

AD-A255 498



(12)

DTIC

SEP 1 1992

NAVSWC TR 91-586

MISSION CRITICAL SYSTEM DEVELOPMENT: DESIGN VIEWS AND THEIR INTEGRATION

BY NGOCDUNG HOANG AND STEVE HOWELL (NAVSWC)
NICHOLAS KARANGELEN (TRIDENT SYSTEMS INC.)

UNDERWATER SYSTEMS DEPARTMENT

OCTOBER 1991

Approved for public release; distribution is unlimited.



NAVAL SURFACE WARFARE CENTER

Dahlgren, Virginia 22448-5000 • Silver Spring, Maryland 20903-5000

92 9 15 066

412546 92-25286



7428

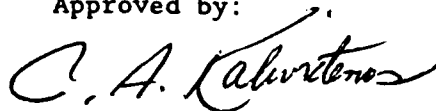
FOREWORD

This document describes a formalism for the design of large, complex systems that can be captured and analyzed to better meet the systems' requirements, resulting in better systems and avoiding the development of inferior systems, possible massive redesign, and reimplementation. This formalism will also increase both the management and the understanding of large, complex systems, reducing the risk of error in the design of the system. Open research issues and further directions of research are also discussed.

The capability to capture and analyze a system's behavior in real-time operation will help analysts to predict possible failures of the system in critical conditions and to refine the requirements throughout development. The ability to integrate the capture, analysis and simulation segments of the design phase will reduce the risk of costly reimplementation of complex components and greatly increase the productivity of system maintenance. From the systems level, well-determined and tight specifications can be passed to engineering sub-environments, such as hardware engineering environments (HWEE), software engineering environments (SEE) or human-computer-interface engineering environments (HCIEE).

The authors would like to thank their sponsor, the Office of Naval Technology, especially Cmdr. Jane Van Fossen, USN Ret., and Elizabeth Wald. The authors would also like to thank Phil Hwang, Cuong Nguyen, and all who provide technical support to refine this report; and Adrien Meskin for her editorial support.

Approved by:



C.A. KALIVRETENOS, Deputy Head
Underwater Systems Department

ABSTRACT

The purpose of this document is to describe a formalism that allows for the capture and analysis of very large, computer-based, real-time, mission critical computer resource (MCCR) systems. The formalism covers all aspects of the system including functional (the functions a system performs) and non-functional (characteristics of system performance) attributes. Non-functional attributes of the systems, such as timing, dependability, security, and reliability, should be captured and analyzed at the early stage of the system development process to guarantee the correctness of a system's performance. The current proposed formalism captures the system design in five different views such that analysis can be correctly performed. The views are Informational, Functional, Behavioral, Implementation, and Environmental. Each view explores different aspects of the system and all five in total provide a more complete understanding of the system. Even though the ultimate goal is to capture all aspects of the system in the five views, at this time there is no perfect vehicle for analyzing system attributes, such as hard real-time, security, reliability, and dependability. However, as the formalism matures, all aspects that affect the system will be covered in detail.

CONTENTS

<u>Chapter</u>	<u>Page</u>
1.0 INTRODUCTION	1-1
1.1 PROBLEM	1-1
1.2 BACKGROUND	1-2
1.3 DESCRIPTION OF EXAMPLE	1-3
2.0 OVERVIEW OF DESIGN CAPTURE	2-1
2.1 CAPTURE METHODOLOGY INTRODUCTION	2-1
2.2 SYSTEM DESIGN VIEW ANNOTATION	2-3
3.0 ESSENTIAL SYSTEM DESIGN VIEWS	3-1
3.1 INFORMATIONAL VIEW	3-1
3.1.1 EXAMPLE METHOD	3-1
3.1.2 EXAMPLE CAPTURE	3-4
3.1.3 SHORTFALLS	3-8
3.2 FUNCTIONAL VIEW	3-8
3.2.1 EXAMPLE METHOD	3-8
3.2.2 EXAMPLE CAPTURE	3-11
3.2.3 SHORTFALLS	3-11
3.3 BEHAVIORAL VIEW	3-12
3.3.1 EXAMPLE METHOD	3-12
3.3.2 EXAMPLE CAPTURE	3-20
3.3.3 SHORTFALLS	3-28
3.4 IMPLEMENTATION VIEW	3-28
3.4.1 EXAMPLE METHOD	3-28
3.4.2 EXAMPLE CAPTURE	3-29
3.4.3 SHORTFALLS	3-33
3.5 ENVIRONMENTAL VIEW	3-33
3.5.1 EXAMPLE METHOD	3-34
3.5.2 EXAMPLE CAPTURE	3-35
3.5.3 SHORTFALLS	3-36
4.0 INTEGRATION AND TRANSITION BETWEEN VIEWS	4-1
4.1 DESIGN DOMAIN RELATIONSHIPS AND DEPENDENCIES	4-1
4.1.1 RELATIONSHIP	4-1
4.1.2 DEPENDENCIES	4-2
4.2 SYSTEM DESIGN VIEWS ANNOTATION AND INTEGRATION	4-6
4.2.1 MANUAL ANNOTATION	4-7
4.2.2 FORWARD ANNOTATION	4-7
4.2.3 BACKWARD ANNOTATION	4-8
4.2.4 FULL INTEGRATION	4-8
4.2.5 STATE OF INTEGRATION IN MECHANIZATION SUPPORT	4-8
4.3 AUTOMATED SYSTEM DESIGN CAPTURE AND ANALYSIS TOOL DEVELOPMENT OPPORTUNITIES	4-9

CONTENTS (Cont.)

<u>Chapter</u>	<u>Page</u>
4.3.1 DESIGN CAPTURE	4-9
4.3.2 INTRA- AND INTER-VIEW CONSISTENCY AND COMPLETENESS CHECKING	4-9
4.3.3 DESIGN SIMULATION	4-10
4.3.4 DOCUMENT GENERATION	4-10
5.0 CONCLUSION	5-1
REFERENCES	6-1
APPENDIX A--SYSTEM DESIGN FACTORS	A-1
DISTRIBUTION	(1)

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
1-1	ENGINEERING OF COMPLEX SYSTEMS BLOCK	1-4
1-2	SYSTEMS DESIGN SYNTHESIS PROJECT	1-5
3-1	NOTATION OF OBJECTS	3-2
3-2	EXAMPLE OF RELATIONSHIP	3-3
3-3	EXAMPLE OF SUPERTYPE/SUBTYPE RELATIONSHIP	3-4
3-4	THE INFORMATIONAL VIEW OF THE CONTEXT DIAGRAM OF THE PASSIVE SONAR SYSTEM	3-5
3-5	DESCRIPTION OF THE PASSIVE SONAR OBJECT	3-6
3-6	DESCRIPTION OF THE SUPPORT SYSTEM SERVICE OBJECT	3-7
3-7	EXAMPLE OF A PROCESS	3-9
3-8	EXAMPLE OF FUNCTION DECOMPOSITION	3-9
3-9	EXAMPLE OF DATA FLOW DECOMPOSITION	3-10
3-10	NOTATION OF DATA STORE	3-11
3-11	THE FUNCTIONAL VIEW OF THE CONTEXT DIAGRAM OF A PASSIVE SONAR SYSTEM	3-13
3-12	DATA FLOW DIAGRAM OF THE PASSIVE SONAR SYSTEM PERFORMANCE PROCESS	3-14
3-13	DATA FLOW DIAGRAM OF THE SIGNAL PROCESSING PROCESS	3-15
3-14	DATA FLOW DIAGRAM OF THE SIGNAL CONDITIONING PROCESS	3-16
3-15	P-SPEC OF THE DETECTION PROCESS	3-17
3-16	EXAMPLE OF DECISION TABLE	3-18
3-17	NOTATION OF STATE TRANSITION DIAGRAM	3-18
3-18	EXAMPLE OF STATE TRANSITION DIAGRAM	3-19
3-19	EXAMPLE OF STATE TRANSITION MATRIX	3-20
3-20	THE BEHAVIORAL VIEW OF THE CONTEXT DIAGRAM OF A PASSIVE SONAR SYSTEM	3-21
3-21	DATA/CONTROL FLOW DIAGRAM OF THE PASSIVE SONAR SYSTEM PERFORMANCE PROCESS	3-22
3-22	DATA/CONTROL FLOW DIAGRAM OF THE SIGNAL PROCESSING PROCESS	3-23
3-23	DATA/CONTROL FLOW DIAGRAM OF THE SIGNAL CONDITIONING PROCESS	3-24
3-24	DATA/CONTROL FLOW DIAGRAM OF THE ANALYSIS AND CLASSIFICATION PROCESS	3-25
3-25	STATE TRANSITION DIAGRAM FOR THE ANALYSIS STATE	3-26
3-26	PROCESS ACTIVATION TABLE FOR THE ANALYSIS STATE	3-27
3-27	CANDIDATE RESOURCE MODEL OF THE PASSIVE SONAR SYSTEM	3-30
3-28	EXAMPLE OF ALLOCATION OF FUNCTIONS TO RESOURCES	3-31
3-29	EXAMPLE OF ALLOCATION OF DATA FLOWS TO RESOURCES	3-32
4-1	TOP LEVEL DESIGN DOMAIN RELATIONSHIPS	4-3
4-2	FORWARD AND BACKWARD ANNOTATION	4-7

TABLES

<u>Table</u>	<u>Page</u>
2-1 SYSTEM VIEWS	2-2

CHAPTER 1

INTRODUCTION

The purpose of this document is to describe a formalism that allows for the capture and analysis of very large, computer-based, real-time, mission critical computer resource (MCCR) systems. The formalism covers all aspects of the system including functional (the functions a system performs) and non-functional (characteristics of system performance) attributes. Non-functional attributes of the systems, such as timing, dependability, security, and reliability, should be captured and analyzed at the early stage of the system development process to guarantee the correctness of a system's performance. The current proposed formalism captures the system design in five different views such that analysis can be correctly performed. The views are Informational, Functional, Behavioral, Implementation, and Environmental. Each view explores different aspects of the system and all five in total provide a more complete understanding of the system. Even though the ultimate goal is to capture all aspects of the system in the five views, at this time there is no perfect vehicle for analyzing system attributes, such as hard real-time, security, reliability, and dependability. However, as the formalism matures, all aspects that affect the system will be covered in detail.

1.1 PROBLEM

The requirements for large-scale, mission critical systems have increased drastically in response to technology gains and advanced threats. These requirements, which include real-time, time-critical, fault tolerant and security issues, have put large burdens on the systems engineers in designing these highly complex systems. Even though significant increases have been made in the capacity and capability of hardware, software, and human resources humware (i.e., human computer interface) resources, the engineering capability to engineer systems which effectively embody these components has not kept pace, resulting in a failure to fully utilize their potential capabilities.

A variety of methodologies has been described for characterizing and capturing complex system designs. These methodologies have been developed by systems engineers and academicians from both theoretical and practical viewpoints. Several different perspectives of the systems engineering design and capture problem exist. One popular perspective is the Yourdon-DeMarco style Structured Analysis methodology which defines a hierarchy of functions and data flows to describe the system.¹ Real-time extensions, such as those developed by Hatley and Pirbhai,² and Ward and Mellor,³ integrate control flow with data flow. An object-oriented approach is embodied in the Information Model approach described by Shlaer and Mellor.⁴ Hardware and software engineers often perceive the system through the various hardware development

formalisms (e.g., VHDL⁵) and software development methods and tools⁶ which address the design and analysis of specific implementations. Personnel with operational experience using earlier generations of the system under design (such as naval officers in the case of an antisubmarine warfare (ASW) combat system) tend to see the system from an operator's point of view and within the context of one or more operational scenarios.

Unfortunately, current methodologies do not fully support large time-critical, real-time systems. These methods fail to adequately address critical information that must be specified in order to understand and analyze the system under design. Particularly, non-functional attributes and estimation parameters for analysis are not formally specified in current methods. Part of this stems from the fact that many of the methods were originally constructed for software development for systems developed in uniprocessor environments. Large, complex, real-time MCCR systems are typically developed on parallel and distributed hardware, which are a combination of computers, sensors, and weapons. These methods tend to de-emphasize non-functional (performance-oriented) hardware, and the distribution aspects that make it difficult to analyze the systems.

Scalability of the methods (and their automation) is an important issue for large, complex MCCR systems. Methods that are applicable for small and moderate size systems become inadequate for large systems which are typically developed by hundreds of systems developers and implemented in millions of lines of source code. Reasons for the failures of the methods include the structure and mechanization of the methods and the complexity of the algorithms within the methods.

Additionally, most of the current methods are being developed independently. While each method explores certain aspects of the system, each has its own format, notation, and conceptual basis. Transition techniques, which allow the information to be retained when moving from one method to another, are not available. This limitation forces systems engineers to completely redevelop the system in order to extract different views. The effort involved in redevelopment is time consuming and costly; therefore systems engineers are prevented from viewing the complete system.

1.2 BACKGROUND

This research was performed under the Real-Time System Design Capture and Analysis task within the Systems Design Synthesis Technology (SDST) project. The project is part of the Engineering of Complex Systems (ECS) technology block. The work is sponsored by the Office of Naval Technology (Code ONT-227). The overall objective of the ECS program is to provide an integrated, system level, full life cycle engineering methodology, supporting mechanization and engineering environments for the next generation of MCCR systems. Within this objective, the SDST project is attempting to develop and enhance the system level ability to specify, capture, synthesize, analyze, and prototype the design of complex, real-time, time-critical, fault tolerant and secure systems.

This research is tightly associated with other ongoing efforts within the ECS program. Currently the closest coordination is linked to the Requirement Specification and Traceability task. The system requirements are organized in a clear structured format by the Requirement Specification and Traceability task before they are fed into this task as input. The Real-Time Dependable Systems Design task, the Design Structuring and Optimization Allocation task within the SDST project, and the Systems Modeling Technology task within the System Evaluation and Assessment Technology (SEAT) project also have close relationships with this task. Figure 1-1 shows the different tasks within the ECS block and Figure 1-2 shows the relationship between tasks within the SDST project.

The ongoing research is exploring ways to allow systems engineers to formally specify a system's design given a set of requirements. Capabilities to examine and analyze the consistency, completeness and correctness of the design is being matured. Techniques for providing the necessary detailed information and allowing the system design to move through the design process (whether it is the Waterfall, Modified Waterfall, Spiral, etc.) are being developed. The effort hopes to also allow systems engineers to optimize the design's distribution, decomposition, and allocation from requirements before deciding which parts will be in hardware, software or performed by humans (humware).

1.3 DESCRIPTION OF EXAMPLE

An example of a passive submarine sonar system will be used throughout this document to illustrate the various capture techniques. This example was developed by Molini⁷ to provide a meaningful test problem for research of real-time distributed systems. The following paragraphs briefly describe the passive sonar system example.

Sonar equipment is used for determining the presence, location, or nature of objects in the sea from the sound that the objects emit or reflect. Active sonar transmits an acoustic signal which, when reflected from a target, provides the sonar receiver with a signal. Based on this received signal, detections and position estimates are made. Passive sonar bases its detection and estimation on sounds that emanate from the target itself.

In passive sonar, the received acoustic waveform from each hydrophone consists of one or more signals and background noise. The hydrophone converts the acoustic waveform to an electronic signal. The signals are amplified, filtered, sampled, and digitized in a signal conditioner. The digitized hydrophone outputs from the signal conditioner are combined by a digital beamformer to form a set of beams. (A beam emphasizes the sound from the beam direction and reduces the sound from other directions.) Beam data are then processed to obtain detection and estimation statistics. Based on the values of these statistics, the system decides where targets are located. Detected targets are tracked by modifying the beamformer parameters and steering a beam toward the target.

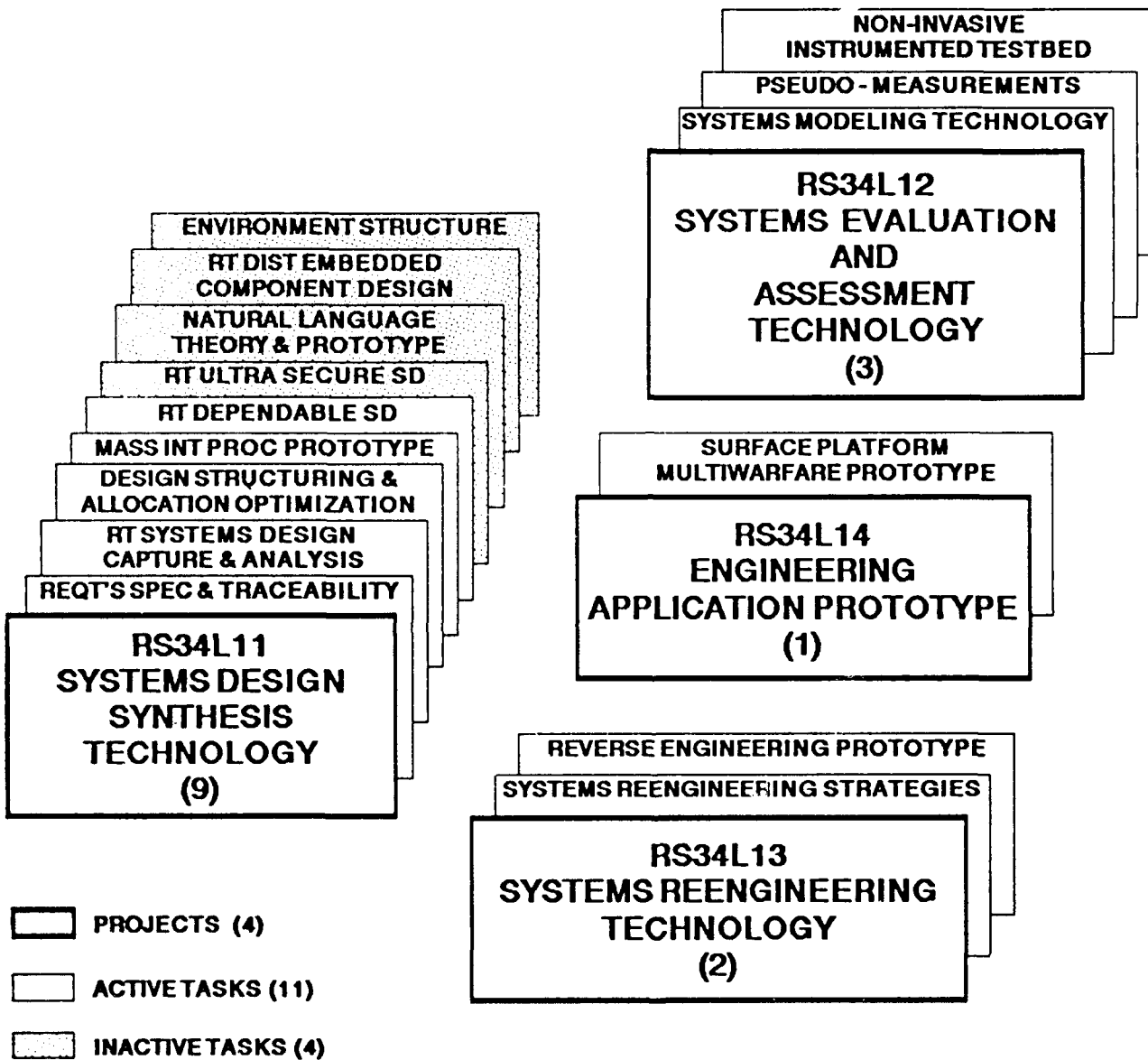
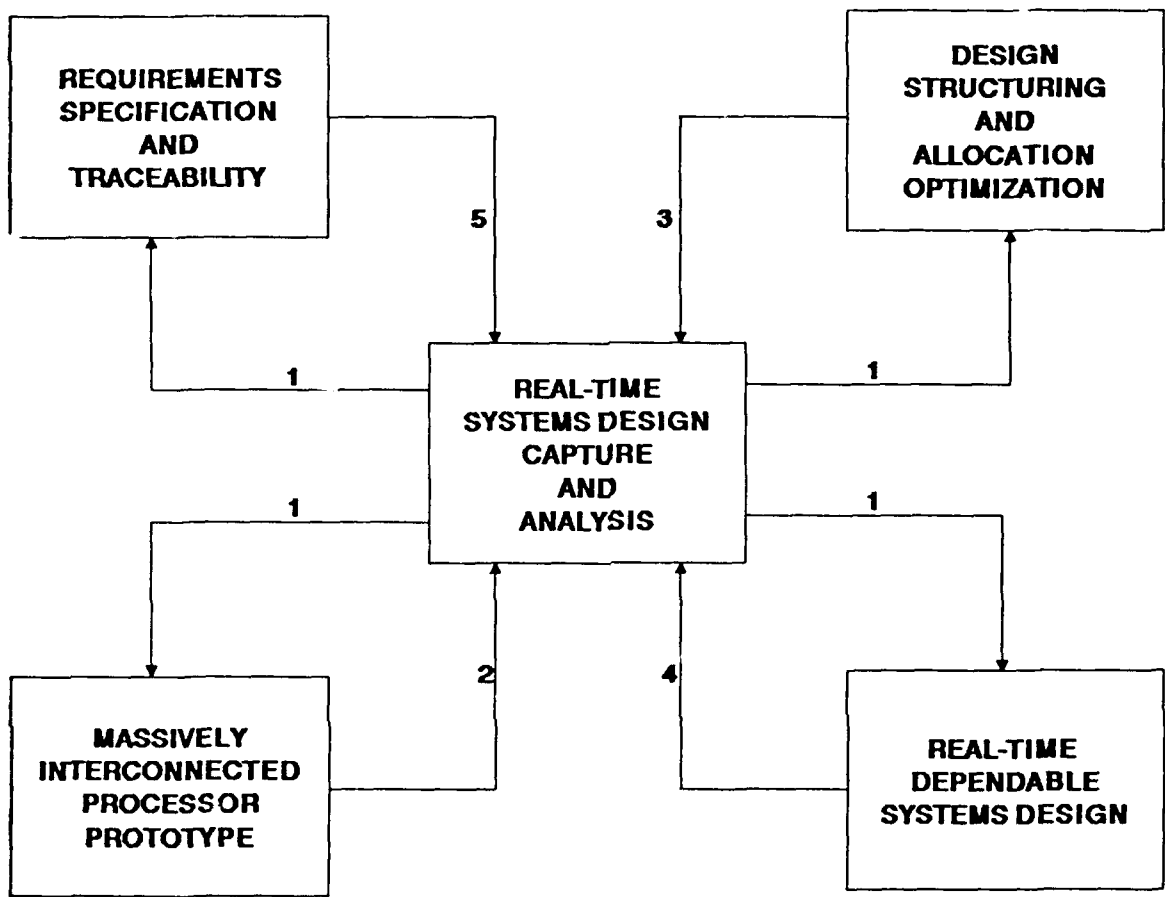


