

DESIGN, VERIFICATION, AND VALIDATION PLAN

FOR THE

SOUTH TEXAS PROJECT

QUALIFIED DISPLAY PROCESSING SYSTEM

Design Specification Number 955842

Rev. 3

8509300185 850924
PDR ADOCK 05000498
F PDR

TABLE OF CONTENTS

- 1.0 Introduction
 - 1.1 Purpose
 - 1.2 System Functions
 - 1.2.1 Data Acquisition and Display
 - 1.2.2 Qualified Control
 - 1.2.3 Steam Generator Water Level Compensation/Temperature Averaging Scheme
 - 1.3 System Architecture
- 2.0 References
- 3.0 Definitions
- 4.0 System Development
- 5.0 System Verification
 - 5.1 Introduction
 - 5.2 Verification Philosophy
 - 5.3 Verification Techniques
 - 5.3.1 Reviews
 - 5.3.1.1 Design Specification Reviews
 - 5.3.1.2 Source Code Reviews
 - 5.3.1.3 Functional Test Reviews
 - 5.3.2 Software Testing
 - 5.3.2.1 Structural Testing
 - 5.3.2.2 Functional Testing
 - 5.4 The Verification Matrix
 - 5.4.1 Classification
 - 5.4.2 Demonstrability of System Functions
 - 5.4.3 Hierarchical Level of Software Components

5.4.4 Justification of Matrix Elements

5.4.5 Application of the Matrix to the QDPS

6.0 System Validation

6.1 Validation Philosophy

6.2 Validation Testing Overview

7.0 Organization

7.1 Development Team

7.1.1 Chief Programmer

7.1.2 Programmers

7.2 Verification Team

7.2.1 Chief Verifier

7.2.2 Verifiers

7.2.3 Librarian

1.0 INTRODUCTION

1.1 Purpose

The purpose of this plan is to provide a description of the design, verification, and validation process and the general organization and activities that are being used in these areas in the Qualified Display Processing System for South Texas. The ideas contained herein are modeled after the guidance provided in (a) the 414 Integrated Protection System Prototype Verification Program, which was presented to the NRC in 1977 as part of the Westinghouse RESAR 414 system and (b) ANSI/IEEE-ANS-7-4.3.2-1982.

1.2 System Functions

The Qualified Display Processing System (QDPS) for South Texas Units 1 and 2 performs three major functions:

1. Data acquisition and qualified display per Regulatory Guide 1.97.
2. Qualified Control of valves necessary to maintain the plant in a safe shutdown condition.
3. Temperature compensation of the steam generator narrow range level signals and calculation of reactor coolant system hot leg temperature average per loop.

The system is composed of three subsystems, each of which is dedicated to performing one of the functions listed above.

1.2.1 Data Acquisition and Display

To perform the data acquisition and display function, a subsystem referred to as the Plant Safety Monitoring System (PSMS) is used. PSMS consists of Remote Processing Units (RPUs), Database Processing Units (DPUs), an Auxiliary Shutdown Panel Demultiplexing Unit (ASP Demux), a Main Control Board Demultiplexing Unit (MCB Demux), a Recorder Demultiplexing Unit, and eight plasma displays.

The Remote Processing Unit (RPU) is an intelligent, qualified, modular data gathering unit. The RPU can receive inputs from process sensors and from other qualified systems such as the process protection racks. In addition, the RPU can receive inputs from other qualified digital systems via datalink. The RPU samples the input data, converts it to process units, formats the data and transmits it to the Database Processing Units and to the Emergency

Response Facility computer System. RPU communication is via isolated, redundant RS-422 data links.

The Database Processing Unit (DPU) is an intelligent microprocessor based unit that receives isolated data link inputs from each RPU and uses the data to provide:

- Analog outputs to drive conventional indicators and recorders
- Contact outputs to provide qualified status information
- Datalink outputs to drive the display modules and recorder demultiplexer

The DPU performs redundant sensor algorithms and other necessary calculations. Each RPU transmits its data to both DPUs in the system. Therefore, each DPU maintains the entire database. Either DPU can provide the operator with the best information available from the entire data base. Comparison of the data between DPUs prevents the possibility of erroneous display information.

The display modules are qualified graphic/alphanumeric devices which provide comprehensive displays without requiring large amounts of control board space. Each display module interfaces to both Database Processing Units to provide full redundancy and meet single failure requirements. The display itself is a 512 by 512 dot matrix plasma display (with an active area of approximately 8.5 inches square) providing a flat screen, extreme ruggedness, and easy to read orange-on-black images that are flicker free.

The display module is human engineered for ease of operation and provides functional pushbuttons for individual display selections. The functional keys allow the operator to move easily from one page to another to display specific information.

During the design of the graphic display pages, South Texas Nuclear Plant Operations personnel have been involved in the establishment of criteria and mimic design. A checklist was developed that was based upon a review of the Westinghouse Owner's Group Emergency Response Guidelines (ERG's) which stipulated the plant

process variable characteristics that must be displayed. Examples of variable characteristics determined by the review of the guidelines include the following: value, range, prediction, trend, and pattern recognition. Grouping the variables and factoring in the location of the QDPS displays on the control board determined the structure of the displays that needed to be developed. Many iterations occurred with the South Texas Operations Department during this development process. Utilizing the plant ERGs and the input from operations personnel ensures that the QDPS will present the data in a clear and concise form.

