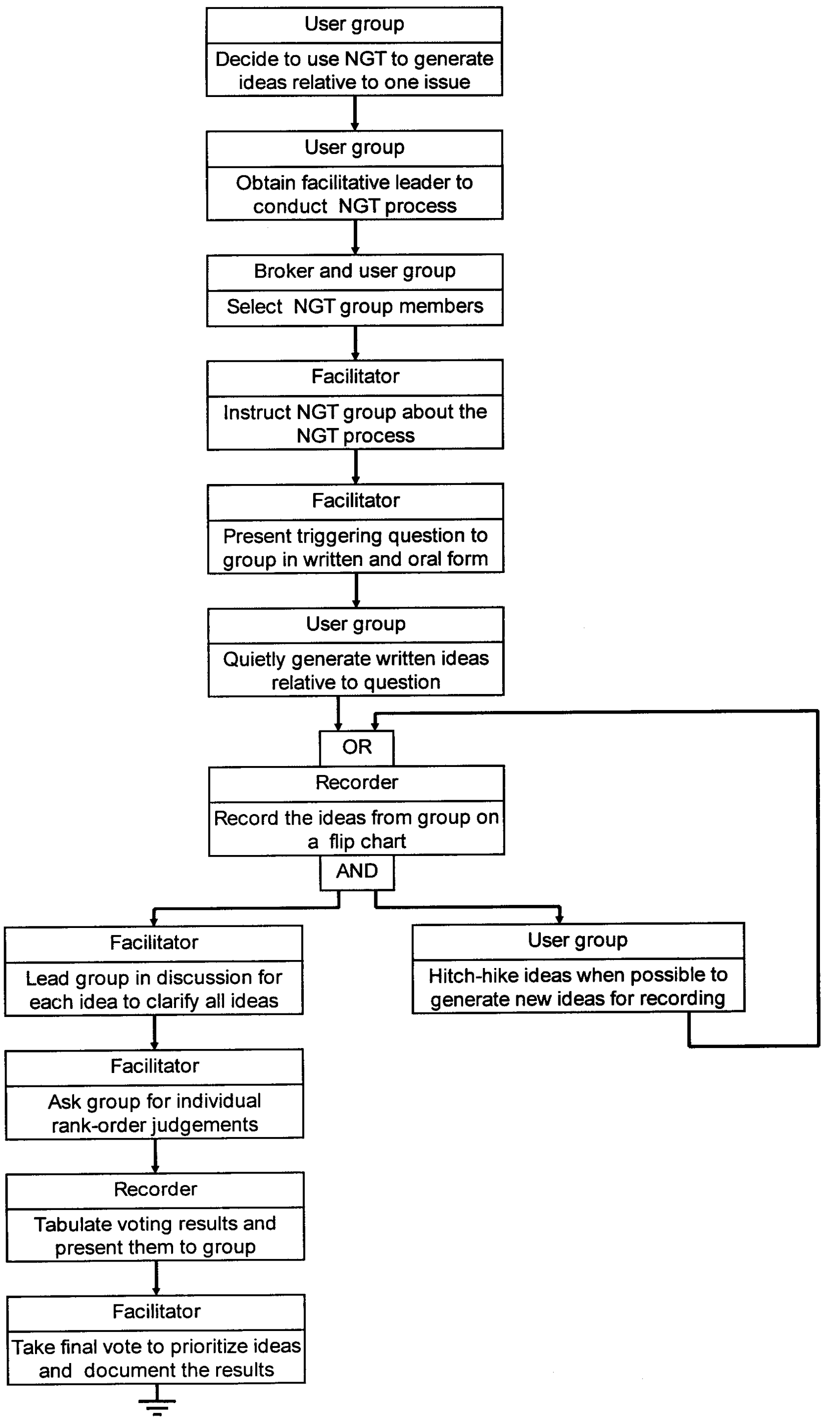

User group
Decide to use NGT to generate ideas relative to one issue
User group
Obtain facilitative leader to conduct NGT process
Broker and user group
Select NGT group members
Facilitator
Instruct NGT group about the NGT process
Facilitator
Present triggering question to group in written and oral form
User group
Quietly generate written ideas relative to question
OR
Recorder
Record the ideas from group on a flip chart
AND
Facilitator
Lead group in discussion for each idea to clarify all ideas
User group
Hitch-hike ideas when possible to generate new ideas for recording
Facilitator
Ask group for individual rank-order judgements
Recorder
Tabulate voting results and present them to group
Facilitator
Take final vote to prioritize ideas and document the results

Figure 7.13 DELTA chart of nominal group technique. Copyright © 1982 SGSR.

