

NSWC TR 83-360

12

AD-A146 460

**A FRAMEWORK FOR THE DEVELOPMENT OF
NAVY COMPUTER-BASED SYSTEMS:
COMBAT SYSTEM ARCHITECTURE AND COMBAT
SYSTEM ENGINEERING PUT INTO PERSPECTIVE**

BY JEFFREY FRANKLIN

UNDERWATER SYSTEMS DEPARTMENT

30 DECEMBER 1983

Approved for public release, distribution unlimited

DTIC
ELECTE
OCT 9 1984
S B

DTIC FILE COPY



NAVAL SURFACE WEAPONS CENTER

Dahlgren, Virginia 22448 • Silver Spring, Maryland 20910

84 10 05 074

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC TR 83-360	2. GOVT ACCESSION NO. AD-A146460	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A FRAMEWORK FOR THE DEVELOPMENT OF NAVY COMPUTER-BASED SYSTEMS: COMBAT SYSTEM ARCHITECTURE AND COMBAT SYSTEM ENGINEERING PUT INTO PERSPECTIVE		5. TYPE OF REPORT & PERIOD COVERED
7. AUTHOR(s) Jeffrey Franklin		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center (Code U32) White Oak Silver Spring, MD 20910		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work request: N0002482 NR-10462 task #24444
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE 30 December 1983
		13. NUMBER OF PAGES 26
		15. SECURITY CLASS. (of this report) Unclassified
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Combat System Engineering, Combat System Architecture, System Engineering Computer System Development		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Navy is funding two areas called combat system architecture (CSA) and combat system engineering (CSE). The question most often asked about these two areas is where does CSA stop and the design begin, and what are the speci- fic functions of CSE. This report attempts to answer these questions by in- tegrating CSA and CSE into a comprehensive system development process. In addition the report recommends which agency should be assigned specific responsibilities within the development process.		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 66 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

The Navy is funding two areas called combat system architecture (CSA) and combat system engineering (CSE). The question most often asked about these two areas is where does CSA stop and the design begin, and what are the specific functions of CSE. This report attempts to answer these questions by integrating CSA and CSE into a comprehensive system development process. In addition the report recommends which agency should be assigned specific responsibilities within the development process. The author wishes to thank Dennis Mensch (Code N13, NSWC) for his thorough review and suggestions concerning this document. The work has been supported by Work Request N0002482WR-10462, task assignment #24444.

Approved by:

J. E. Goeller
J. E. GOELLER, Head
System Engineering Division

DTIC
ELECTE
S **OCT 9 1984** **D**
B



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist	Special
A-1	

CONTENTS

<u>Chapter</u>		<u>Page</u>
1	INTRODUCTION	1
2	WHAT IS SYSTEM ENGINEERING?	2
3	A FRAMEWORK FOR THE DEVELOPMENT PROCESS	3
4	ASSIGNMENT OF RESPONSIBILITIES FOR THE ITEMS IN THE SYSTEMS ENGINEERING PLAN	15

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
1	METHODOLOGY STEPS AND TOOLS OF NSWC ASSESSMENT METHODOLOGY . . .	9
2	SUMMARY OF FRAMEWORK	10
3	RELATIONSHIP OF SYSTEM ANALYSIS MODELS TO CSE MODELS	11
4	COMBINING SYSTEM ARCHITECTURE AND SYSTEM ANALYSIS	12

TABLES

<u>Table</u>		<u>Page</u>
1	QUANTITATIVE SCENARIO EXAMPLE	4

CHAPTER 1

INTRODUCTION

There are several system engineering/computer system development standards (see References 1 through 3) and/or guidelines. These standards can be complied with, but the resulting system still may suffer from the same deficiencies that the standards were supposed to prevent. The reason for this is: (1) the standards do not explain how to actually generate the items they require, (2) the standards do not explain the detailed properties of the standard's required items, and (3) the standards do not specify what tests will be applied to show the required items are satisfactory. Thus the developer may meet the standard in the letter of the law but not the spirit of the law.*

It is the purpose of this report to present a comprehensive system development framework. Within this framework the existing Navy system engineering (etc.) standards can be implemented in a fashion to more fully comply with their original reason for being written.

Experience has shown that some aspects of the standards development process need increased emphasis while other parts need decreased emphasis.** In addition, this experience has led to the development or use of a set of automated tools by which the standard's required items can be defined and meaningfully developed. (Where standard development documents (e.g., Top Level Requirement Document) are referenced, no detailed explanations will be given because these can be found in the standards themselves.)

*A specific example of meeting the letter of the law but not the spirit of the law is the Navy TADSTAND5 (Navmat Letter MAT-09Y:CFH, Serial 134) which requires a computer to be no more than 80% loaded. Since the TADSTAND does not specify how this 80% loading is to be measured the system developer is free to do anything from a back of the envelope calculation to a wraparound simulation driven detailed measurement.

**See Chapter 3 for specifics.

CHAPTER 2

WHAT IS SYSTEM ENGINEERING?

The very first system engineering task on any project is to define what activities fall under the auspices of system engineering and how the system engineering activities relate to the other activities of the project. Experience has shown that systems engineering activities done by the Navy R&D Community need to be improved so as to provide quantitative reliable studies upon which system design decisions can be made.