FIGURE 1-1. ENGINEERING OF COMPLEX SYSTEMS BLOCK



- 1 DESIGN METHODOLOGY CAPTURE INFORMATION
- 2 RESOURCE LIBRARY AND MIM CAPTURE INFORMATION
- 3 LESSONS LEARNED FROM STRUCTURING/CAPTURE INFORMATION FOR ALLOCATION/STRUCTURING
- 4 DEPENDABLE ANALYSIS RESULTS AND NEEDED DEPENDABLE CONSTRUCTS
- 5 CAPTURE INFORMATION FOR TRACEABILITY

FIGURE 1-2. SYSTEMS DESIGN SYNTHESIS PROJECT

A detected target is also analyzed by classification. This includes distinguishing a signal returned from a target with regard to the type of target that produced the signal and by a signal frequency spectrum and dynamics (e.g., a school of fish versus a submarine) of its target signal.

The system is designed to detect and track submarine targets that can operate at any speed from 0 to 15 knots at any depth from 0 to 500 feet. The target sound source is assumed to consist of vibrational harmonics from rotating machinery with a nominal rotation rate of 2,400 rpms. Therefore, it is expected that the return signal from a submarine contains a fundamental 40-hertz component and the first three harmonics.

The data processing load for a sonar system is roughly proportional to the number of hydrophones, number of beams, and the sample rate (i.e., the larger the array of hydrophones, the greater the beamforming gain and the greater the detection range; the finer the beam width, the finer the display resolution that assists in resolving multiple targets even though they appear close together). Some of the beams are used for detection displays, some for tracking targets, and some just for listening. Typically, the number of detection beams is fixed, so the detection processing varies little over time. The number of steered tracker beams may change as targets come and go. When the number of tracker beams is changed, the tracking load changes proportionally.

CHAPTER 2

OVERVIEW OF DESIGN CAPTURE

In order to completely capture a design, a well defined methodology should be provided to systems engineers. This methodology not only captures the design, in general, but also includes the detailed information of each design element such that analysis can be performed.

2.1 CAPTURE METHODOLOGY INTRODUCTION

The methodology for system design capture and analysis described in this technical report addresses each of the key system design perspectives and provides a mechanism for constructing a consistent, structured, multi-domain representation of the design. The overall goal is to provide a method with the following capabilities which are critical to support efficient capture and analysis of large, complex, computer-based systems:

Scalability of the method to very large complex problems and management of system complexity and size through automation.

Unambiguous representation of system performance requirements and concept of operations providing a common means of communication between the sponsor, users, architects, analysts and engineers.

Separation of functional and implementation design issues and support for simulation and analysis of candidate functional designs and candidate resource implementations.

Cross-linking of design capture information in rigorous relationship definitions to avoid redundant data capture and to support intensive automated bookkeeping tasks.

Consistency checking and traceability across design capture elements and domains.

Generation of system documentation from information captured in the design database.

Applicability of the methodology to full-scale automation of system design capture and analysis process.

The central element of the multi-domain design capture and analysis methodology includes definitions of the multiple design domains or views which address the principal system design perspectives: (1) Environmental, (2) Informational, (3) Functional, (4) Behavioral, and (5) Implementation. The

capture approach for each design domain or view share a common hierarchical structure which supports management of the magnitude and complexity associated with a large system design. Flat representations of complex system designs rapidly become unwieldy as the design detail unfolds. A hierarchical structure allows the system views to be represented at various levels of detail from a broad top level, which encompasses the breadth and scope of the system and its external interfaces, to very low levels which describe the details of a particular segment of the system design.

The five system views partition the system design into logical segments which correspond to key perspectives of the system design. These five views are distinct but related representations of key aspects of the system. They are summarized in Table 2-1.

TABLE 2-1. SYSTEM VIEWS

DESIGN VIEW	VIEW OBJECTIVES	DESIGN ELEMENTS
Environmental View	Establish Conditions and Events Constraining System Operations Specify Performance MOEs and Conditions of Measurement	Environmental Conditions and Event Descriptions Design Req and Constraints System Initial Conditions Measures of Effectiveness
Informational View	Characterize System Concept of Operations Represent System Components in Abstract Terms	Entity - Relationship Diagrams Attribute/Method Descriptions
Functional View	Define System Functions and Decompositions Specify Data Flow Requirements	Function/Data Flow Diagrams Process Specifications Data Dictionary
Behavioral View	Define System States and Triggers Specify System Behavior Characteristics	Control Flow Diagrams State Transition Diagrams Control Specifications
Implementation View	Define the Physical Hardware, Software and Human Resources Which Make Up the System Specify System Physical Interconnectivity	Hardware, Software and Human Resource Descriptions Performance Parameters and Resource Characteristics Function-Resource Mapping

The Environmental View captures the system under design from the user's or operator's point of view. This view describes the situation(s), environment(s), expected events and other factors that make up the conditions under which the system is operating. This includes the initial state of the system under design; environmental conditions including acoustic, electromagnetic, and meteorological conditions; hostile threat platform types and locations (in a military application); operational constraints; likely strategic and tactical considerations and other pertinent items; and concept of operations. The measures of effectiveness (MOEs) which characterize system performance and establish the "mission success criteria" for the system are also specified together with the conditions under which the MOEs are measured.

The Environmental View also specifies the guidance and constraint of the environment which the design itself must face.

The Informational View captures a conceptual representation of the system under design in abstract terms. This view captures all components that make up the system and the interaction between these components. This allows for a description of the intended system concept of operations without implying or constraining the physical implementation of the system under design.

The Functional View establishes the functional structure of the system. It specifies how the functions are decomposed and how the information is transformed through these functions. This provides a better understanding of the system's functional decomposition.

The Behavioral View describes the system's attributes over time and describes the event or time-driven aspects of the system. This view allows for specification of the system's behavior at different times and under various conditions and situations.

The Implementation View defines the physical hardware, software and human resources which comprise the system and its connectivity to external systems.

An attempt to address all of the issues associated with these views simultaneously or without a structured methodology is a multi-dimensional problem of a magnitude which exceeds the capacity of most, if not all, systems engineers. Each of these views provides key information concerning particular aspects of the system under design. Taken individually the views allow the systems engineer to partition the design of a proposed or existing system into manageable parts. The five system views introduced in this section will be described in detail in Sections 3.1 through 3.5.

As MCCR systems increase in complexity and size, the capturing design methods have to expand extensively so that they can be able to capture different aspects of the system in a complete and systematic manner. Deciding what aspect of the system needs to be captured and to what level of detail is a very difficult task. Sometimes the same group of attributes can be used to completely capture one type of system, but for another type it becomes irrelevant or incomplete. Therefore, the ideal situation is to define the capturing method for all aspects of the system and, depending on the system requirements, emphasize or deemphasize certain system views.

2.2 SYSTEM DESIGN VIEW ANNOTATION

A design element is a partitionable component of the specification. It can be informational, functional, behavioral, implementation, and/or environmental and can describe function(s), behavior(s), data, object(s), relationship(s), physical implementation(s), etc. As part of the system design views, the systems engineer must not only capture information about how the design elements interrelate, but also have extensive information about the

design elements themselves. Relationships between the various system design factors can be expressed formally so that effective trade-offs can be performed between the various design factors.

Design factors are attributes that are attached to design elements. Appendix A is a list of the design factors that may typically need to be specified within a mission critical system design. This list is initial and is expected to be expanded/modified over time. These design factors are typically non-functional in nature; they describe how the design component will operate, not what the design element will do. For example, in the sonar system, the detection function could be annotated with the throughput, response time, and availability factors in addition to the description of the detection functionality.

Typically, a design element related to a particular design factor may be annotated in three ways: (1) requirements/budgets, (2) predicted/expected, and (3) actual.

The first category is the required need for that design factor within the design element, either specified by the original requirements, or budgeted by the designer in order to meet an overall requirement. The second type of annotation describes what has been allocated by the systems engineer to the design element. This is realistically what the system design expects to be able to meet in the design. The allocated value of a design element should specify a better or equal performance than the required value. The third category describes actual measurements for the design factor's value. This is used in the validation of the system design. Actual measures at the "leaf" design elements can propagate to the other design elements, yielding the verification and validation of various portions of the design. Earlier in the design process, the information available (through rapid prototyping or existing candidate hardware/humware characteristics) can be used to guide design decisions.

Additionally, the design elements should be annotated with a description of a design rationale which lead to the design of that particular element. This provides background for future modification or selection of alternative design options.

CHAPTER 3

ESSENTIAL SYSTEM DESIGN VIEWS

In order to design any type of system, systems engineers should be able to understand clearly the components that make up the system. Defining a proper method that can correctly capture different types of system components becomes a critical issue for the methodologist. The following description of the five system views and their supporting methods is an attempt to formally and completely capture the system under design.

3.1 INFORMATIONAL VIEW

The Informational View captures all things that make up the system and shows how those things interact with each other. All components of the system are represented as abstract objects and they are associated with each other by relationships. Depending on their performance, objects and relationships at the boundary of the system will also be studied to complete the Informational View.

3.1.1 Example Method

Over the past years, the information modeling concept has been used in the development of systems and has had some success. As systems change their size and mission, different or additional information modeling concepts are developed to support them. The object-oriented approach to analyze systems by Shlaer and Mellor⁴ is one of the most well known methods. It uses abstract objects to represent real world components of the system and uses the entity relationship symbols to model the connectivity of the objects. Shlaer and Mellor⁴ believe that a large, flat graphical information model can carry enough information to represent any system. Unfortunately, this method is insufficient for Navy systems due to their size and complexity. To solve the above problem, the Method and Tools Working group⁶ adopted and modified the Shlaer and Mellor concept by introducing complex objects. This change allows for objects to be represented at different detail levels which provides more understanding about the system. The Hierarchical Information Model (HIM) which basically is the incorporation of the hierarchy concept into the Shlaer and Mellor object-oriented approach becomes sufficient in the capturing of the Informational View.

3.1.1.1 Object. In general an object is defined as an abstract representation of the real thing that makes up the system. Each object has a unique name and, depending on whether it is simple or complex, its characteristics will be captured by a list of attributes or by another set of objects.

All information that reflects the characteristic of an object or can be used to identify an object should be represented in the object attributes. Some system design factors that are listed in Appendix A are very good examples of likely object attributes. Each attribute should only capture a single relevant piece of information to the object being defined and take on the values independently. To minimize the confusion between simple and complex objects, "object" refers to a simple object and "cluster" refers to a complex object. Figure 3-1 summarizes the notation of different types of objects

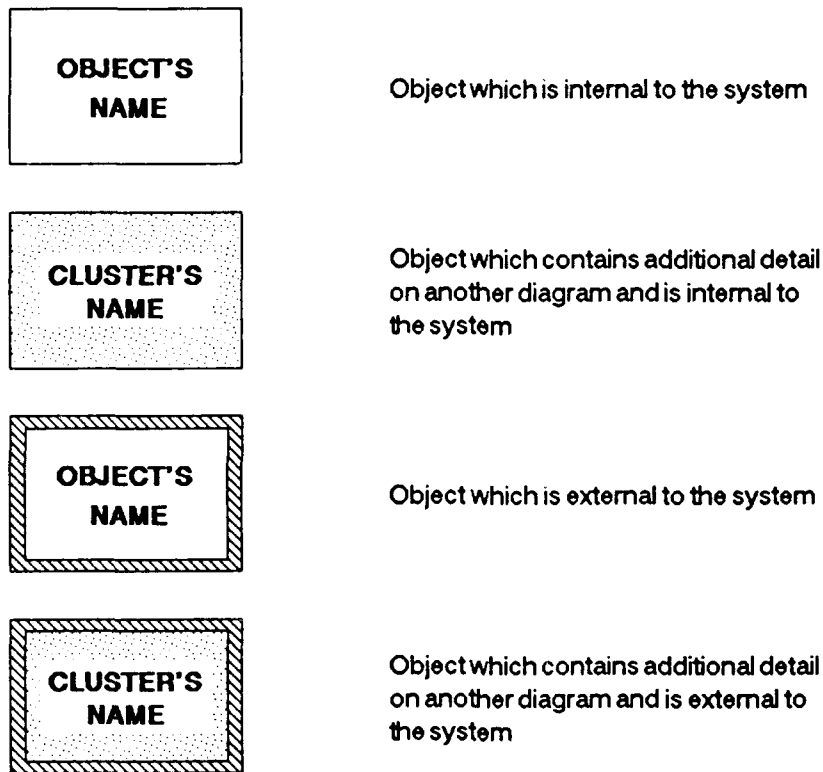
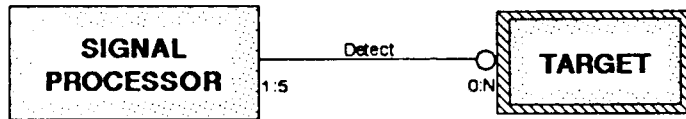


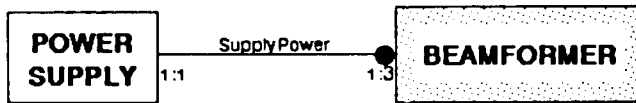
FIGURE 3-1. NOTATION OF OBJECTS

3.1.1.2 Relationship. Relationship shows how objects are associated to one another. To emphasize that relationship does not signify data flow, it is represented by a straight line that ends by a white or black ball (see Figure 3-2). The ball indicates the "inferior" side of the relationship. While a white ball indicates that simultaneous existence of the inferior and superior objects are not required at all times, a black ball indicates that the existence of the inferior objects is dependent on the existence of the superior objects. Attached to each end of a relationship is a set cardinal number which is used to indicate the number of relationships that may exist simultaneously between the objects. The cardinal number is expressed as "min.max" where min is always less than or equal to max and can have a value as small as zero.

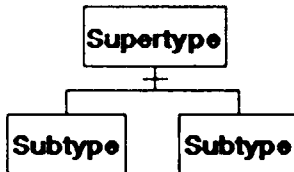
A special type of relationship between objects is the subtype/supertype. Object SHIP in Figure 3-3 is an example. All ships will carry the same attributes; however, in the real world, ships can be classified by different types (i.e., merchant ships, combat ships, and cruise ships) and in some situations can be viewed as different objects. To represent this type of relationship, the supertype/subtype object is introduced. Merchant, combat, and cruise ships are called the subtype objects and will inherit all attributes of their supertype object ship. In addition, each subtype object will have its own additional attributes to represent itself. For example, all types of ships will have a serial number, size, and destination; but combat ships will have different types of weapons and missions as attributes to distinguish between themselves. Consequently, the subtype object inherits everything, without exception, from their supertype object. Inherited attributes can be modified by subtype objects; and if the next level of subtype exists, it will also inherit the modifications.⁹



A SIGNAL PROCESSOR can detect many targets
 A TARGET can be detected by 1 to 5 SIGNAL PROCESSORS



A POWER SUPPLY Supplies Power to from 1 to 3 BEAMFORMERS at a time
 A BEAMFORMER is supplied power by only 1 POWER SUPPLY
 The existence of the BEAMFORMER depends on the existence of the POWER SUPPLY



Subtype objects inherit everything, without exception, from their supertype object. Additional attributes may be added to the subtype object. If so, subsequent subtypes inherit the additional attributes.