Figure 15 Nominal Group Technique Process DELTA Chart [Warfield]

The NGT was used to generate and clarify ideas regarding improvement of the SE IV&V curriculum.

5.6 Interpretative Structural Modeling (ISM)

> "In application, ISM provides the means to formulate a pattern or structure of elements associated with issue formulation. The element may include needs, constraints, objectives or options in a variety of fields such as education, public facility planning, city budget-cutting, or system design." [Warfield]

Interpretive Structural Modeling (ISM) is used to probe issues with interactions among diverse elements, focus group discussion on a specific issue and development of a multi-level relation map. The resources ISM uses to accomplish these tasks include the use of a computer-assisted learning process.

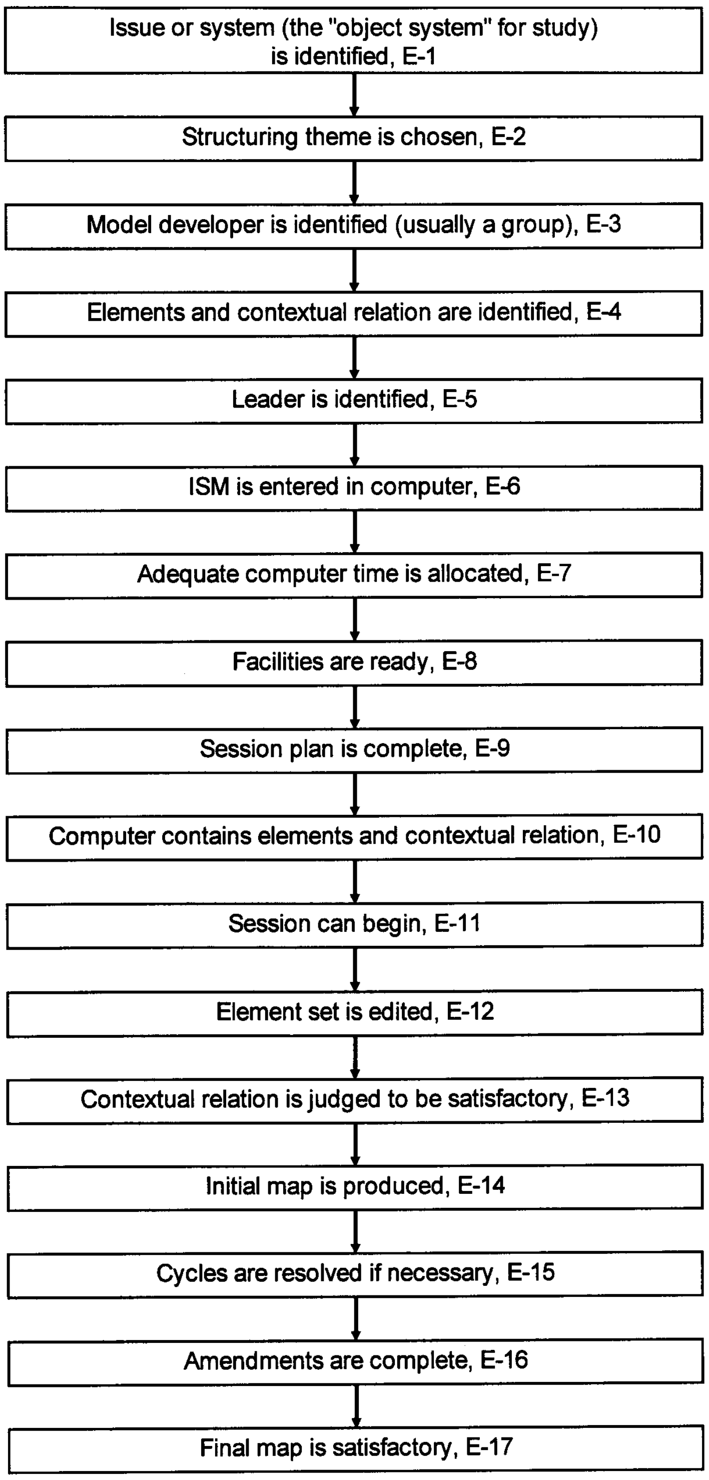

Issue or system (the "object system" for study) is identified, E-1
Structuring theme is chosen, E-2
Model developer is identified (usually a group), E-3
Elements and contextual relation are identified, E-4
Leader is identified, E-5
ISM is entered in computer, E-6
Adequate computer time is allocated, E-7
Facilities are ready, E-8
Session plan is complete, E-9
Computer contains elements and contextual relation, E-10
Session can begin, E-11
Element set is edited, E-12
Contextual relation is judged to be satisfactory, E-13
Initial map is produced, E-14
Cycles are resolved if necessary, E-15
Amendments are complete, E-16
Final map is satisfactory, E-17

Figure 7.16 DELTA chart of ISM process.