The viewpoint of this report is that systems engineering is the technical arm of the funding agency/project office. Thus, its job is to prepare a plan that will define all the technical activities needed to develop the system. This plan should be as detailed as possible. Most important the plan should show how the inputs and outputs of each activity are to be used by succeeding activities, who is to be responsible for performing the activity and what methodology will be used to perform the activity. The name of this plan - - - the Systems Engineering Plan - - - of course.

It should be noted that even though systems engineering defines all the activities it does not necessarily mean that system engineering is responsible for or performs all the activities. What systems engineering is responsible for is to ensure that the system, when produced, meets its required specifications.* It may happen that no system level test can be formulated to test certain requirement specifications. For these situations, the system engineer may impose on the activities not performed by the systems engineering staff certain subsystem tests or certain design/development methodology standards.

In summary, systems engineering will be defined by a systems engineering plan. The plan will define the system development process from the system's earliest phases of TOP LEVEL REQUIREMENTS (TLR) setting to its final phase of sign-off/acceptance. Certain tasks in the plan may be assigned the responsibility of the systems engineer while other tasks may be assigned to other agencies (e.g., the development agency). In spite of which tasks are assigned to the systems engineer, the systems engineer will monitor all tasks to ensure deliverables meet requirements, specifications, and system engineering guidelines/standards defined in the plan.

Chapter 3 will present a generic systems engineering plan that satisfies the above definitions.

*This implies that all requirements/specifications should be testable. It also implies that the test plan be written almost simultaneously with requirements/specifications; i.e., to ensure requirements/specifications are truly testable.

CHAPTER 3

A FRAMEWORK FOR THE DEVELOPMENT PROCESS (i.e., A GENERAL SYSTEMS ENGINEERING PLAN)

A. REQUIREMENTS GENERATION

Inputs

- Tactical Scenario

Outputs

- Top Level Requirements Document (Sensor and weapon characteristics, etc.),
- systems analyses justification document

The TLR quantitatively and qualitatively defines the tactical scenario. The tactical scenario comprises the operating environment, enemy capabilities, own ship capabilities, tactics, etc. (See Table 1.)

The problem with some TLRs is that they are not developed from a quantitative base where the interrelationship between the requirements have been developed or worked into a complete system. For these cases the TLR represents a qualitative wish list made up by either the command organization or the operating forces. There is a distinct difference between a wish list and a set of requirements.

In order to transform a wish list into a good TLR, standard systems analysis techniques (e.g., stochastic analysis, optimization, error analysis) need to be employed. A two-sided analysis needs to be done; i.e., the enemy system is modeled to the same level of detail as own forces and "exchange ratios"* used as one of the main measures of effectiveness (MOE). The sensors and weapons can be described with generic capabilities or in terms of actual equipment.

It should be noted that at this time in the development process only items that are related to enemy interaction; i.e., sensors, weapons, operating environment, and tactics need be modeled. Computer, electronic, and human intensive items like command and decision or control systems need not be included. Systems analyses to select or analyze algorithms associated with sensors and weapons can be done at this stage or postponed till the next stage of development. (See Section B below.) However, in general, the earlier in the development that development activities can be done, the more advantageous.

*Exchange ratio--ratio of friendly to enemy killed.

TABLE 1. QUANTITATIVE SCENARIO EXAMPLE

- Number of each type of attacking platform that is to be defeated.
- Dynamic Geometry of attacking platforms.
- Number and type of weapons of attacker.
- Arrival rates of the enemy weapons.
- Launch distances of the enemy weapons.
- Time between launch of enemy weapons and arrival.
- Number of targets in training scenario.

etc.

(Qualitative Scenario Definition.

Defines requirements that have no quantitative component (e.g., battle planning) or amplifies the quantitative requirements.)

Thus in conclusion, the TLR (or the Requirements Document) should not be issued unless it includes or references an in-depth systems analysis which shows how the requirements were arrived at and where all the data used in the systems analysis were obtained. This document is called the Systems Analysis Justification Document.

B. SYSTEMS* ARCHITECTURE GENERATION

Input

- Results of systems analysis, (i.e., sensors and weapons (characteristics) of the combat system).
- TLR/Requirements Document

Output

- The architecture, (interconnected functions allocated to interconnected servers that perform the functions of the combat system), (i.e., Type A Spec).

This phase is actually the beginning of the design process. Thus any attempts at defining generic architectures will not be useful or productive; i.e., designs are not done generically but specifically to meet a set of requirements.

1. Function Generation.

The TLR has defined the sensors, weapons, tactics, operating environment, etc. What it has not defined is how sensors and weapons will be integrated into a functioning integrated system. In order to accomplish this integration a set of functions must be developed, which when executed will give outputs/results that meet the TLR.

The functions that are developed are not just for the items considered in the TLR but for all the functions that the combat system (CS) needs to do to meet the TLR. Thus it is at this phase of development that all the remaining items of the CS will begin to emerge, i.e., the control systems, etc. However, their details need not be firmly decided on until later in the development.