1.2.2 Qualified Control

The second subsystem, Qualified Control, performs the valve control functions, including control for the steam generator Power Operated Relief Valves (PORV), the Auxiliary Feedwater Throttle Valves, and the Reactor Vessel Head Vent Valves. The control units will be channel oriented in exactly the same way as the RPUs and will share the same cabinets with the RPUs. However, operation of the control unit will be independent of the RPU.

1.2.3 Steam Generator Water Level Compensation/Temperature Averaging Scheme

The third subsystem includes the Steam Generator Water Level Compensation Subsystem (SGWLCS) and the Temperature Averaging Scheme (TAS). The SGWLCS temperature compensates steam generator narrow range water level channels for level induced errors as a result of changes in reference leg temperature. The TAS calculates the loop average hot leg temperature from the three narrow range fast response RTD's per loop. The SGWLCS/TAS subsystems are housed in the same cabinets with the RPUs and the Qualified Control units. Like the Qualified Control subsystem, the operation of SGWLCS/TAS are independent of the RPU.

1.3 System Architecture

A block diagram of the QDPS system is shown in Figure 1. Each of the hardware units depicted contains one or more microprocessors. Each Auxiliary Process Cabinet (APC) shown on the left of Figure 1 contains an RPU, a Qualified Control unit and a SGWLCS/TAS unit as shown in Figure 2. Figures 1 and 2 illustrate the various inputs, outputs and data communication paths associated with the QDPS subsystems.

2.0 REFERENCES

The following is a list of relevant industrial standards which were considered in the development of this plan:

1. ANSI/IEEE-ANS-7-4.3.2.-1982, "Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Generating Stations"
2. IEEE Std. 279-1971, "Criteria for Protection Systems for Nuclear Power Generating Stations"
3. IEEE Std. 603-1980, "Criteria for Safety Systems for Nuclear Power Generating Stations"
4. WCAP 9153, "414 Integrated Protection System Prototype Verification Program," Westinghouse Electric Corp., August 1977.
5. WCAP 9740, "Summary of the Westinghouse Integrated Protection System Verification and Validation Program," Westinghouse Electric Corp., September 1984.
6. Regulatory Guide 1.97, Rev. 2, "Instrumentation for Light-Water-Cooled Nuclear Power Plants to Assess Plant and Environs Conditions During and Following an Accident," December 1980
7. ANSI/ASME NQA-1-1983, "Quality Assurance Program Requirements for Nuclear Power Plants"
8. IEEE Std 729-1983, "Standard Glossary of Software Engineering Terminology"
9. IEEE Std 730-1981, "Standard for Software Quality Assurance Plans"
10. IEEE Std 828-1983, "Standard for Software Configuration Management Plans"
11. IEEE Std 829-1983, "Standard for Software Test Documentation"
12. IEEE Std 830-1984, "Guide to Software Requirements Specifications"
13. NBS Special Publication 500-75 (February 1981), "Validation, Verification and Testing of Computer Software"
14. NBS Special Publication 500-93 (September 1982), "Software Validation, Verification, Testing Technique and Tool Reference Guide"
15. NBS Special Publication 500-98 (November 1982), "Planning for Software Validation, Verification and Testing"
16. IEC SC 45A/WG-A3 (January 1984), "Draft: Software for computer in the Safety System of Nuclear Power Stations"

3.0 DEFINITIONS

The definitions in this section establish the meaning of words in the context of their use in this plan.

COMPUTER SOFTWARE BASELINE - The computer program, computer data and computer program documentation which comprises the complete representation of the computer software system at a specific stage of its development.

DEVELOPMENT TEAM - A team of individuals or an individual assigned to design, develop and document software as outlined in this plan. Generally it comprises team coordinator, programmers and a librarian.

DESIGN REVIEW - A meeting or similar communication process in which the requirements, design, code, or other products of a development project are presented to a selected individual or group of personnel for critique.

FUNCTIONAL TESTING (FT) - Exercise of the functional properties of the program as specified in the design requirements.

FUNCTIONAL TEST REVIEW (FTR) - A review which is performed on the documented functional tests that were run by the developer on his code.

INSPECTION - An evaluation technique in which software requirements, design, code, or other products are examined by a person or group other than the designer to detect faults, violations of development standards, and other problems.

INTEGRATION TESTS - Tests performed during the hardware-software integration process prior to computer system validation to verify compatibility of the software and the computer system hardware.

MODULE (M) - Refers to a significant partial functional capability of a subprogram. Modules are usually stand-alone procedures or routines which may call other lower level modules or units.

PEER REVIEW - An evaluation technique in which software requirements, design, code, or other products are examined by persons whose rank, responsibility, experience, and skill are comparable to that of the designer.

PROGRAM - Totality of software in a system or one independent part of software of a distributed system.

SOFTWARE DESIGN SPECIFICATION (SDS) - A document which represents the designers' definition of the way the software is designed and implemented to accomplish the functional requirements, specifying the expected performance. An SDS can be for a system, subsystem, module, or unit.