Figure 16 Interpretive Structural Modeling (ISM) Process DELTA Chart [Warfield]

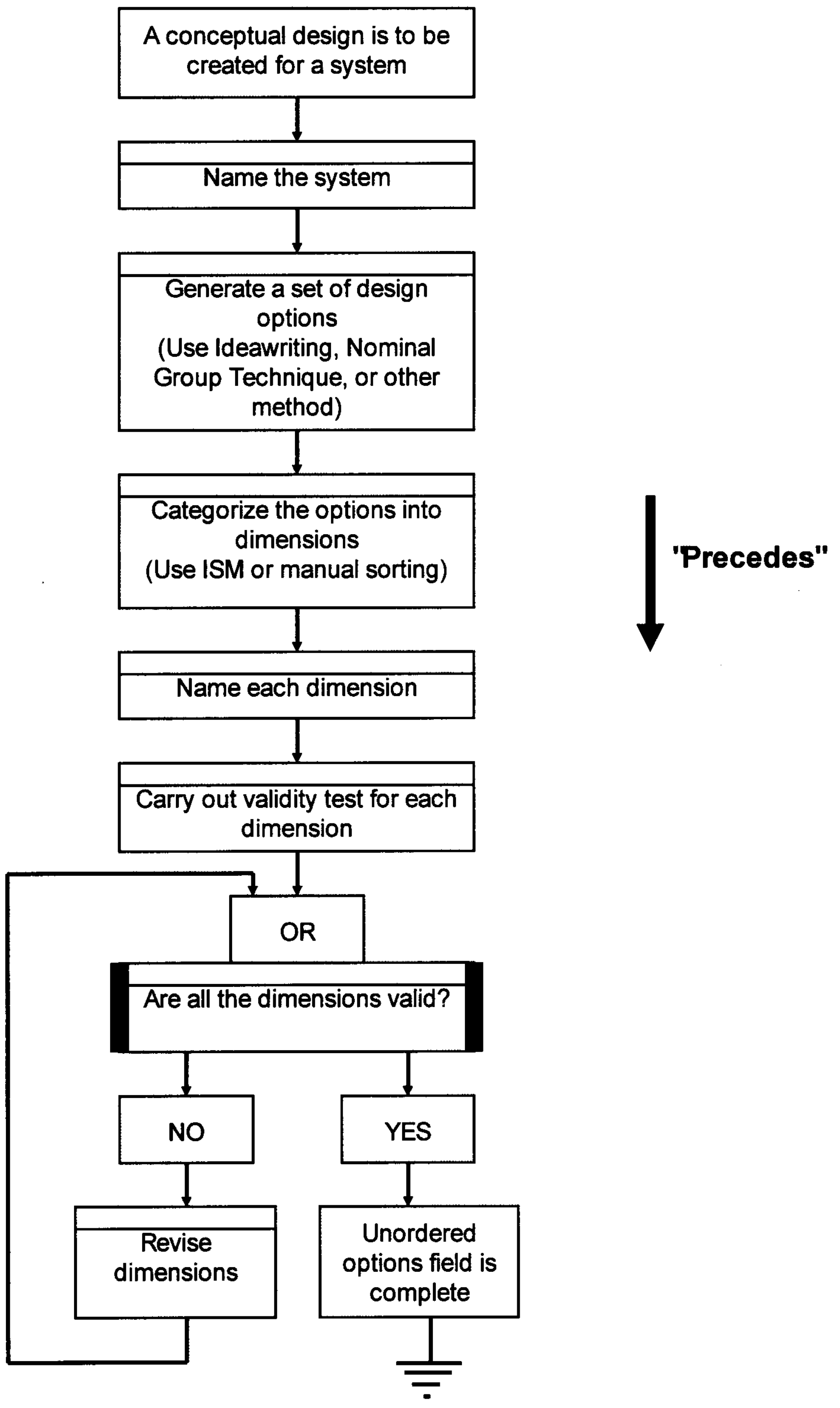
Among the resources required are the following:

1) A set of elements relating to the issue
2) A time-shared digital computer that contains the programs for structuring
3) A contextual relationship, which is appropriate to interrelate the elements
4) Up to 8 willing and able participants, as well as a group leader familiar with interpretive structural modeling, and a computer operator.

Aside from the lack of resources, this technique can be performed on a small scale in the application section of the paper.