Current systems engineering methodology advocates developing the functions starting from the general and progressing to more and more detailed functions in a series of steps called tiers; i.e., a top down design. This is in contrast to starting with detailed functions and combining them to form higher/more general functions; i.e., bottom up design. A third possibility is to think about both the bottom and the top functions simultaneously; i.e. "end in analysis." "End in analysis" is found to be superior because it ensures that the top functions can be realistically built by integrating knowledge of bottom level functions gained from already operational or prototype systems, and because it allows the system to be quantitatively assessed early in the development process. The use of this "end in" technique for system assessment is described in Reference 4.

*Throughout the report the general term "system" has been used vice the specific term "combat system". The reason being that what is being reported is true for systems in general, as well as to specifically combat systems.

It should be noted that with both "end in" or "top down" techniques there is no reason to keep the level of detail of all the functions to the same tier level. Thus at any given point in the development some functions will be described possibly to tier 5 or 6 while others may be only at tier 1 or 2. In point of fact the earlier in the development greater detail can be given, the more advantageous to the systems engineering process.

1.1. Functional Analysis.

Input - Functions developed in 1.

Output - Function Analysis Justification Document
- Function I/O.

Functional Analysis, an analysis that shows that there exist a set of functions that satisfies each TLR; that the functions flow together (both intra and inter system) smoothly. In short, this analysis shows exactly how the system will functionally operate. The results of this functional analysis are documented and justified in a justification document.

Erroneously, "functional analysis" is what is usually thought of as systems engineering.

1.1.1. Function Input/Output (I/O) Analysis.

Input - Functions developed in 1. Functions analysis document developed in 1.1

Output - Interface Requirements Document (IRD)
- Function I/O

The IRD is a high level Interface Design Document (IDS). It can be thought of as a precursor or requirements document for the IDS. The IRD is developed by asking the question of each function: what data will it need and where will the needed data come from? The answer to this question will yield a set of function I/O. The I/O analysis will thus show what kind of top level information needs to be passed between functions, and thus it also will indicate the needed data bases for the system. (The detailed design of the data bases can be worked out later.) This I/O analysis is then summarized into an IRD.

There are a number of tools that can be employed to aid in functional and function I/O analysis, (e.g., PSL/PSA (Reference 4), SADT (Reference 5), HIPO, RSL/RSA (Reference 6), etc.).

From the activities in the above sections (1, 1.1, and 1.1.1) a possible system design will begin to emerge. The reason is that the need for common information/functions will become apparant. Thus rather than calculating common information in separate redundant functions, one common/combined function could be created or assigned the responsibility of one particular subsystem.

2. Top Down Functional Allocation.

The functions are allocated first to major servers (e.g., hull sensor, control system, etc.) and then to servers within major servers (e.g.,

computers/personnel). It is at this stage that functions are assigned to hardware, computer/software, or personnel.

It should be noted that this allocation is only a trial allocation/architecture. The required allocation cannot be decided until several iterations of functional allocation/specification and performance assessment are done. This iteration is discussed in the next section (B3).*

2.1. Alternate Architecture Formulation.

There is no one** architecture/function allocation that will satisfy the requirements, or alternately, it may not be obvious which architecture meets or exceeds the requirements. Thus it is best to formulate several alternative architectures. Examples of different architectures are: using different computer bases (e.g., UYK 7 versus UYK 20), assigning a console for each operator versus sharing common consoles, etc.; i.e., different allocation of functions to different servers.

These alternate architectures are then assessed for performance, life cycle cost, and schedule impact.

Experience has shown that the allocation process tends to become an emotional and drawn out activity. The reason for this is that subsystem (development) agencies perceive the possibility of having functions that were historically developed by them allocated to another agency's subsystem. The system engineer must not waste time by allowing these debates on alternate architectures. The systems engineer should allow the alternate architectures to be submitted and then quickly move to the assessment phase (described in the next section); i.e., alternate architectures may be submitted but not subject to prolong qualitative debate. If the assessment shows that all architectures are equal then selection of an architecture can be made based on programmatic or life cycle considerations.

3. Iterated Architecture Performance Assessment/Modeling.

Input - Functional Allocation, TLR

Output - Queuing MOE's, updated Systems Analysis, Architecture model

*"Display sharing" is an example of the type of question that can be decided during the allocation and performance assessment stage.

**From a strict systems engineering viewpoint it makes no difference if the architecture is technically innovative/state of the art or technically antiquated as long as the architecture will satisfy the requirements. However, other non-system engineering considerations like cost or maintenance may show the technically innovative architecture to be more favorable.

Using the NSWC Assessment Methodology (Reference 7 and summarized in Figure 1 and Figure 2) or some equivalent methodology (i.e., a methodology where servers and functions are quantitatively specified and interrelated), assess the performance of the architecture.

If the results of the first performance assessment indicate that the architecture is either overloaded or lightly loaded then appropriate function reallocation to servers will need to be done on successive iterations of the architecture/design.

The queuing MOE of response time can be incorporated into the SA studies to determine the effect on P_{Hit} (i.e., the ballistic solutions will be in error by an amount of time equal to the response time of the combat system architecture). If P_{Hit} is adversely affected, then some remedial designing will be needed.*

It should be noted the servers modeled include both computer and human servers. To date no validated data on human functions are known. Thus only estimates of operator times for button pushing and display comprehension can be used in the queuing models of the assessment methodology. In principle the results of any human engineering can be evaluated at this development stage.