FIGURE 3-2. EXAMPLE OF RELATIONSHIP

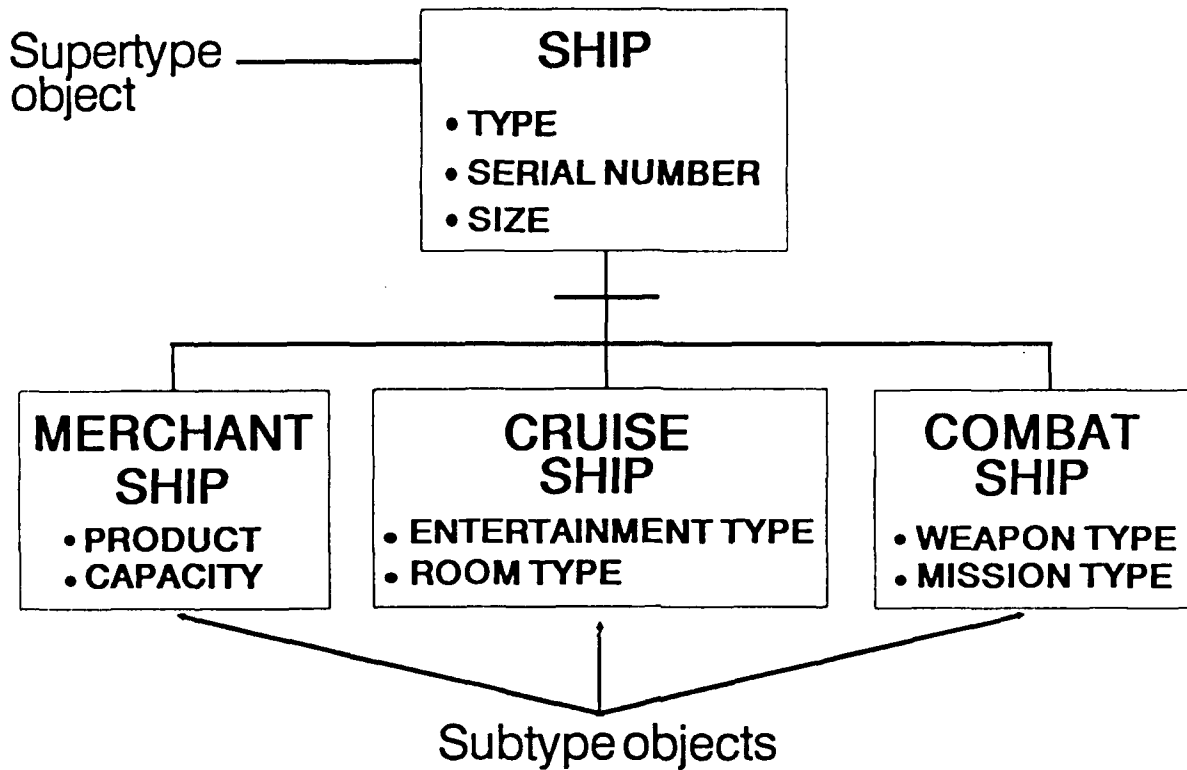


FIGURE 3-3. EXAMPLE OF SUPERTYPE/SUBTYPE RELATIONSHIP

3.1.2 Example Capture

Figures 3-4 through 3-6 show a portion of the HIM of the passive sonar system. The objective of this example is to demonstrate how a real world problem is captured and the usefulness of the hierarchical concept in information modeling. The first level of the HIM, Figure 3-4, is the context diagram which is usually named after the design system. In the lower levels each diagram will be labelled both by the level number (starting from 0) and by the object's name where it is represented. For example Figure 3-5 is a more detailed description of the PASSIVE SONAR object and Figure 3-6 is a detailed description of the SYSTEM SUPPORT SERVICE object that was found in Figure 3-5. The description of the attributes that should be attached to each object and relationship are not shown in this example.

Context Diagram: Passive Sonar System

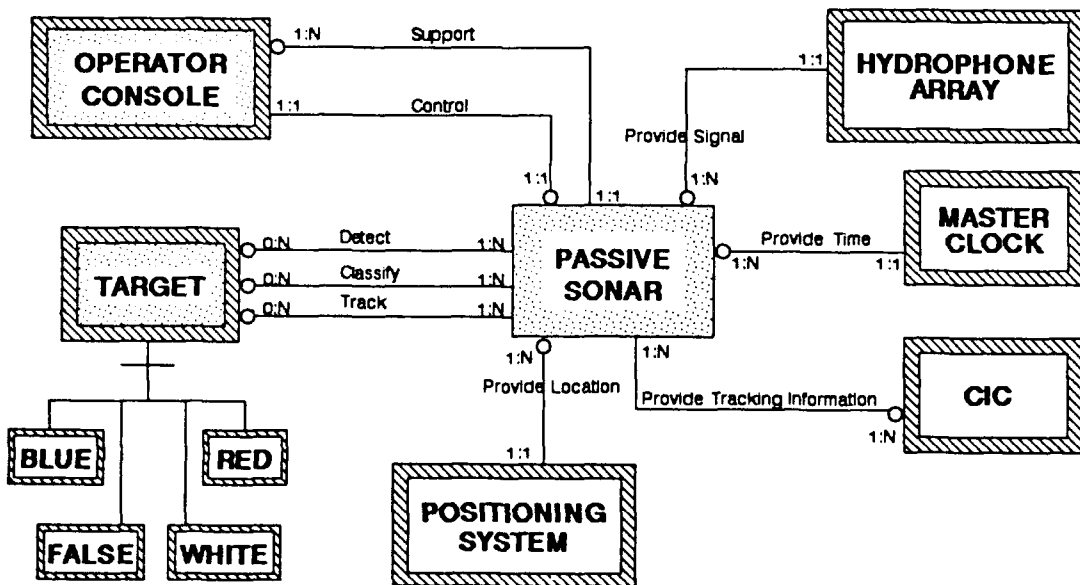


FIGURE 3-4. THE INFORMATIONAL VIEW OF THE CONTEXT DIAGRAM OF THE PASSIVE SONAR SYSTEM

Passive Sonar

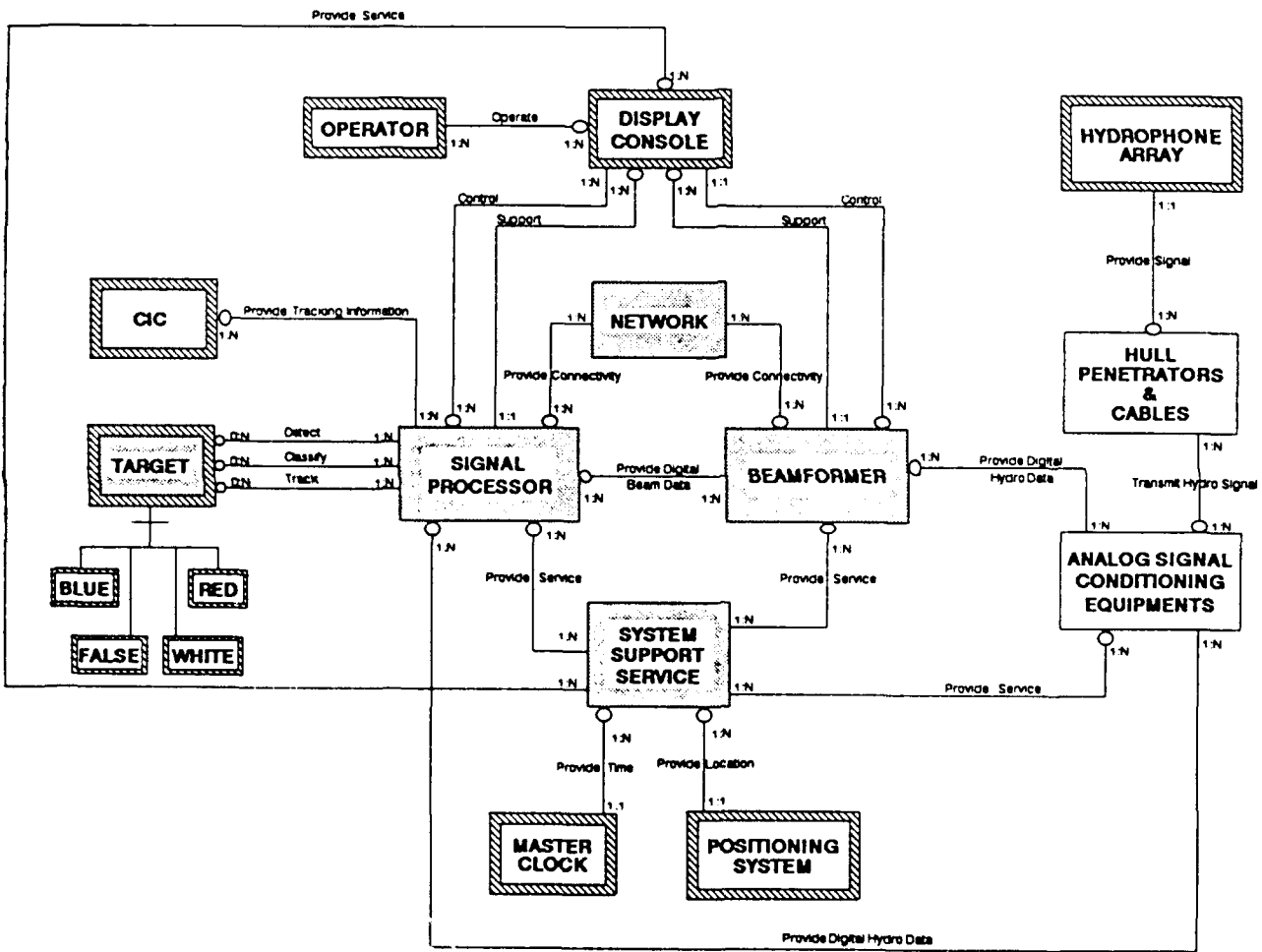


FIGURE 3-5. DESCRIPTION OF THE PASSIVE SONAR OBJECT

System Support Service

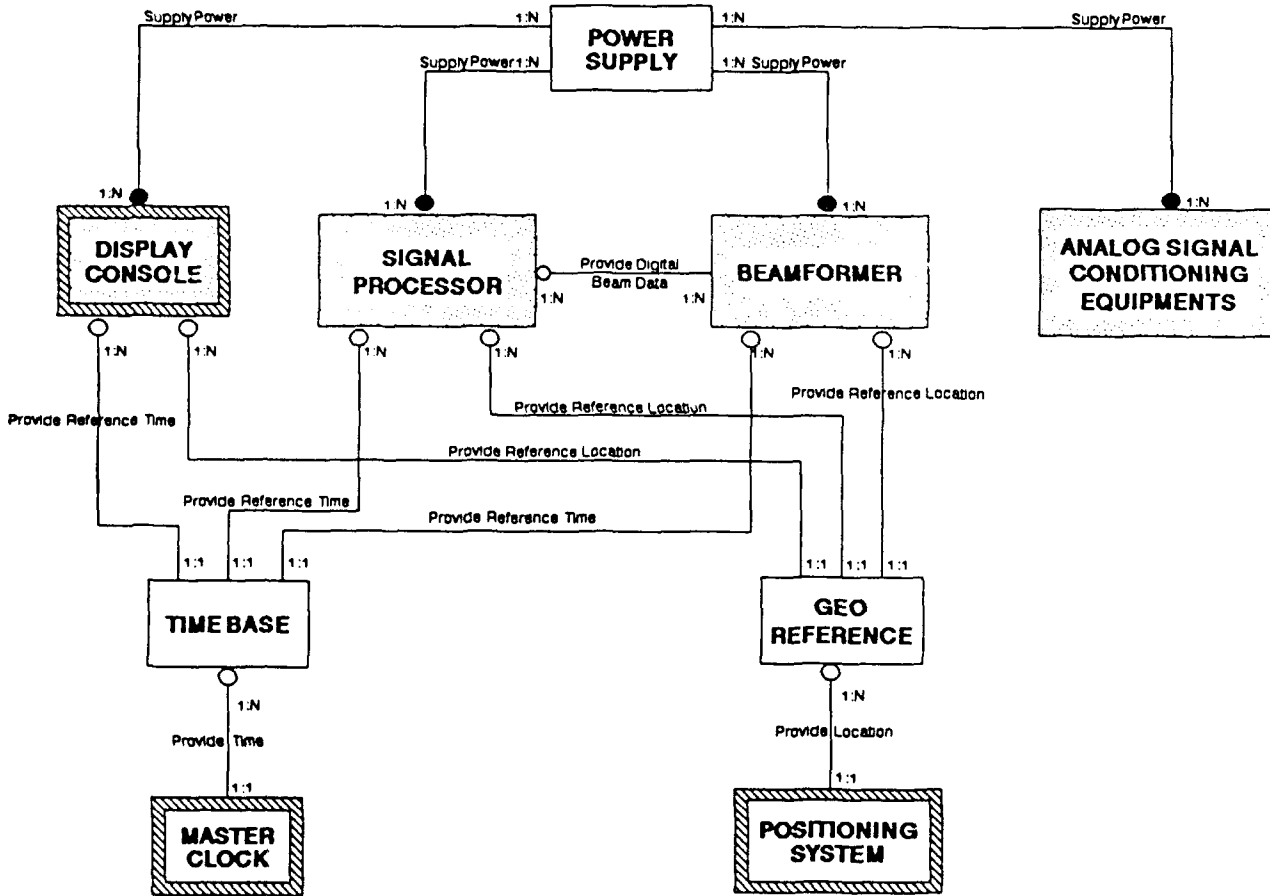


FIGURE 3-6. DESCRIPTION OF THE SYSTEM SUPPORT SERVICE OBJECT

3.1.3 Shortfalls

Even though the concept of HIM has been introduced, it is still immature and more effort is required to complete this work. Besides solving the complexity and scalability issues of the Information Modeling techniques, there is still a need to define the way to capture the non-functional attributes of the objects.

3.2 FUNCTIONAL VIEW

The Functional View is the representation of the system structure: how functions in the system are decomposed and how they interact with each other. The Functional View is represented by data flow diagrams.

In designing large-sized and complex systems, engineers do not have the luxury to build and test the complete system as in designing small-sized systems. It is important to study and estimate correctly what the system will do so that errors can be prevented early in the development process. System functions need to be organized in a clear, systematical, and hierarchical manner so that engineers can see how the functions are arranged, how they interact with each other, and what other options are available in partitioning these functions.

3.2.1 Example Method

The Yourdon-Demarco¹ Structured Analysis method is used as a base in capturing the Functional View. Functions and the data flow between them are represented in a graphical and hierarchical manner such that engineers can analyze the functional construction of the system.

3.2.1.1 Data Flow Diagram (DFD). The DFD is a network of processes and data flows. Requirements are partitioned into processes (functions), and the information that passes between processes is represented by data flows. The DFD usually includes the following components: processes, process specifications (P-Specs), data flows, data dictionary entries, and data stores.

3.2.1.1.1 PROCESSES/FUNCTIONS. Process represents the transformation of information using incoming data to produce outgoing data. Each process is represented by a circle and has a unique name. The process name usually corresponds to the function that it performs such that engineers can have a general idea of what the system is doing. Figure 3-7 shows an example of a process.

To express the transformation in more detail, engineers decompose the process into many different processes. There is no limitation on the level of decomposition; however, all lower levels of decomposition of a process must reflect the same transformation. Also, not all processes have the same level of decomposition. Functions that are more complex require extra partitioning levels. Figure 3-8 shows how some functions are decomposed.

When a process reaches its lowest level of decomposition, where its function is simple enough such that it can be textually expressed without raising any confusion or misunderstanding, it is called a primitive process. Each primitive process must have an associated P-Spec which, textually or mathematically, explains how the process operates. A sample of P-Spec is included in Section 3.2.2.

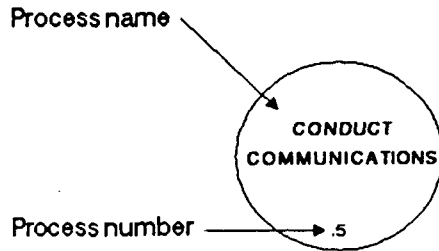


FIGURE 3-7. EXAMPLE OF A PROCESS

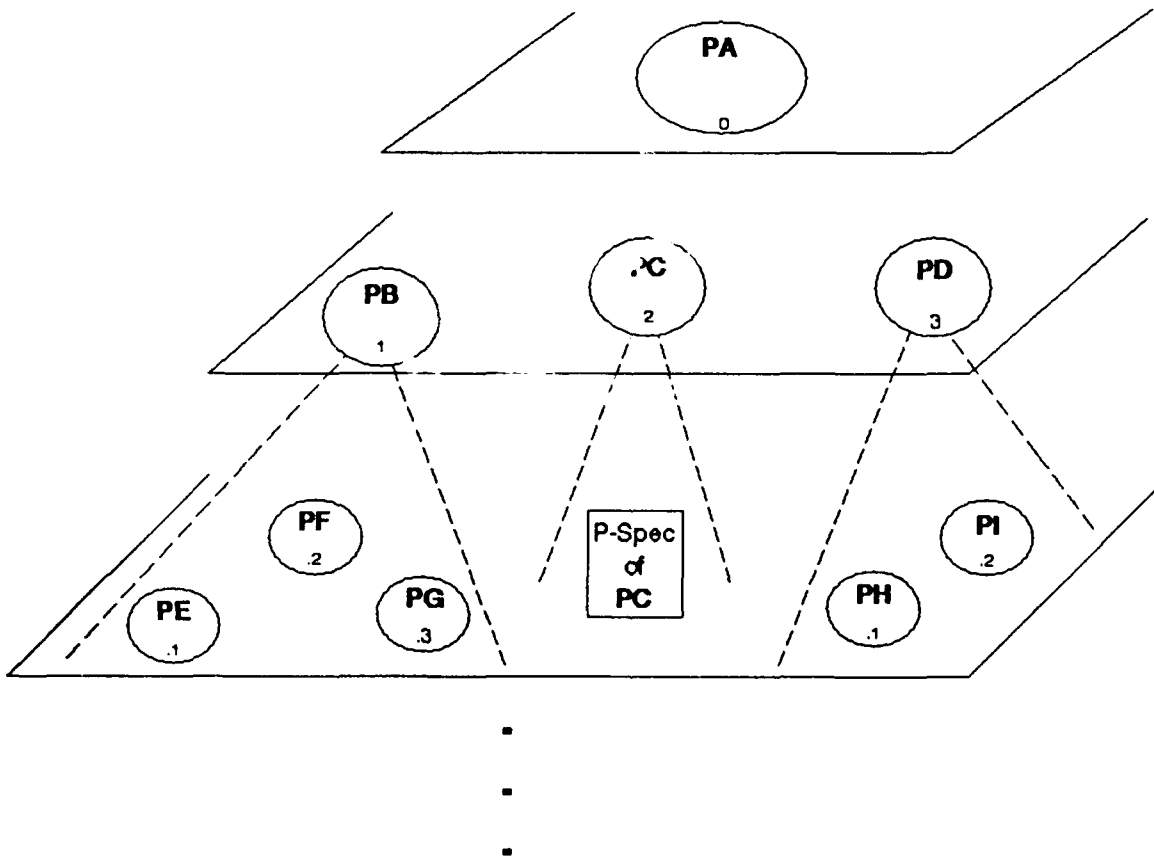
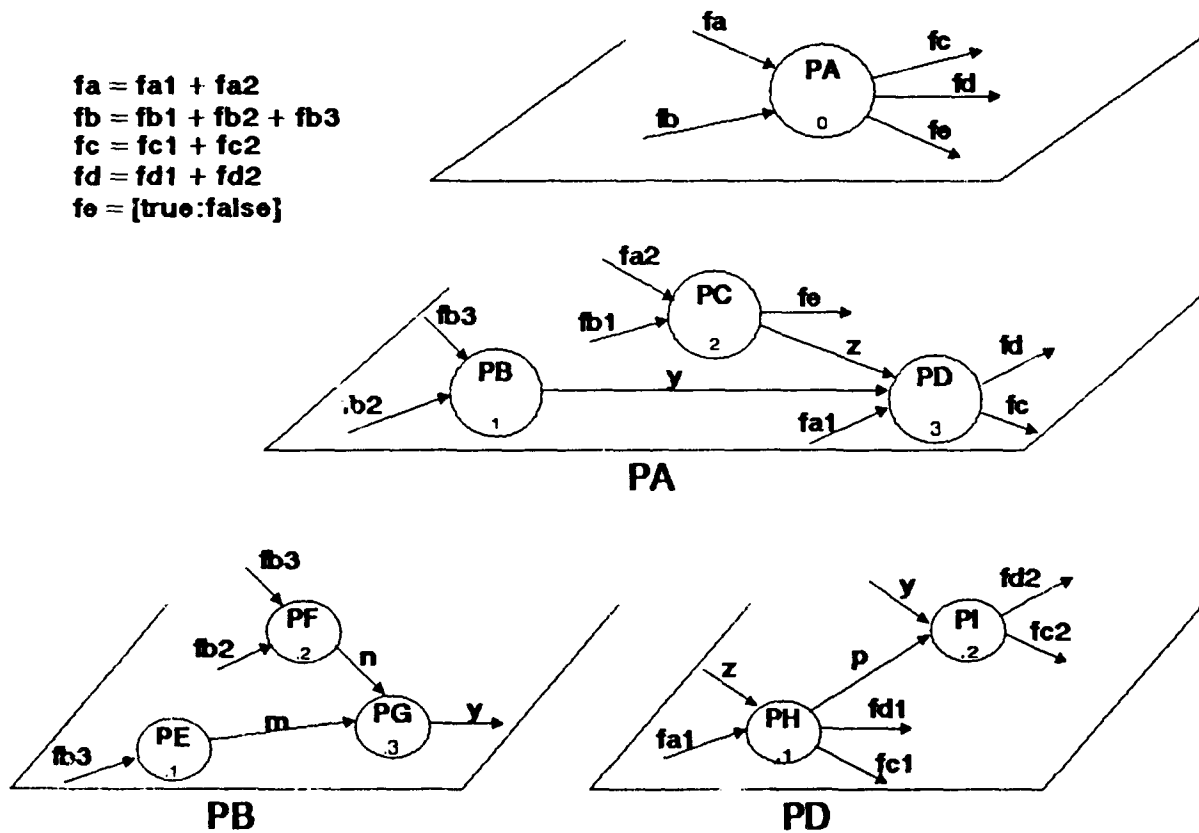


FIGURE 3-6. EXAMPLE OF FUNCTION DECOMPOSITION

3.2.1.1.2 DATA FLOW. Data Flow represents the flow of information between processes. Data flow is represented by an arrow going from one process to another. Data flow's name is unique and reflects the information that it carries. Like process, each data flow can be a composite data flow or a primitive data flow. A composite data flow is a combination of other composite data flows, or primitive data flows, or both. Each primitive data flow has a Data Dictionary Entry to explain the characteristic of the flow.