5.7 Options Field Methodology (OFM) / Options Profile Methodology (OPM)

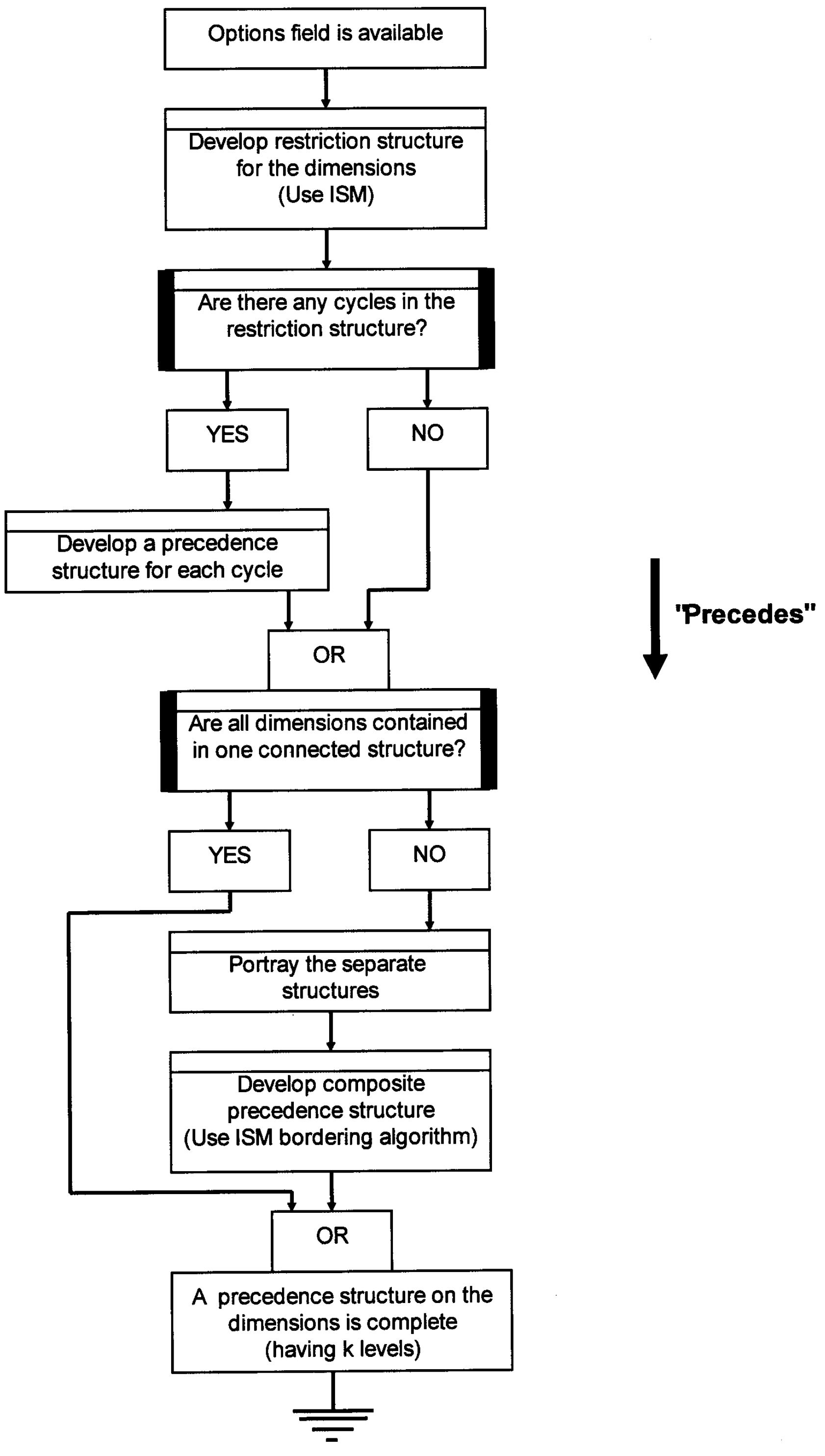
The Options Field Methodology and the Options Profile Methodology are closely tied together. These methodologies provide a way to thoroughly develop various Design Situation descriptions and design Target descriptions. OFM and OPM involve discovery and identification of the dimensions of a situation. Once the dimensions are identified, OFM and OPM facilitate the corresponding of dimensionality of the Target with the dimensionality of the Design Situation. This satisfies the Law of Requisite Variety. NGT and ISM are components of OFM and OPM.


A conceptual design is to be created for a system
Name the system
Generate a set of design options (Use Ideawriting, Nominal Group Technique, or other method)
Categorize the options into dimensions (Use ISM or manual sorting)
Name each dimension
Carry out validity test for each dimension
OR
Are all the dimensions valid?
NO
YES
Revise dimensions
Unordered options field is complete
"Precedes"

(a)

Figure 7.20 DELTA chart of options field methodology. (Continued on next page.)