Experience has shown that not enough time/emphasis is spent on performance assessment. The reason is that up to now, no adequate technique has been available. Another reason is that the assessment was usually assigned to the design agency, and they do not have enough funds/personnel to do the design development and the performance assessment** (i.e., when the design agency has to choose between expending resources on assessment or developing the design, design development is chosen). However, if an adequate performance assessment is not done then the systems engineering/ system design is seriously compromised.

C. DETAILED DESIGN AND SYSTEM IMPLEMENTATION

Inputs - Type A Spec,
- IRD,
- architecture model

Outputs - B Specs e.g., PPS, PDS, PDD, DBDS, IDS, etc.,
- updated architecture model
- the system

*This effect of response time on P_{Hit} is the bridge that joins systems analysis studies with architecture studies. See Figure 3 and Figure 4.

**It is desirable that the design agency not evaluate its own design in order to maintain the highest standards of objectivity. On the other hand, it is sometimes difficult for an independent design evaluator to get the required data or to obtain an in depth understanding of the design necessary to do a good assessment.

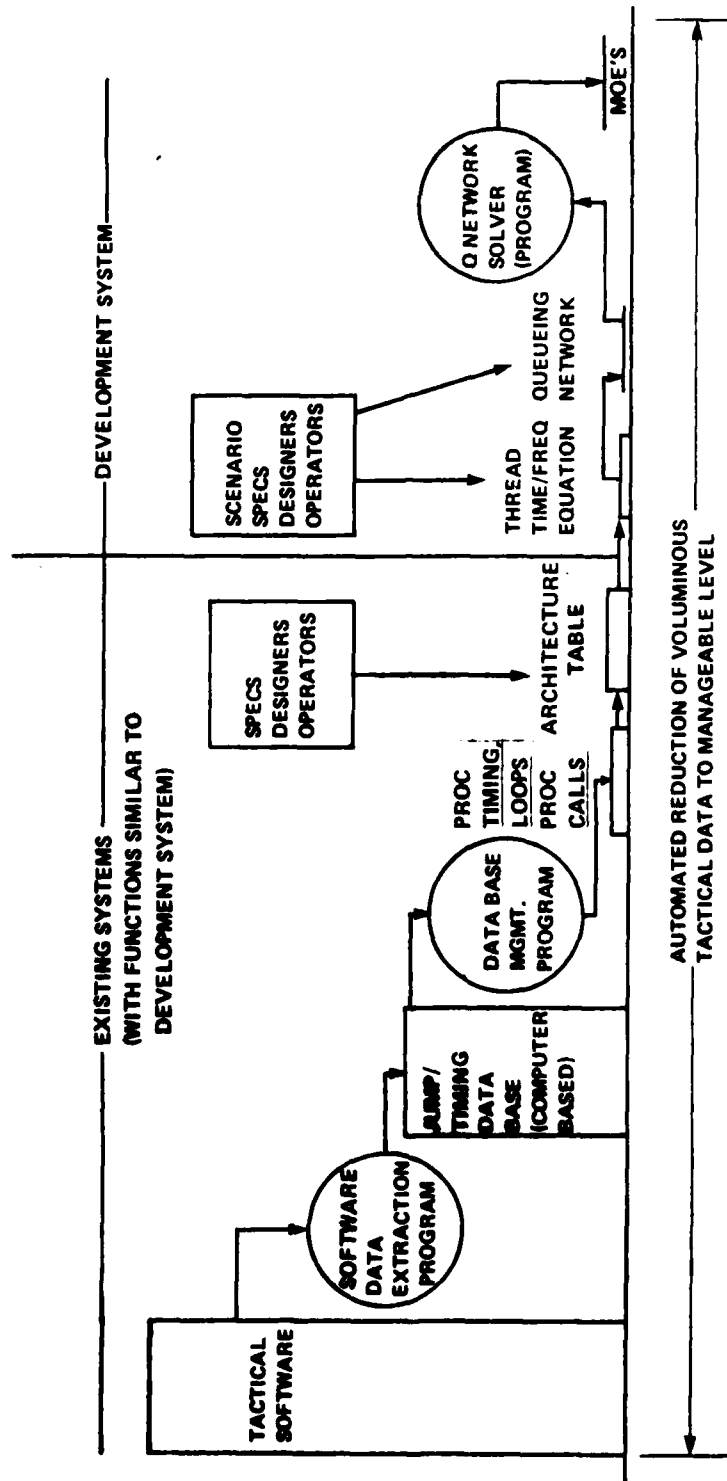


FIGURE 1. METHODOLOGY STEPS AND TOOLS OF NSWC ASSESSMENT METHODOLOGY

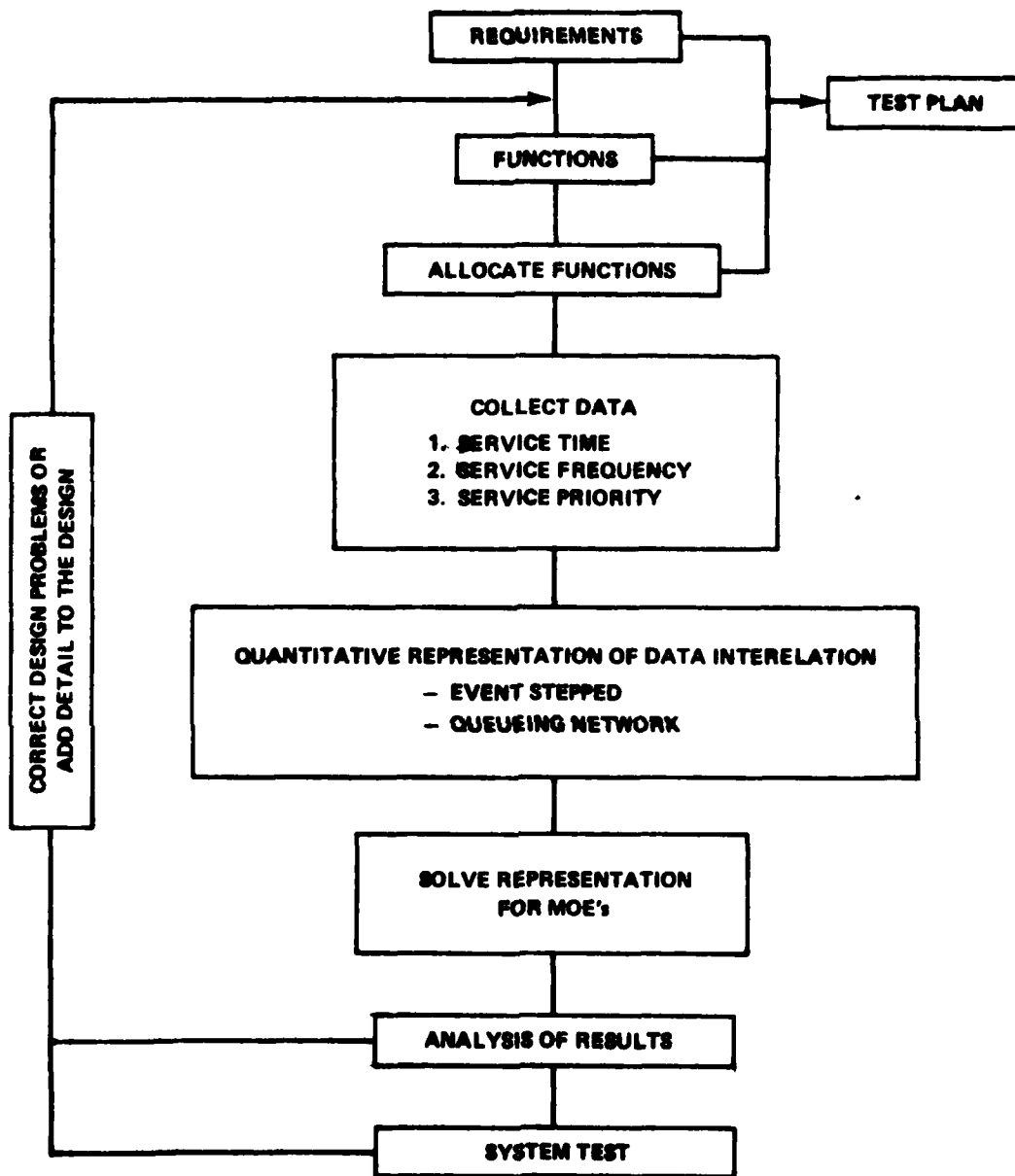


FIGURE 2 SUMMARY OF FRAMEWORK

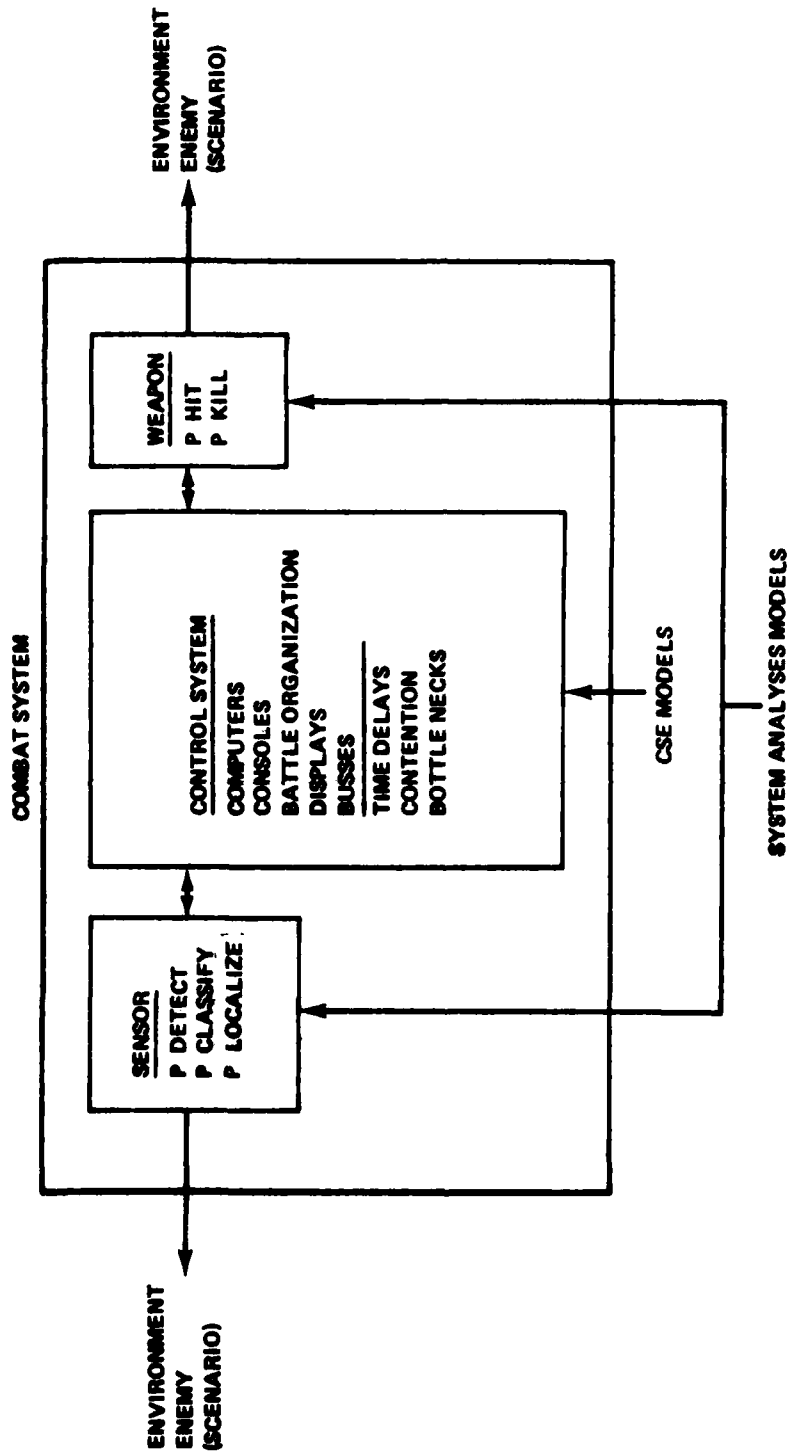


FIGURE 3. RELATIONSHIP OF SYSTEM ANALYSIS MODELS TO CSE MODELS

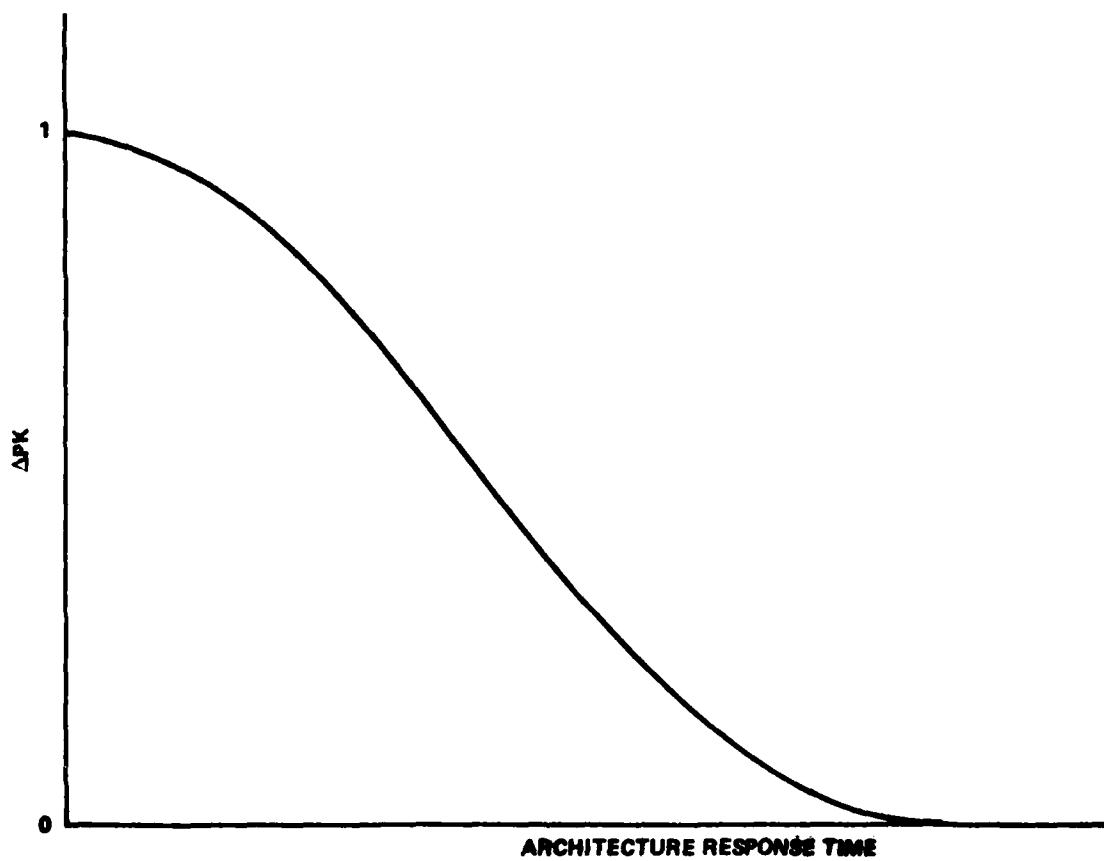


FIGURE 4. COMBINING SYSTEM ARCHITECTURE AND SYSTEM ANALYSIS

The point at which detailed design (i.e., the design architecture) begins and the (functional) architecture (defined in Section B) ends is not usually distinct. In practice the two activities are smooth and continuous. One reason given for making a distinction between functional architecture and design architecture is that as development proceeds, some functions of the functional architecture may be combined, and thus there then will not be a one-to-one mapping of the functional architecture onto the design architecture. However, the need to combine or split functions as indicated by past experience and the performance assessment is better viewed as a continuous/more detailed iteration of the function and allocation process, as described in section B, rather than the beginning of a new phase of development.

With this point of view the distinction between a (functional) architecture and a design disappears. They are both just functional allocations; the design being a more detailed functional allocation than the architecture.* The culmination of the design activity is the B Specs; e.g., PPS, PDS, etc.

1. Updated Performance Assessment.

As the development progresses the design becomes more detailed; i.e., more detailed functions are added or functions are combined, etc. The assessment model thus has to be kept current with the development. However, the assessment does more than pace the development, it helps drive the development. It does this by indicating the overload points of the design. Knowing the weak points of the design usually indicates in which direction the design needs to be changed.**

2. The System (i.e., software, hardware, etc.).

For each of the above outputs, format, content, and methodology need to be defined. These will not be discussed in this report since there exist numerous techniques for preparing B specifications, developing Software Development Plans, and developing software.

D. TEST AND EVALUATION

1. Iterated Software and Operator Performance Assessment.

This phase is actually the beginning of testing. However, it is done before the system is completed rather than after the system has been built as with traditional testing.

*It may be true that a new set of personnel (i.e., programmers vs. system analysts/engineers) do this design detailing, but this in no way changes the conceptual process. In point of fact the function/allocation process even extends into the coding stage of development; i.e., a computer instruction can be thought of as just a very detailed low level subfunction allocated to a very low level server like a CPU.

**This is an example of how an iterated analysis technique is an approximate synthesis technique. This is discussed in more detail in Reference 7.

In this phase, portions of the assessment model's service times are iteratively replaced with the actual service times of the software as the software becomes available. The service time of the software can either be measured or calculated (using Reference 7's automated timing tools). Both CPU and I/O service times are developed. Operator service times can be measured as the console software or a suitable simulator for the consoles/Man Machine Interface (MMI) becomes available.

2. Test Plan.

Input - TLR, PPS, Architecture Model

Output - Test Plan that describes tests that will show that all requirements in TLR and PPS are met. It defines the acceptance criteria that must be shown in order to satisfy each requirement.

2.1 Testable Requirements - As was noted in Chapter 2, when a requirements document is issued it is reviewed to ensure that the requirements it contains are testable. Thus the Test Plan can be issued when these requirements are known usually around the PPS stage of development. The Test Plan is then updated to keep current with the requirements documents.

3. System Testing.

This is the implementation of the test plan which is covered in detail in numerous other reports. Thus it will not be covered in this report.

4. Reliability and Maintainability Analyses (RMA).

RMA is included as part of system testing because it is an analytic test that ensures the system will meet RMA requirements.

A summary of the above framework is given in Figure 2.

CHAPTER 4

ASSIGNMENT OF RESPONSIBILITIES FOR THE ITEMS IN THE SYSTEMS ENGINEERING PLAN

The term "assignment of responsibility" is not meant in the sense of what agencies issue or sign off on the information, but in the sense of the required skills of the person(s) creating the information. Where the person resides, (e.g. headquarters, government lab, industry, etc), or to which functional group (e.g. system engineering, design, analysis, test) he belongs is of little importance. What is important is that personnel with the required skills be available to work as a team.

(If the system engineer is viewed as the technical arm of the developing/funding agency the specification of required personnel skills and how they function as a team could be made part of the systems engineering plan).

A. REQUIREMENTS GENERATION

Inputs are needed from the operating forces to delineate deficiencies in the present systems, define the operating environment, and develop a wish list for the future system.

Inputs are needed from the intelligence community to define the enemy's capabilities in the time frame of interest.

Inputs are needed from the present systems maintenance and test agency to delineate deficiencies in the present system.

The above inputs are then quantitatively combined into a model by systems analysts who actually produce the written requirements as discussed in Chapter 3. If possible, the algorithm analysis will also be done by the systems analyst.

B. SYSTEMS ARCHITECTURE GENERATION

It should be noted in reading this section that the point of view that the system engineer takes the lead in creating the systems architecture (even the very top level architecture) is not taken. The viewpoint taken is that design personnel and the system engineer produce the systems architecture in consort as defined in the succeeding sections.

1. Function Generation

Since as discussed in Chapter 3, Section C, this is the beginning of design these functions are best generated by the design agent.*

1.1 Functional Analysis.

The Functional Analysis can be done by the design agent following the standards/requirements set down in the Systems Engineering Plan.

1.1.1 Function I/O Analysis.

The I/O Analysis is performed by the design agent for each subsystem. The results are then used by the systems engineer in order to architect the total system.

2. Top Down Allocation.

Top Down Allocation is performed by the design agent for each subsystem. The results are then used by the systems engineer to architect the total system.

2.1 Alternate Architecture Formulation

Alternate architecture formulation is performed by the design agent for each subsystem. The results are then used by the systems engineer in architecting the total system.

3. Architecture Performance Analysis

A computer performance assessment analyst models the system as described in Chapter 3. The assessment is given to the subsystem designer, the system analyst and the system engineer who jointly improve (or select) the architecture on each iteration. (See footnote **p8 in Chapter 3 for what agency the assessment analyst is assigned to.)

C. DETAILED DESIGN AND SYSTEM IMPLEMENTATION

Detailed design and system implementation is best done by the designer for each subsystem but checked by the systems engineer from a total systems design point of view.

1. Updated Performance Assessment.

A computer performance assessment analyst models the system as described in Chapter 3. The assessment is given to the subsystem designer, the system analyst and the system engineer who jointly improve the architecture on each iteration.

*Design agent is a person(s) who has knowledge of the detailed design of present systems and will be responsible for the detailed design of the development system/subsystem.

D. TEST AND EVALUATION

1. Iterated Software and Operator Performance Assessment.

Iterated Software and Operator Performance Assessment is performed by a computer performance analyst giving the results to the designer and system engineer for the purpose of improving the current design.

2. Test Plan.

The Test Plan is written by the system engineer and test personnel. The reason the systems engineer is involved in the test plan is because he is responsible for ensuring that the system will meet its requirements.

3. System Testing.

The system testing is performed by test personnel and results approved by the system engineer and the head test assessment engineer.

4. RMA.

RMA is performed by RMA engineers.

REFERENCES

1. SECNAVINST 3560.1 of 8 Aug 1974, Tactical Digital Systems Documentation Standards.
2. Military STANDARD Engineering Management, MIL-STD-499A (USAF).
3. DOD INSTRUCTION 5000.29 of 26 Apr 1976, Management of Computer Resources in Major Defense Systems.
4. ISODOS INC ANN ARBOR Michigan 48106, Problem Statement Language (PSL), Language Reference Summary, Jul 1983.
5. Ross, D., Structured Analysis For Requirements Definition, IEEE Transaction on Software Engineering, Vol SE-3, No 1, Jan 77.
6. Software Requirements Engineering Methodology Notebook, Final Report for DAHC60-71-C-00449, Ballistics Missile Defense advanced Technology Center, 31 Oct 1974.
7. Franklin, J., Gray, C., Wrenn, A., Architecture, Design, and System Performance, Assessment and Development Methodology, for Computer Based Systems, NSWC TR 83-324, (in publication).

DISTRIBUTION

	<u>Copies</u>		<u>Copies</u>
Commander		Defense Technical Information	
Naval Underwater Systems Center		Center	
Attn: Code 33B (R. Prager)	1	Cameron Station	
(B. Thorpe)	1	Alexandria, VA 22314	12
New London, CT 06320			
Commander		General Electric Corporation	
Naval Ocean Systems Center		Attn: Andy Rasi, Bldg. 1, Rm. R5	1
Attn: Code 82 (Dr. R. Kolb)	1	Farrell Road Plant	
Code 6201 (J. Reardon)	1	Syracuse, NY 13221	
Code 62 (R. Thulen)	1		
Code 6211 (M. Stonebreaker)	1	<u>Internal distribution:</u>	
Code 6201 (D. Callabro)	1	E431	9
San Diego, CA 92152		E432	3
		K54	1
		N	1
RADC/CO		N04 (Dr. H. Crisp)	1
Attn: COL J. Marcinak	1	N05 (C. Yarbrough)	1
Griffis AFB		N10	1
Rome, NY 13441		N13 (D. Mensch)	1
		N14 (W. Martin)	1
Commander		N20	1
Naval Sea Systems Command		N21 (M. Masters)	1
Attn: SEA-61V (R. Wilson)	1	N30	1
SEA-61V (D. Perrill)	1	N51 (E. Price)	1
PMS-400 (R. Hill)	1	N53 (D. C. Hill)	1
PMS-400 (CAPT Donegan)	1	N302 (R. Cullen)	1
SEA-61R2 (P. Andrews)	1	N305	1
PMS-400	1	N307 (J. Straub)	1
PMS-409	1	U	1
PMS-411	1	U04	1
PMS-411B	1	U05	1
PMS-411C	1	U12 (B. Podolsky)	1
PMS-411G	3	U20	1
PMS-411E	1	U23 (J. Cottrell)	1
Washington, DC 20362		U30	1
		U31	1
Commander		U31 (T. Ng)	1
Naval Electronic Systems Command		U31 (J. Simpson)	1
Attn: Elex 814 A (LCDR M. Gehl)	1	U32	1
Washington, DC 20363		U32 (J. Franklin)	1
		U32 (C. Gray)	1
Library of Congress		U32 (H. Herring)	1
Attn: Gift and Exchange Division	4	U32 (R. Timberlake)	1
Washington, DC 20540			