When the processes are decomposed, their associated data flows will have to be balanced at different levels of decomposition. Figure 3-9 graphically explains how data flows are decomposed and recomposed.



DATA STORE

FIGURE 3-10. NOTATION OF DATA STORE

3.2.1.2 Context Diagram. Context Diagram is a special case of the DFD. It represents the top level of the design where the boundary of the system under design is defined. At this level the whole system is represented by one process; any other outside elements that interact with the system will be represented by terminators. Using the current method, terminators only communicate directly with the system. Research shows that if the interfaces between terminators affect the system's performance, then they have to be captured and studied to a certain extent.

A terminator is represented by a square box and its name should reflect the function or information that it carries. Data flow is still used to show the flow of information both from and to the terminators. Figure 3-11 is an example of the context diagram.

3.2.2 Example Capture

Figures 3-11 through 3-15 capture a portion of the functional view of the passive sonar system. Figure 3-11, the context diagram of the passive sonar system, defines the system's interface. Figure 3-12, one level below the context diagram, shows how the system is functionally decomposed. Figure 3-13 shows that the SIGNAL PROCESSING function is decomposed into six other functions, and figure 3-14 specifies that the SIGNAL CONDITIONING is performed by four different functions: AMPLIFY, FILTER, A/D CONVERSION, and AUTOMATIC GAIN CONTROL. Finally, Figure 3-15 is the P-Spec of the DETECTION function.

3.2.3 Shortfalls

A major shortfall of the capturing method of the Functional View is that it does not address the non-functional attributes of the system, such as dependability, security, and reliability, which are critical to the system's performance. Detailed information of the non-functional attributes can be found in Appendix A and systems engineers should be able to tag this information directly to the functions at any level of decomposition. One potential approach to solve this problem is to make the P-Specs available for all processes in the DFDs--not just the primitive processes.

In addition, automation support often has limited capabilities which make it inefficient when developing large models. For example, two problems were discovered with the Teamwork CASE tool's automation of the Yourdon-

Demarco method. First, Teamwork does not allow for automatic replication of identical parts of a system. When simulating a large system, it becomes necessary to replicate parts of the system, and therefore, it is important that parts of the model can be replicated efficiently. The second shortfall is that it does not offer automatic simulation. A different tool, such as ADAS, has to be used for full simulation. The ad hoc transformation of information from the Functional View to the Environmental View is time consuming and error prone.

3.3 BEHAVIORAL VIEW

The Behavioral View captures the operation of the system functions at different times, under different situations and conditions. Finite state machines have been efficiently used to represent the system's behavior.

Real time systems dynamically change over time and under different conditions. In order to correctly predict the performance of the system, it is essential for systems engineers to be able to capture the causes that make the system change its mode of operation and describe how the system behaves afterward. Following is the overview of different types of finite state machines.

3.3.1 Example Method

Finite state machines have been used in conjunction with other capturing methods, such as Structured Analysis or Information Modeling, to represent the behavior of the system. This section will emphasize different types of the finite state machines that are used to represent the behavior of the system's functions.

3.3.1.1 Decision Table. Decision Table, Figure 3-16, is a tabular representation of the control input and output signals. All inputs are placed on the left side of the table and all outputs are on the right. Each row in the decision table represents a unique logical combination of inputs which provides either a single or a combination of output situations. Note that in the decision table, the output only depends on the current input.

3.3.1.2 Process Activation Table. Process Activation Table (PAT) is used to show how processes are activated by the control input. The process activation logic is enable (1), disable (0), enable in sequence (1,2,...), or unchange (UC). If other processes in the same DFD are not controlled by the PAT, they are considered to be always active. The left side of the table shows the control inputs and the right side shows the processes and how they are activated. A sample of the PAT is included in Section 3.3.2.

3.3.1.3 State Transition Diagram. When a system has more than one mode of operation, a state transition diagram is used to specify the current state of operation of a process, the condition causing the transition from one state to another, and the action required on the change of state. Figure 3-17 summarizes the notations and Figure 3-18 shows an example of the state

transition diagram. A process activate table can also be used in conjunction with the state transition diagram to show the activation of processes on the data flow diagram at different states. (See samples in Section 3.3.2.)

Context-Diagram:5
PASSIVE_SONAR_SYSTEM

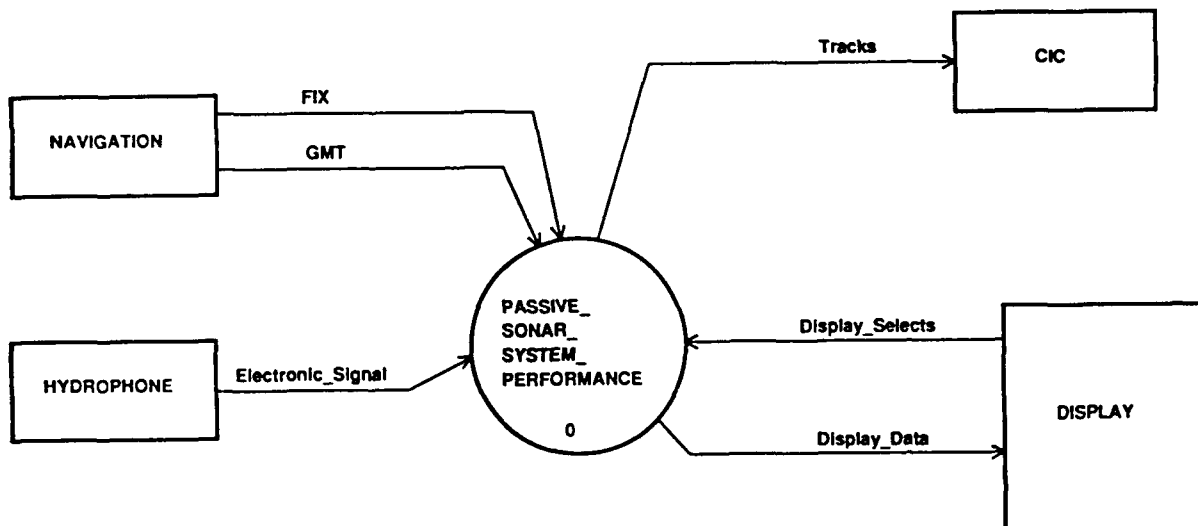


FIGURE 3-11. THE FUNCTIONAL VIEW OF THE CONTEXT DIAGRAM OF A PASSIVE SONAR SYSTEM

0.5
PASSIVE_SONAR_SYSTEM_PERFORMANCE

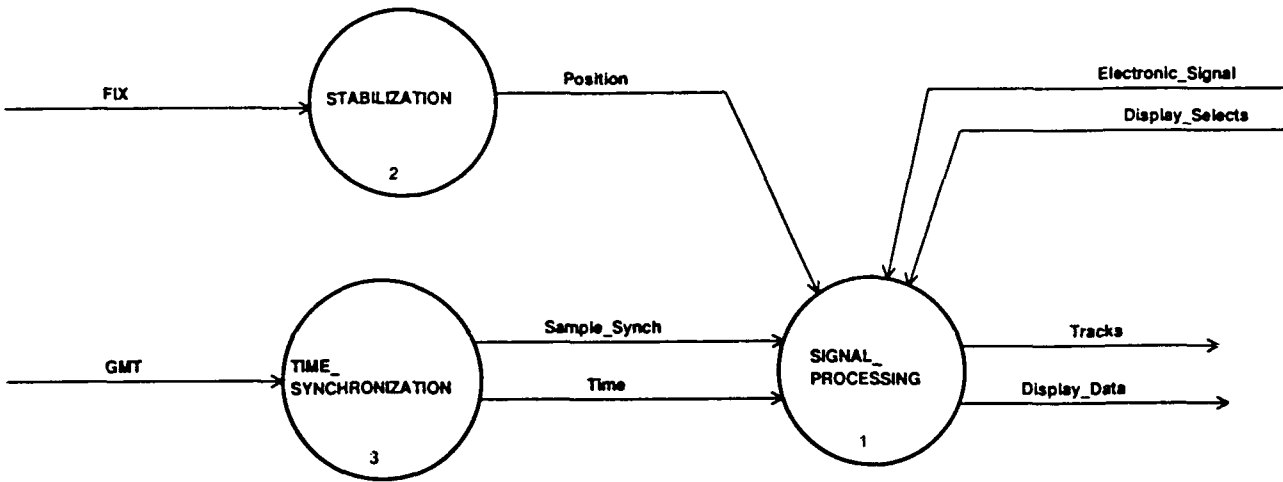


FIGURE 3-12. DATA FLOW DIAGRAM OF THE PASSIVE SONAR SYSTEM PERFORMANCE PROCESS

1:9
SIGNAL_PROCESSING

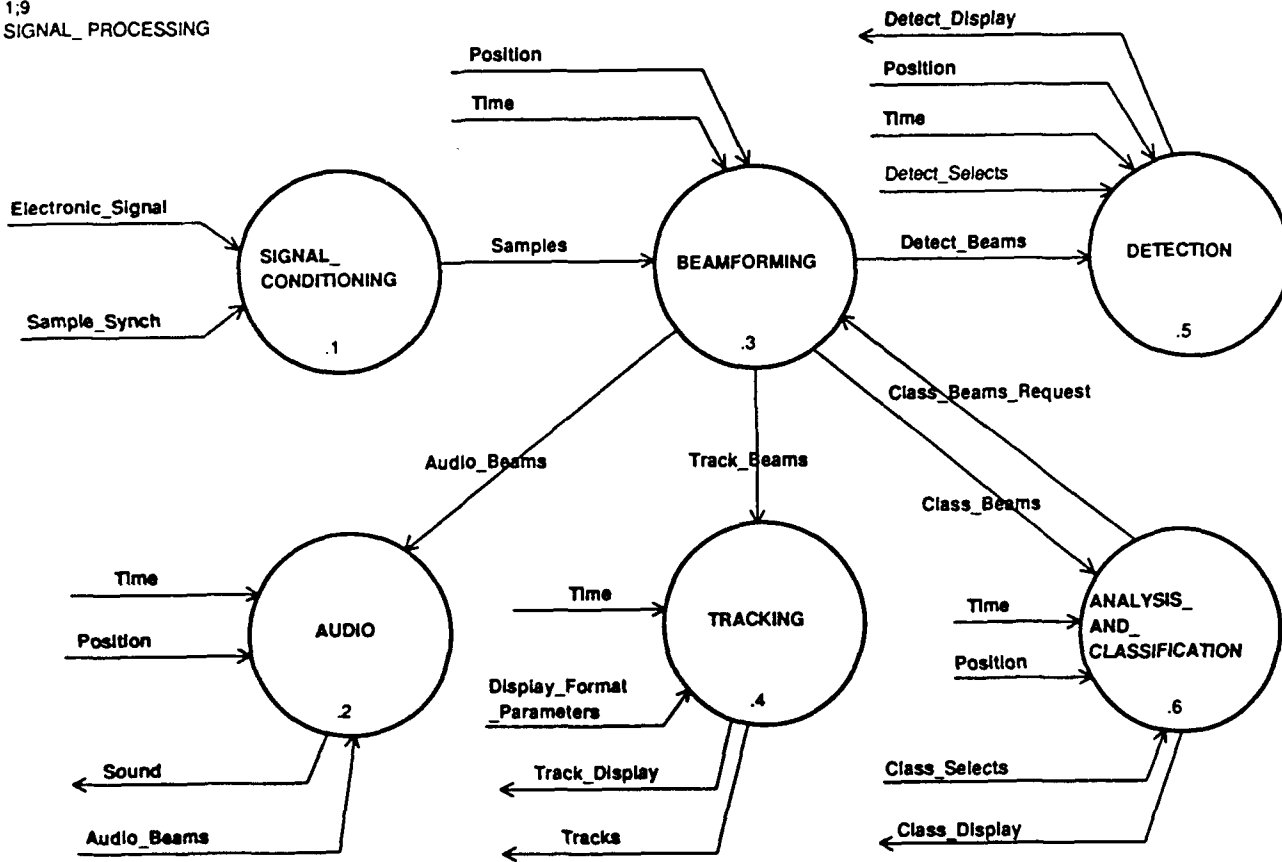


FIGURE 3-13. DATA FLOW DIAGRAM OF THE SIGNAL PROCESSING PROCESS

1.1.3
SIGNAL_CONDITIONING

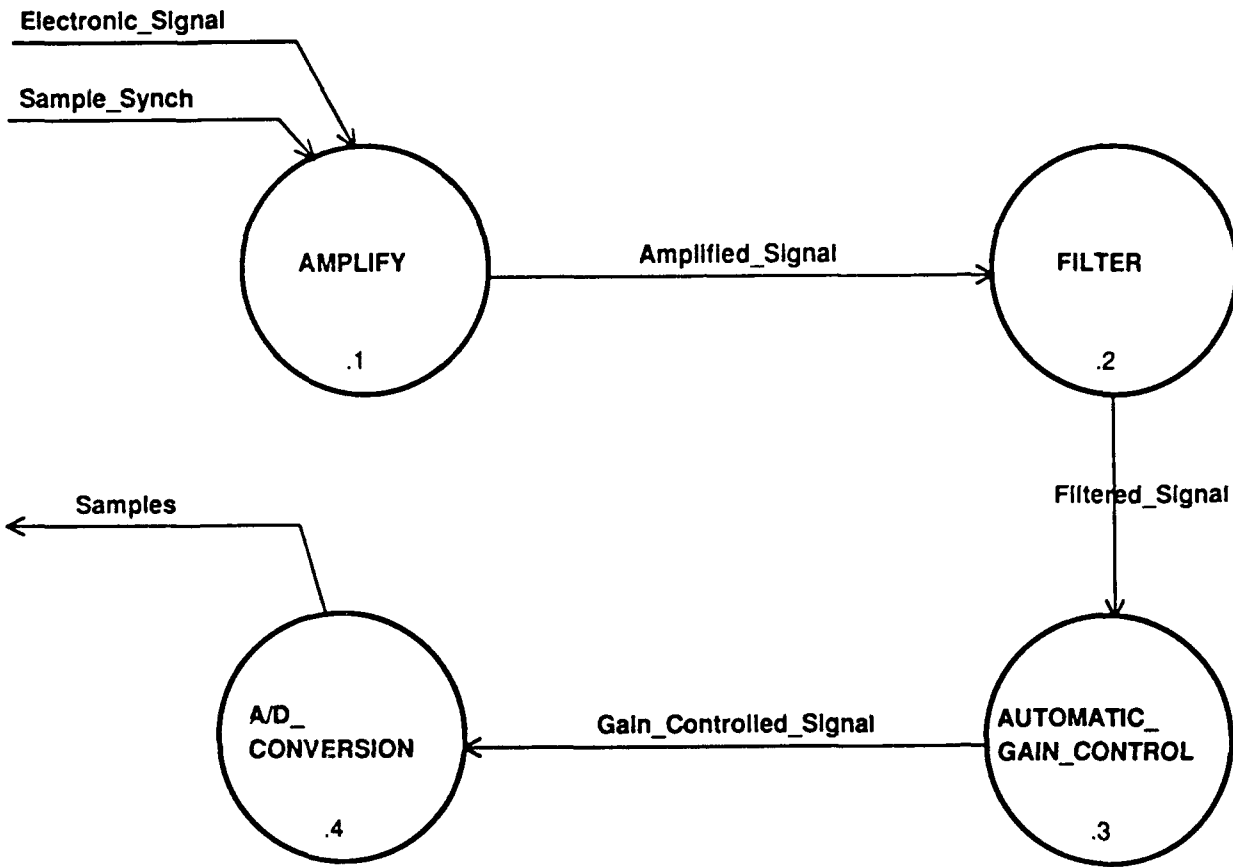


FIGURE 3-14. DATA FLOW DIAGRAM OF THE SIGNAL CONDITIONING PROCESS

NAME:

1.5;2

TITLE:

DETECTION

INPUT/OUTPUT:

Position : data_in

Time : data_in

Detect_Beams : data_in

Detect_Display : data_out

Detect_Selects : data_in

BODY:

The DETECTION function compares the amplitude of the beams to select thresholds, changing the data units to levels or quanta. This requantized data is integrated over time, filtered, and formatted for display. Only a portion of the information about particular beams is provided to the display when the operator makes a selection with the display cursor. Processing will require 10*Nd thousand instruction per second (KIPS).

Timing Requirements: Detection display data will be sent at least four times a second. The display of detection data will be synchronized to within 100 milli-seconds of the audio data.