SOFTWARE TEST SPECIFICATION (STS) - A document detailing the tests to be performed, test environment, acceptance criteria and the test methodology. Approved receipt of the SDS document by the chief verifier forms the basis for the STS.

SOURCE CODE REVIEW (SCR) - A review which is performed on the source code.

SUBPROGRAM (SP) - Refers to a major functional subset of a program and is made up of one or more modules. A subprogram is typically represented by the software executed by a single processor.

STRUCTURAL TESTING (ST) - Comprehensive exercise of the software program code and its component logic structures.

UNIT (U) - The smallest component in the system software architecture, consisting of a sequence of program statements that in aggregate perform an identifiable service.

VALIDATION - The test and evaluation of the integrated computer system to ensure compliance with the functional, performance and interface requirements

VERIFICATION - The process of determining whether or not the product of each phase of the digital computer system development process fulfills all the requirements imposed by the previous phase.

VERIFICATION TEAM - A team of individuals or an individual assigned to review source code, generate test plans, run tests and document the test results for a computer system. It is comprised of verifiers led by a chief verifier.

VERIFICATION TEST REPORT (VTR) - A document containing the test results. In conjunction with the Software Test Specification it contains enough information to enable an independent party to repeat the test and understand it.

4.0 SYSTEM DEVELOPMENT

The development of the QDPS, as shown in Figure 3, involves four stages:

1. Definition
2. Design
3. Implementation
4. System Integration and Testing

A brief description of each stage is given below:

The definition stage is characterized by the statement of the problem to be solved, the construction of an initial project plan, and a high-level definition of the system. During this stage, the overall functional requirements of the system are identified. Within Westinghouse, these requirements are brought together in a System Design Specification.

The design stage is characterized by the decomposition of these System Design specifications into Hardware Design Specifications and Software Design Specifications of sufficient detail to enable the implementation of the system. The Software Design Specifications for the system are then further decomposed into subsystem, module and unit specifications.

The implementation stage is characterized by the actual construction of the hardware and the coding of the various software entities. The software development team is responsible for the writing, assembling, testing, and documenting the computer code. As the software entities are completed, beginning at the unit level, they are officially turned over to the verification team for final independent review and/or testing as specified in Section 7.0.

Software development can be viewed as a sequence of well-defined steps similar to system development. The System Design Specification is used to generate Software Design Specifications which in turn are used to develop high level language programs. These programs are converted by the compiler into assembly language, then by the assembler into machine code. The linker combines groups of assembled code with the library to produce relocatable object code for input to the loader. The loader generates the absolute code which is then burned into read only memory (ROM).

The use of a high level language allows the engineer to express his ideas in a form that is more natural to him. The computer adjusts to his language and not he to the language of the computer. Software written in a high level language is more readily reviewed by an independent party who may not be familiar with the computer instruction set. Some features of the high level language aid the development of reliable software. For example, block structuring helps identify and reduce the number of possible execution paths.

The system integration and testing stage is where the various hardware components and software entities are assembled in a stepwise manner with additional testing at each step to ensure that each component performs its required function when integrated with its associated components.

The final activity associated with the system integration and testing stage is the testing of each subsystem and final testing of the QDPS. A system test plan is derived from the system functional requirements and System Design Specifications to confirm that the QDPS exhibits a level of functionality and performance which meets or exceeds the stated requirements. This final system test is referred to as the factory acceptance test.

Several design assurance techniques are utilized throughout all stages of the development process to ensure that the hardware and software components meet the required specifications.

Formal design reviews are held within Westinghouse to ensure that the System Design Specifications meet the System Functional Requirements. The design review team consists of a group of multidisciplinary engineers to ensure that all aspects of the design are reviewed.

During the implementation stage, acceptance testing and review are conducted by the designers on the hardware components, circuit boards, and subsystems to ensure they exhibit a level of functionality consistent with the Hardware Design Specifications and Software Design Specifications.

The final design assurance technique utilized is the conduct of the system factory acceptance test to ensure the system performance meets the system functional requirements and system design specifications.

5.0 SYSTEM VERIFICATION

5.1 Introduction

With the application of programmable digital computer systems in safety systems of nuclear power generating stations, designers are obligated to conduct independent reviews of the software associated with the computer system to ensure the functionality of software to a level consistent with that described in the system requirements.

Section 5.2 provides an overview of the verification philosophy. Section 5.3 describes the verification techniques utilized in performing the verification process, Section 5.4 describes the matrix that the verification team uses for determining the level of verification that should be applied to each software entity. The section concludes by defining the application of the verification matrix to the QDPS.

5.2 Verification Philosophy

Figure 4 illustrates the integration of the system verification and validation process with the system design process. As discussed in Section 4.0 during the implementation stage, when the writing, testing, assembling, and documenting associated with each software entity (beginning at the unit level) is completed by the design team, the software entity is officially turned over to the

verification team. At this point the verification team performs an independent review and/or testing of the software entities to verify that the functionality of the software entities meet the applicable Software Design Specifications. After the verification team is satisfied that all requirements are met, the software is configured for use in the final system and subsequent system validation process.