Figure 17 Options Field Methodology (OFM) Process DELTA Chart Part 1 [Warfield]


Options field is available
Develop restriction structure for the dimensions (Use ISM)
Are there any cycles in the restriction structure?
YES
NO
Develop a precedence structure for each cycle
OR
Are all dimensions contained in one connected structure?
YES
NO
Portray the separate structures
Develop composite precedence structure (Use ISM bordering algorithm)
OR
A precedence structure on the dimensions is complete (having k levels)
"Precedes"

(b)

Figure 7.20 DELTA chart of options field methodology. (Concluded.)

Figure 18 Options Field Methodology (OFM) Process DELTA Chart Part 2 [Warfield]

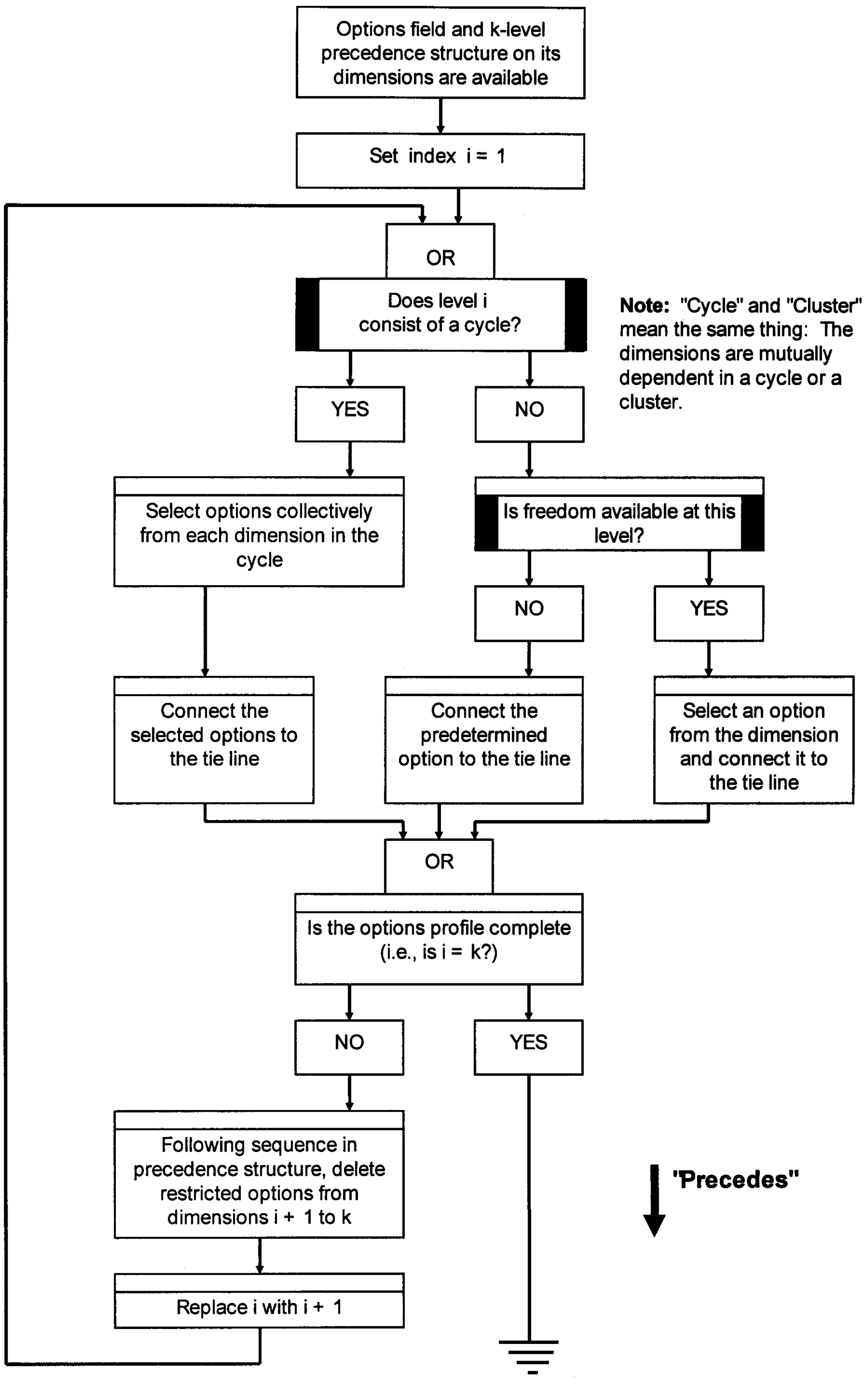

Options field and k-level precedence structure on its dimensions are available
Set index i = 1
OR
Does level i consist of a cycle?
Note: "Cycle" and "Cluster" mean the same thing: The dimensions are mutually dependent in a cycle or a cluster.
YES
NO
Select options collectively from each dimension in the cycle
Is freedom available at this level?
NO
YES
Connect the selected options to the tie line
Connect the predetermined option to the tie line
Select an option from the dimension and connect it to the tie line
OR
Is the options profile complete (i.e., is i = k?)
NO
YES
Following sequence in precedence structure, delete restricted options from dimensions i + 1 to k
Replace i with i + 1
"Precedes"

Figure 7.22 DELTA chart of options profile methodology. Copyright © 1982 SGSR

Figure 19 Options Profile Methodology (OPM) Process DELTA Chart [Warfield]

In order to conduct an OFM/OPM certain conditions must exist. In the case of the OFM, the most important condition is that there must be a requirement to develop a design. Next, a group generally consisting of 6 or more participants must agree on the requirement to develop a top down design with various design choices. In addition, it is also important to maintain continuous display of pas design decisions. OPM contains similar conditions for use in addition to having a facilitator is available to lead the group design effort.

Although the textbook description of this methodology recommends 6 or more participants, through imagination and creativity, it is possible to generate a sufficient number of options.

5.8 Tradeoff Analysis Methodology (TAM)

After applying select methodologies to identify options, the Tradeoff Analysis Methodology (TAM) offers a means of choosing systematically one alternative. This methodology may also use the NGT process as a component to develop criteria for making choices. The ISM may also be used as a component to prioritize those criteria.

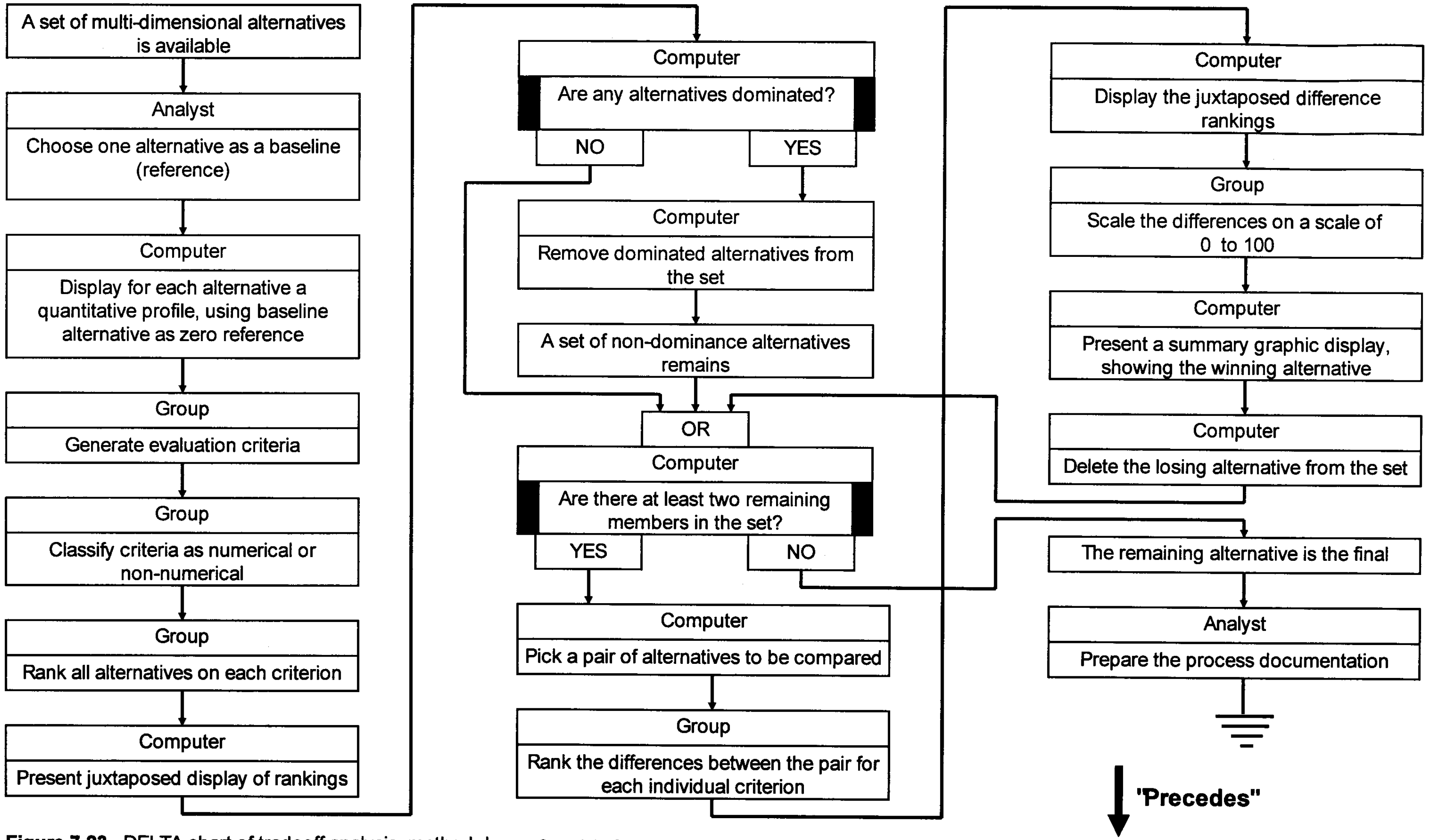

A set of multi-dimensional alternatives is available
Analyst
Choose one alternative as a baseline (reference)
Computer
Display for each alternative a quantitative profile, using baseline alternative as zero reference
Group
Generate evaluation criteria
Group
Classify criteria as numerical or non-numerical
Group
Rank all alternatives on each criterion
Computer
Present juxtaposed display of rankings
Computer
Are any alternatives dominated?
NO
YES
Computer
Remove dominated alternatives from the set
A set of non-dominance alternatives remains
OR
Computer
Are there at least two remaining members in the set?
YES
NO
Computer
Pick a pair of alternatives to be compared
Group
Rank the differences between the pair for each individual criterion
Computer
Display the juxtaposed difference rankings
Group
Scale the differences on a scale of 0 to 100
Computer
Present a summary graphic display, showing the winning alternative
Computer
Delete the losing alternative from the set
The remaining alternative is the final
Analyst
Prepare the process documentation
"Precedes"

Figure 7.23 DELTA chart of tradeoff analysis. methodology. Copyright © 1982 SGSR.

Figure 20 Tradeoff Analysis Methodology (TAM) Process DELTA Chart [Warfield]

The TAM is a relatively complex technique that when used under the proper conditions yield a decision between complex alternative solutions. Some application areas of this methodology include environmental decision, involving forestry, and large investment decisions, where there are competing alternatives.

Although this technique would be useful in virtually any design, due to the lack of resources such as special-purpose software to facilitate the use of the method and generation of the appropriate displays, results of the TAM from our generic design can only be hypothesized. Additionally, to obtain a well rounded decision on alternatives and allow for full development of issues or topics, a more diverse group of domain experts would be needed.