FIGURE 3-15. P-SPEC OF THE DETECTION PROCESS

ATTACK CAPABILITY	WITHIN KILLING RANGE	DIRECT INCOMING	COUNTER ATTACK	TRAIL
F	F	F	F	DC
F	F	T	F	T
F	T	T	F	T
T	T	T	T	T

FIGURE 3-16. EXAMPLE OF DECISION TABLE

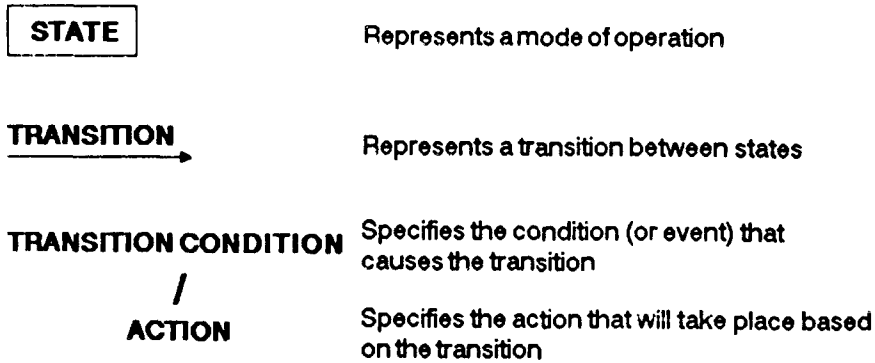


FIGURE 3-17. NOTATION OF STATE TRANSITION DIAGRAM

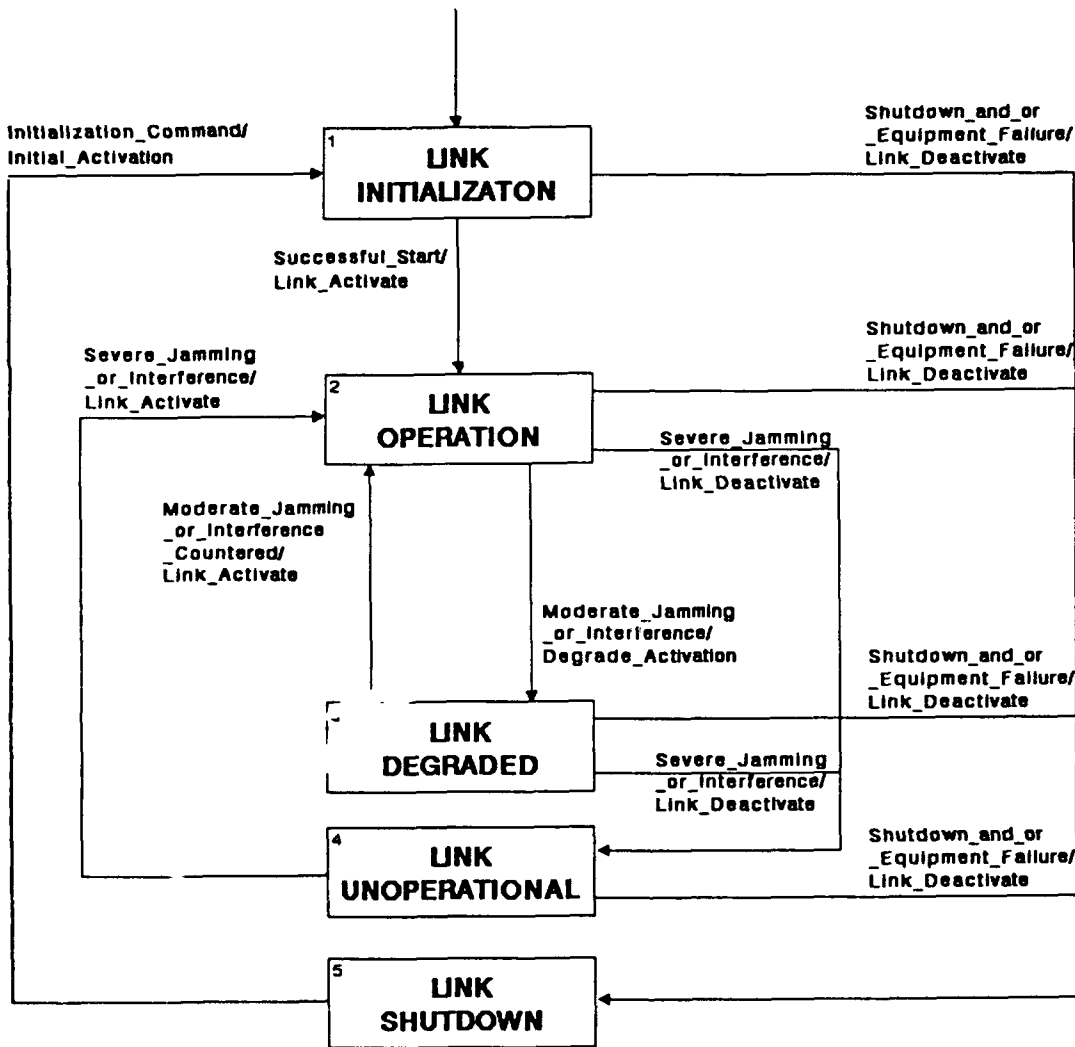


FIGURE 3-18. EXAMPLE OF STATE TRANSITION DIAGRAM

3.3.1.4 State Transition Matrix. When the number of states and/or transitions becomes large and complex, a state transition diagram is replaced by a state transition matrix, Figure 3-19, for better understanding. There is more than one format for the state transition matrix and its use is based on convenience.

Figure 3-19 can be read "if the process is in the 'TRANSIT' state and the 'Arrival' event occurs, then the 'In_Port_Activation' action will take place and the process is transferred into the 'IN PORT' state."

EVENT STATE	Arrival	On_Station	Contact	Trail_Cmd	Within_Range	Fired_Upon	Transit_Order	Target_Gone	Evade_Disrupt Complete_With_Closure	Evade_Disrupt Complete_Without_Closure	Attack_Without Closure_After Evade_Disrupt	SLCM_Launch Detected	Major_Casualty Damage
IN PORT		"Search_Activation" SEARCH					"Transit_Activation" TRANSIT					"In_Port_Activation" DISRUPT	"OOA_Activation" OOA
TRANSIT	"In_Port_Activation" IN PORT					"Evade_Activation" EVADE							"OOA_Activation" OOA
SEARCH			"Track_Activation" TRACK			"Evade_Activation" EVADE	"Transit_Activation" TRANSIT					"In_Port_Activation" DISRUPT	"OOA_Activation" OOA
TRACK			"Close_Activation" CLOSE			"Evade_Activation" EVADE	"Transit_Activation" TRANSIT	"Search_Activation" SEARCH				"In_Port_Activation" DISRUPT	"OOA_Activation" OOA
CLOSE				"Trail_Activation" TRAIL		"Evade_Activation" EVADE	"Transit_Activation" TRANSIT	"Search_Activation" SEARCH			"Attack_Activation" ATTACK	"In_Port_Activation" DISRUPT	"OOA_Activation" OOA
TRAIL						"Evade_Activation" EVADE	"Transit_Activation" TRANSIT	"Search_Activation" SEARCH				"In_Port_Activation" DISRUPT	"OOA_Activation" OOA
EVADE									"Attack_Activation" ATTACK	"Search_Activation" SEARCH			"OOA_Activation" OOA
DISRUPT									"Attack_Activation" ATTACK	"Search_Activation" SEARCH			"OOA_Activation" OOA
ATTACK								"Search_Activation" SEARCH					"OOA_Activation" OOA
OOA													"OOA_Activation" OOA

FIGURE 3-19. EXAMPLE OF STATE TRANSITION MATRIX

3.3.2 Example Capture

Behavior information is incorporated into the data flow diagrams of the passive sonar system to show how a system's behavior is represented. Figures 3-20 through 3-26 capture the behavior aspect of the passive sonar system. The processes that were presented in the example of the Functional View are repeated here with the description of their behavior. Dashed lines in the example represent the control signals that were used to trigger the change of the function's behavior. Since this is only a portion of the model, not all information is shown.