Figure 5 illustrates the philosophy utilized in conducting the verification process. The verification process begins at the unit software level, i.e., the simplest building block in the QDPS software. After all software units that are utilized in a software module are verified, the verification team proceeds to verify that module. Not only is the software module verified to meet the module Software Design Specification, but the verification team ensures that the appropriate units are utilized in generating the software module.

After all software modules necessary to accomplish a software subprogram are verified to meet the applicable Software Design Specifications, the verification team proceeds to verify that subprogram. As in the case of the software module, the verification team not only verifies that the subprogram meets the applicable Software Design Specifications, but the team verifies that the appropriate software modules were utilized in generating the subprogram entity. This verification philosophy ensures that the verification team tests and/or reviews the interface between the software unit, module and subprogram entities.

Depending upon the hardware implementation, the verification process may utilize system hardware in the verification of the software modules and subsystems.

5.3 Verification

Verification techniques used in software development fall into two basic categories: review and testing.

5.3.1 Review

There are three types of reviews used in the verification of software: Design Specification reviews, code reviews and functional test reviews.

5.3.1.1 Design Specification Review

This activity involves the comparison of a Software Design Specification for a subsystem, module, or unit to the Design Specification of the component above it to ensure that all of the performance requirements stated in the higher level document are met.

5.3.1.2 Source Code Review

Source code review, as opposed to code testing, is a verification method in which the software is examined visually. The operation of the software is deduced and compared with the expected operation. In effect, the operation of the software is simulated mentally to confirm that it agrees with the specification.

Source code reviews will be used to verify the transformation from a Design Specification into high level code. High level code is easy to read and understand, and therefore full inspection at that level is feasible.

5.3.1.3 Functional Test Review

A functional test review is a review by the verifier of the documentation associated with the functional tests which were performed by the designer. This review will provide a high degree of assurance that the software performs the functions specified in the design requirements.

5.3.2 Software Testing

Software tests can be divided into two categories: structural and functional.

5.3.2.1 Structural Testing

Structural testing, which attempts to comprehensively exercise the software program code and its component logic structures, is usually applied at the unit level. The functionality of the program is verified along with the internal structure utilized within the program to implement the required function. The expectation is that most of the errors will be discovered and corrected at this level, where the cost of doing so will be minimal.

Structural testing requires that the verifier inspect the code and understand how it functions before selecting the test inputs. The test inputs should be chosen to exercise all the possible control paths within the software component. If this is not possible, the test inputs should be chosen to exercise every statement within the component. For example, if a trigonometric function is calculated in several different ways, depending on the range

of the input argument, then the test inputs include tests for the argument in each of these ranges, as well as on the boundaries between ranges. In particular, they exercise the upper limit, the lower limit, and at least one intermediate value within each range.

5.3.2.2 Functional Testing

In the functional approach to program testing, the internal structure of the program is ignored during the test data selection. Tests are constructed from the functional properties of the program which are specified in the Design Specification. Functional testing is the method most frequently used at the module or subsystem level. Examples of functional testing include random testing and special cases by function.

Random testing is the method of applying a test input sequence chosen at random. The method can be used in the following circumstances: to simulate real time events that are indeed random; to increase the confidence level in the correctness of a very complex module; to test a subsystem or a system where it is not necessary to test all the possible paths; to get some quantitative measure on the accuracy of a numeric calculation; or to get a measure of the average time required by some calculation.

Special cases by function can be deduced from the Design Specification of the module and will determine some test cases. For example, a subroutine for matrix inversion should be tested using almost-singular and ill-conditioned matrices. Subroutines which accept arguments from a specified range should be tested with these arguments at the extreme points of the range. An arithmetic package should be tested with variables which have the largest and smallest mantissa, largest and smallest exponent, all zeroes, and all ones.

5.4 Verification Matrix

The choice of particular verification techniques to be utilized on a system component is a function of the following parameters:

1. The safety classification of the system
2. The demonstrability of the system functions : Visual or non-Visual
3. The hierarchical level of the software component (unit, module or subprogram)

5.4.1 Safety Classification

The safety classification of an item is defined according to IEEE-279-1971 and IEEE Std 603-1980. In general, the safety classification of the system establishes the verification requirements for the system. However, since all the components contained in the system do not necessarily perform equal safety functions, a higher or lower level of verification may be assigned to specific system components depending on the exact functions performed. If a different level of verification is assigned to a component, the interactions between that component and the other components in the system must be carefully considered and reviewed.

5.4.2 Demonstrability of System Functions

The method of testing for a software item depends upon the visual demonstrability of its function within the system. If the system function can be illustrated by a simple model as in Figure 6, then the method of testing required need not be as comprehensive as that for software items whose functions cannot be visually recognized. The verification team will be responsible for determining whether a component has a visual or non-visual function.

5.4.3 Hierarchical Level of Software Components

For software that is organized in a hierarchical structure, the intricacies of the actual code can not be easily grasped at the upper levels. For all but simple systems it is prudent to approach verification in a progressive manner, beginning at the unit level. It is at the unit level that the code can be most easily inspected or comprehensively tested.

As the software is built up into higher level components during the integration stage, it becomes possible to demonstrate complete processing functions. This process allows the validation of functional performance requirements. Thus, validation testing assumes a functional theme, with the main emphasis on the interaction between subsystems and their interfaces.

At the system level, subtle errors resulting from the complex interaction between pieces of software never before interconnected may be exposed. The deterrent to any major problems in this phase of testing is thorough and comprehensive verification at lower levels.

5.4.4 Justification of Matrix Elements

Considering the parameters detailed above, different verification methods are required for different subsystems and software components. Figure 7 illustrates, in tabular

form, the levels of verification being proposed for various systems. The safety classification column identifies different categories of subsystems in relation to their safety classification. The software component columns identify the levels of software found within each system. Each element of the matrix specifies the type of testing or review that will be performed on the software component within that safety classification. The justification of each matrix element follows.

The software associated with actuation and/or implementation of reactor trip and engineered safeguards (IEEE-279-1971) must receive the highest level of verification identified. As such, all software at the unit level must be structurally tested to ensure that all lines of the unit indeed meet the intended design specification. Since the control room operators rely upon the automatic actuation of the plant reactor trips and/or engineered safeguards actuations, the highest level of confidence must be afforded the operator. Examples of the South Texas Project system application that meets this definition are the SGWLCS/TAS and the auxiliary feedwater throttle valve qualified control loop.

Depending upon the demonstrability of the identified module and subprogram segments, these segments must either be structurally or functionally tested. Generally, if the module level becomes too complicated, one or more additional unit segments will be identified, which incorporate many of the operations included in the original complicated module or subprogram segment. Therefore, the subdivision will be designed to maximize the demonstrability of the module and subprogram segments. If the verifiers are successful in this task, the module and subprogram tasks are only required to be functionally tested. If the module and/or subprogram segments are visual, a functional test provides essentially the same level of verification as a structural test. Furthermore, for the verifiers to make an appropriate decision concerning the demonstrability of the module and/or subprogram, a detailed source code review (SCR) must be conducted which necessitates a thorough review of each line of code.

Regulatory Guide 1.97, Rev. 2 Type A, B and C Category 1 variables are defined as those key variables which are necessary for the operator to: (1) diagnose the accident; (2) take the preplanned manually controlled actions for which no automatic control is provided; (3) take the plant to a safe shutdown condition; (4) monitor the status of the plant critical safety functions; and (5) monitor the potential for breach of a fission product barrier. These variables are identified as only those necessary for the operator to achieve and maintain the plant in a safe

T23817-1

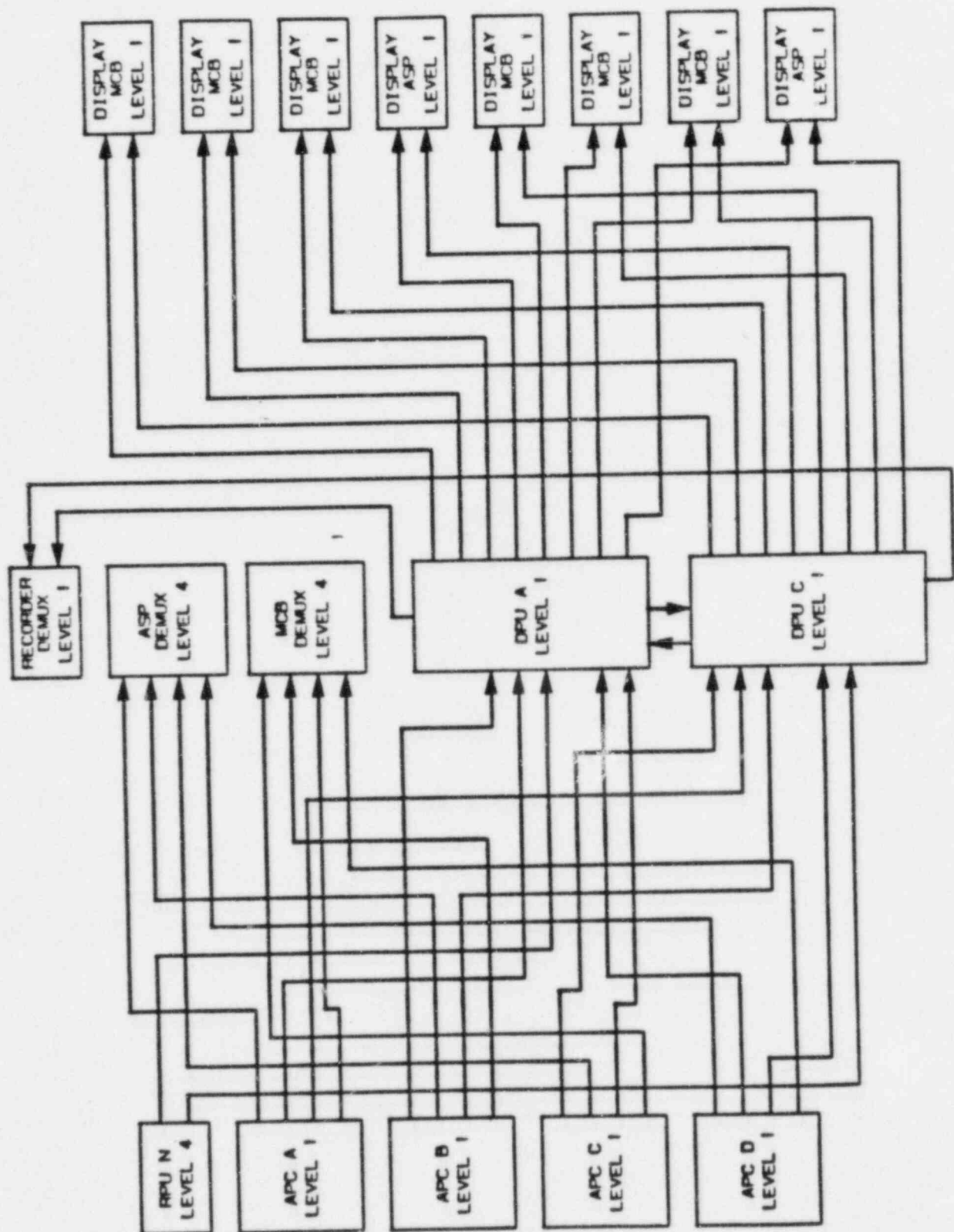