5.9 Application of Select Methodologies

5.9.1 Nominal Group Technique (NGT)

Following the NGT process we must first generate a trigger question.

Trigger Question: What problems or issues must be dealt with in the future in order to make SE IV&V highly ineffective in providing a background for training future Test engineers?

- Not give enough examples
- Perspective is more of a Lead/project management scope
- Does not identify how to tailor the process to meet project specific needs
- Does not address how to deal with standards or where to find them
- Does not clearly address roles and responsibilities
- Does not address special concerns of COTS and GOTS integration
- Does not identify participation in requirement decomposition to provide a set of requirements, which are bounded, complete and testable.
- Does not show clear examples of VTM generation
- Needs to detail test plans and planing responsibilities no checklist
- Needs to detail complete test activities
- Does not adequately address test metrics
- Does not address change management and discrepancy reporting processes
- Does not identify possible contingency plans when schedule gets pulled tight
- Does not address the needs of cross training to build test team knowledge
- Does not address support needs at each phase of testing for multiple build complex products.
- Does not adequately address test cases, implementation, contents and use
- Does not describe the process behind each step in the test process
- Does not address reliability
- Does not address availability
- Security concerns
- Life cycle concerns

- Automated testing details
- Software testing techniques
- Dealing with fast turn around needs in complex systems
- Identifying a clear test process and proper implementation and tailoring options

Ranking these in order the five most important topics are:

- Identifying a clear test process, proper implementation and tailoring
- Not give enough examples
- Perspective is more of a Lead/project management scope
- Does not identify how to tailor the process to meet project specific needs
- Does not clearly address roles and responsibilities

5.9.2 Options Field Methodology (OFM) / Options Profile Methodology (OPM)

From the list of ideas generated through NGT, we have categorized them into Clusters, Dimensions and Options as shown in Table 4 SE IV&V Field / Option Profile below.

As shown in the table the dimensions, and clusters are sequenced in the following manner:

DIMENSIONS	CLUSTERS
A1. Exercises	Course Data
A2. Course Perspective	Test Activity
B1. Process	Resources
B2. Engineering Activities	
C1. Team Members	
C2. Resources	
C3. Support Material	
D. Special Concerns	Miscellaneous

The Options Field shows the optimum set of curriculum changes to obtain the objective of providing a better course for a diverse group test engineers. As you can see, this method lends itself to the generation of a modular set of topics which can be tailored to the audience. It is important to note here that sufficient detail in all sections would create a very large course and for this reason, it may be better to provide a complete overview with references to more details.

Table 4 SE IV&V Field / Option Profile

COURSE DATA

A1. Exercises
- Metrics
- Checklists
- Test Plans
- Test Procedures
- Process on a small scale
- Test Case

A2. Course Perspective
- Lead
- Project manager
- Roles and responsibilities
- Engineer

TEST ACTIVITY

B1. Process
- Change Management
- Discrepancy reporting
- Software turnover
- Test Process
- Planning activities

B2. Engineering Activities
- Engineering activities
- Requirements Review
- Test Plans
- Test Procedures
- Test case
- Automated testing

TIE LINE

RESOURCES

C1. Team Members
- Program Manager
- Chief/Test Engineer
- Facilities
- Systems Engineer
- Software Engineer
- Hardware Engineer
- Infrastructure Engineer
- Procurement
- Finance
- Business Development
- Configuration Management
- Quality
- Customer
- Data Management
- Contracts
- Legal
- Misc. Group Participants

C2. Resources
- Personnel
- Scheduling
- Training
- Cross training

C3. Support Material
- Standards
- COTS/GOTS
- Web sites
- Certifications

MISCELLANEOUS

D. Special Concerns
- Availability
- Reliability Maintainability
- Life cycle
- Security

C H A P T E R 6
CONCLUSION

In the design of a course curriculum, it is important to consider all the factors that go into Systems Engineering Integration, Verification and Validation. Through observation and an initial brainstorming activity in the NGT, it was revealed that many modifications to the curriculum were needed. The NGT permitted the generation of many options. The selection of the optimum design approach was selected in order to provide the most efficient and cost-effective solution. Measuring the effectiveness of the modifications can be done through future evaluations of the course presentation.

The steps of the Systems Engineering process can be applied to any project – even writing a training curriculum. After going through the stages of the Systems Engineering process a set of verifiable requirements along with an IMP and IMS were generated. The development of these artifacts demonstrates the Systems Engineering Process.

Obstacles encountered during the development of the course included obtaining resources, coordination of schedules and most of all, including enough reference material and examples to allow students to apply what was learned. Most importantly, this study demonstrated the diversity of Systems Engineering as a discipline and reveals a need for training engineers in a variety of disciplines. It is important for each engineer to have an awareness of the complete Systems Engineering process as utilized and applied in a given program permits the growth of a strong organization along with a mature product.

Often, as a symptom of growth, compartmentalization is necessary to meet immediate needs, however, in the long run, cross-training of individuals provides stability for the future organization and minimizes the number of failure points. In addition, removing the culture of compartmentalization, allow many engineers experience growth in diverse areas of systems engineering. This is idea lends itself to many benefits including giving individuals a sense of the "big picture" and team values.

Overall, the outcome of IV & V is dependent on the outputs of the requirements design and development. Without sound requirements, testing of any kind is nearly impossible. Any short comings of requirements development must be caught in the early stages of IV & V otherwise any testing that is to be accomplished is only a formality and not truly verifying or validating the product.

R E F E R E N C E S

[Montgomery] Montgomery, Douglas C., 1996 Introduction to Statistical Quality Control, Third Edition, John Wiley & Sons, Inc., New York, Chichester, Brisbane, Toronto, Singapore

[Blanchard/Fabrycky] Blanchard and Fabrycky, 1981,1991, Systems Engineering and Analysis, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey

[Kaner] http://www.kaner.com
(In this web-site, in the white papers there were numerous white papers regarding the testing process including automated testing pros and cons. Some of these were used when collecting background material for the project).

[UNISA] http://louisa.levels.unisa.edu.au/se1/testing-notes/test01.html
(In this web-site, these notes were used when collecting background material for the project).

[SQA-Test] http://www.sqa-test.com/articles.html
(In this web-site, there were links to numerous articles regarding testing techniques and automated testing)

[Logica] http://www2.logica.com/papers/gui-testing.html
(In this web-site, in the white papers there were numerous white papers regarding the testing process including automated testing pros and cons. Some of these were used when collecting background material for the project).

[Dacsdtic] http://www.dacsdtic.mil/awareness/newslettters
(In this web-site, there were many links to articles containing software testing related topics including software reliability. Some of these were used when collecting background material for the project.)

Blanchard & Fabrycky, 1990,1981. *Systems Engineering and Analysis*, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey

Raytheon, 1997. *Systems Engineering Manual*, Revision 2.1.

Raytheon, 2001. *Introduction to Systems Engineering Course 5000*

Raytheon, 2000. *Integrated Product Development System*, Release 2.06