Context-Diagram;3
 PASSIVE_SONAR_SYSTEM

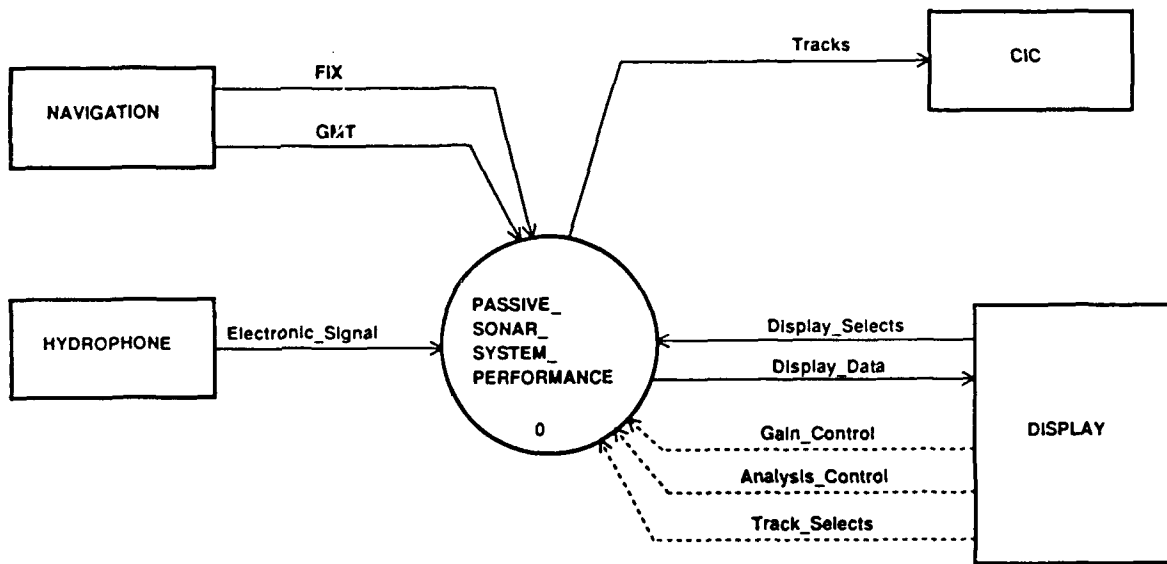


FIGURE 3-20. THE BEHAVIORAL VIEW OF THE CONTEXT DIAGRAM OF A PASSIVE SONAR SYSTEM

0.3
PASSIVE_SONAR_SYSTEM_PERFORMANCE

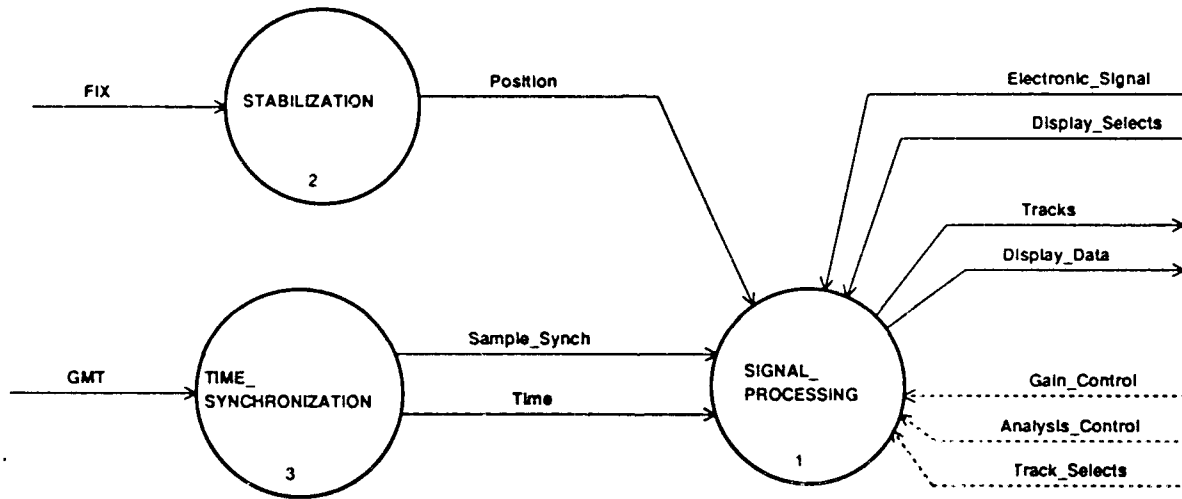


FIGURE 3-21. DATA/CONTROL FLOW DIAGRAM OF THE PASSIVE SONAR SYSTEM PERFORMANCE PROCESS

1.5
SIGNAL_PROCESSING

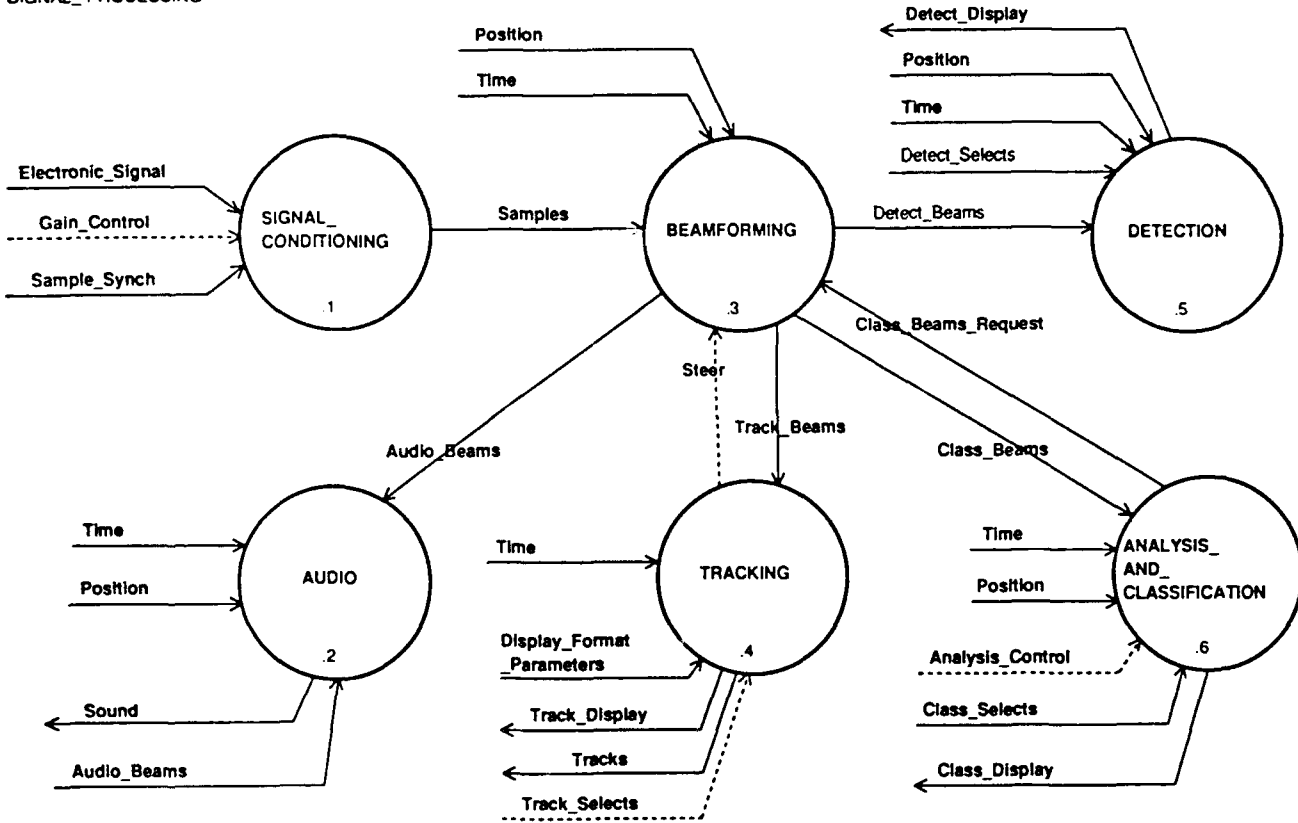


FIGURE 3-22. DATA/CONTROL FLOW DIAGRAM OF THE SIGNAL PROCESSING PROCESS

1.1:2
SIGNAL_CONDITIONING

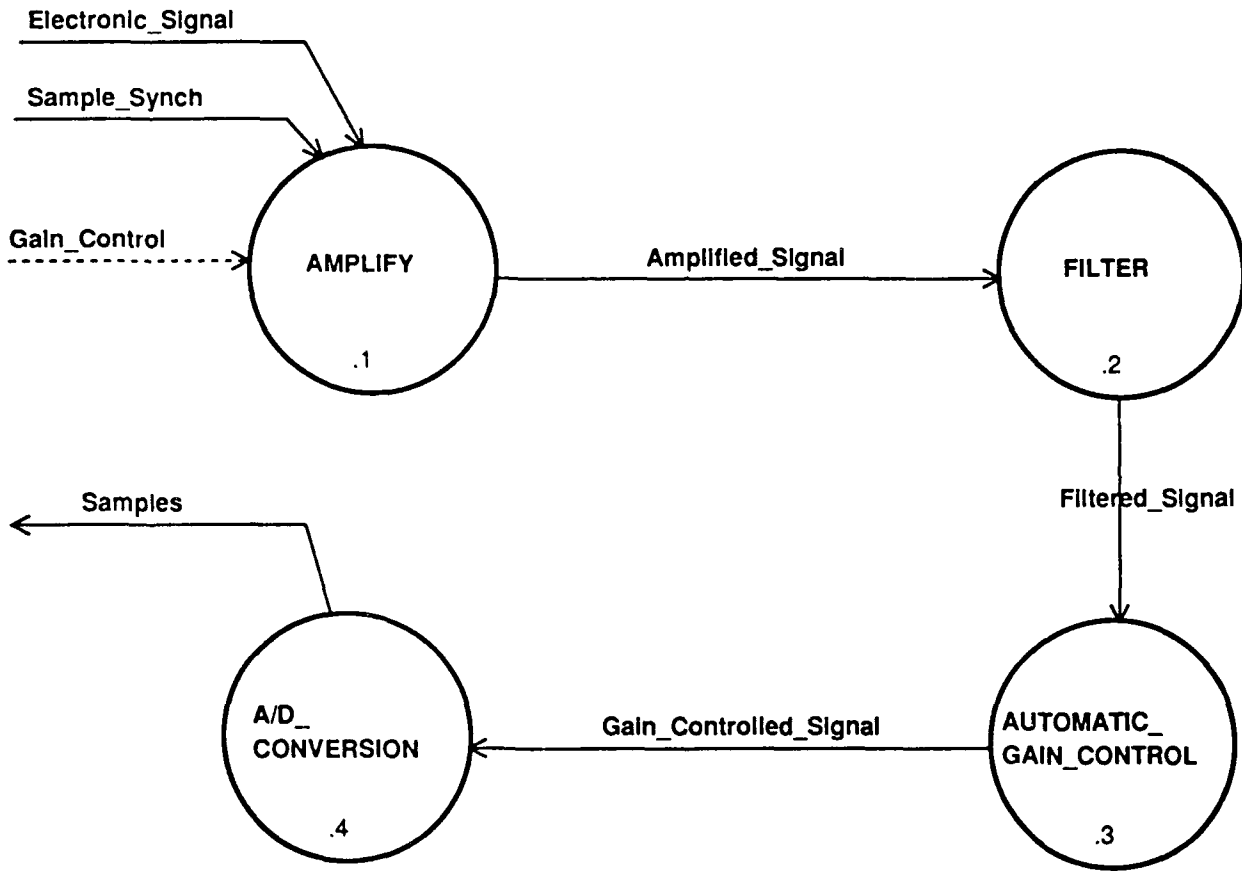


FIGURE 3-23. DATA/CONTROL FLOW DIAGRAM OF THE SIGNAL CONDITIONING PROCESS

1.6.3
ANALYSIS_AND_CLASSIFICATION

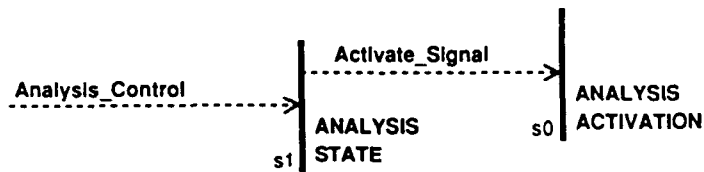
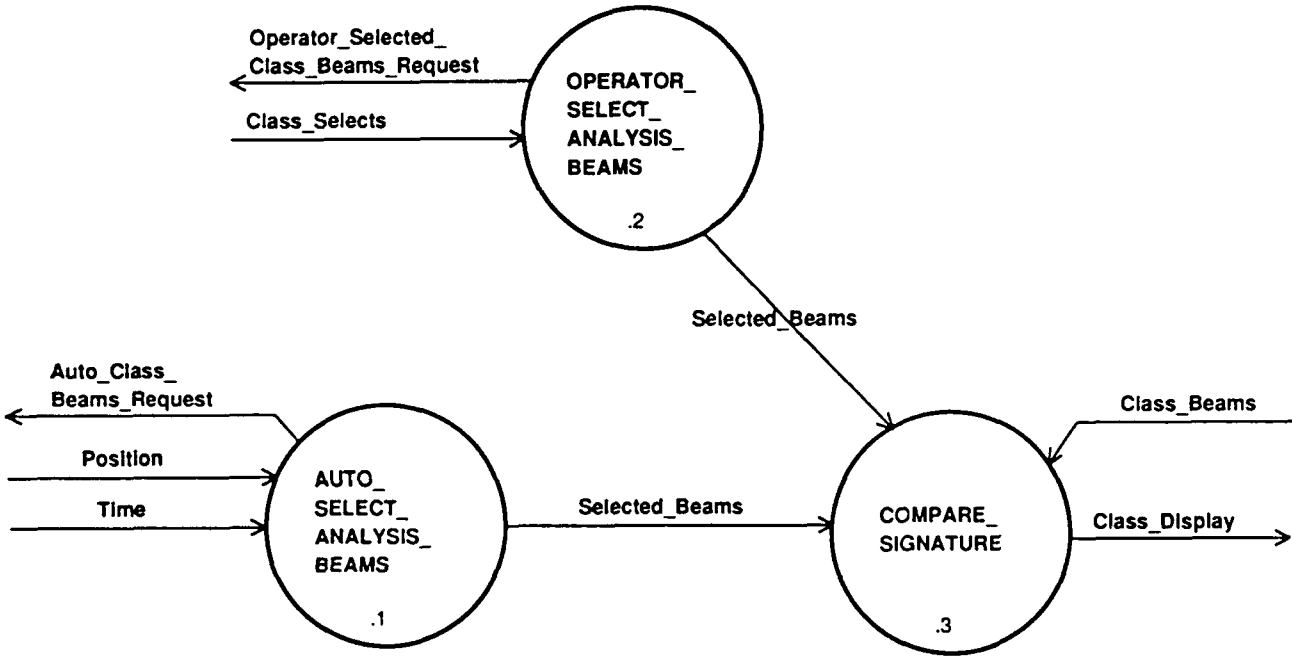


FIGURE 3-24. DATA/CONTROL FLOW DIAGRAM OF THE ANALYSIS AND CLASSIFICATION PROCESS

1.6-s1;2
ANALYSIS STATE

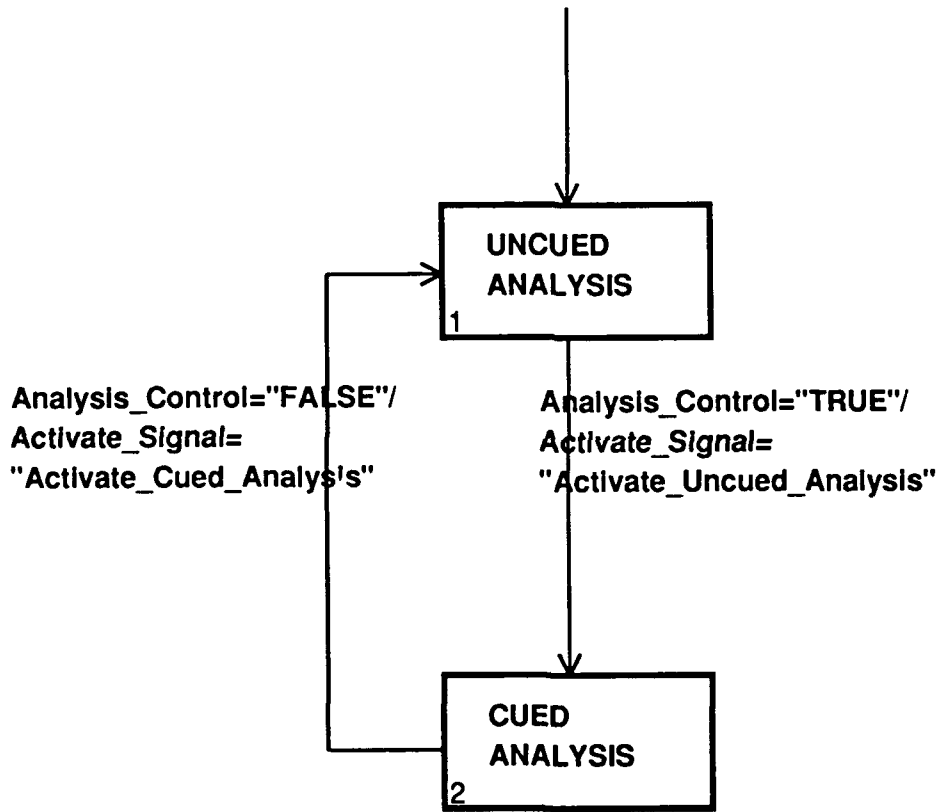


FIGURE 3-25. STATE TRANSITION DIAGRAM FOR THE ANALYSIS STATE

1.6-s0;2
ANALYSIS ACTIVATION

Activate_Signal	1	2	3
"Activate_Uncued_Analysis"	1	0	2
"Activate_Cued_Analysis"	0	1	2
	AUTO SELECT ANALYSIS BEAMS	*OPERATOR SELECT ANALYSIS BEAMS*	*COMPARE SIGNATURE*

FIGURE 3-26. PROCESS ACTIVATION TABLE FOR THE ANALYSIS STATE

3.3.3 Shortfalls

Since the Behavioral View is tightly related to the Functional View, it shares some common shortfalls with that view in terms of the capturing method as well as the supporting tool. In addition, the capturing method of the Behavioral View has its own major shortfall: the inability to capture the critical real-time attributes. Current methods do not allow the capturing of deadline issues or the reconfiguration of the system after failure. Extension of the methods to cover this critical real-time information has to be developed.

3.4 IMPLEMENTATION VIEW

The Implementation View represents the architectural descriptions and performance capabilities of all hardware, software and hardware resources that perform the system functions as well as the interconnection between all resources. To guarantee the consistency of the development process, a Resource Model which can be mapped one-on-one to the Logical Model (see Chapter 4 for detailed information on models) will be captured and analyzed.

After the system's functions and behaviors are identified, to avoid massive failure in the design phase, it is important to understand the resources that will be used to perform these functions. The capability to capture and analyze different resources to achieve high performance at low cost in the early stage of the design cycle will yield maximum payoffs in the production and maintenance phases. The analysis of the resources also reduces the complication in upgrading systems when better technology is introduced. In reality, most resources already exist and their performance is documented one way or another. If the resource representations are formalized, then engineers can study the trade-offs of one resource versus the other, and can identify which resources can be modified or will need to be built for the system under design.

3.4.1 Example Method

The Resource Model and the mapping method of functions to resources are usually used to represent the Implementation View.

3.4.1.1 Resource Model. The Resource Model includes a graphical representation of the proposed system architecture. Required resources should be defined from the system's requirement and from the Functional View. All candidate resources and their interconnections are formally captured so that analysis can be performed correctly. Once the proposed Resource Model is complete, its analysis will identify all resource constraints, and decisions will be made on whether new resources should be built, existing resources should be modified, interconnectivities between resources should be made, or even the required or defined function itself should be changed. With such high impact on the design, it is important to define a formal method to capture resources.

In the past, resources were known as the system's physical hardware and software. As the system expands its size and complexity, many human decisions become part of the system--rather than its interface; hence, the concept of humware (or humanware) is introduced. Depending on the method and/or how they were used, hardware, software, and humware were defined differently. In this paper, only the general concept of capturing resources will be discussed. Detailed description and definitions of each particular type of resource will be defined in the future.

Like many other aspects of the system, it is necessary to represent resources in a hierarchical manner. This requirement is mainly derived from the fact that current advanced technology allows engineers to develop very high performance resources which can perform many functions at the same time. In order to maximize the utilization of such resources, engineers should understand how they operate as a whole as well as how they are partitioned to support various functions. In addition, since the functions and behaviors of the system are hierarchically captured, the resources that were used should be represented by the same method.

Beside the system's resources, the Resource Model also includes any resources that are used to support the interested external functions and behaviors that are captured in the external Logical Model. Consequently, the external resources will have different notations. Also, the issue of capturing resources that are used to support both internal and external functions at the same time has been raised. Figure 3-27 shows an example of a Resource Model.

3.4.1.2 Mapping Function To Resource. After the Resource Model is constructed, functions and behaviors of the system will be allocated to resources. The tabular representation is the most popular current approach of mapping of functions to resources. Functions and information flows in the Logical Model are listed with their associated required performance and the resource that was dedicated to support them. This information can then be used to support the system's analysis. Since this method is a straight static allocation, many mapping options should be constructed to support the analysis which will guarantee the correctness as well as the highest performance of the system.

Even though the allocation of function to resource is performed in every design process, it has never been formalized at the complex system level. The mapping table is usually customized for a single project. There are no rules to define what it should look like or what type of information is contained within it. An example of this table is shown in Figure 3-28.

3.4.2 Example Capture

Figures 3-27 through 3-29 show a candidate Resource Model of the passive sonar system and the mapping of functions and data flows to resources. Although the resources (CPU 1, CPU 2, etc.) in Figure 3-27 are annotated with two attributes, in the full description of the Resource Model, of course, all resources attributes will be captured.

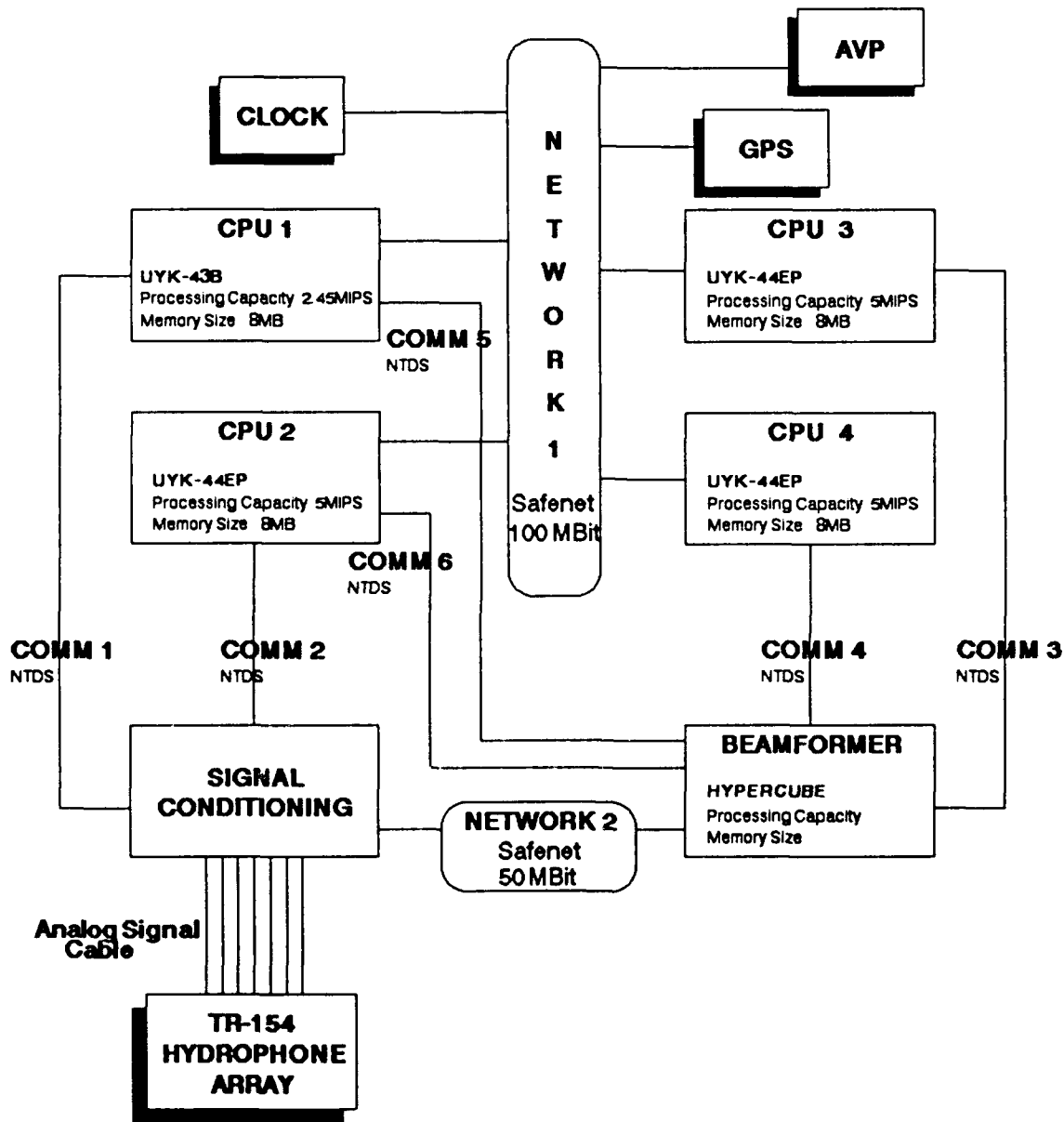


FIGURE 3-27. CANDIDATE RESOURCE MODEL OF THE PASSIVE SONAR SYSTEM

FUNCTION	PERFORMANCE REQUIRED		RESOURCE OPTION 1	RESOURCE OPTION 2	RESOURCE OPTION n
	SPEED	MEMORY			
DETECTION	.8MIPS	14MB	CPU 3, CPU 4	CPU 2, CPU 3	
TRACK	.2MIPS	6MB	CPU 1	CPU 2	
ANALYSIS	.5MIPS	6MB	CPU 2	CPU 4	
AUDIO	.1MIPS	2MB	CPU 4	CPU 1	
SIGNAL CONDITIONING	SPECIAL		SIGNAL CONDITIONER	SIGNAL CONDITIONER	
BEAMFORMING	SPECIAL		BEAMFORMER	BEAMFORMER	
.

FIGURE 3-28. EXAMPLE OF ALLOCATION OF FUNCTIONS TO RESOURCES

CONNECTIVITY	PERFORMANCE REQUIRED	RESOURCE OPTION 1	RESOURCE OPTION 2	RESOURCE OPTION n
Electronic_Signal	Analog Low Pass below 512 Hz	CABLE	CABLE	
Sample_Synch	$\frac{32 * 1}{\text{Sec}}$	CABLE	CABLE	
Samples	$\frac{32 * 512 \text{ bits}}{\text{Sec}}$	NETWORK2	NETWORK2	
Detect_Beams	$\frac{32 * 80 * 512}{\text{Sec}}$	COMM3, COMM 4	COMM 2, COMM 3	
Track_Beams	$\frac{32 * 10 * 512}{\text{Sec}}$	COMM 1	COMM 1	
Audio_Beams	$\frac{32 * 2 * 512}{\text{Sec}}$	COMM 1	COMM 1	
Class_Beams	$\frac{32 * 10 * 512}{\text{Sec}}$	COMM 1	COMM 1	
.

FIGURE 3-29. EXAMPLE OF ALLOCATION OF DATA FLOWS TO RESOURCES

3.4.3 Shortfalls

Even though different techniques are being developed and used to capture various types of resources in the Implementation View, they do not represent the resources at the system level. The ability to capture resources at different detail levels allows systems engineers to perform analysis on system functions at the high level of their decomposition. In addition, the mapping techniques do not reflect the dynamic allocation from functions to resources. This limitation restricts the analysis of the full utilization capability of system resources.

3.5 ENVIRONMENTAL VIEW

The Environmental View includes a representation of the outside world that will interact with the system under design. The intention here is not to model something that is not included in the system but to study the external interface of the system such that any real situation that may affect its performance is taken into consideration. The current approach is to extend the other four views beyond the boundary of the system under design and capture other environmental information in a documented form. It is important to understand that the captured external information is only required for the analysis; therefore, the external views only include the information that directly relates to the system under design.

The Environmental View establishes the conditions and environment for the system under design including the initial state of the system, external interfaces to related systems, environmental conditions (including acoustic, electromagnetic and meteorological conditions), operational constraints, likely strategic and tactical considerations and other pertinent items.

The following scenario elements illustrate a sampling of the types of categories that may be defined when developing the approach for defining the Environmental View. Items necessary for scenario development include force intentions, composition, and concept of operations. The description of force intentions involves planned motion, targets of interest, pre-planned responses, and other intended actions. These scenario elements are applicable to the design and analysis of the complex computer systems supporting a large naval force.

Specification of the geographic areas of interest could serve to further focus in on appropriate subsets of the threat force, geopolitical influence, and the rules of engagement databases. Environmental conditions including acoustic, electromagnetic, and meteorological conditions also need to be taken into account when developing a geographic scenario specification.

Concept of operations specification will involve, at a minimum, the elements of rules of engagement, transit planning, communications planning, and tactical planning.

3.5.1 Example Method

Historically, the Environmental View has been described in an ad hoc manner through test plans, context diagrams, simulation/stimulation (sim/stim) descriptions, and system modeling/simulation scenario descriptions. Since the capturing methods of this view are very immature, this paper proposes a potential approach to capture this view.

The Environmental View is captured in two principal parts: first, as a structured Environmental View document, and second, as an external model which represents the environment and all the external interfaces and other relationships which originate and terminate outside the system boundaries.

3.5.1.1 Environmental View Document. The Environmental View document is the principal view for capturing system requirements, boundaries, design constraints and other issues relating to the system under design. It is the document which links high-level requirements (e.g., the Top Level Warfare Requirements and Top Level System Requirements for a naval system) to the ultimate system design. The Environmental View document provides a precise statement of the requirements and assumptions for the architecture.

The Environmental View document has six goals:

Summarizes the architecture's high-level performance requirements. These requirements are based on the Top Level Warfare Requirements (TLWR). The TLWR's Mission Success Criteria (MSC) quantify the acceptable architecture performance. These critical performance areas must be defined and prioritized according to the goals of the sponsor.

Describes the architecture's scope and boundaries. The design boundaries and scope of the architecture are established through discussions with the architectural sponsor. The scope limits the design space by defining the external boundaries, interfaces and relationships between the system and any entities outside the system.

Collects the high-level design guidance. Guidance is provided by the architectural sponsor and other high-level sources for future use in the design process after publication of the System Requirement Documentation. Included in this high-level guidance are system design requirements and the performance requirements stated in the TLWR.

Summarizes the threat forces which the architecture must face. For military systems, areas for consideration include expected threat operations, organizations, platform order of battle, expected platform capabilities and perceived objectives. This summary of information establishes the tactical framework in which the system has to function.