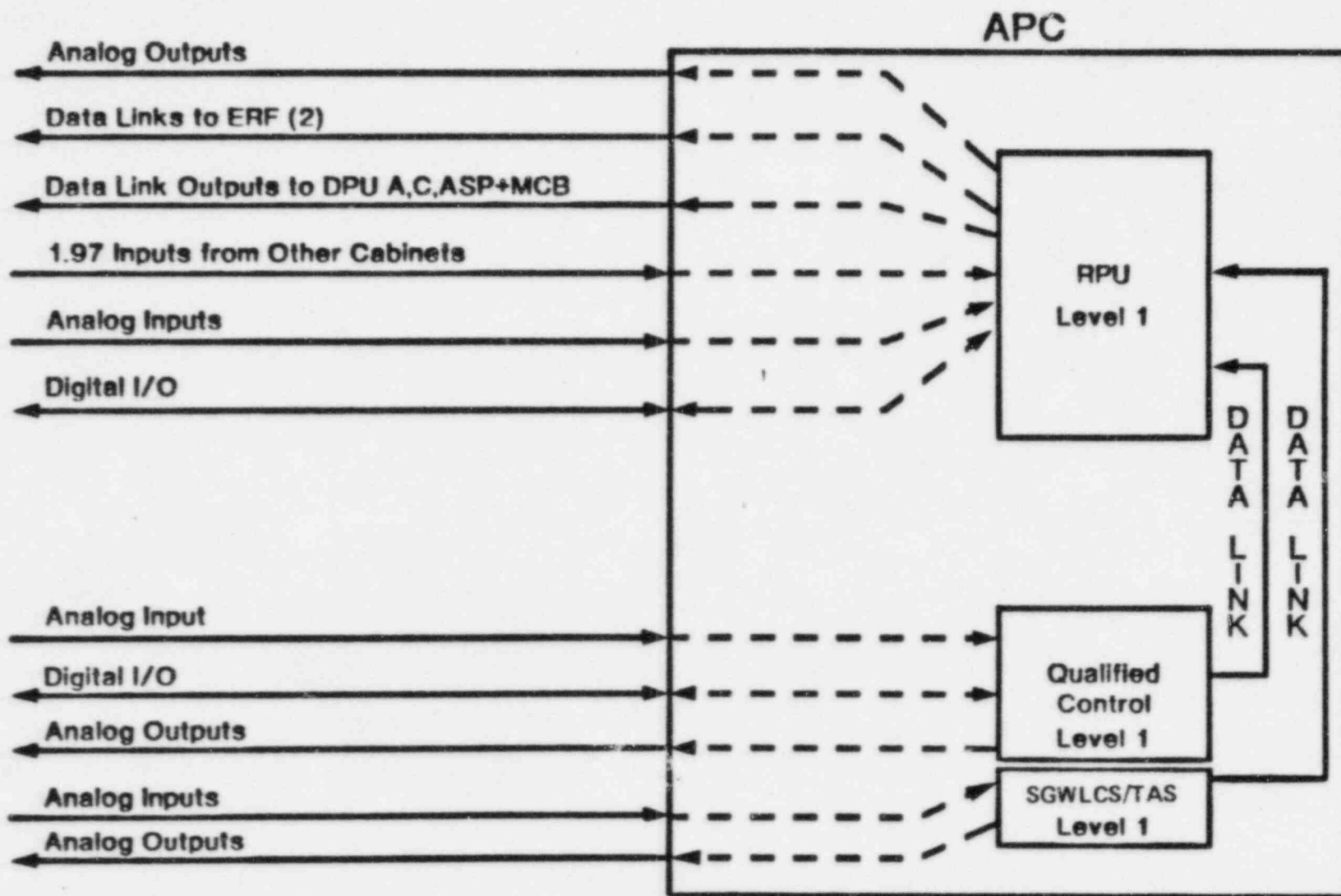
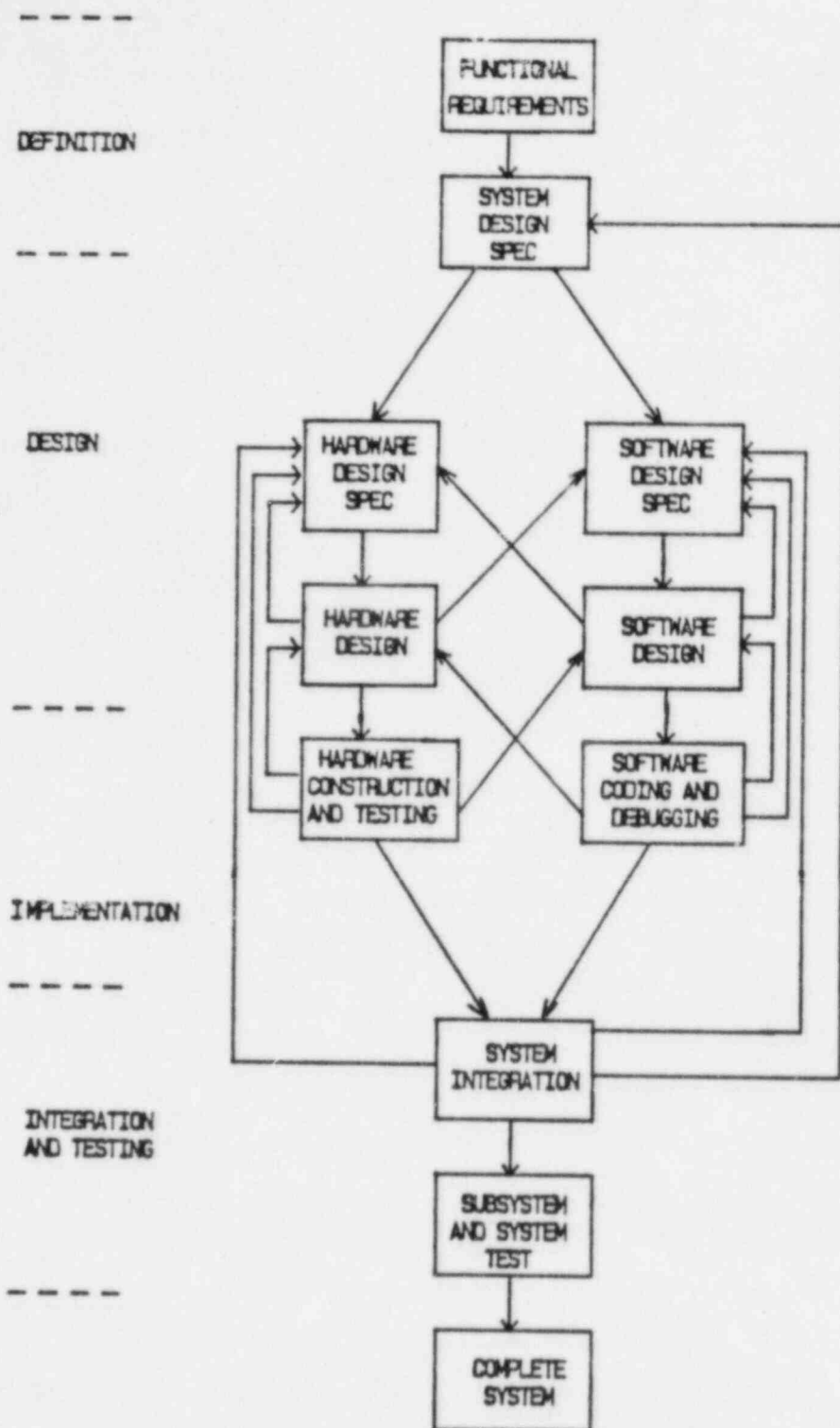


Figure 1.
QDPS System Block Diagram



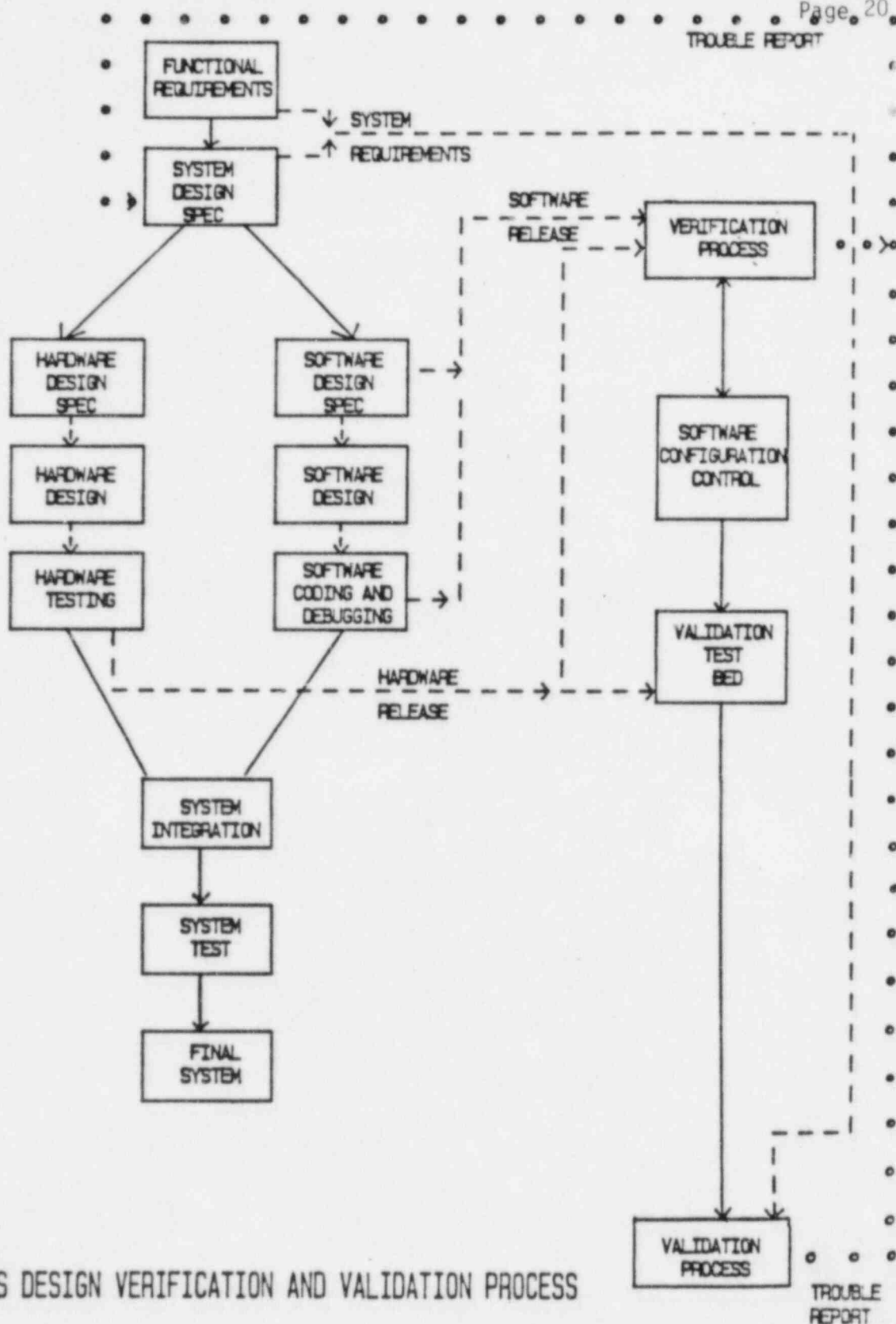
BASIC FEATURES AND I/O OF A CLASS 1E AUXILIARY
PROCESS CABINET (APC)

Figure 2



GDPS DESIGN PROCESS

FIGURE 3

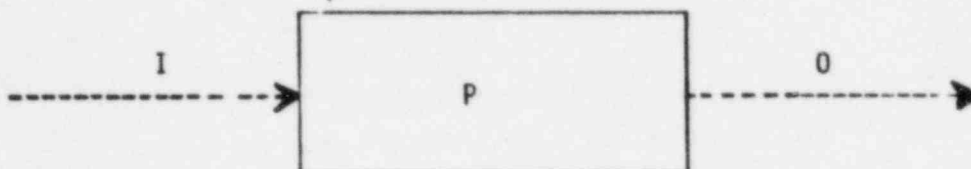


QGPS DESIGN VERIFICATION AND VALIDATION PROCESS

FIGURE 4

DESIGN DOCUMENT	COMPONENT	VERIFICATION			VALIDATION	
		UNIT	MODULE	SU PROGRAM	SUBSYSTEM	SYSTEM
SYSTEM FUNCTIONAL REQUIREMENT	GDPS					
SUBSYSTEM FUNCTIONAL REQUIREMENT	PSMS . CONTROL . SGMLC S/ TAS					
SUB SYSTEM DESIGN SPEC						
SOFTWARE DESIGN SPEC	SUBPROGRAM					
	MODULE					
	UNIT					

FIGURE 5



- Either (1) The system function has only one input and one output and for each input value/state there is one and only one expected output value or state. The transfer function in this case could be either linear or non-linear.
- or (2) The system has more than one input and/or more than one output but they are disjunct and only specific inputs contribute to a specific output or a cluster of outputs and for each input cluster of values/states there is one and only one expected output value or state for a specific output or an output cluster. The inputs/outputs relationship can be represented in a simple tabular form and easily understood.

Figure 6. Model of a Visual Function

SOFTWARE VERIFICATION LEVEL	SAFETY CLASSIFICATION	SOFTWARE COMPONENT			
		UNIT	MODULE	SUBPROGRAM	
1	IEEE 279	ST	ST / FT	ST / FT	↑ INDEPENDENT TESTING ↓
	IEEE 603 PARA 5 B 1				
2	QUALIFIED CONTROLLERS	ST / FT	FT	FT	
3	IEEE 603 PARA 5 B 2 CLASS 1E	FT	FT	FT	↑ SEPARATE REVIEW ↓
4	IEEE 603 PARA 5 B 2 NON / CLASS 1E	FTR / SCR	FTR / SCR	FTR	

FT = FUNCTIONAL TESTING
ST = STRUCTURAL TESTING
SCR = SOURCE CODE REVIEW
FTR = FUNCTIONAL TEST REVIEW
XX / XX = NON - VISUAL / VISUAL

SOUTH TEXAS VERIFICATION MATRIX

FIGURE 7