Identifies architectural performance modeling considerations. The MOEs and the measures of performance (MOPs) provide the guidelines by which the architecture's requirements and performance are evaluated. The level of fidelity required for each analysis must be determined. The

performance modeling parameters to be used in the modeling efforts are also identified. These parameters may also need supplementary analysis.

Enumerates issues that affect the design, performance, cost, and risk of the architecture. These issues relate to technology development and capabilities, national policy matters, and strategic/tactical considerations. This goal helps quantify the technical and design risks of the architecture.

The Environmental View is captured as a structured requirements document. It is expected to evolve as system requirements are analyzed and as logical and implementation design options are examined. It may also need updating as perceptions of the threat and expected engagement scenarios change throughout the development process.

3.5.1.2 External Model. Two types of external models have been identified: external Logical Model and external Resource Model. The external Logical Model includes Informational, Functional, and Behavioral Views of the external entities that influence the performance of the system. Logically, the external Logical Model is gradually constructed in conjunction with the system. For example, when the Informational View is captured, it also includes, not all, but some external objects that are directly related to the objects within the system. The detail descriptions of these external objects may not be completed, but their existence will notify engineers about their effect on the system. The same process happens when the Functional and Behavioral Views are defined. Capturing the system under design jointly with its external entities is also very helpful for engineers when defining the boundary of the system.

An external Resource Model will be defined based on the external Logical Model. The method to represent external resources is the same as representing internal resources, except that the existence of the shared resources (i.e., resources that are used by both external and internal functions) has to be taken into consideration during the mapping from functions to resources.

3.5.2 Example Capture

The following is a list of the documents which might be used in development of the example passive sonar system Environmental View. They illustrate the type of documents used as references in developing the Environmental View:

- * ASW Top Level Warfighting Requirements Document
- * AN/BSY-2 Submarine Sonar Operations Manual
- * ASW Master Plan, dated Mar 88
- * Area ASW Architecture Preliminary draft
- * System Threat Assessment Report (STAR)

- * Acoustic Threat Characteristics
- * System Threat Assessment Report - Submarine Systems Vol I
- * System Threat Assessment Report - Submarine Systems Vol II

Since the external information is captured in conjunction with internal information, the external model should only be the extension of the internal model. The context diagram of the Informational View of the passive sonar system (Figure 3-4) shows how some external objects are captured.

3.5.3 Shortfalls

The supporting method of the Environmental View is very ad hoc and informal. Currently, this view is captured in part in various design activities, such as test plans or test scenarios. In the near future a complete and formal method needs to be defined to achieve the goals of this view.

CHAPTER 4

INTEGRATION AND TRANSITION BETWEEN VIEWS

Although the design has to be captured in many different domains, there is a definite interconnection among them. Some may have stronger relationships than others; however, they must all be understood clearly for better analysis. In addition, due to the complexity and the large size of the system, the automation support to transfer information from one design activity to another becomes critical and requires automation.

4.1 DESIGN DOMAIN RELATIONSHIPS AND DEPENDENCIES

Each design view represents the system from a particular perspective and highlights different aspects of the design; however, they are not independent. The views represent different aspects of the same system and therefore must be consistent. The features of a particular view can directly or indirectly impact, to a greater or lesser degree, the design in another view, depending on how the relationship between the views is specified. Relationships defined between the views establish the nature of the inter-view dependencies. If very strong interdependencies and rules for linkage between the views are established, the system design is rapidly constrained when a single view is specified. If no formal linkage is established between the system views, then the views evolve independently which can result in severe inconsistencies. The ideal set of domain relationship definitions would ensure that the five views are entirely consistent with each other, without over constraining the design options available in a given domain.

4.1.1 Relationship

The following paragraphs describe a notional set of domain relationships between the five views. The five views are first organized into three groupings which represent models of the system at different levels of abstraction from highly abstract to complete physical implementation. These three models of the system establish a basis for simulation and analysis of the system under design at the conceptual, logical, and implementation level.

4.1.1.1 Conceptual Model. The Conceptual Model includes the Environmental View and the Informational View. It highlights the system concept of operations from an operator's perspective using abstract object-oriented constructs and captures the expected operational environment, system requirements and system design constraints. The Conceptual Model provides a vehicle for the customer and the systems engineering team to form a clear mutual understanding of the proposed system operational requirements and constraints early in the design process and provides a mechanism for relating

all aspects of the design back to those requirements and constraints through all phases of the design process. The Conceptual Model also provides a basis for developing a simulation supporting system requirements analysis within the context of the environment and any larger mega-system of which it is an element.

4.1.1.2 Logical Model. The Logical Model which includes the Functional and the Behavioral View captures the system in terms of the functions, data and control without regard to a specific set of physical resources. The Logical Model provides a vehicle for the systems engineering team to specify and capture the system design in terms of what the system must do functionally versus how it is accomplished in hardware and software. This partitioning allows the systems engineering team to deconvolve design issues and to address the logical (or non-implementation) aspects of the design. The Logical Model provides a basis for developing a simulation which can be used to analyze many key aspects of the design such as the functional correctness of the design.

4.1.1.3 Implementation Model. The Implementation Model, which is essentially the Implementation View, includes resource descriptions and a mapping which relates the functions, data flows and control from the Logical Model directly to the resources captured in the Implementation View. The Implementation Model provides a vehicle for the systems engineering team to specify and capture the system design in terms of the specific hardware, software and human resources which perform the functions specified in the Logical Model. The Implementation Model provides a basis for developing a simulation which can be used to estimate the system performance for a given implementation option.

4.1.2 Dependencies

Figure 4-1 groups the five views and summarizes the top level relationships between the three models of the system. The relationships between the models are categorized as either weak linkages or strong linkages. Weak linkage from the Conceptual Model to both the Logical and Implementation Models implies a requirement for consistency checking and formal traceability between two models without close coupling of the design elements within the models. Strong linkage from the Logical Model to the Implementation Model represents a direct mapping or correlation between the functions and data flows of the Logical Model to the specific resources which perform each function and provide the communication connectivity for each data flow.

4.1.2.1 Conceptual Model to Logical Model Linkage. The linkage between the Conceptual Model and the Logical Model is largely characterized as the traceability of requirements, constraints, and system concept of operations captured in the Conceptual Model to the functions, data, and control features of the design captured in the Logical Model. The Informational and the Environmental View provide the systems engineer with a basis for developing design features in the Logical Model. In general, groupings of objects and relationships in the Informational View and groupings of requirements and constraints in the Environment View may suggest required functions, data flow

and control features to the systems engineer for inclusion in the Logical Model.

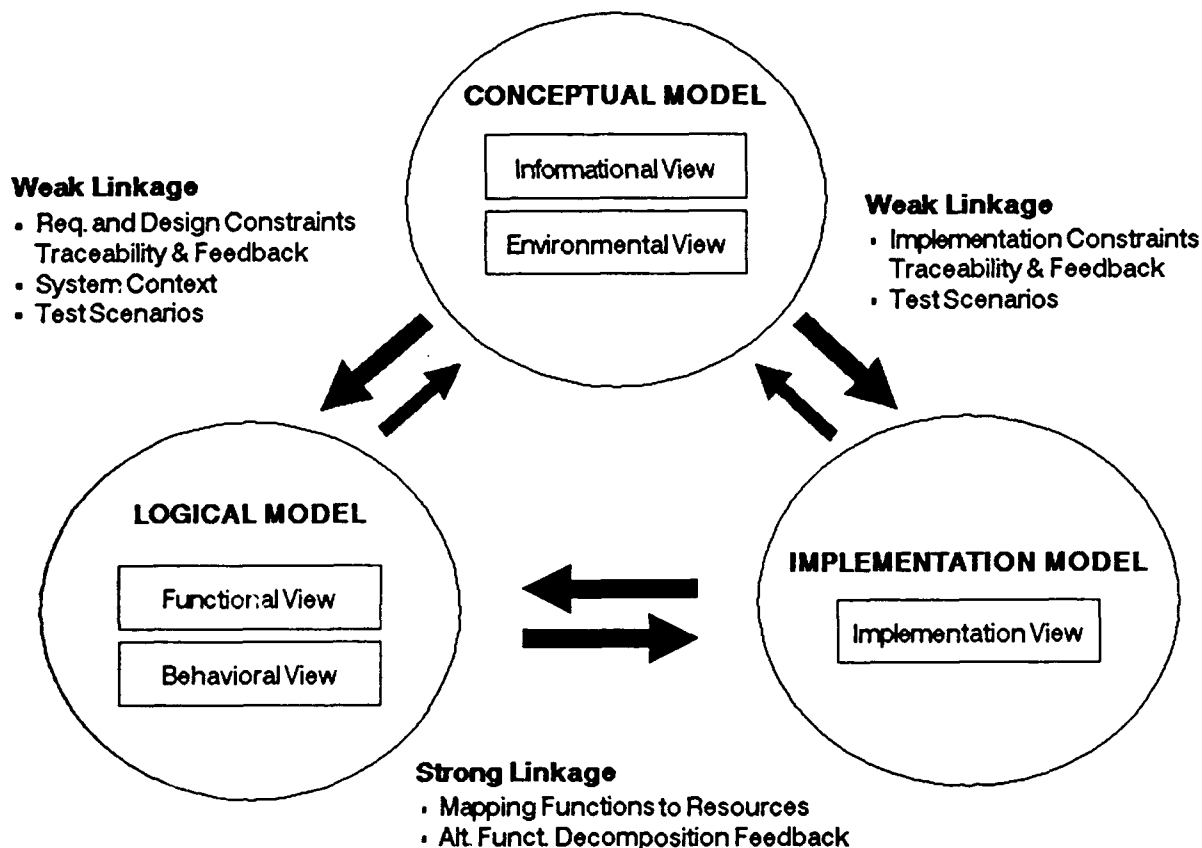


FIGURE 4-1. TOP LEVEL DESIGN DOMAIN RELATIONSHIPS

A clear structured picture of the system boundaries and external relationships as defined and captured in cooperation with the system sponsor(s) and potential users resides in the Informational View. This provides a strong basis for establishing a Functional View context diagram. No formal linkage is currently proposed explicitly linking the Informational View boundary objects and external relationships with the external terminators in the Functional View context diagram; however, the systems engineer can draw heavily on the features of the Informational View in creating the Functional View context diagram.

The requirements and constraints captured in the Environmental View provide the design rationale for the creation of functions, data flow and control features in the Logical Model. Logical groupings of design requirements and constraints established by the systems engineer are linked to one or more functions establishing traceability from the Conceptual Model to the Logical Model. This linking of requirements and constraints to functions, data and control provides a mechanism for completeness and consistency checking between the two models.

The following rules summarize a minimal formalism for linkage between the Conceptual and Logical Models:

Every requirement and design constraint captured in the Environmental View must be linked to at least one function in the Functional View;

Every object and relationship in the Informational View must be linked to one or more functions and/or data flows in the Functional View;

Every function and data flow in the Functional View which is not linked to one or more requirements or design constraints in the Environmental View, and one or more objects and/or relationships in the Informational View must have a clear rationale as a derived requirement or functionally decomposed element of the design.

This mandatory association of the two models supports development of a complete Functional Model which addresses all the system requirements and concept of operations as captured in the Conceptual Model. This type of traceability does not guarantee a complete Functional Model. It can, however, identify objects and relationships from the Informational View and requirements and constraints from the Environmental View which have not been addressed in the Logical Model. This traceability also serves to identify functions and data flows which do not have a clear basis for inclusion in the logical design.

4.1.2.2 Conceptual Model to Implementation Model Linkage. The linkage between the Conceptual Model and the Implementation Model is characterized as the traceability of requirements and constraints captured in the Environmental View to the hardware, software, and human operators which make up the Implementation View. The Environmental View captures two key areas which are linked directly to the Implementation Model: first, the specific implementation requirements and constraints which are linked directly to the applicable resources captured in the Implementation View, and second, the limiting scenarios and MOEs which are used to analyze the Implementation Model and generate system performance estimates.

High-level design guidance and constraints provided by the system sponsor (i.e., specific external interface requirements and protocols, use of standard computer hardware and software development standards, manning constraints, etc.) which are captured in the Environmental View must be linked to the specific elements of the Implementation Model which address them. In general, each implementation constraint will impact an entire class of resources such as a requirement for Mil-Spec computer hardware or use of Ada in the development of new software. The Environmental View also establishes the top level system performance requirements, limiting scenarios for system operation, system performance MOEs, and the conditions of measurements for those MOEs. These key elements establish the basis for simulation and analysis of the Implementation Model and must be accounted for in the overall test plan for performance estimation of a candidate system implementation.

No direct formal linkage is established between the objects and relationship of the Informational View to the features of the Implementation

View. This is to ensure that the object constructs developed in capturing the abstract representation of the system in the Informational View do not unnecessarily constrain the system implementation design.

The following rules summarize a minimal formalism for linkage between the Conceptual and Implementation Models:

Each implementation design constraint in the Environmental View must be linked to all resources related to that design constraint in the Implementation View;

Each test scenario, MOE, top level system performance requirement, and condition of measurement captured in the Environmental View must be linked to one or more features of the Implementation Model analysis.

This mandatory association of the two models demands that the systems engineer directly address and trace the implementation requirements and constraints captured in the Conceptual Model during development of the Implementation Model. This type of traceability does not guarantee satisfaction of all design requirements and constraints; however, it does identify those which have not been addressed in the Implementation Model.

4.1.2.3 Logical Model to Implementation Model Linkage. The linkage between the Logical Model and the Implementation Model is characterized by a strong association of each function and data flow in the Functional View with one or more resources in the Implementation View. The resource which a function or data flow is mapped to must have the necessary characteristics and capacity to perform that function or to achieve the data flow connectivity as illustrated in the following example: the function `Display_Graphic_Image` which might include a real-time color graphic display of certain system parameters must be mapped onto a resource which is capable of providing color displays at the update rate and resolution specified in the function specification. Obviously, a track ball or keyboard is not an appropriate match. More subtle differences in display types such as the number of colors available can also serve to eliminate certain implementation options.

The Implementation View must therefore contain resources which can perform the various functions specified in the Functional View and provide physical paths for the specified data flows. However, the Functional View does not constrain how the functions and data flow are accomplished. A wide range of implementation options typically exist for any given functional system description. The mapping relationship between the Functional and Implementation View also allows multiple Implementation Views to be defined for a given Functional View. This provides the systems engineering team with the ability to consider simultaneous alternative implementation designs.

The following rules summarize a minimal formalism for linkage between the Logical and Implementation Models:

Each function in the Functional View must map to at least one resource in the Implementation View which is compatible with that function. (Compatible is defined as having the appropriate capabilities and

capacities to perform the function such as a computer with certain characteristics);

Each data flow in the Functional View must map to at least one resource in the Implementation View which provides the physical path (or paths) for that data flow in the system design;

Each resource in the Implementation View must have at least one function or data flow from the Functional View mapped onto it or have a clear rationale as an implementation specific element of the design.

This requirement for a strong mapping between the Logical Model and the Implementation Model supports development of a design which accommodates all required system functions and data paths. This mapping does not guarantee that a particular implementation in resources will accomplish the functional design requirements; however, it does identify those functions which have not been addressed in the Implementation Model and serves to identify and eliminate resources which do not have a firm basis in supporting one or more functional elements of the design.

4.2 SYSTEM DESIGN VIEWS ANNOTATION AND INTEGRATION

A key to the usefulness of any design view in the design process is to allow the view of system design to be formally attributed. Relationships between the various system design factors can then be expressed so that effective trade-offs can be performed between the various design factors.

Effective trade-offs between mechanization support can only be accomplished if the design factors become formalized into the methods themselves. Today, the system methods only allow informal definition of the design factors for a design element (e.g., *Teamwork's* implementation of structure analysis allows unstructured notes to be tied to functions/processes). This makes the extracting of the information between methods difficult and limits the ability to determine consistency of design factor specification. Also, these methods make it extremely difficult to share the design element attribute information between system methods and forces manual annotation to occur.

For large complex systems the ability to mechanize (automate) methods is critical. The integration methods employed between two design activities can be classified as supporting forward annotation, backward annotation, or full integration. In each successive mechanization listed above, the information between the activities is successively more unified.

The annotation identified in this section refers to the ability to add/modify information from one design activity to another, not the particular information needed to effectively annotate a design element. Design activities may be particular design methods (e.g., structured analysis), design tools (e.g., CASE tool), design views (e.g., behavioral) or design processes (e.g., test). The critical aspect to analysis is how information can effectively flow between the various parts that make up the design.

It is critical for the integration of design activities that the proper mechanization of the methods is supported. This is especially true for large, complex, dependable, real-time, time-critical systems. These large systems make it intractable to manually reenter information in multiple system design tools.

4.2.1 Manual Annotation

Manual annotation transfers information between design activities via the system designer by manually determining and entering the proper information. The specification of the system design is separate from any of the specifications for system modeling, system validation, system test, etc. Updates and consistency is purely a manual process.

4.2.2 Forward Annotation

Forward annotation is the process of transferring the pertinent information uni-directionally from one design activity to another. While forward annotation occurs to some extent during the design of all systems, one key to a successful design process is for efficient, robust forward annotation to be included in the process. Figure 4-2 pictorially shows the process. In an automated system design process, forward annotation often refers to the movement of information from a CASE tool to a system modeling tool.

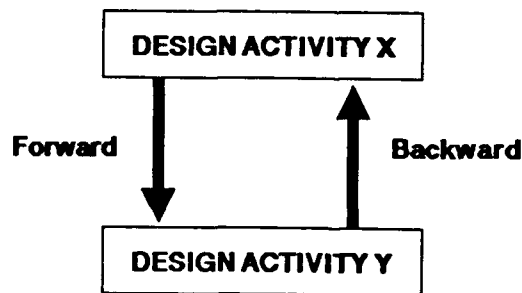


FIGURE 4-2. FORWARD AND BACKWARD ANNOTATION

Forward annotation is restrictive in the sense that the information flow is one way. If the information sent is changed, the total design specification becomes inconsistent. If the information is changed in the initial specification, then the forward annotation would need to be performed again. Also, by its nature, forward annotation involves multiple copies of the same information (possibly formatted or represented differently) to exist in various design activities. Because of this, forward annotation puts a significant strain on the configuration management activities of the system.

Full forward annotation into a design activity would occur if all pertinent information for the activity can be derived from another activity (automatically). For example, if a CASE tool captures a complete system

design, it could be forward annotated into a system modeling tool and provide all the necessary modeling specification information concerning the system.

4.2.3 Backward Annotation

As shown in Figure 4-2, backward annotation is the reverse process from forward annotation. Given information that has been forward annotated from Design Activity X to Design Activity Y and modified in Design Activity Y, then backward annotation is the process of updating the information in Design Activity X. Automated backward annotation does not currently exist to any large extent to today's development of systems. However, backward annotation is critically important in the development of complex systems, especially to maintain consistency between design activities.

4.2.4 Full Integration

Full integration has consistency of information maintained without a designer's intervention. In the integration process, only the common information needed by the design activities are shared.

The ultimate goal is to have one representation (capture) of the information concerning all factors and aspects of the system's specification. A forward annotation capability would then allow this information to be propagated to the appropriate design activities when deemed appropriate by the systems designer. If information currently defined by the systems engineer is not complete, the designer could add the information and the information would be forward and backward annotated as necessary. During the design process, information necessary for a type of systems modeling (not to be performed on the system) need not be specified.

4.2.5 State of Integration in Mechanization Support

Many of the activities in today's systems development are using manual annotation between design activities. For the most part, specifications of the system design (such as Teamwork, Software through Pictures (STP), Statemate) and system modeling approaches (e.g., ADAS, SES Workbench, QASE) are entered separately. Some systems do allow limited access. For instance, Statemate has a built-in modeling system. Teamwork's SA (structured analysis) module with certain restrictions can be forward annotated into the ADAS simulation tools. The same is true for STP specification to SES workbench. However, typically multiple modeling approaches for differing pieces of the system design need to be performed in order to fully evaluate the design.

While many of the barriers to more complete integration are technical, there are also corporate, competitive reasons why the integration of methods has not occurred rapidly.

4.3 AUTOMATED SYSTEM DESIGN CAPTURE AND ANALYSIS TOOL DEVELOPMENT OPPORTUNITIES

Successful employment of a multi-domain design capture and analysis methodology in supporting a complex system design is largely a function of the degree of mechanization which can be achieved. The size and complexity of large-scale advanced computer systems render manual application of any design process or method unusable. Considerable potential benefits can be gained from automation which supports a disciplined structured capture of the initial iteration of a system design and subsequent editing of that capture. Further significant efficiencies can be gained through automated consistency and completeness checking within and between the five system views which represent the captured design. However, in light of the overwhelming systems engineering task represented by analysis of an advanced complex system design, the most significant productivity gains are in automated support for design simulation and analysis within an integrated and highly automated design capture and analysis environment. The following paragraphs suggest opportunities for applying automated support to the methodology described in this paper.

4.3.1 Design Capture

An interactive graphic user interface with convenient pull-down menus and advanced editing functions such as cut, paste, replicate and append for complex design capture constructs is vital in capturing and modifying large, complex designs. The ability to manipulate entire sections of functional flow data diagrams and resource connection diagrams should be provided. Find and replace is also a powerful feature which can significantly reduce the editing time required to address the inevitable design iterations of a large, complex design.

Hierarchical design capture techniques offer significant advantages in managing the complexity of large computer system designs through information hiding. A hierarchical structure allows the system views to be represented at various levels of detail, from a broad top level which encompasses the breadth and scope of the system and its external interfaces to very low levels which describe the details of a particular segment of the system design. Automated support for capturing the design views in a hierarchical manner and in navigating and accessing the various levels of the design represents a relatively low risk tool development effort.

4.3.2 Intra- And Inter-View Consistency And Completeness Checking

Consistency and completeness checking within the five views is an area where automation can significantly reduce the effort required to maintain the design. Rules can be specified for checking the consistency and completeness within each design view. An example of the level of automation which is achievable in single domain consistency and completeness checking is the *Teamwork* CASE tool by Cadre.¹⁰ *Teamwork* provides automatic checking for functional flow data diagrams (FFDDs) which inform the user of unconnected

data flows, functions without inputs and/or outputs, functions without process specifications, inconsistencies between levels of decomposition and data dictionary related problems. This type of automated internal view checking should be provided to the greatest extent possible within each of the five design views.

Consistency and completeness checking between the five views represents a large step toward a truly integrated design capture and analysis methodology. Rules specifying the consistency criteria between the various views, such as the ones proposed in the preceding section, would be checked in a manner similar to the intra-view checking described above.

Beyond the predefined checking criteria which can be built into a CASE tool, such as the *Teamwork* FFDD capabilities, is an ad hoc ability to check both individual design views and across design views. The ad hoc capability would allow a knowledgeable operator to specify certain rules for checking the design capture. This capability would provide the user significant flexibility in establishing the relationships between the views and would allow the user to establish liberal or strict relationships between the views.

4.3.3 Design Simulation

Automatic generation of executable simulations from the design capture and external environment descriptions contained in the five views represents potentially the largest efficiency gains which an effective methodology automation may provide. The cost of capturing and analyzing multiple design iterations and options is potentially prohibitive without the leverage provided by automation of the simulation and analysis process. Completed capture of the five views contains much of the information necessary to develop a simulation of both the system under design (i.e., the internal model) and the external environment (i.e., the external model) which is required to stimulate the system's internal simulation. Efficient use of the information captured in the five design views for automatic or semi-automatic generation of a system simulation represents a key capability in realizing the overall goal of efficient large scale complex system design development.

4.3.4 Document Generation

Automated generation of required system documentation from the captured design data also represents an area where existing design capture information can be efficiently employed to reduce the manual labor required to support the design development process. A variety of CASE tools currently provide this capability particularly in the software development arena. The automated document generation facility should provide standard formats (such as the MIL-STD-2167A DIDs) and also allow the user to specify the format and organization of the documents produced by the system.

CHAPTER 5

CONCLUSION

Despite their differences, all five system views share at least one shortfall: the lack of capability to capture their non-functional attributes. Starting with the design factors list in Appendix A, the Real-Time System Design Capture and Analysis task will identify the critical non-functional attributes of the system and try to define their representing methods so that system views can be completely captured. In addition, the Environmental and Implementation Views have to be further expanded. The formal methods to classify and represent various types of system resources will be defined. Information that should be included in the Environmental View to complete its representation will be further refined.

In addition to the representation of the system views, the integration across different views will be studied. The ability to move from one design activity to another is so critical that systems engineers cannot afford to miss it. Without the integration between methods, the concept to provide a seamless process from requirement capture through both analysis and design will never exist.

REFERENCES

1. DeMarco, Tom, *Structured Analysis and System Specification*, Prentice-Hall, Inc., Yourdon Press, Englewood Cliffs, NJ, 1979.
2. Hatley, Derek J. and Pirbhai, Imtiaz A., *Strategies for Real-Time System Specification*, Dorset House Publishing, New York, NY, 1987.
3. Ward, Paul T. and Mellor, Stephen J., *Structured Development for Real-Time Systems*, Prentice-Hall, Inc., Yourdon Press, Englewood Cliffs, NJ, 1985.
4. Shlaer, Sally and Mellor, Stephen J., *Object-Oriented Systems Analysis: Modeling the World in Data*, Prentice-Hall, Inc., Yourdon Press, Englewood Cliffs, NJ, 1988.
5. *IEEE Standard VHDL Language Reference Manual*, IEEE Standard 1076-1987.
6. *Interactive Development Environments, Software Through Pictures Reference Manual, Ver. 4.2*, Interactive Development Environments, Inc., San Francisco, CA, 1988.
7. Molini, J.J, Maimon, S.K., and Watson, P.H., *Real-Time System Scenarios*, Real-Time Systems Symposium, Florida, Dec 1990.
8. Methodology and Tools Working Group, *ASW Architecture Development and Analysis Methodology, Ver. 1.0*, Technical Report, Honeywell Marine System Division, Everett, WA, Feb 1990.
9. Trident Systems Incorporation, *Software Requirements Specification for the System Engineer's Associate, A Computer-Aided Hierarchical Information Model Development and Analysis Tool*, Trident Systems Inc., Fairfax, VA, Aug 1991.
10. Technical Documentation Department of Cadre Technologies Inc., *Teamwork/SA User's Guide, Rel. 4.0*, Cadre Technologies Inc., Providence, RI, Dec 1990.

APPENDIX A

SYSTEM DESIGN FACTORS

1 PERFORMANCE

- 1.1 RESPONSE TIME
- 1.2 CAPABILITY
- 1.3 RELATIVE ACTIVITY
- 1.4 SPEED
- 1.5 THROUGHPUT
- 1.6 LATENCY
- 1.7 LOAD BALANCING
 - 1.7.1 INFORMATION OVER LOAD
 - 1.7.2 PROCESSING OVER LOAD
- 1.8 GRACEFUL DEGRABILITY
- 1.9 EFFICIENCY
- 1.10 PREDICTABILITY

2 REAL-TIME

- 2.1 HARDNESS
- 2.2 HARD DEADLINES
 - 2.2.0.1 PERIODIC
 - 2.2.0.2 APERIODIC
 - 2.2.0.3 SPORADIC
- 2.3 SOFT DEADLINES
 - 2.3.0.1 PERIODIC
 - 2.3.0.2 APERIODIC
 - 2.3.0.3 SPORADIC

3 COMPUTATION/PROCESSING REQUIREMENTS

- 3.1 IMPORTANCE
- 3.2 USEFULNESS
- 3.3 PRIORITY
- 3.4 (COMPUTING) PORTABILITY
- 3.5 INTERRUPT/RESET CAPABILITIES
- 3.6 MEMORY SPACE

4 DEPENDABILITY

- 4.1 RELIABILITY
- 4.2 ACCURACY
- 4.3 FAULT TOLERANCE
- 4.4 GRACEFUL DEGRABILITY
- 4.5 REDUNDANCY
 - 4.5.1 STATIC
 - 4.5.2 DYNAMIC
- 4.6 AVAILABILITY
 - 4.6.1 INHERENT AVAILABILITY
 - 4.6.2 ACHIEVED AVAILABILITY

- 4.6.3 OPERATIONAL AVAILABILITY
- 4.6.4 EASE OF REPLACEMENT
- 4.6.5 CRASH RECOVERABILITY
- 4.6.6 COMPUTATION HEAVY PROCESS EFFECTS
- 4.7 QUALITY

SECURITY

- 5.1 CLASSIFICATION
 - 5.1.1 TOP SECRET
 - 5.1.2 SECRET
 - 5.1.3 CONFIDENTIAL
 - 5.1.4 UNCLASSIFIED
- 5.2 TYPE OF DATA
 - 5.2.1 LEVEL I (CLASSIFIED)
 - 5.2.1.1 TOP SECRET OR ABOVE
 - 5.2.1.2 SECRET
 - 5.2.1.3 CONFIDENTIAL
 - 5.2.2 LEVEL II (SENSITIVE)
 - 5.2.2.1 PRIVACY ACT/FINANCIAL
 - 5.2.2.2 FOR OFFICAL USE ONLY
 - 5.2.2.3 SENTITIVE MANAGEMENT
 - 5.2.2.4 PROPRIETY/PRIVILEGED
 - 5.2.3 LEVEL III (NONSENSITIVE)
 - 5.2.3.1 (OTHER--NOT CATEGORIES IN LEVEL I AND II)
- 5.3 PERCENTAGE OF PROCESSING TIME
- 5.4 ENCRYPTION TYPE REQUIREMENTS
- 5.5 IMPLEMENTATION TECHNIQUES REQUIREMENTS

HUMANWARE

- 6.1 EASE OF USE
- 6.2 POTENTIAL OPERATOR DECISIONS
- 6.3 1.OPERATOR DELAY / 2.USER RESPONSE TIME
- 6.4 OPERATOR ACTION(S)
- 6.5 1.REQUIRED NUMBER OF OPERATORS / 2.NUMBER OF SIMULTANEOUS USERS
- 6.6 USER INTENSITY
- 6.7 AVERAGE TIME FOR EACH CATEGORIES
- 6.8 POTENTIAL ERRORS

PHYSICAL REQUIREMENTS

- 7.1 SIZE REQUIREMENTS
 - 7.1.1 HEIGHT
 - 7.1.2 WIDTH
 - 7.1.3 LENGTH
 - 7.1.4 DEPTH
 - 7.1.5 AREA
 - 7.1.6 VOLUME
- 7.2 WEIGHT REQUIREMENTS
- 7.3 RUGGABILITY (RUGGEDIZED)
- 7.4 SURVIVABILITY
- 7.5 (PHYSICAL) PORTABILITY
- 7.6 ENERGY REQUIREMENTS
 - 7.6.1 (ENERGY) CONSUMPTION

- 7.6.1.1 ELECTRICAL (ENERGY CONSUMED)
- 7.6.1.2 FUEL (ENERGY CONSUMED)
- 7.6.1.3 OTHER (ENERGY CONSUMED)
- 7.6.2 (ENERGY) DISSIPATED
- 7.7 LOCATIONAL OPERATING ENVIRONMENT
 - 7.7.1 GEOGRAPHICAL LOCATION
 - 7.7.2 INDOORS/OUTDOORS
 - 7.7.3 TEMPERATURE
 - 7.7.4 HUMIDITY
 - 7.7.5 ACOUSTICAL NOISE
 - 7.7.6 AIR PURITY/QUALITY
 - 7.7.7 EXPOSURE TO WIND
 - 7.7.8 EXPOSURE TO WATER
 - 7.7.9 EXPOSURE TO ELECTROMAGNETIC RADIATION
 - 7.7.10 VIBRATIONS/STABILITY
- 7.8 CLIMATE CONTROL
 - 7.8.1 COOLING
 - 7.8.2 HEATING
 - 7.8.3 HUMIDITY CONTROL
 - 7.8.4 ACOUSTICAL NOISE SUPPRESSION
 - 7.8.5 AIR PURITY/QUALITY CONTROL
 - 7.8.6 MOTION STABILIZATION
 - 7.8.7 LIGHTING
- 7.9 MANUFACTURING CONSIDERATIONS
 - 7.9.1 PRODUCTION CAPACITY
 - 7.9.2 PRODUCTION TIME

8 FINANCIAL REQUIREMENTS

- 8.1 COST TO DEVELOP
- 8.2 COST TO PROTOTYPE
- 8.3 COST TO PRODUCE
- 8.4 COST TO TEST
- 8.5 COST TO PURCHASE
- 8.6 COST TO OPERATE
- 8.7 COST TO MAINTAIN
- 8.8 COST TO REPAIR
- 8.9 COST TO INCLUDE SECURITY CAPABILITY
- 8.10 PRODUCTIVITY

9 TIME PROJECTED

- 9.1 ESTIMATED TIME TO DEVELOP
- 9.2 ESTIMATED TIME TO PROTOTYPE
- 9.3 ESTIMATED TIME TO PRODUCE
- 9.4 ESTIMATED TIME TO TEST
- 9.5 ESTIMATED TIME TO PURCHASE
- 9.6 ESTIMATED TIME TO OPERATE
- 9.7 ESTIMATED TIME TO MAINTAIN
- 9.8 ESTIMATED TIME TO REPAIR
- 9.9 ESTIMATED TIME TO INCLUDE SECURITY CAPABILITY

10 LIFE CYCLE

- 10.1 TESTABILITY

- 10.2 MAINTENANCE
 - 10.2.1 EASE OF MAINTENANCE
 - 10.2.2 NUMBER OF PERSON NEED TO MAINTAIN
 - 10.2.3 NOTIFICATION
 - 10.2.4 FREQUENCY
 - 10.2.5 MAINTENANCE DOWNTIME/DURATION
 - 10.2.6 DEGREE OF SYSTEM DISABILITY
 - 10.2.7 WHEN MAINTENANCE COMES DUE
 - 10.2.8 DURING MAINTENANCE
 - 10.2.9 WEAR LIFETIME
- 10.3 OBSOLESCENCE LIFETIME
- 10.4 REUSE-ABILITY

- 11 FUTURE NEEDS CONSIDERATIONS
 - 11.1 ADAPTABILITY/FLEXIBILITY
 - 11.2 EXPANDABILITY
 - 11.3 COMPATIBILITY
 - 11.4 INTERGRABILITY
 - 11.5 INTEROPERABILITY
 - 11.6 INTEGRITY

DISTRIBUTION

	<u>Copies</u>		<u>Copies</u>
Defense Technical Information Center Cameron Station Alexandria, VA 22314	12	Naval Postgraduate School Attn: Code EC/LE (Chin Hwa Lee) Monterey, CA 93943	1
Library of Congress Attn: Gift and Exchange Division Washington, DC 20540	4	NUWC Attn: John Rumbut Jr. Bldg. 1171-3 Newport, RI 02841-2047	1
Naval Air Development Center Attn: Code 7033 (Dr. C. Schmiedekamp)	1	Trident Systems Incorporated Attn: Nicholas Karangelen	1
(P. Zombori)	1	10201 Lee Highway Suite 300 Fairfax, VA 22030	
Warminster, PA 18974-5000			
Office of Naval Technology Attn: Code 227 (Elizabeth Wald)	1	Research Triangle Institute Attn: Geoffrey Frank	1
(Cmdr. Gracie Thompson)	1	P.O. Box 12194 Research Triangle Park, NC 27709-2194	
800 N. Quincy Street Arlington, VA 22217-5000			
Center for Naval Analyses 4401 Fort Avenue P.O. Box 16268 Alexandria, VA 22302-0268	2	Advanced Technology & Research Corp. Attn: Adrien J. Meskin Gorge Stathopoulos	5 1
Naval Research Laboratory Attn: Code 5534 (Connie Heitmeyer)	1	14900 Sweitzer Lane Laurel, MD 20707	
(Bruce Labaw)	1	Internal Distribution:	
Washington, DC 20375		D4	1
Naval Research Laboratory Attn: Code 5543 (Catherine Meadows)	1	E231	2
Washington, DC 20375		E232	3
Naval Postgraduate School Attn: Code AS/RA (Balasubramaniam Ramesh)	1	E342 (GIDEP)	1
Administrative Sciences Department Monterey, CA 93943		F01	1
		G07 (F. Moore)	1
		G42 (C. Yeh)	1
		K02	1
		K14 (D. Clark)	1
		K52 (W. Farr)	1
		N15 (M. Wilson)	1
		N30 (H. Crisp)	10
		N35 (M. Masters)	1
		N35 (F. Riedl)	1
		R44 (E. Cohen)	1

DISTRIBUTION (Con't)

	<u>Copies</u>
R44 (H. Szu)	1
U	1
U02	1
U042	1
U10	1
U20	1
U23 (W. Dence)	1
U23 (J. Horner)	1
U23 (P. Winters)	1
U25	1
U25 (D. Bergstein)	1
U25 (E. Hein)	1
U30	1
U31 (R. Scalzo)	1
U33	1
U302 (P. Hwang)	20
U33 (D. Choi)	1
U33 (M. Edwards)	1
U33 (N. Hoang)	10
U33 (S. Howell)	10
U33 (M. Jenkins)	1
U33 (T. Moore)	1
U33 (C. Nguyen)	1
U33 (T. Park)	1
U33 (H. Roth)	1
U33 (M. Trinh)	1
U40	1

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE October 1991	3. REPORT TYPE AND DATES COVERED	
4. TITLE AND SUBTITLE Mission Critical System Development: Design Views and Their Integration			5. FUNDING NUMBERS PE - 0602234N PR - RS34P11 TA - Task 2	
6. AUTHOR(S) Ngocdung Hoang and Steve Howell (NAVSWC) Nicholas Karangelen (Trident Systems Inc.)				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Surface Warfare Center (Code U33) 10901 New Hampshire Avenue Silver Spring, MD 20903-5000			8. PERFORMING ORGANIZATION REPORT NUMBER NAVSWC TR 91-586	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The purpose of this document is to describe a formalism that allows for the capture and analysis of very large, computer-based, real-time, mission critical computer resource (MCCR) systems. The formalism covers all aspects of the system including functional (the functions a system performs) and nonfunctional (characteristics of system performance) attributes. Nonfunctional attributes of the systems, such as timing, dependability, security, and reliability should be captured and analyzed at the early stage of the system development process to guarantee the correctness of a system's performance. The current proposed formalism captures the system design in five different views such that analysis can be correctly performed. The views are Informational, Functional, Behavioral, Implementation, and Environmental. Each view explores different aspects of the system and all five in total provide a more complete understanding of the system. Even though the ultimate goal is to capture all aspects of the system in the five views, at this time there is no perfect vehicle for analyzing system attributes such as hard real-time, security, reliability, and dependability. However, as the formalism matures, all aspects that affect the system will be covered in detail.				
14. SUBJECT TERMS Large Scale, Mission Critical, Real-Time Systems; Analysis; Design Capture; Representation; Methodology; Integration.			15. NUMBER OF PAGES 75	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR	