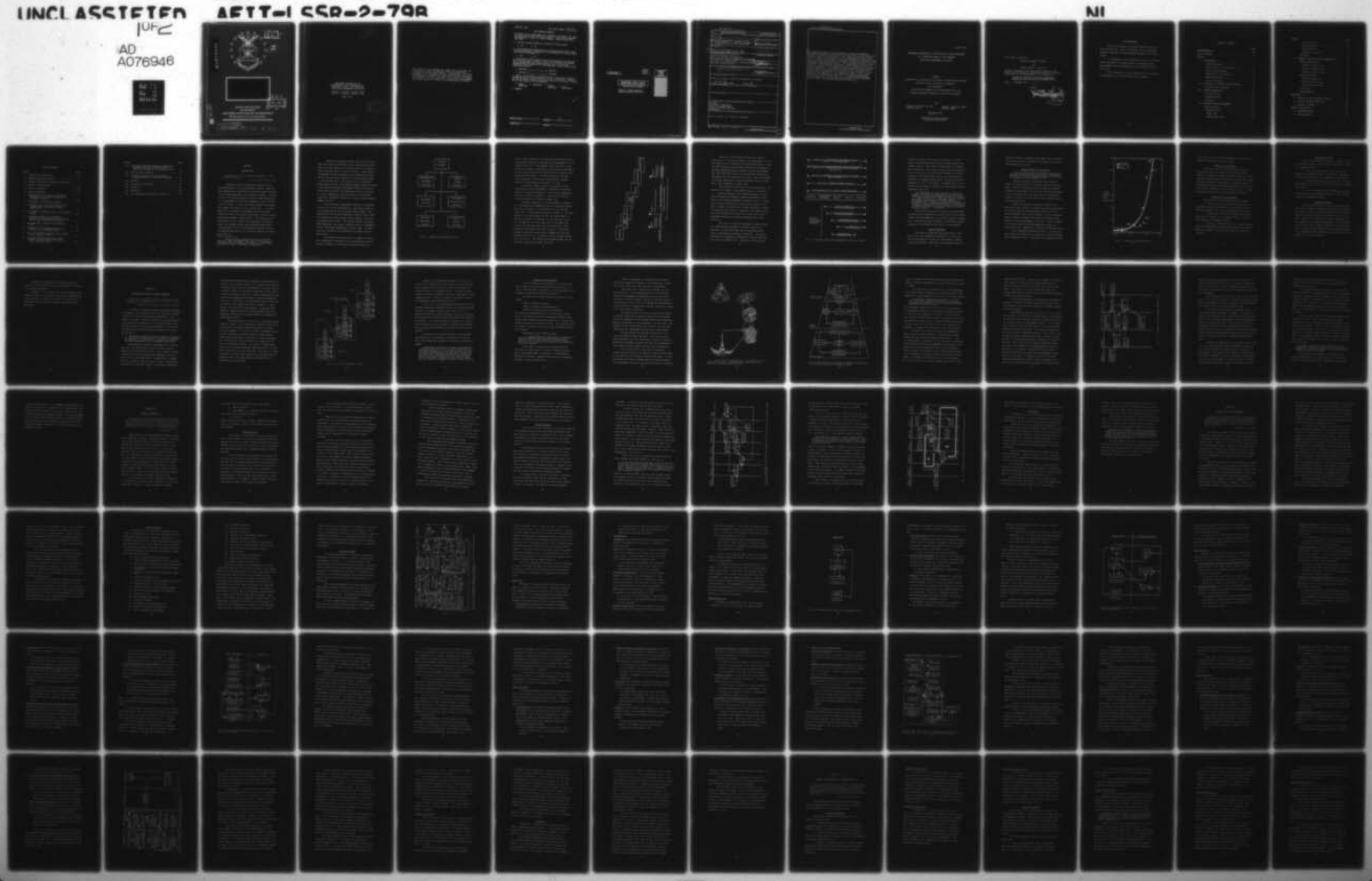
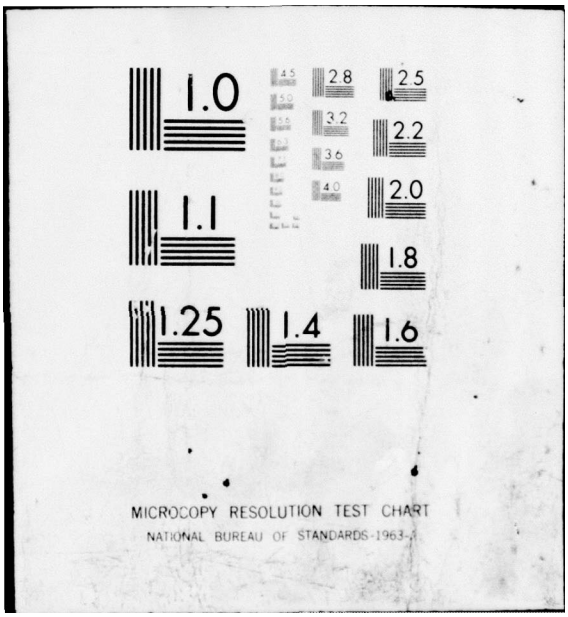
AD-A076 946 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/G 5/2
MANAGEMENT CYBERNETICS: AN APPLICATION TO THE DEVELOPMENT OF A --ETC(U)
SEP 79 R E PESCHKE , M L SHERRILL

UNCLASSIFIED AFIT-LSSR-2-79B NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A076946

LEVEL II

3



DDC
RECEIVED
NOV 20 1979
A

UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DDC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

79 19 9 053

(3)

MANAGEMENT CYBERNETICS: AN APPLICATION TO THE DEVELOPMENT OF A CONCEPTUAL MODEL OF THE SOFTWARE ACQUISITION MANAGEMENT DISCIPLINE

Richard E. Peschke, Captain, USAF
Marcus L. Sherrill, Captain, USAF

LSSR 2-79B

DDC
RECEIVED
NOV 20 1979
A

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

The contents of the document are technically accurate, and no sensitive items, detrimental ideas, or deleterious information are contained therein. Furthermore, the views expressed in the document are those of the author(s) and do not necessarily reflect the views of the School of Systems and Logistics, the Air University, the Air Training Command, the United States Air Force, or the Department of Defense.

USAF SCN 75-20B AFIT Control Number LSSR 2-79B

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: AFIT/ LSH (Thesis Feedback), Wright-Patterson AFB, Ohio 45433.

1. Did this research contribute to a current Air Force project?

   a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

   a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of man-power and/or dollars?

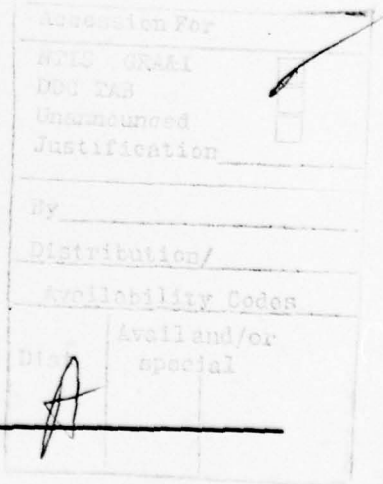
   a. Man-years _____ $ _____ (Contract).

   b. Man-years _____ $ _____ (In-house).

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3 above), what is your estimate of its significance?

   a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments:



Name and Grade Position

Organization Location

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 73236 WASHINGTON D.C.

POSTAGE WILL BE PAID BY ADDRESSEE

AFIT/LSH (Thesis Feedback)
Wright-Patterson AFB OH 45433

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: LSSR 2-79B	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): MANAGEMENT CYBERNETICS: AN APPLICATION TO THE DEVELOPMENT OF A CONCEPTUAL MODEL OF THE SOFTWARE ACQUISITION MANAGEMENT DISCIPLINE	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis / 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Richard E. Peschke, Captain, USAF; Marcus L. Sherrill, Captain, USAF	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Graduate Education Division, School of Systems and Logistics, Air Force Institute of Technology, WPAFB OH	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Department of Communication and Humanities, AFIT/LSH, WPAFB OH 45433	12. REPORT DATE: September 1979 / 13. NUMBER OF PAGES: 118
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): AFIT-LSSR-2-79B	15. SECURITY CLASS. (of this report): UNCLASSIFIED / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

JOSEPH D. HIPPS, Major, USAF 1 OCT 1979

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)

SOFTWARE
MANAGEMENT CYBERNETICS
ACQUISITION MANAGEMENT
EMBEDDED COMPUTER SOFTWARE
MANAGEMENT

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Thesis Chairman: Mr. Daniel E. Reynolds

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

012 250

The underlying thesis of this research is that the management cybernetics paradigm, developed by Stafford Beer, can be applied to Software Acquisition Management. The authors studied the current software acquisition process, how it is employed and controlled, and the current problem areas, and then developed a conceptual model that is capable of improving the process through the application of management cybernetic principles. The conclusions of this research are that a viable Software Acquisition Management Discipline is needed to gain control of acquisition costs, schedules and specification attainment. The conceptual model presented, applying management cybernetics, provides the manager with a systems view of what factors are affecting the acquisition process and how these factors are interrelated. This conceptual model, useful today, provides the framework for developing an effective management information system through the use of a computerized management control model that will enhance the managers' ability to effectively control the Software Acquisition Process.

LSSR 2-79B

MANAGEMENT CYBERNETICS: AN APPLICATION TO THE DEVELOPMENT OF A CONCEPTUAL MODEL OF THE SOFTWARE ACQUISITION MANAGEMENT DISCIPLINE

A Thesis

Presented to the Faculty of the School of Systems and Logistics

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Logistics Management

By

Richard E. Peschke, BA, MSA
Captain, USAF

Marcus L. Sherrill, BSME
Captain, USAF

September 1979

Approved for public release;
distribution unlimited

This thesis, written by

Captain Richard E. Peschke

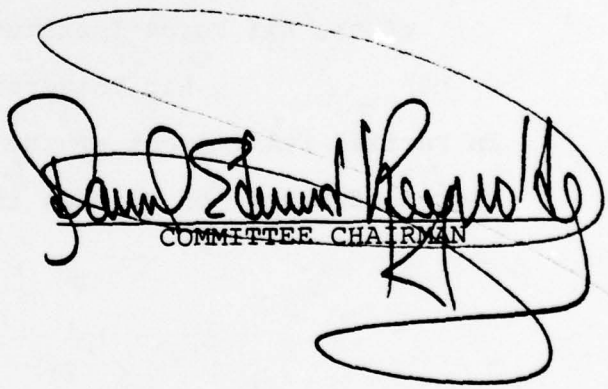
and

Captain Marcus L. Sherrill

has been accepted by the undersigned on behalf of the faculty of the School of Systems and Logistics in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN LOGISTICS MANAGEMENT
(CONTRACTING AND ACQUISITION MANAGEMENT MAJOR)

DATE: 7 September 1979



COMMITTEE CHAIRMAN

ACKNOWLEDGMENTS

We wish to express our sincere appreciation to our thesis advisor, Mr. Daniel E. Reynolds, for his ceaseless interest, guidance and understanding throughout this thesis effort.

A very special note of appreciation is due our wives, Jeanne and Linda, for their patience, understanding and encouragement which made bearable a very difficult year in our lives.

Final thanks go to Phyllis Reynolds for her patience and competence in the final typing of this thesis.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS iii

LIST OF FIGURES vi

Chapter

I. INTRODUCTION 1

Problem Statement 8

Justification of the Research 9

Scope of the Research 11

Objectives of the Research 11

Research Questions 12

Plan of the Report 12

II. THE NATURE OF SYSTEMS SCIENCE RESEARCH 14

Systems Science Paradigm 18

III. RESEARCH METHODOLOGY 28

Data Collection 29

Conceptual Model 32

Validation 37

IV. CONCEPTUAL MODEL DEVELOPMENT 39

Data Collection 42

Conceptual Model 44

Model List 46

Supplementary List 48

Chapter Page

First Extension 53

Second Extension 60

Third Extension 67

Summary of the Model 73

Validation 74

V. SUMMARY, CONCLUSIONS AND RECOMMENDATION 77

Research Objectives 77

Research Objective One 78

Research Objective Two 78

Research Objective Three 79

Research Questions 79

Research Question One 80

Research Question Two 81

Research Question Three 82

Conclusions 83

Recommendations 85

APPENDICES 88

A. INTRODUCING THE CORPORATE PARADIGM 89

B. LIST OF INTERVIEW QUESTIONS 105

C. GLOSSARY OF TERMS 109

SELECTED BIBLIOGRAPHY 113

A. REFERENCES CITED 114

B. RELATED SOURCES 117

LIST OF FIGURES

Figure		Page
1.	Standard SPO Organization	3
2.	Software Life-Cycle Phases	5
3.	Software Acquisition Management Directives	7
4.	Growth in DOD Software	10
5.	Levels of Recursion	16
6.	Cones of Resolution	20
7.	Application of the cones of resolution technique to the software acquisition management process	21
8.	An example of an Influence Diagram	24
9.	An example of an influence diagram with linking arrows and variable change indications	34
10.	An example of feedback loops in an influence diagram	36
11.	A conceptual model of the software acquisition management process utilizing the influence diagramming technique	45
12.	The Model List column of the conceptual model	49
13.	The Model List and Supplementary List columns of the conceptual model	52
14.	The First Extension and Model List columns of the conceptual model	57
15.	The Second and First Extensions and an excerpt from the Model List columns of the conceptual model	64

Figure		Page
16.	The Second and Third Extension columns and excerpts from the First Extension and Model List columns of the conceptual model .	70
A-1.	The Corporate Paradigm	91
A-2.	Exploded diagram of the brain showing classification as a five-tier hierarchy . .	93
A-3.	System I	95
A-4.	Activities of the Firm	96
A-5.	System II	98
A-6.	System III	100
A-7.	The Organization Interface of System IV . . .	102

CHAPTER I

INTRODUCTION

Software is the most expensive component in the systems procurement.

— *Dr. Ruth M. Davis [10:19]*

Computers and their associated software are an ever-growing part of our technological society. They are being marketed for personalized use and are being utilized in an ever-growing number in our defense systems. As early as 1975, 115 different defense systems, either operational or in development at that time, employed *embedded computer systems*[1] (30:43). *Software* for these computers has become the largest cost factor in total system cost. It is estimated that the software development costs now consume almost 90 percent of the total acquisition costs of a fully operational computer system (10:18). This percentage is significant in that, while total acquisition costs have risen, the hardware dollars have decreased dramatically. A microprocessor that can be purchased today for $20 has the computational power of a $1 million computer of twenty years ago (15).

---

[1]Certain words or phrases have been italicized, when first used, throughout this report. This action is intended to key the reader to the fact that these words have been defined in a glossary in Appendix C.

There have been many studies, done both by government and contractor personnel, that investigated the problems of various phases of the process used by the Department of Defense (DOD) to acquire software. These studies recommended various solutions to the identified problems but did not provide one aspect that is definitely needed in this complex arena. The aspect that is lacking was recognized by the Air Force Systems Command (AFSC) Director of the Computer Resource Development Policy and Planning Office when he stated: "The remaining task is to provide an overall perspective or architecture to the software acquisition management discipline in the Air Force Systems Command [19:36]."

Most software for embedded computers is purchased as a part of a given subsystem for a new weapon system or a major modification to an existing weapon system. The organization used by the Department of the Air Force (DAF) for acquisition of these major weapon systems is the system program office (SPO). The program manager, appointed under AFR 800-2, has overall responsibility for implementation of the Program Management Directive (PMD). The PMD is the authorization to proceed with the weapon system acquisition.

The standard SPO organization is shown in Figure 1. This organization, developed for the acquisition of major hardware systems, is functionally organized along the

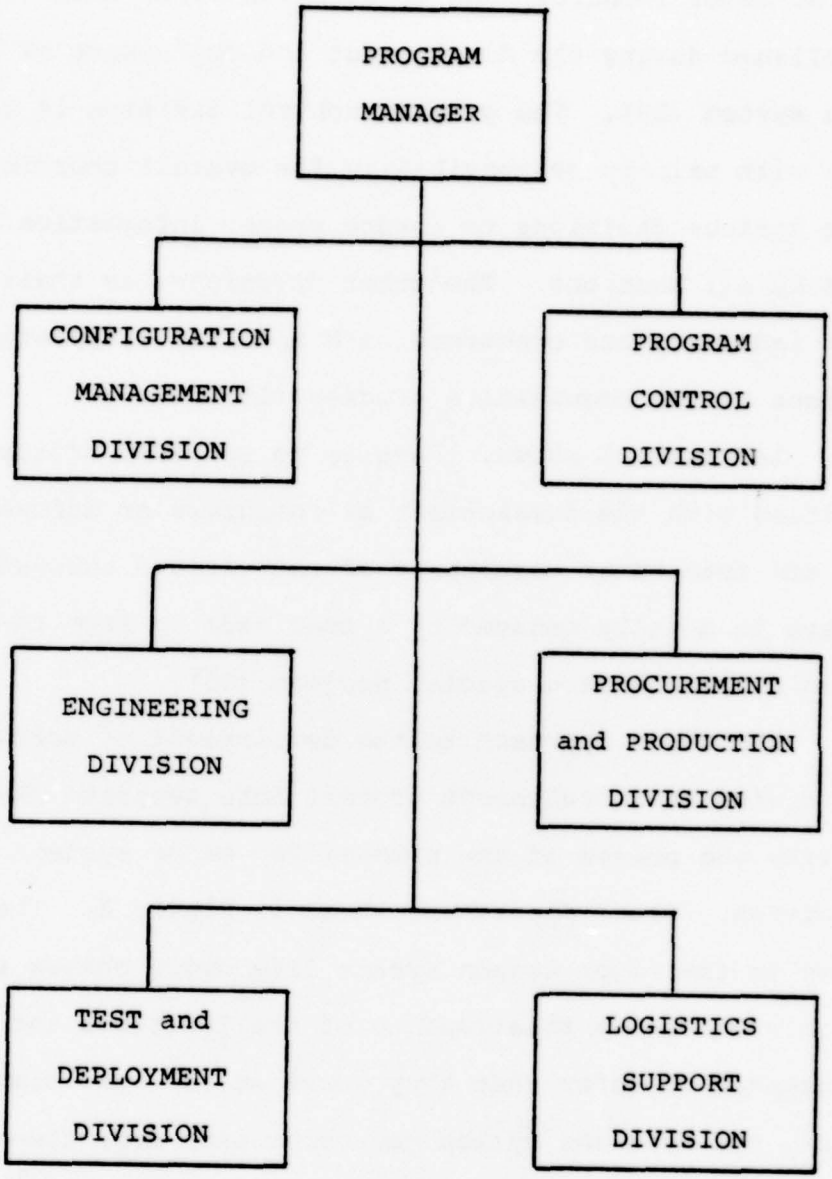


Fig. 1. Standard SPO Organization (20)

lines of major technical and acquisition milestones to be accomplished during the development and deployment of a new weapon system (20). The program control division is the office with primary responsibility for overall coordination of the various divisions to assure proper information flow needed by all sections. The other divisions, as their titles indicate, are concerned with specific, separate functions of the acquisition process (15; 20).

As Figure 1 shows, there is no specific office identified with the development of computers or software. These are treated as components of the various subsystems. Software is usually managed by a team made up from the various divisions as a special project (15).

The basic approach to the development of software is to divide the development process into separate phases much like the phases of the process for major systems acquisition. These phases are shown in Figure 2. The comparison to the major weapon system life cycle phases is done only to show a relationship of the functions and is not intended to infer that they occur at the same time. In fact, the software cycles may occur many many times throughout the system life cycle (21:4-6). The Defense Systems Acquisition Review Council (DSARC) has the final approval authority for all phases of major program acquisitions (21:15-20). The software development phases are not generally controlled by DSARC decisions.

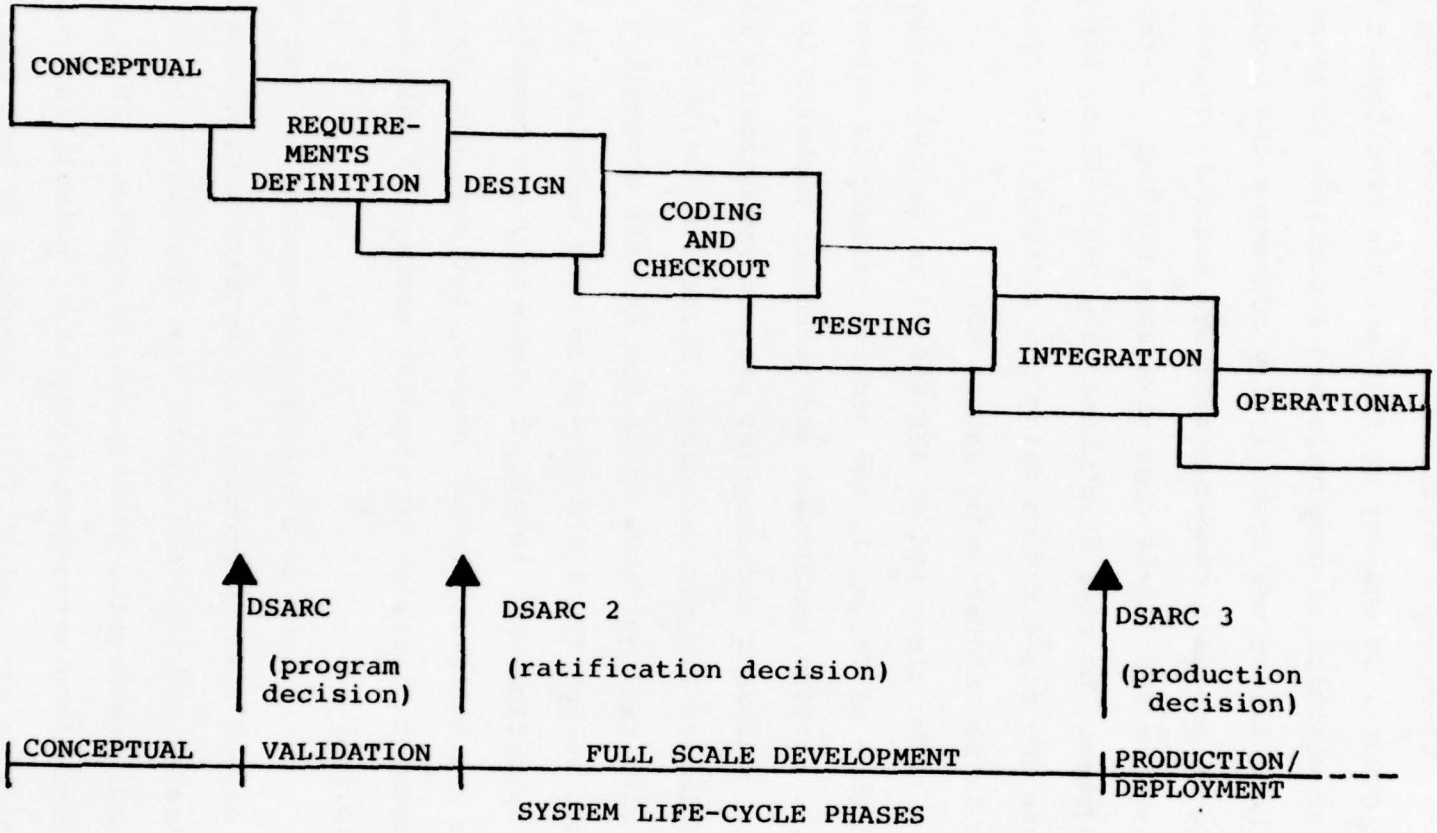


Fig. 2. Software Life-Cycle Phases (1:4)

There are a multitude of directives, Figure 3, that govern, in one way or another, the development of, and the acquisition of computers and associated software. None of these directives specifically addresses the problems of managing embedded computer software acquisition programs and, while AFR 800-14 does give some minimal reference to management, no real discipline is given. Also, the problem of the SPO organization using the software life cycle process is not addressed in detail (26).

The direction of AFR 800-14 is at best minimal. In Section B of Volume I, the program manager is directed to ". . . provide management and technical emphasis to computer equipment and computer program requirements identified in the Program Management Directive [34:3]." It does not give real guidance as to how he will accomplish this task. Volume II of AFR 800-14 is more technical in nature and specifies what levels of visibility the computer hardware and software should receive, but does not list any management direction to actually accomplish this task (32:p.8-2).

This lack of a specific management control system has caused serious problems in software acquisition. Typically, if the team leader has an engineering background, engineering principles are applied. If he has a configuration management background, principles from that discipline are applied (18). Overruns in both cost and

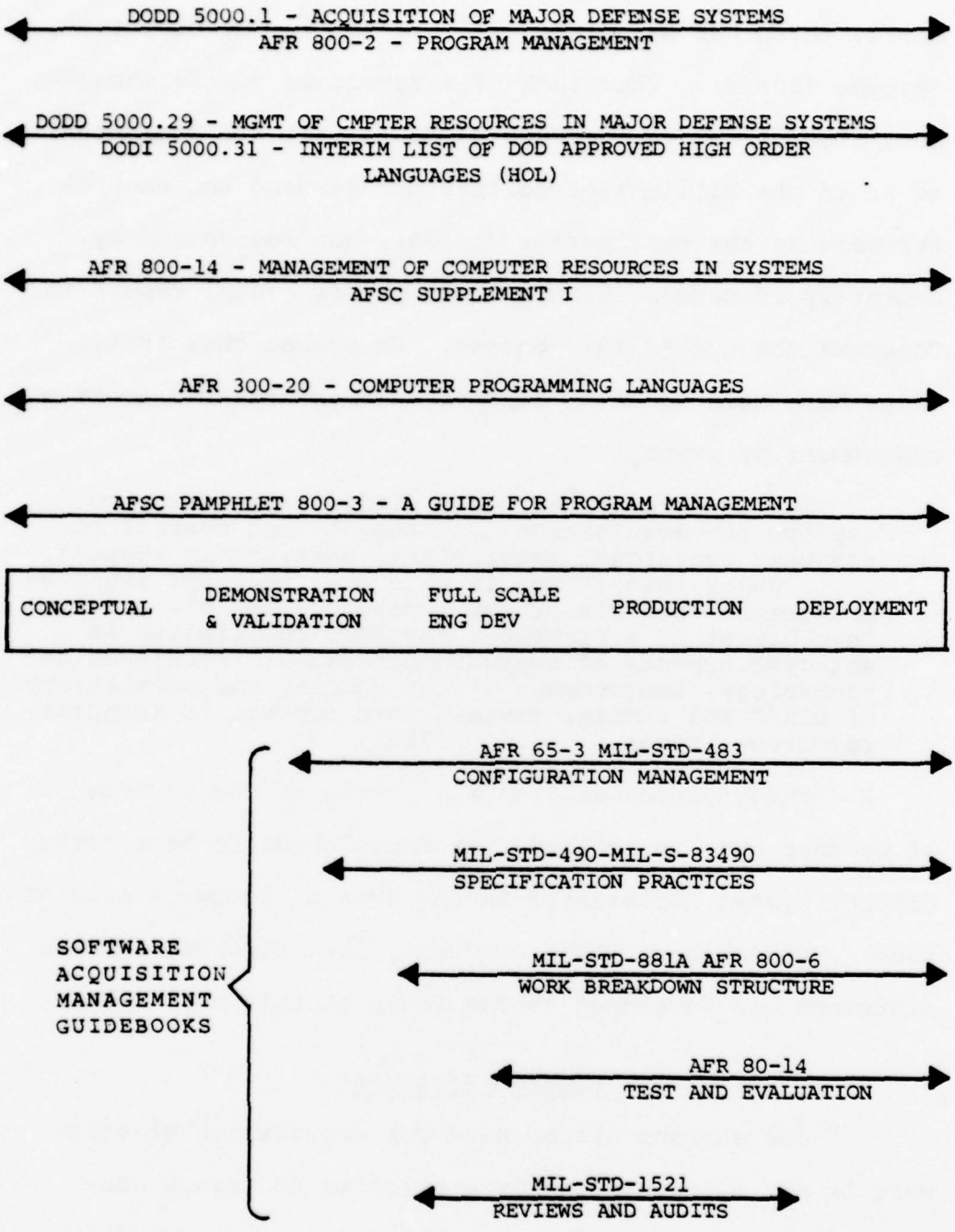


Fig. 3. Software Acquisition Management Directives (19:34)

schedule estimates have reached 100 percent. In some cases, there has been total failure to ever develop the systems (10:19). This lack of a specified *viable* management discipline, in the face of costs that are estimated to be in the billions of dollars for command and control software in the next decade (10:29), was recognized by Secretary of Defense Harold Brown in his annual report to Congress for the FY 1979 budget. He showed that initiatives have been taken to improve this and related areas of management by saying:

> The department has taken significant action to improve the acquisition, management, and control of computer resources, particularly software in weapons. . . . Major initiatives in this area include: improved management controls of our research efforts . . . , development of a framework for NATO cooperation in selected aspects of software management techniques and technology, improvement of the quality and consistency of DSARC and similar reviews with respect to computer resources issues, . . . [33:359].

Thus, personnel from all levels of the Department of Defense have recognized that there needs to be a better control system implemented in the area of software acquisition. Motivated by these concerns, the following problem statement was developed as the focus of this research.

Problem Statement

The current system used for acquisition of software is not being adequately controlled to insure consistent satisfaction of technical performance, schedule, and cost objectives. Therefore, a requirement exists to

develop a dynamic management system model that is capable of providing control information to managers of the software acquisition process.

Justification of the Research

> For most weapon system development programs that incorporate both hardware and software, the computer software is a critical component relative to the overall operation of the system [17:13].

Figure 4 portrays dramatically the tremendous growth in system software in the last twelve to fourteen years. The 1977 estimated cost of software development, testing, and maintenance for the entire federal government is $4 billion per year, and these costs are expected to climb to a figure ten times greater than hardware costs in the years ahead (23).

Today, according to Lieutenant Colonel John J. Marciniak, USAF, Director of Computer Resources Development Policy and Planning, DSC/Development Plans, Air Force Systems Command, "The software problem is seen as excessive costs, schedule slippages, and reduced performance compared to initial requirements [19:32-33]." What is needed is a clearly defined software management discipline. A discipline which, according to Lieutenant Colonel Marciniak, ". . . is not adequately described [19:34]." If the software acquisition problem is to be solved it must be brought under management control. One method of accomplishing this

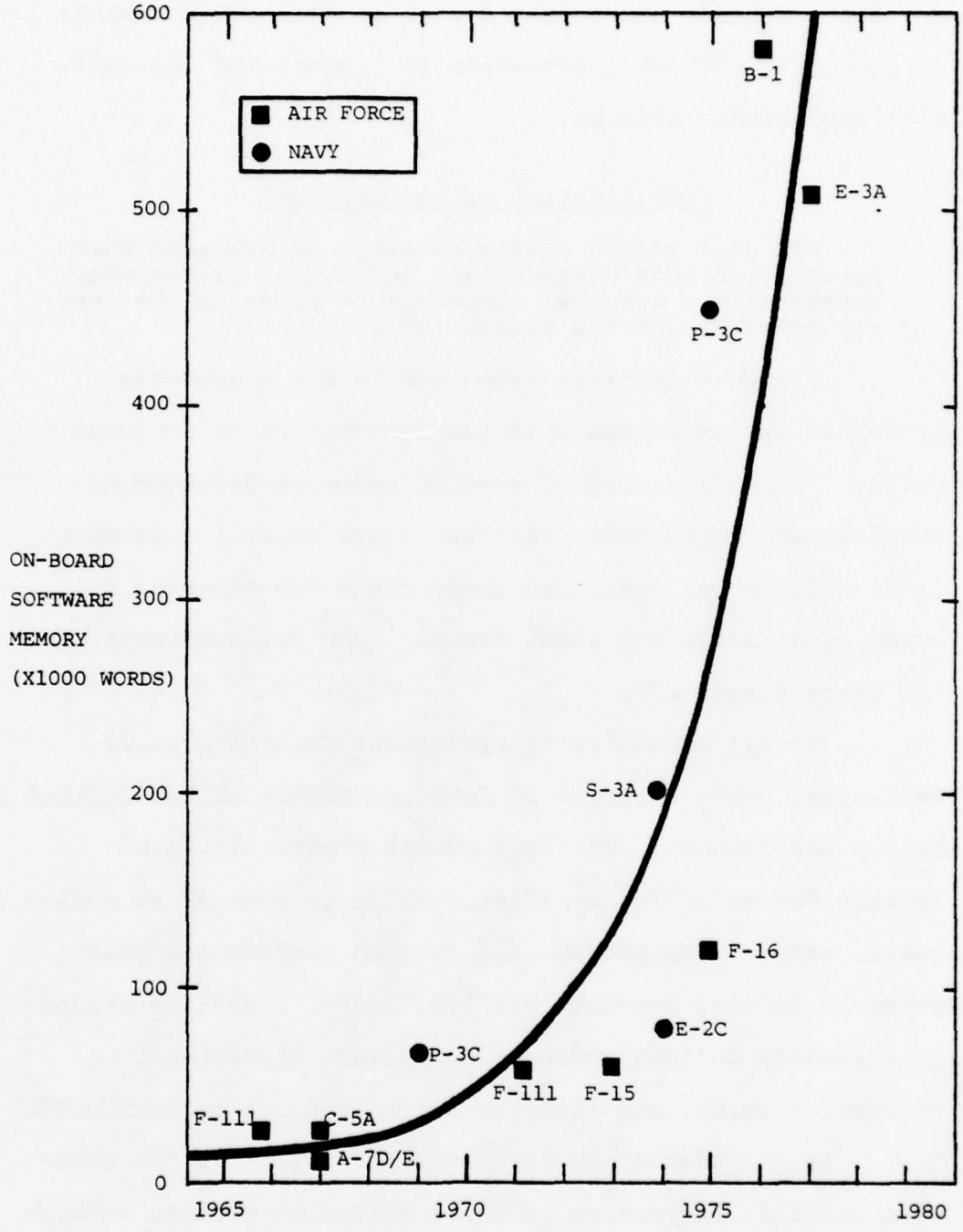


Fig. 4. Growth in DOD Software (23)

is to clearly define the relationships of the activities within the software acquisition process.

Scope of the Research

The scope of this research has been limited to identifying a set of variables (activities) which are required to adequately control software acquisition, operationally defining these variables (activities), and, using cybernetic principles, developing a conceptual model of the software acquisition management process that will provide control information to the manager in terms of these variables (activities) and their interrelationships.

Objectives of the Research

The objectives of this research project were to:

1. Investigate the current software acquisition management process with emphasis on the elements that are creating problems in the areas of employment and control.

2. Identify and define the variables (activities) which must be included in a conceptual model of the software acquisition management process.

3. Develop a conceptual model of a viable software acquisition management process which explains the system's behavior in terms of the activities and their interrelationships.

Research Questions

The following research questions, compatible with the problem statement and the research objectives, guided this research effort:

1. What are the elements within the software acquisition management process, as currently employed, which are contributing to the problems that now exist?

2. What are the variables (activities) that must be included in a model of the software acquisition management process?

3. Can a conceptual model be developed that can accurately portray the dynamic behavior of the software acquisition management process?

Plan of the Report

The following chapters, using the research questions and the research objectives, presented in Chapter I, as a guide, develop a proposed solution to the management problem that exists within the software acquisition process. Chapter II is devoted to a general discussion of the systems science research methodology. This discussion is little more than a summary statement of the techniques used in systems science research and is presented to provide a background for understanding the results and conclusions of this research project.

Chapter III discusses the methodology that was applied in this research project and introduces the concept of influence diagramming.

Chapter IV is concerned with the development of the conceptual model of the software acquisition management process and Chapter V presents the specific conclusions and recommendations for further research that resulted from that model.

CHAPTER II

THE NATURE OF SYSTEMS SCIENCE RESEARCH

> *Control can be obtained only if the variety of the controller is at least as great as the variety of the situation to be controlled.*
>
> — *Ashby's Law of Requisite Variety [4:53-54]*

In order to successfully apply the systems approach to management, the organization must be viewed as a system. A system is a "body of interrelated components." This sequence of terms is important because the systems approach first develops an understanding of the whole (body), then analyzes the parts of the whole (components), and then the interrelationships among the parts and between the parts and the whole.

> The systems approach discounts simplistic statements of "principles of organization" and reflects the search for patterns of relationships, configurations within and among subsystems, and a contingency view [16:23].

To develop the required understanding of the whole one could, and indeed should, start with the universe. Clearly the universe qualifies as a "whole system;" however, a study of the software acquisition management process at this level is quite cumbersome. To concentrate a systems study in a specific area of interest without violating the requirement to study the "body of interrelated components"

the Recursive System Theorem developed by Stafford Beer is applied. This theorem states, "if a viable system contains a viable system, then the organizational structure must be recursive [4:287]." What this theorem says is that within a viable organization there is a viable suborganization and within that suborganization there is another suborganization, etc., right down to the individual worker at the lowest level of the organization who is, himself, a viable system, and therefore a whole system. By applying this theorem, a detailed analysis of the software acquisition management process can be accomplished at the appropriate organizational level.

The application of this *cybernetic* principle is presented in Figure 5 which represents a portion of the recursive pattern of material acquisition within the United States Government. The *niveau* in level I represents one of the many functional divisions within a system program office (SPO) within AFSC. The level II niveau represents one of the project functions which is an element of the level I functional division. The level III niveau represents one of the many subproject functions which is an element of the level II project function. The level II niveau (referred to as the Prime Niveau) is the specific level of recursion within the organizational structure that is addressed by this research.

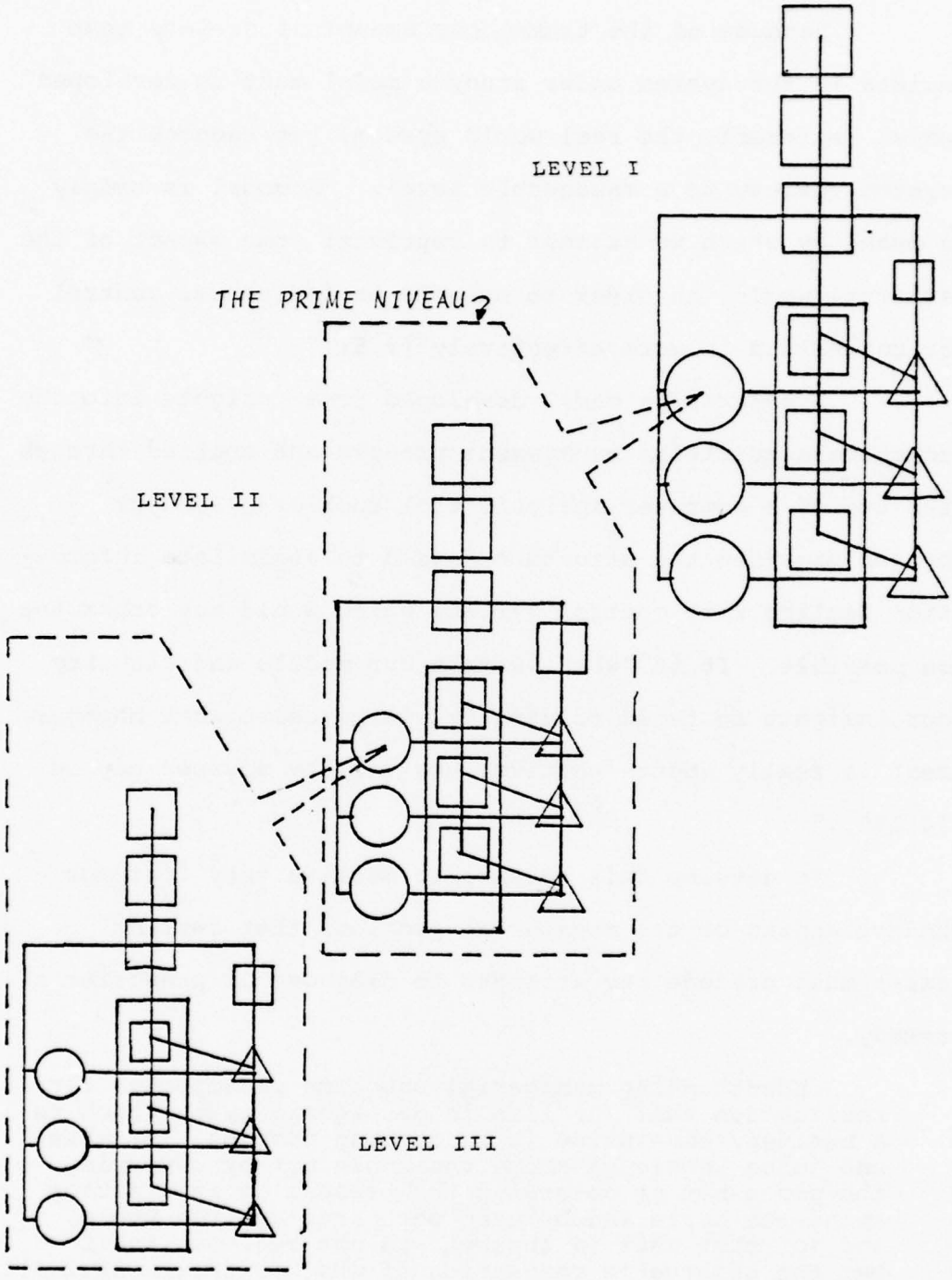


Fig. 5. Levels of Recursion (4:200)

Because of the tremendous amount of *variety* that exists in the system under study a model must be developed which represents the real world system, yet reduces the system variety to a manageable level. "A model is simply a means by which we attempt to represent some aspect of the external world, in order to be able to influence, control or understand it more effectively [9:5]."

A *scientific model* developed from insights into the software acquisition management process and applied through the use of a computer analysis tool such as *Q-Gert* or Dynamo provides the structure needed to assimilate information dealing with control variety which would not otherwise be possible. It is "when we make our models and classify our insights in terms of variety, we perceive what management is really about--whatever the variety sources may be [4:290]."

To develop this scientific model a very thorough understanding of the managerial problems that currently exist must precede any attempts to diagnose or prescribe a remedy.

> Understanding managerial problems presupposes the realization that (a) life in an organic system such as a business enterprise is an ongoing process, (b) that one gains knowledge about the whole not by observing the parts but by observing the process of interaction among the parts and between the parts and the whole, and (c) that what is observed is not reality itself but the observer's conception of what is there [29:247].

Systems Science Paradigm

Once an adequate understanding of the whole relevant system is developed the systems science *paradigm* is used to complete basic milestones in the development of the software acquisition management discipline.

By applying the systems science paradigm in three phases:

Phase I--Conceptualization,

Phase II--Analysis and Measurement,

Phase III--Computerization (29:254-259),

the perceived overwhelming task of holistically investigating the software acquisition management process, which exists under constantly changing conditions, is facilitated through the modelling process (29:254). This modelling process begins in Phase I with a very rough conceptualization of the system.

Conceptualization in Phase I means,

> . . . understanding and organizing the interactions among the elements making up the phenomenon under scrutiny into a logical network of relationships in such a way as to reveal the direction of the underlying structure [29:249].

In an effort to develop this understanding and organizing of interactions, a series of modelling activities is undertaken which is arranged in a thoroughness-abstraction hierarchy. Stafford Beer calls this hierarchy of models a "Cone of Resolution."

Figure 6 graphically illustrates the increase in detailed information available to the researcher, or manager, as he or she moves down the cone of resolution.

The first step in applying the concept of "cones of resolution" to the conceptualization of the software acquisition management process begins at the top of the cone where the most abstract thinking is applied (Figure 7). The objective at this level is to develop an understanding of the acquisition organization, the SPO, and how it relates to its environment in general.

The second level in the cone of resolution focuses in greater detail on the software acquisition environment and how the elements of the software acquisition management process interrelate to the environment. At this point, the elements within the SPO which work directly with the software acquisition management process become identifiable.

Through observation and data collection at this level the research effort develops a list of activities and/or variables that are determined to affect the software acquisition management process, and then the conceptualization effort can move further down the cone of resolution.

At the third level in the cone of resolution research efforts focus on developing a logical network model of the control and information channels that exist. At this point facts have been collected about the software acquisition management process and the activities that are required

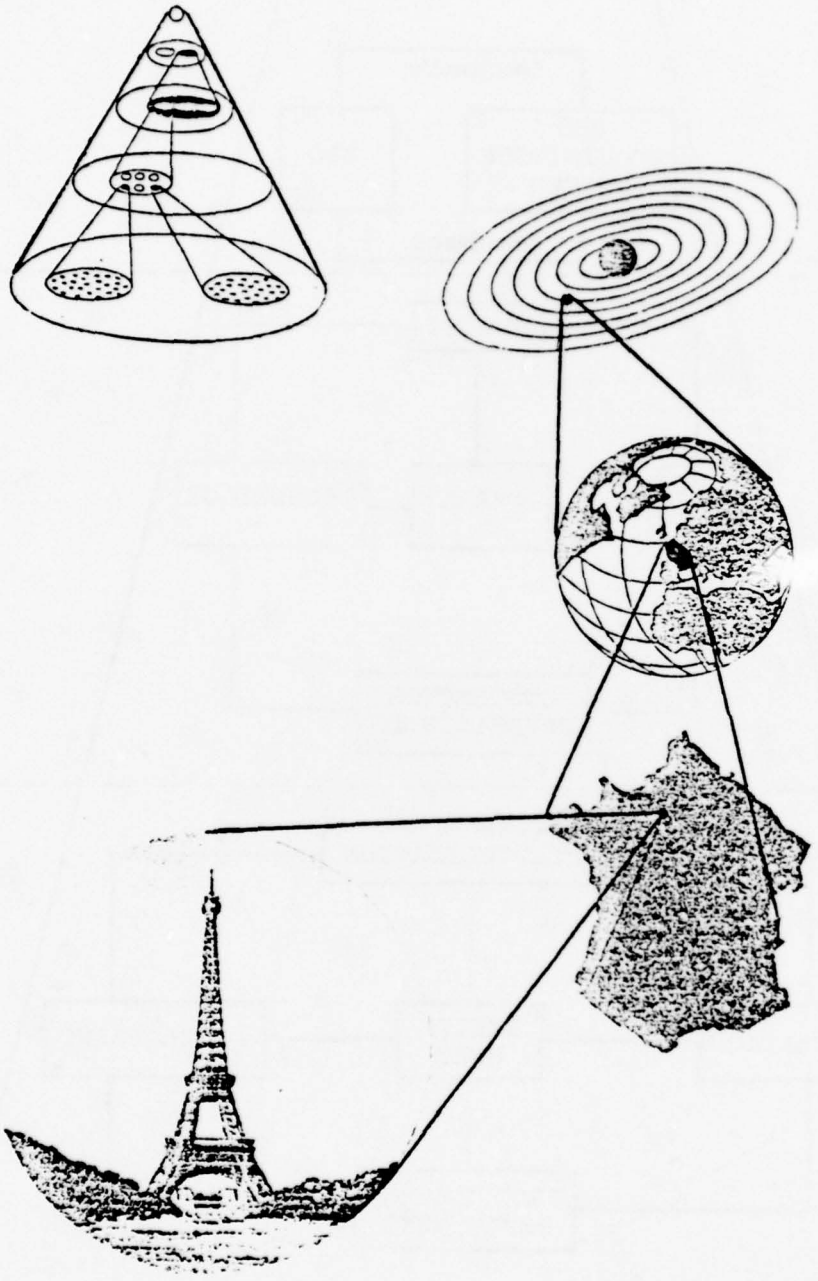


Fig. 6. Cones of Resolution. Each feature at one level may represent a tremendous amount of detail when examined on a larger scale (29:248)

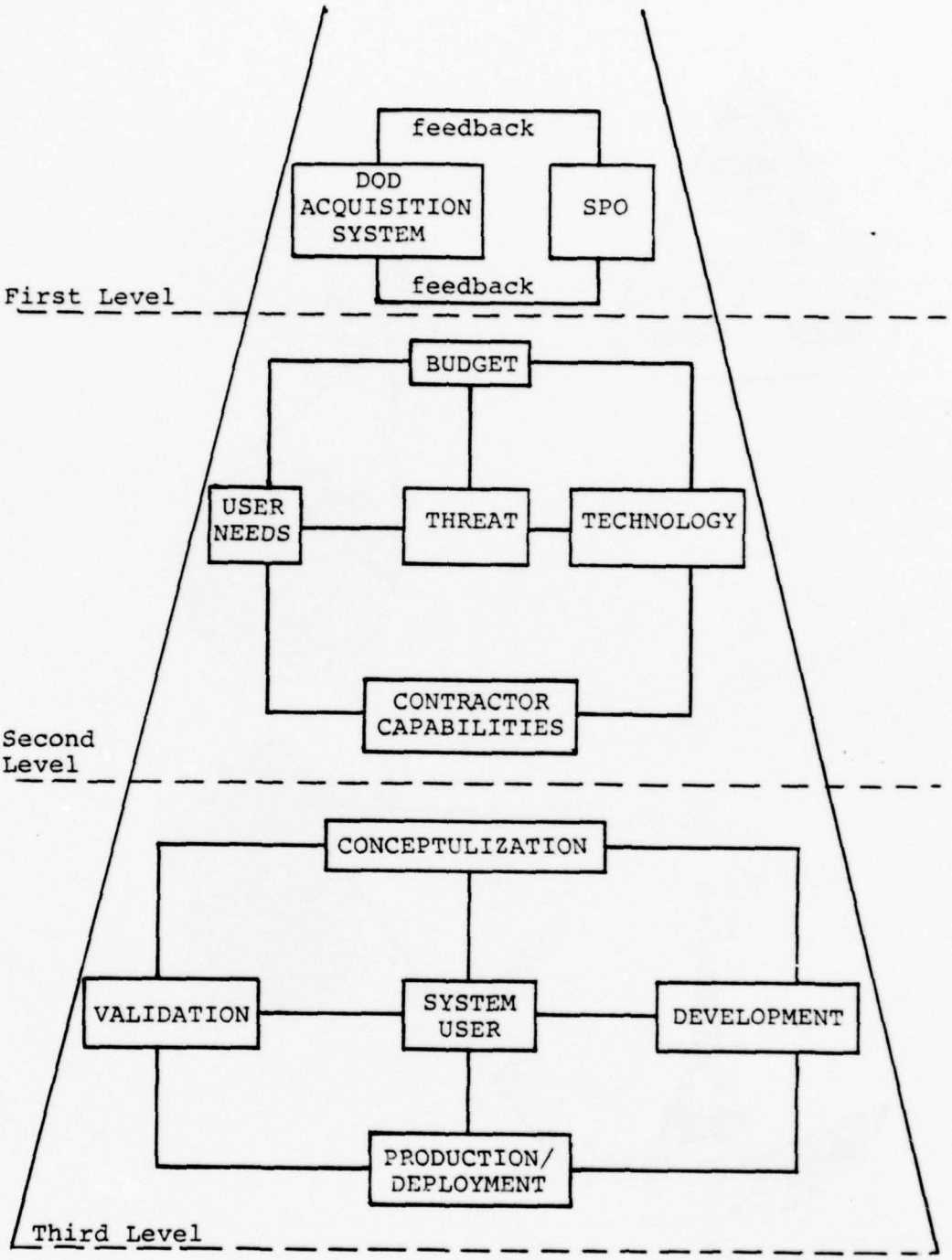


Fig. 7. Application of the cones of resolution technique to the software acquisition management process

within a viable system structure have been identified even though a complete understanding of these activities is still not known.

The next step is to explain the interrelationships of the activities by advancing the conceptual model of the system into a more detailed *homomorphic model* of the process.

> The process of homomorphic modelling is at the least a heuristic method for inferring the existence of structure of systems of which the complexity defeats isomorphic modelling [6:125].

This statement implies that the first act in the process of homomorphic modelling is to reduce the tremendous amount of variety in the system, but to reduce it in a way that will maintain enough detail to make it possible to deal with the situation at hand. Also, the model must be constructed in such a way as to allow some of the lost variety to be replaced, if needed, at some later time during simulation.

The basic tool used to develop this homomorphic model was the influence diagram. "The influence diagram records the way in which the system works [9:63]." This is accomplished by listing all of the variables (activities) that have been identified as elements of the system at the appropriate level of resolution and then linking each of these variables (activities) so as to show how each affects, or is affected by, the other variables (activities) in the model. To simplify the linking of each of the variables the list is written out across the page rather than in the

usual vertical column. Figure 8 is an example of an influence diagram of a production-inventory model that was developed using the horizontal listing and linking technique. What results is a structural model which very clearly portrays the structure of the system and the manner in which the system functions.

A detailed discussion of the influence diagramming process is included in Chapter IV where the actual influence diagram of the software acquisition management process has been developed.

The modelling process at this point is not a mathematical model, but rather a structural model of the stochastic processes that have been identified through research of the software acquisition management process. It is within this structural model that the capacity to replace reduced variety has been incorporated. This structural model contains four forms of suppressed variety. First, none of the variables have any numerical data values assigned to them. Second, the relationships between variables will differ in quantity, such as those which represent different activities yet are all defined by the same quantifier--years of experience, for example. Relationships which differ in natural features are the third form of suppressed variety present in the model. An example of this is the interrelationship between the timing of the development of an operational definition and the accuracy of that definition.

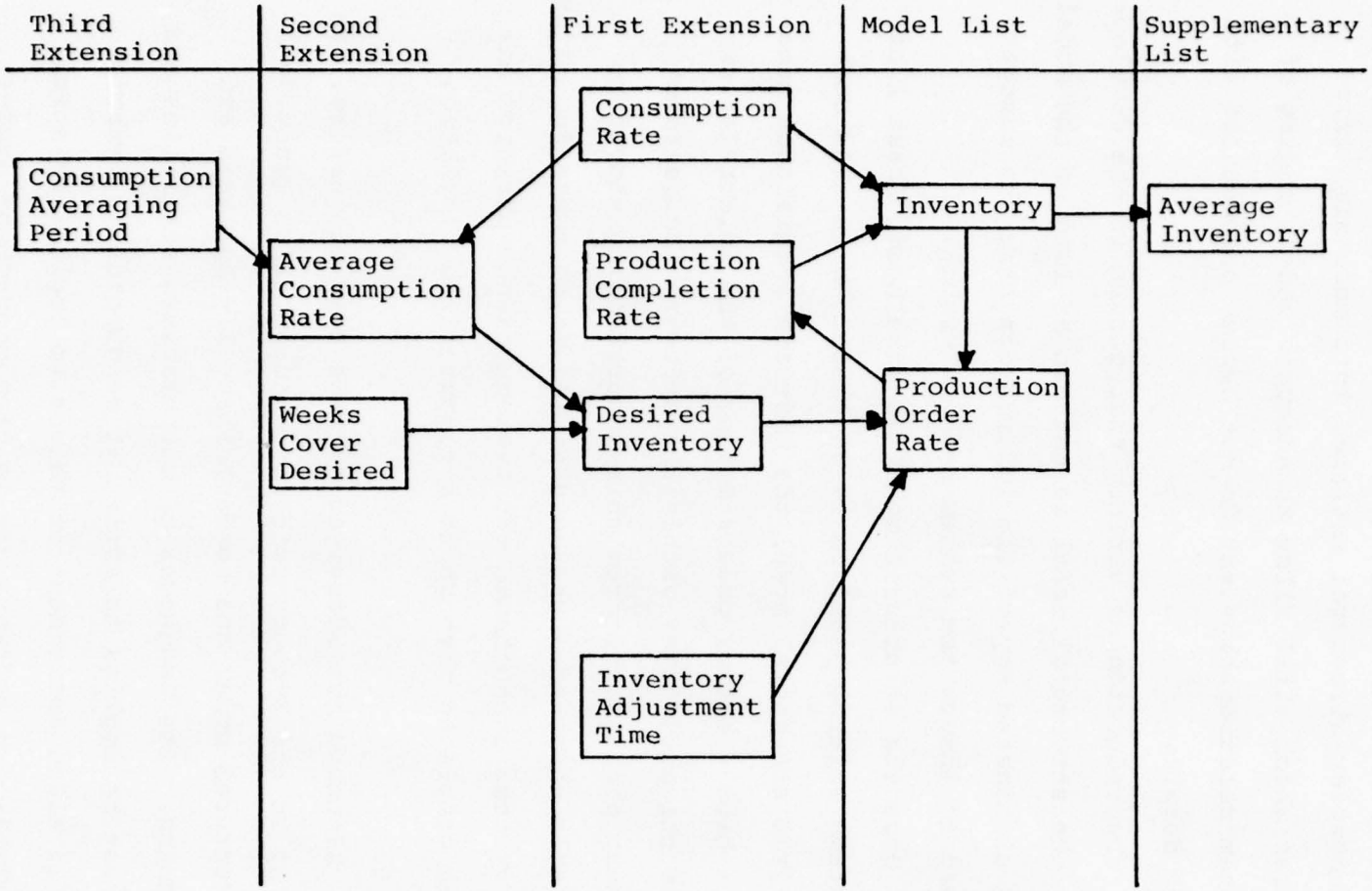


Fig. 8. An example of an Influence Diagram (9:74)

The final form of suppressed variety is the reduction of certain complex structural entities into more simplified structures which still allow an adequate understanding of the system relationships yet do not become preoccupied with too much detail.

The reduction of variety that occurs in the development of the structural model is not bad as long as the model has not eliminated any of the information that the manager will need to insure the system remains viable.

Once the structural model is developed, Phase I of the systems science paradigm is complete. With this conceptualized structural model the software acquisition manager will have a better understanding of the interrelationships of the activities within the software acquisition management process and from this understanding should be better able to control the acquisition of software for major weapon systems even though the system science paradigm has not been completed through to a computerized simulation model.

Although not addressed in this research effort, the next step in the systems science paradigm is to quantify the structural model and conduct Phase II--Analysis and Measurement. The languages of mathematics, statistics and logic must be applied to arrive at meaningful information that will allow assignment of values to logical relationships within the software acquisition management process,

analyze activities and assess those that are critical to system viability, and determine the accuracy and control potential of the model.

The ultimate product of Phase II of the systems science paradigm is a mathematical model that is then translated into a computer project (29:259). This translation comprises Phase III--Computerization of the systems science paradigm. The computerization, whether Dynamo, Q-Gert, or some other more appropriate method, will be a simulation model of the homomorphic structural model that was developed in Phase I.

The advantages gained from simulation modelling are three-fold. First, simulation provides an artificial experience of the real system much more quickly than could otherwise be obtained. Second, there are no risks involved in gaining this experience. Third, it is possible to make changes to the present system and project possible outcomes under the new system (6:231). Simulation allows determination of what would happen as a result of certain actions or policy decisions.

> Scientific analysis confirms what common sense comprehends, namely that information cannot be had for nothing, and a predictive model is only as good as the information fed to it [6:326].

The principles and concepts of systems science research presented in this chapter provide a brief introduction to the systems approach of analyzing and managing

a complex organization. In the chapters that follow, this systems approach will be applied to the identification and definition of the variables (activities) that exist in the software acquisition management process, and the development of a structural model through the use of influence diagramming that presents to the manager a clear picture of the interrelationships that exist in the software acquisition management system.

CHAPTER III

RESEARCH METHODOLOGY

For the first time in the history of man, science can do whatever can be exactly specified. Then, also for the first time, we do not have to be scientists to understand what can be done. It follows that we are no longer at the mercy of a technocracy which alone can tell us what to do. Our job is to start specifying.

— *Stafford Beer [7:56]*

When stated in the very simplest of terms, the two steps required to develop a conceptual model of any system are to write down the names of all the activities that impact that system and connect them with arrows to show what effects they have on one another. A third step, generally felt to be required of all modelling exercises, is to, in some manner, validate the model to insure that all the variables are present and that the proper interrelationships have been identified. Unfortunately, the procedures are much more complicated when applied to a real world, complex system. In fact, entire research efforts could be done on only one of these steps when the ultimate objective is to model an extremely complex system.

In that much of the theory discussed in Chapter II is an approach to thinking about a problem, it will not be reiterated. However, it has been applied rigorously

throughout the three research tasks listed below.

1. Data collection

2. Development of a conceptual model of the software acquisition management process

3. Validation

Each of these three tasks, stated so simply in the beginning, is discussed in detail in the remaining portions of this chapter.

Data Collection

Since there is no accepted list of activities that have a direct impact on the costs or schedule of software acquisition, the first step in modelling the process was to compute a list of all the activities that have an impact on the successful acquisition of embedded computer software programs.

In that there was no acceptable consolidated list of these activities that could be used as a starting point, the data had to be gathered essentially from ground zero. The only easily accessible place this data is held is in the minds of those personnel that are, or have been, involved in this process during the acquisition of current weapon systems. Thus, the only course open was to interview personnel in an attempt to identify those activities which have an impact on the acquisition process.

Initial research into this area provided a list of several activities which, in the contributors' opinions (18; 26), have had a direct impact on programs in the recent past.

These activities included determination of central storage core size, availability of support software to contractors, availability of government-furnished software, early determination of weapon system software requirements, level of expertise in the government contract monitoring office, and the importance of a particular computer language.

With this list as an initial set of contributing activities, the interview guide in Appendix B was used to interview the Commander, Air Force Logistics Command; the Chief, Software Support Center Branch, WR-ALC/MAIT; the personnel of the Operational Flight Program software support office, WR-ALC/MMEC; the Chief, Embedded Computer/ Software Group, AFALD; personnel from the Simulator System Program Office in ASD; the Chief, Computer Resources Division, AFLC/LOEC; the Avionics Integration Engineering Advisor, ASD/ENA; and a Software Project Manager in ASD/YYM in an effort to develop a comprehensive set of contributing activities. In addition, telephone communications were conducted with others involved in software acquisition to insure a cross-section of those involved in software

acquisition had the opportunity to make inputs to this set of contributing activities.

These interviews were not intended to gather the same type of data that might be gathered in behavioral science data assimilation interviews where distinctive areas of interest have been predetermined. Therefore, only a small list of questions was compiled to help structure the interviews. These questions were used to expand the research questions listed in Chapter I. This group of questions provided adequate guidance during the interviews to be able to determine those activities that are significant to the software acquisition management process.

The interviews were purposely conducted in both Air Force Systems Command and Air Force Logistics Command in order to determine activities that impact throughout the life cycle of the software acquired. It does no good to acquire software at a very low dollar value when it will cost a small fortune to maintain it over the useful life of the weapon system. Also, since software problems are beginning to impact all levels of our organizations, these interviews attempted to cover various eschelons of the commands in order to discover all of the activities that lead to difficulties in Air Force software acquisition programs.

Once the list of activities had been compiled and reviewed, an operational definition was proposed for each activity and the interviews were repeated to provide

additional feedback on these definitions. The ultimate intent of these interviews was to obtain some concensus as to what the operational definitions were for the activities identified. Again, by going across commands and levels of the organizations, these definitions should contain the elements important to most offices involved in this process.

Conceptual Model

Once the activities influencing the software acquisition management process were identified and defined, the next step was to develop a conceptual model, using influence diagramming, to properly display how all the activities interact and where they make their impacts on the overall process.

The particular niveau of the organization investigated was an element of the system program office since most acquisition programs are controlled from that level. The corporate paradigm, as proposed by Stafford Beer, was used to identify just what functions, within that element, were to be included in the model directly. Therefore, only a portion of the SPO functions were considered in diagramming the software acquisition management process.

As stated before, the first two steps in construction of a conceptual model are to write down all those activities that impact the system and then connect them by an arrow or influence link using the influence diagram

technique. In reality, the second step is not any easier than was collection of data for the first step.

Prior to drawing any influence lines, the activities are sorted into categories including the model list, the supplementary list, and the extension lists. Those limited activities at which control is aimed, are placed in the model list. It is good to limit these, even in a completed model, in order to have a reasonably clear and coherent purpose for the model. Secondly, those activities which most immediately affect the model list are placed in the first extension. Those activities that most immediately offset those of the first extension are placed in the second extension and so on. Those activities that simply act as indicators of system performance and which play no other part in the system or its control policies are placed in the supplementary list (9:70-73).

After each list is filled the influence lines are drawn between activities (Figure 9). The rule is:

> If the head variable [activity] changes in the same direction as the tail variable [activity], use a + [plus] sign, but if it changes in the opposite direction, use a - [minus] sign; if the result is sometimes in the same direction and sometimes in the opposite direction use an asterisk [9:63].

After all of the influence lines are drawn between successive extension lists, the entire listing must be reviewed to include those activities which also influence some others not in an adjoining list, i.e., an activity in the

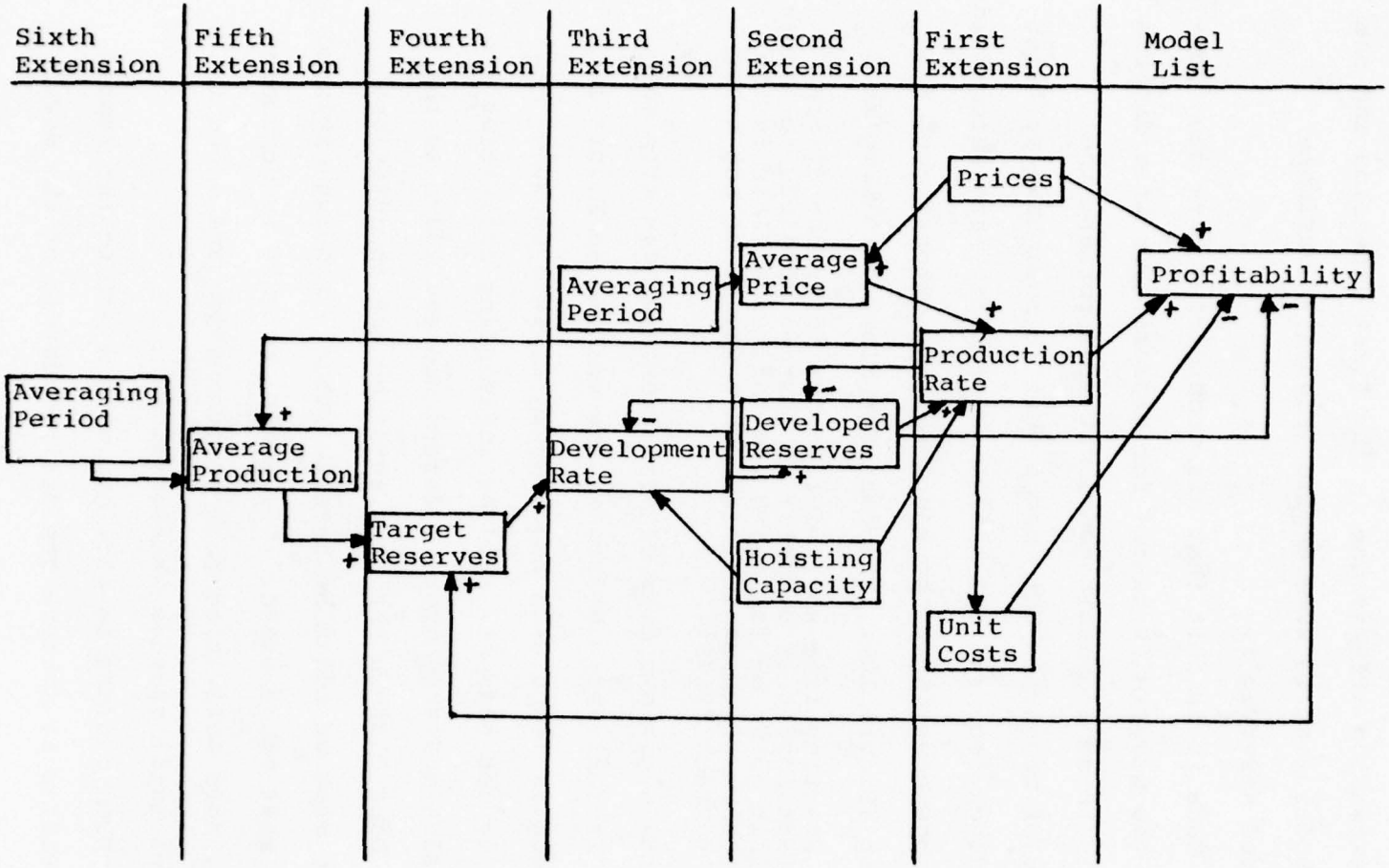


Fig. 9. An example of an influence diagram with linking arrows and variable change indications (9:76)

second extension affects one in the first extension and one in the model list to some degree (Figure 9; variable "Developed Reserves").

Once it is felt that all connections have been completed, the diagram is tested for closure, for if a diagram is really to be a dynamic model, it must possess the property of closure. This means that it must have at least one feedback loop and all activities except those determined to be exogeneous inputs or supplementary outputs must lie on a loop (Figure 10). The test for closure is simple:

> Starting from any point in the influence diagram [except inputs or outputs] it must be possible to return to that point by following the influence lines in the direction of causation, in such a way as to not cross one's track [9:70].

Thus, with a given number of variables, one can stop when closure is achieved. Notice in the example in Figure 10, four feedback loops were required to achieve closure. If closure is not present, all the influencing activities are not present and they must be determined and added to the model. Once closure has been attained and an additional variable must be added because the model is being expanded, closure must be retested. If, again, closure is no longer present, more activities must be identified and added to the model until closure is reattained.

When closure is attained with a sufficient number of activities to achieve the desired detail of the model,

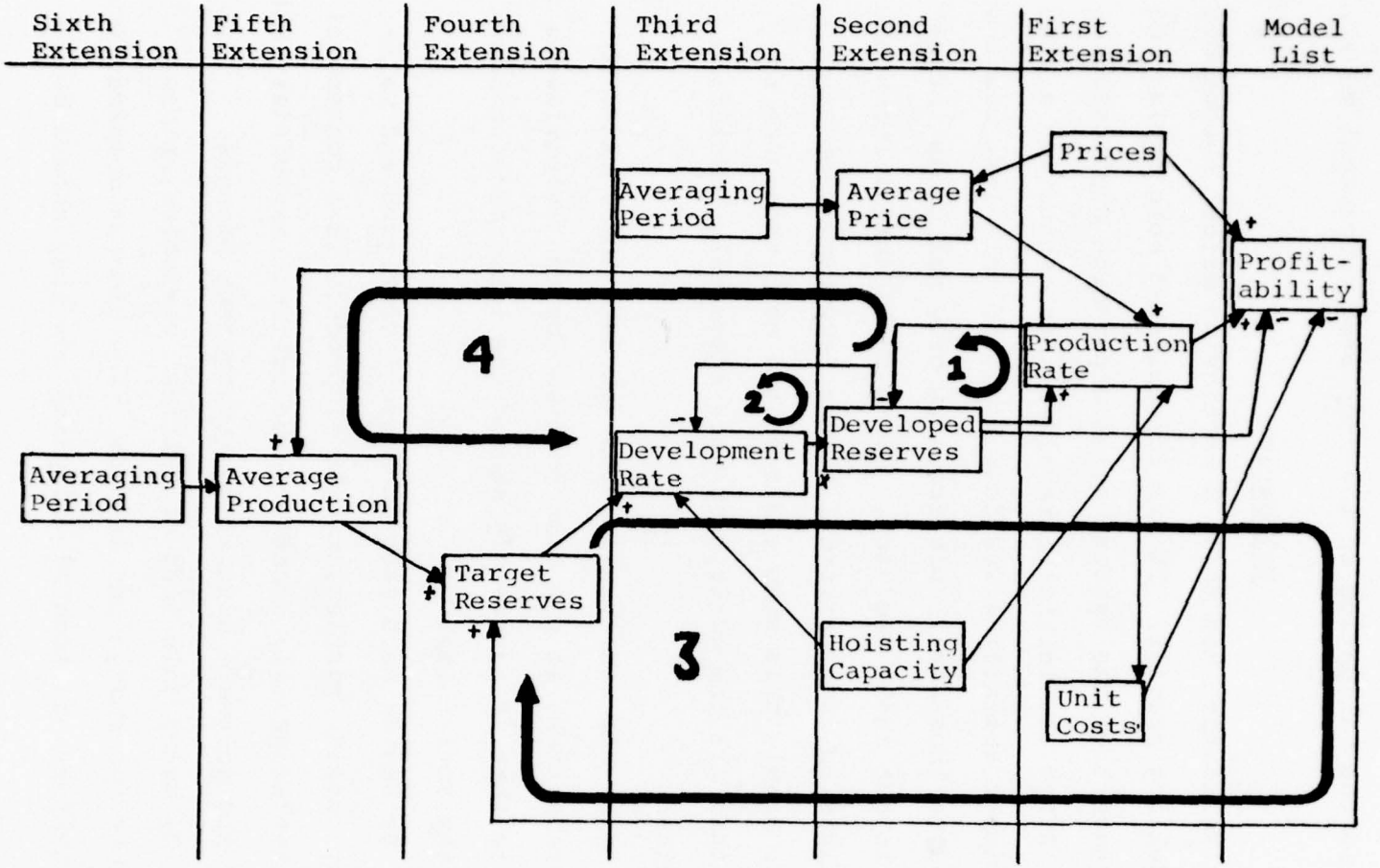


Fig. 10. An example of feedback loops in an influence diagram (9:76)

the process is completed. The remaining task, at this point, is the validation of the completed conceptual model.

Validation

Validation had to be a continuing effort throughout the modelling process. It was accomplished both during and after Phase I of the systems science paradigm application.

After the initial interviews were conducted, a list of pertinent activities was compiled, and this list was then returned to those available from the original group interviewed to get their feedback. Since the identification of the activities is so critical to the development of an accurate model, this step was repeated more than once to ensure that all the activities were adequately identified and defined.

Once the conceptual model was developed, based on these activities, it was also reviewed by the interviewees to ensure that it adequately showed the interrelationships of all the activities.

By having each step in Phase I critiqued and validated by outside parties, the final product is a conceptual model that adequately describes the critical activities that make up the software acquisition management process.

By using this list of critical variables, recommendations for change can be made. The process compares how the activities are treated today and how they should be

treated to achieve the maximum management control of the system. This may lead to significant changes in current procedures or may simply require that a factor be monitored in the future even though it was neglected in the past.

This chapter has presented a step-by-step application of the systems science research theory to the development of a conceptual model that will help in improving management control of the software acquisition process.

> The essential importance to the manager himself of a cybernetic control system is that it automatically filters the vast amount of proliferating information about the world situation that is accessible, and can present him with that very small proportion which is of real importance [6:342].

The model that is developed in Chapter IV addresses this small proportion of important information and provides the software acquisition manager the control capability needed to insure consistent satisfaction of technical performance, schedule and cost objectives.

CHAPTER IV

CONCEPTUAL MODEL DEVELOPMENT

The essential importance to the manager himself of a cybernetic control system is that it automatically filters the vast amount of proliferating information about the world situation that is accessible, and can present him with that very small proportion which is of real importance.

— *Stafford Beer [3:342]*

As stated in Chapter I, most software for embedded computers is purchased as a part of a given subsystem for a new weapon system or a major modification to an existing weapon system. Seldom is it ever the one and only purpose of a major contract effort. Because of this low level of visibility that software has received in the past, there has never been a large push to develop an overall perspective for management of the software acquisition process (19:35-36).

Many times in the past, all that was done to manage the acquisition process was to monitor the progress of the contractor on each of the seven functional activities that occur in development of any software program. These seven functions, conceptualization, requirements definition, design, coding and checkout, testing, integration, and maintenance, shown previously in Chapter I, Figure 2, are the mechanical steps a developer goes through to convert

an idea into lines of code that will operate in some central processing unit. Monitoring these functions only does not really provide an overall perspective of how the software development is proceeding except in the area of a deadline schedule. Problems with cost factors for the life of the software and accurate specification attainment may well be lost in this type of management approach.

By looking at any system in its entirety, one can see not only the problems in certain parts of the system, but also the ultimate output of the system and how these outputs will interface with other systems. That is the purpose of Systems Dynamics and the ultimate goal of the model presented in this chapter. The variables that were determined to have a significant impact on the system are not necessarily tied to one specific function, but instead are generalized activities that can impact across the entire acquisition process. This approach insures that the entire system is looked at whenever a problem arises. Also, taking the systems approach to viewing the software acquisition process allows managers of the process to understand how policy dictated from above will impact their ability to get the software development accomplished within their given constraints of dollars, specifications, and time.

The model presented here, when finally computerized, will encompass the information now collected concerning how well the contractor is meeting the milestone

schedule for the seven functional steps of the development. This information will eventually be used as inputs into the variables of this conceptual model. These variables were not subdivided further for this conceptual model as too much detail at this point in development would have defeated the purpose of attaining a proper system perspective of the acquisition process.

Thus, the conceptual model presented here was developed to give the overall systems perspective needed by the manager to adequately control the process and to filter information needed by managers at higher levels. The model, as presented, does not provide finely detailed information that might be needed on a daily operating basis, but instead shows the impacts certain decisions or policies will have in the long-range stability of the software acquisition process.

To completely understand the model as presented, one must be familiar with the interrelationships of all the activities that are involved in the process. The next sections discuss the activities found by this research project, the definitions that were used for development of the model, and an explanation of the interrelationships of the activities within the conceptual model itself.

Data Collection

Through all of the interviews conducted in this project, twenty-eight activities were determined to have significant impacts on the successful acquisition of embedded computer software. These twenty-eight could have been broken down into a much larger list but it was felt that sufficient detail was achieved with these activities to meet the purpose of the model being developed. Those activities identified are:

1. Early involvement of AFLC personnel
2. Unique support requirements definition timing
3. The amount of government-furnished software (26)
4. The age of the software being modified (when applicable)
5. Timing of the operational requirements definition
6. Allowed development time*
7. Accuracy of operational requirements definition
8. Standardization of code developed
9. Support software available for development
10. Core size of the processor to be used (26)
11. Timing cycle (26)
12. Other hardware constraints*
13. Difficulty factor*
14. Requirement for transportability
15. Degree of development entropy (25)
16. User involvement in development

17. SPO/AFPRO expertise
18. Contractor expertise
19. External influences
20. Planning for reprogramming capability
21. Computed development time*
22. Computer resources required for development
23. Criticality of software being developed (26)
24. Test/verification requirement timing
25. Risk analysis*
26. Timely and complete documentation
27. Verification and validation*
28. Formal reviews and audits (15)

The activities above, when appropriate, have been footnoted to show when one individual originally was responsible for their identification or if they were derived from the literature review portion of this research project. Those activities shown with an asterisk were derived by the authors from experience in software development or by subdividing an activity, identified during the interviews, for purposes of model clarity and closure. The remaining activities on the list were contributed and corraborated by all of those interviewed in AFLC, AFSC, and AFALD. Whenever a list such as this is compiled, it should contain the researchers' operational definitions of each activity to ensure that all concerned are in accord as to exactly what is meant by the titles applied to the activities. Since

these activities must eventually have some vehicle by which they can be measured or judged, it is encumbent on the initial researchers to specify, if at all possible, what the appropriate vehicles are. The definitions used in this research, and the measurement vehicles that were possible to determine, are presented with each phase of the model's development.

Conceptual Model

The conceptual model of the software acquisition management process developed as a result of this research is presented in Figure 11. This model presents a conceptualization of the overall architecture that is required to bring under control the Software Acquisition Management Discipline. The purpose of the model is to explain the variations in software development and to enable the manager to study the problem of devising improved control strategies.

The influence diagramming technique builds the complete model in distinct steps starting with the model list, the supplementary list, and then each required extension until closure is attained. The explanation of the model follows this same building block technique.

To aid in understanding why certain influence line signs were used, arrows have been added to show the desired directions of activities when the outputs of the model are

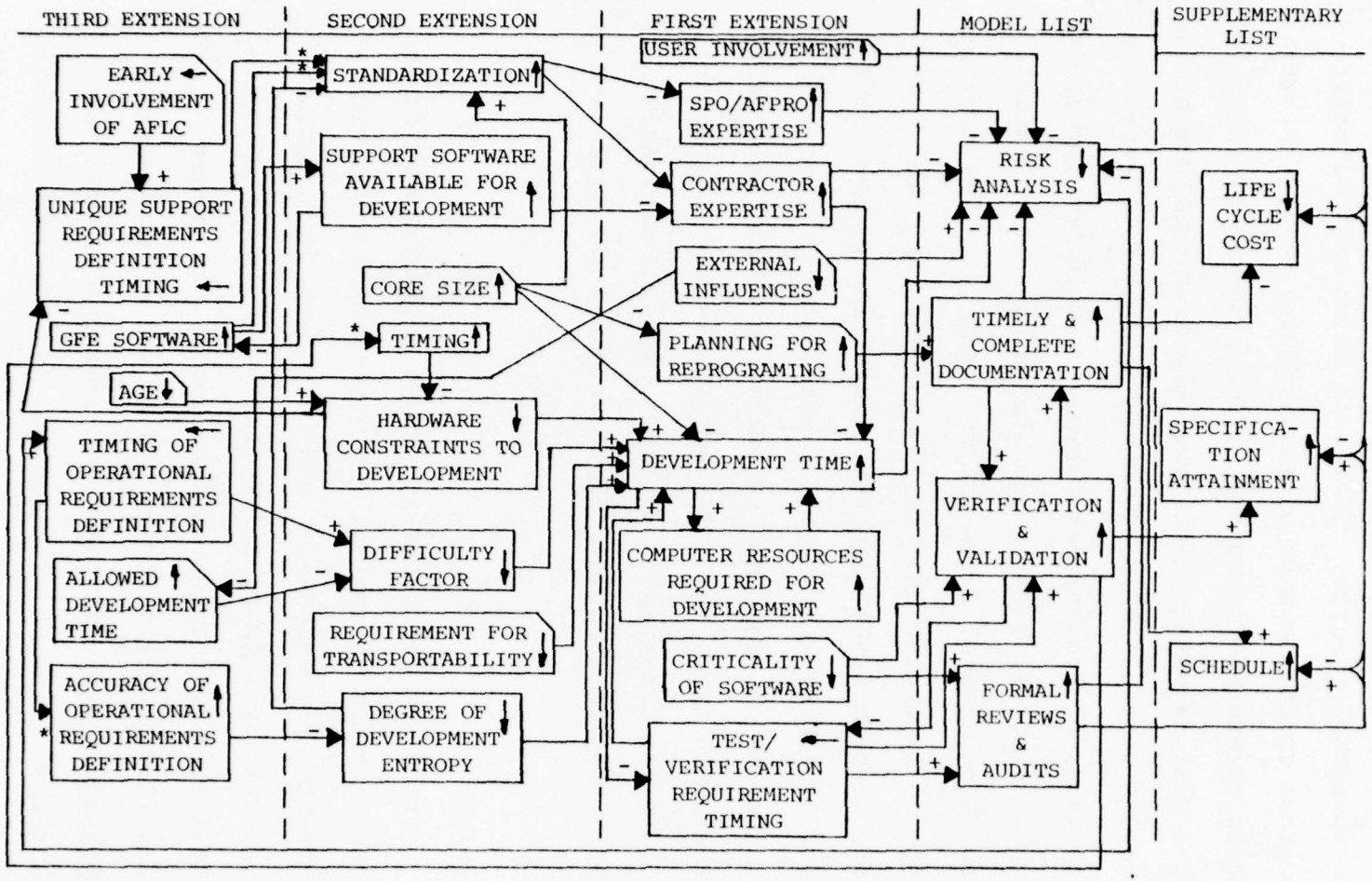


Fig. 11. A conceptual model of the software acquisition management process utilizing the influence diagramming technique

within acceptable limits. These are shown as horizontal or vertical arrows in each of the activity boxes. A vertical arrow upward indicates the activity should be increasing toward a most favorable level when the model outputs are as desired. A vertical arrow downward indicates the activity rating scale, or measurement vehicle, should be approaching zero when the model outputs are within the desired stable ranges. A horizontal arrow in any activity box associated with timing indicates that it is most desirable for that activity to be brought into the acquisition process at the earliest feasible time to assure stability.

To aid in grasping the extensive interrelationships of these activities, the model was subdivided into extension lists for the purpose of discussion and explanation. The activities in the model list extension are discussed first.

Model List

The model list contains those activities which the model is intended to control (9:72). The four activities that comprise the model list extension and their definitions as used in this research effort are presented below. Also presented is a brief explanation of the arrows included within each activity box in the model diagram.

1. Risk Analysis--uncertainty of management parameters expressed in terms of manpower, dollars, and schedule.

(↓) indicates that the smaller risk involved in the system development the greater the probability of completing the project successfully.

2. Documentation

Descriptive--the documentation that explains the inputs, outputs, and purpose of each module (or subroutine) being developed.

User--that documentation that will describe the application of the software to the eventual user of the computer system. It should explain the specific output that can be expected from a specific input.

(↑) More complete documentation will improve the manager's ability to control the program, and future program usability will be increased.

3. Validation/Verification--a review of available documentation to assess logic and thoroughness. After coding is completed a review is conducted to insure system specifications have been met and that performance is satisfactory in the mission environment. Modules must be validated prior to integration into the system and any time changes are made.

(↑) As verification and validation efforts increase the probability of the system performing as desired will also increase.

4. Formal Reviews/Audits--audits are conducted to verify compliance with specifications and other appropriate

contractual guidance. "[It] forces an orderly development of contractor software and provides the Air Force with a visibility of the managerial and technical activities of the contractors [3:23]."

(↑) When reviews and audits become more detailed and formal in their analysis of contractor activities, the manager's confidence of having a successful program will increase.

Figure 12 has taken the model list out of the complete influence diagram and focuses only on these elements and their relationships.

The reduction of risk is a required activity if a manager is to improve his or her control over the software acquisition process. In the model list one sees that the other elements act in concert with risk analysis in the development of improved control strategies. Notice the direct interrelationship of Documentation and Validation and Verification. The documentation is required to effect Validation and Verification, yet the Validation and Verification activities can result in a need for improved documentation to continue the process.

Supplementary List

Although the supplementary list is not a part of the system, it is used in the model to indicate system

MODEL LIST

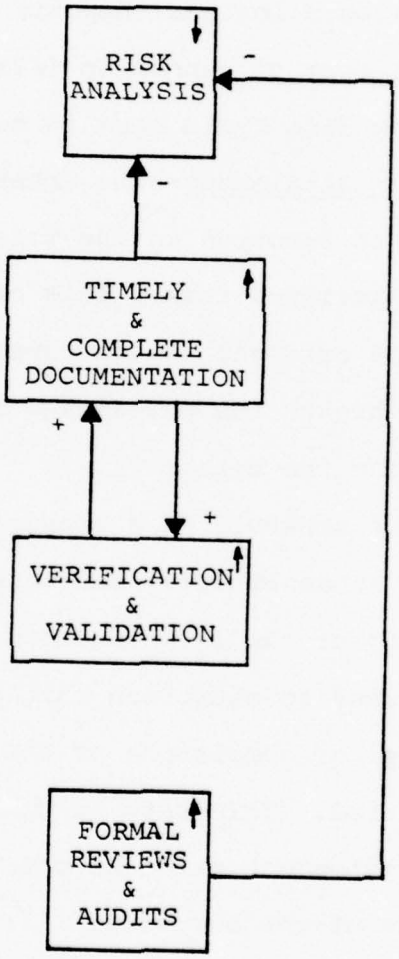


Fig. 12. The Model List column of the conceptual model

performance in an attempt to evaluate various proposed control policies. The three elements of the supplementary list are:

1. Life Cycle Costs--"Those costs, including direct, indirect, recurring, and nonrecurring costs associated with a system's research, development, production, and deployment (operation and support) that are incurred as the total cost of ownership [12:110]."

   (↓) A low life cycle cost is most desirable.

2. Specification Attainment--the extent to which the system is able to function in the mission environment and perform its assigned task. This can be measured as a percentage of original design capability.

   (↑) The higher the percentage of specification attainment the better.

3. Schedule--the schedule is a proposed rate of delivery of modules, subassemblies, and assemblies that is to occur throughout the life of the development program. One method used to establish this schedule is to estimate the rate of completion of the components that are to be delivered. This rate is computed using the expected development time and the total object code requirements of the component (27:17).

   The extent to which the program is on schedule can be evaluated by comparing the total object code

developed to the proposed amount of code that should have been developed.

(↑) The greater the amount of object code developed, especially if it is greater than the proposed amount on a given date, the greater the probability that the program will be completed on schedule.

These three variables reflect the ultimate indication of system performance by evaluating whether the project does what it is supposed to do, when it is needed, at the lowest possible life cycle cost.

Figure 13 shows the supplementary list and the model list and the manner in which the variables are interrelated. This figure shows the direct relationship between Validation and Verification and Specification Attainment. As the Validation and Verification effort increases the quality of the product being produced, the probability that that product will meet the required specifications also increases. Note that documentation does not directly affect whether or not a product meets specifications, but rather through Risk Analysis and Validation and Verification documentation improves the probability of meeting specifications.

Documentation, on the other hand, does directly impact life cycle cost through documentation articles that reduce maintenance costs later in a system's life, and

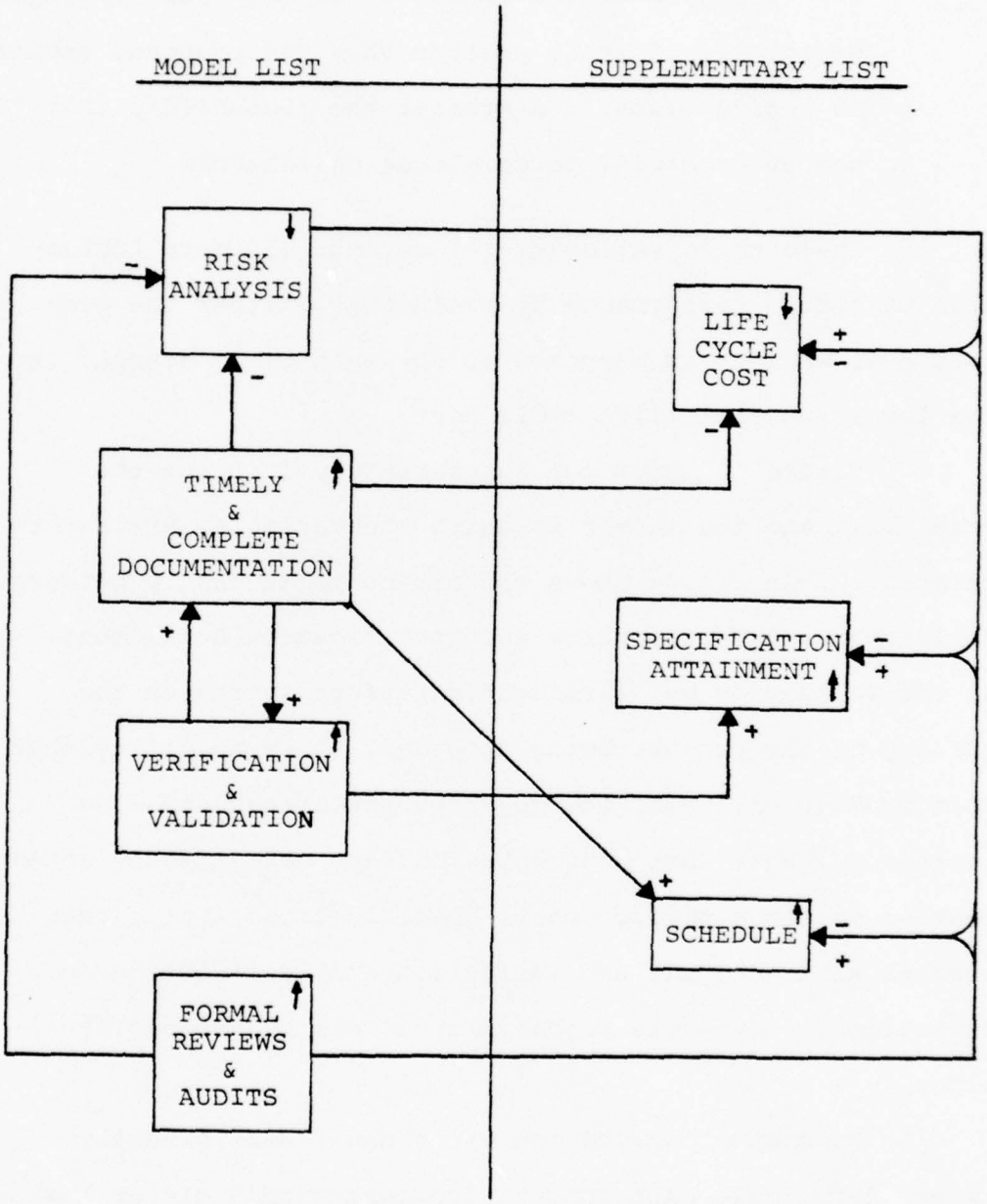


Fig. 13. The Model List and Supplementary List columns of the conceptual model

schedule through documentation articles that provide the manager with assessment capabilities to judge the progress of the development program.

Risk Analysis and Formal Review and Audits affect all three supplementary list variables by providing the visibility necessary to reduce uncertainty and to evaluate the activities of the contractor.

First Extension

The first extension contains those variables that directly affect the variables in the model list. The elements of the first extension and their definitions are:

1. User Involvement in Operations Requirement Definition (I)--the point in the DSARC cycle at which the user has active inputs into the system (and the software subsystems) being acquired.

   (↑) The more active the user is in defining the operational requirements, the more likely the end product will do what it was designed to do.

2. SPO/AFPRO Expertise--the average of the number of years' experience of the personnel with a software background assigned to the SPO and/or AFPRO.

   (↑) SPO/AFPRO familiarity with the software development process is directly related to quality of output, reliability of the system, and operation of the development program.

3. Contractor Expertise--average of the number of years experience of the programmers in specific applications areas similar to the proposed type of software development project.

   (↑) It is more desirable to have a contractor with previous experience in a similar program than a new contractor with no previous experience.

4. External Influences (I)--the extent to which the acquisition program is under scrutiny from outside established acquisition channels. A highly political weapon system may face more changes directed from outside sources than some of the lower priority programs.

   (↓) Less involvement of outside sources will lead to a more stable development program. The ideal situation would exist if no external pressure were placed on the program manager.

5. Planning for Reprogramming Capability (I)--the degree to which it is recognized that the software module under development may have to be significantly modified at some time in the future.

   (↑) Realization early in the development process that a software module may need the capability to be reprogrammed at some future time will allow this capability to be built in, reducing costly follow-on redevelopment efforts.

6. Development Time--the design time (or the time to reach Initial Operating Capability) of a large software project.

   (*) Development times are a result of the size of the system being developed. These times can range from one or two months to two to five years (25:23).

7. Computer Resources Required for Development--determination of the availability and number of CPU/wall-time hours, by machine type, that will be required to support the development effort, and the accessibility of the programmers to the problem center and the services offered.

   (↓) It is desired to provide the maximum degree of accessibility of the people to the services offered in the problem center. This can best be accomplished by having the center as close as possible (28:59).

8. Criticality of the Software Being Developed (I)--recognition of the ultimate use of the output of the software module and how its relative importance will affect other factors in the development and operations cycles; i.e., flight control software will require more test/verification and development time, and have more hardware constraints than will a software module for maintenance support of a simulator processor.

(↓) The less critical the output of the software, the more stable the development process since the pressure for perfection is reduced. The manager cannot control this as it is an exogenous input.

9. Test/Verification Requirement Timing--early recognition of the total testing required to verify the validity of the software module and ensuring that schedules reflect adequate time to react to the results of this process.

(←) The earliest possible realization of the need for specific test is desirable. Early recognition will hopefully lead to planning for complete test/verification procedures and for an adequate time period in which to accomplish these tests.

The "(I)" following certain variables listed above indicates those activities which are exogenous inputs to the system.

Figure 14 shows the relationship of the First Extension activities to those activities in the model list extension. The input activities are denoted in this figure by boxes which have the upper right corner cut on a diagonal.

Early involvement of the using activity in the development of the operational requirements of a proposed system will greatly reduce the risk of acquiring a system that is not capable of meeting its intended use. The

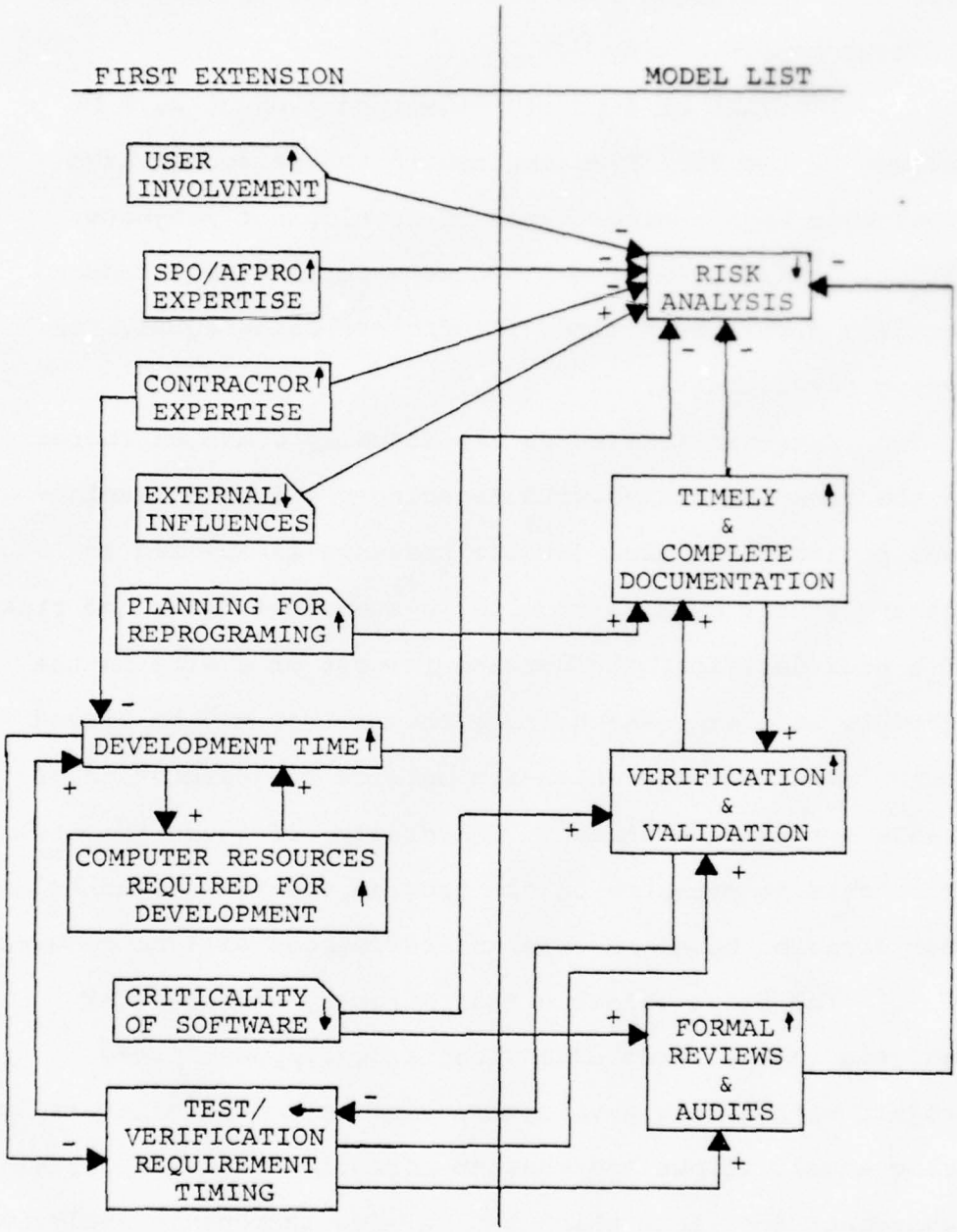


Fig. 14. The First Extension and Model List columns of the conceptual model

user must be brought into the development process at the earliest possible time.

The risk of a poorly developed product will be reduced if the SPO/AFPRO and contractor personnel have experience with similar types of development projects. [Additionally, greater contractor expertise will reduce required development time, all factors being equal, for system development.]

External influences may actually cause an increase in the risk associated with managing a software development project. As more outside pressure is applied to achieve faster results or alter system functions, the risk of a poor decision, an increase in cost or a slip in the schedule will increase because the manager may be forced to implement actions which are outside the parameters of stable system development. Conversely, if there is little or no outside pressure on the program the risk of making a poor decision based on external influences will be reduced.

The final variable that directly affects risk analysis in the first extension is development time. A project with a long development time, all other factors being equal, allows the manager more time to reach certain management decisions which may be more thoroughly analyzed and therefore will reduce the associated risk of making an error in judgement.

A factor which affects and is affected by development time is Computer Resources Required for Development. As development time increases, the requirement for development computer resources also increases to meet the need for additional programmer development activities. Additionally, if requirement for computer resources for development increases but cannot be met, then there will be an associated increase in development time as programmers are forced to wait for access to the available resources.

If it is recognized early in the development process that a software module may require modification at a later date, additional documentation will be required during the development effort to insure the future capability to modify the module as needed.

The criticality of the software being developed will determine the amount of Validation and Verification and the number of Reviews and Audits that are required during the program. For example, a software module for a flight control system will require closer Reviews and more Validation and Verification than a module that will be used in a flight simulator.

Early recognition of the total amount of time required to accomplish the Validation and Verification will impact on development time, the degree of Validation and Verification to be employed, and the amount of program review that will be implemented. Additionally, the

estimated development time will impact on the point in the development program that Validation and Verification activities must be identified and started. Also, the amount of Validation and Verification that will be required to insure successful program completion will determine when Validation and Verification activities must begin if the program is to be accomplished on schedule.

At this point the first test for closure is accomplished to determine if the model is complete. The influence diagram, as presented so far, does not exhibit closure, so attention is moved to the second extension.

Second Extension

The Second Extension contains those variables which most directly affect the variables that are presented in the First Extension of the influence diagram. Those eight variables are presented below along with their respective definitions.

1. Standardization--the extent to which the module under development can be designed under "top-down" structural programming rules. This can be measured as a percentage of the number of modules written following the guidelines out of the total number of modules in the software development package.

   (↑) The higher the percentage, the better. Standardized code is easier, and therefore cheaper, to maintain.

2. Support Software Available for Development--determine if all the required compilers, translators, debugging aids, assemblers, generators, etc., are currently available or if they will have to be developed along with the operational software module.

   (↑) When none of the support software is required to be developed concurrently with the operational software, all efforts can be focused on the object software. Programmers can begin to debug new code immediately.

3. Core Size (I)--the amount of memory available normally measured in thousands of bytes such as 16K bytes = 12K bits of information.

   (↑) Highly sophisticated code, with interwoven routines is not necessary when core size is not a constraint to development. Anything that removes some constraint will lead to a more stable development program.

4. Timing--the cycle time of the programs; i.e., the length of time, in microseconds, allowed for the entire program to execute and interface with other system modules.

   (↑) The longer the timing allowed, the better. Ideally, this number would go high enough to actually be no constraint at all.

5. Hardware Constraints to Development--those requirements placed on the software development effort due to weight and volume constraints of the hardware and/or schedule of hardware development.

   (↓) When there are no hardware constraints, including core size and timing, code does not have to be written to specifically meet these constraints. Zero constraints is the best situation possible.

6. Difficulty Factor--a scaled rating of the programming effort required to produce the software. Scale ranges from very easy to very hard.

   (↓) This factor will never reach zero since all software requires some logical development. The more basic the computer function to be performed, the lower this factor will be.

7. Requirement for Transportability (I)--determination of the requirement that the software module be capable of executing in more than one CPU in sufficient time to design this requirement into the module initially.

   (↓) When it can be assured that the module will never have to be executed on any other CPU, this will go to zero. When it may have to execute on more than one CPU, the design must take that fact into account. This may indeed complicate the design.

8. Degree of Development Entropy

Stand Alone Software--software developed for mainframe (or mini) computers to be used for support functions. Not normally constrained by any physical space limitations.

Rebuild (Extensive Modifications)--programs already in operation that are to be used as a basis for software with a new function. Usually reinstalled in the original machine.

New Software with Interface--software being developed for new systems that will have to interface with other processors in the overall major system.

(↓) A factor that influences the amount of development time that will be required for the program. The various classes require different development times. As this factor approaches zero, risk will decrease as required development time will be reduced.

Figure 15 displays the interrelationships of the Validation and Verification activity from the model list, the activities of the First Extension, and the activities of the Second Extension. The validation and verification activity was included in this figure because a feedback loop exists between it and the timing activity in the second extension.

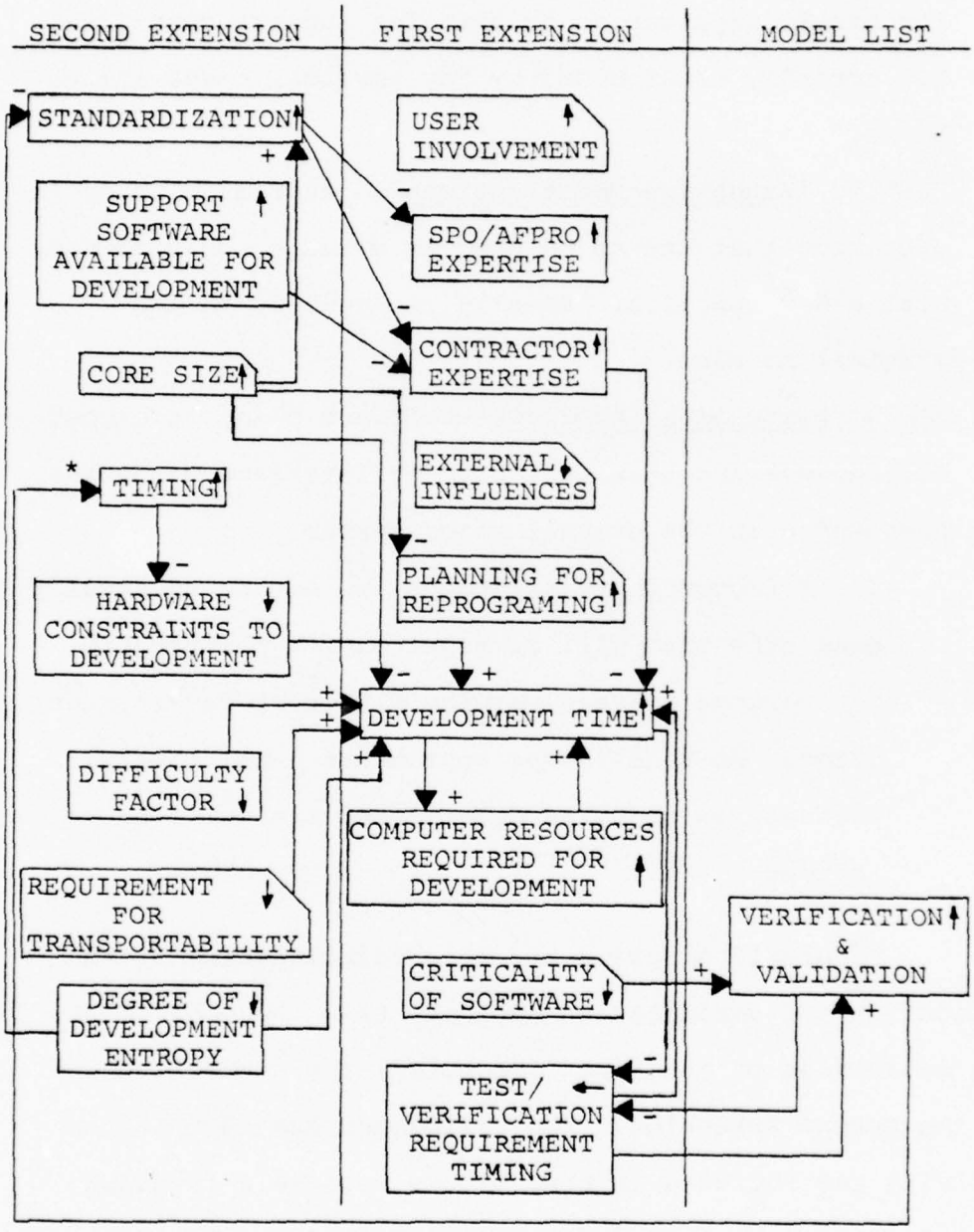


Fig. 15. The Second and First Extensions and an excerpt from the Model List columns of the conceptual model

The ability to apply standardized programming rules in a development project will offset, to some degree, a lack of expertise on a specific project in both the SPO/AFPRO organization and in the contractor's organization.

Support software availability during development may reduce the amount of expertise required of a contractor if he does not have to develop the support software in addition to the operational software modules.

The core size of the computer mainframe impacts three areas, standardization, planning for reprogramming and development time.

If core size is not critical, a greater degree of programming standardization may be employed; whereas if core size is critical, the software modules may have to be designed to conform to the sizing constraints. This will not allow for designs that meet standardized coding structures.

Secondly, the availability of extra core space will enhance efforts to allow for future reprogramming needs.

Finally, if the programmers have adequate core size to write straightforward code, the development time will be reduced significantly by decreasing the potential for errors as the complexity of the design can be reduced. This also reduces the amount of debugging time required generally.

Timing interfaces directly with hardware constraints to development. If information is required at a certain rate to operate a piece of hardware, steps must be taken to insure the software is capable of functioning properly in the required time to interface with the associated hardware.

Validation and Verification reviews system operation to insure the specifications have been met and that the system operates satisfactorily in the mission environment. This review includes an analysis of the timing criteria and how well it accomplishes the software/hardware interface.

Hardware Constraints to Development, Difficulty Factor, Requirement for Transportability and Degree of Development Entropy all have a direct impact on required development time. As any of these factors increase, they tend to increase development time as well.

Degree of Development Entropy also impacts on standardization. As the degree of entropy increases, the ability to apply standardized programming structures becomes less and less. For example, a program with the Rebuild type of Development Entropy may not be able to apply a great deal of standardized programming because new functions are required to fit into old equipment. In this case, core size may be limited or other constraints of the old hardware may impose unique programming restrictions on the software. However, Stand Alone Software can use a

great deal more standardized techniques because it is not normally constrained as are the other types of software programs.

Again, after all the variables have been listed in the Second Extension, the closure test is applied. Although one feedback loop has emerged across extensions, the model is still not closed and, therefore, the Third Extension is addressed.

Third Extension

Here again the process repeats itself. Those variables that directly affect the variables in the Second Extension are listed in the Third Extension and the influence lines are drawn in. The seven activities that make up the Third Extension are:

1. Unique Support Requirements Definition Timing--the point in the DSARC cycle at which it is realized that the hardware package under development will require totally new dedicated support software, not currently available, once the weapon system becomes operational.

   (+) If these requirements can be identified early in the program, more time will be available to develop these unique requirements. There also exists the possibility that other programs will be able to use the same support equipment or vice versa.

2. GFE Software--the number of computer programs supplied by the government to assist the contractor in his software development efforts.

   (↑) It is desirable to utilize as much existing software as possible to reduce costs and aid standardization if possible.

3. AGE (I)--the number of years the software being modified has been in the field.

   (↓) Older software is more difficult to modify and update; therefore, more acceptable results will be realized on newer pieces of software.

4. Timing of Operational Requirements Definition--the point in the DSARC cycle that it is recognized that significant software development will be required to meet the established mission requirements of the system being acquired.

   (←) Defining operational requirements as early as possible in the acquisition cycle will increase the probability of successfully completing the development project.

5. Allowed Development Time (I)--the time in months allowed for software module development prior to system interface testing. Not necessarily equal to Development Time.

   (↑) Allowed Development Time needs to be at least equal to development time and cannot be reduced to

a level below development time without increasing the difficulty of the development effort.

6. Accuracy of Operational Requirements Definition--the degree to which the established mission requirements of the system being acquired reflect, in sufficient detail to insure adequate software development, the actual use to which the weapon system will be employed.

   (↑) The more accurate the operational requirements definition, the greater the probability the completed system will perform as the user intended.

7. Early Involvement of AFLC (I)--the point in the DSARC cycle at which the AFLC personnel who will be responsible for maintenance of the software are brought into the development of the software being acquired.

   (↓) It is desirable to involve AFLC in the acquisition cycle at the earliest possible time to insure adequate planning for maintenance activities on a system life cycle basis.

Figure 16 contains the third and second extensions, the External Influences Variable from the First Extension and the Risk Analysis and Validation and Verification variables from the model list. The three excerpted variables are, again, added to clarify the developing feedback loops within the model.

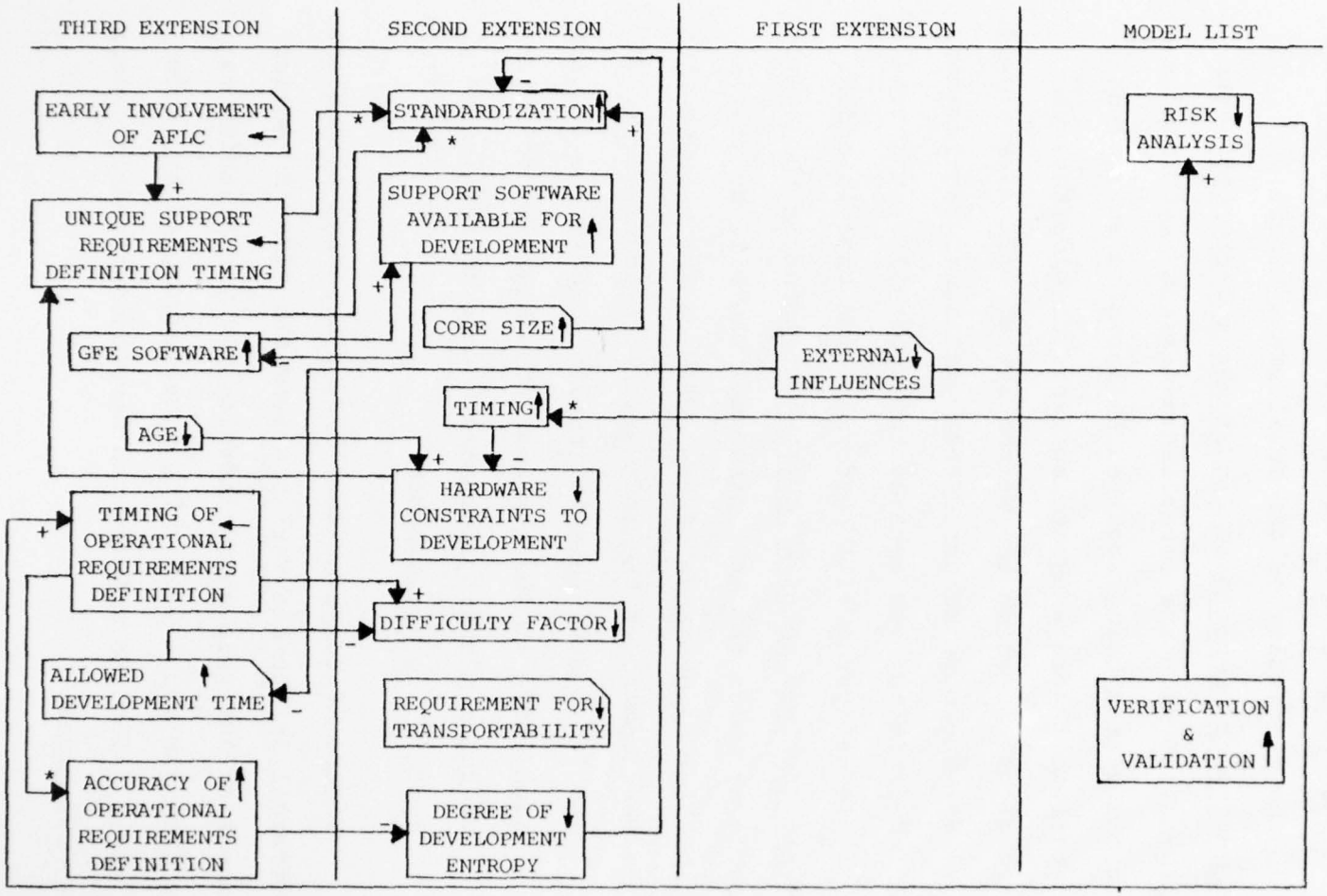


Fig. 16. The Second and Third Extension columns and excerpts from the First Extension and Model List columns of the conceptual model

Early determination of unique support requirements will impact on the number of standardized programming rules that can be employed during development activities. Also, any hardware constraints that are placed on the development project will determine how early unique support software will be required.

If government furnished software modules are available for the development program, the degree of standardization within the project may be increased if the software modules fit directly into the project. However, if these modules must be modified in order to work in the project, the degree of standardization may actually decline.

Also, if the government can furnish additional support software to the project, the total amount of software available will increase. However, if there is an adequate amount of contractor supplied support software then the amount of GFE software needed will be reduced.

Age places an additional constraint on this software/hardware interface in a development project. The older a module is, the more there is a chance of inaccurate or inadequate documentation being available that describes the module. The module may also have been written in an early version of a language that is no longer widely used, or known, and it may no longer have the development support software available that was around when it was written.

Early operational requirements definitions reduce the difficulty factor by allowing more time for complete development. By determining early in the acquisition cycle what the component is to do, risk of system failure is reduced because less pressure is felt by all parties in their efforts to complete development. This early operational requirements definition will improve the accuracy of the definition if the needs of the user are sufficiently detailed to allow for a realistic assessment of mission requirements; however, if the timing is too early and all the mission details have not been worked out, the accuracy of the definition will suffer as changes will be required to correct deficiencies and oversights.

The accuracy of the operational requirements definition will have an impact on the Degree of Development Entropy insofar as the type of software development program will be determined by the operational requirements imposed by the user. Those modules whose functions do not change many times, do not suffer from the inefficiency that the design and redesign cycle can promote.

Allowed development time is most directly affected by external influences. For example, pressure may be directly applied to get a new weapon system operational by a specific date. If this allowed development time is less than the actual development time needed to complete the

project, the difficulty factor is increased and a ripple effect rolls through the entire model.

By involving AFLC in the development process very early, unique support requirements may be addressed from a complete life cycle approach and total system procurement packages may be developed more effectively. Their inputs concerning the maintainability of the software model flow through the First and Second Extensions and hopefully will aid in reducing the life cycle cost of the software.

After all the variables have been entered and linked, closure is again attempted. This time, closure is attained and the influence diagram, which is now also a model, is complete.

Summary of the Model

There are two basic objectives to a System Dynamics analysis no matter which dynamic system it is applied to. The first objective is to explain the system's behavior in terms of its structure and policies. The second objective is to suggest changes that will lead to improved behavior of the system or, alternatively, suggest changes to structure and policy in a small system which will enable it to survive, or even take advantage of what the larger system does to it (9:19).

The model just discussed was the first step in achieving the first objective of influence diagramming.

It shows, not too surprisingly, that the activities that impact the software acquisition management process are extensively interwoven. It does not show, at this stage of development, the technical relationships of the five functional steps in development of software as the model has maintained the systems view of the process. This viewpoint cannot be ignored by the manager of a system if he is to maintain proper control of his process (29:238-243).

To meet the second objective of influence diagramming, the final computerized model must be developed and implemented. This facet of the model development is discussed further in the Recommendations section of Chapter V.

The final step in development of the conceptual model is validation. The remaining portion of this chapter will cover this critical phase.

Validation

Validation of a model, one of the most difficult tasks in systems science research, is "the process by which we establish sufficient confidence in a model to be prepared to use it for some particular purpose [9:181]."

The validation effort for the conceptual model of the software acquisition management process that is presented in this chapter began with repeated interviews during each phase of variable identification and definition, and model building. Two principle questions were addressed

during the validation process to insure the highest degree of confidence in the validity of the proposed model. The first question asked during model evaluation was, "Does the model include the necessary elements of the software acquisition process to enable it to effect the system's behavior?" Initial interviews and literature reviews provided the first set of variables that were to be included in the model. Through the process of return interviews with AFLC, AFSC and AFALD personnel, reevaluation, more interviews, and further reevaluation of the variable list, sufficient confidence was developed in the list containing the final twenty-eight variables to permit development of the conceptual model itself.

Once the model was developed, the second question that was asked to further strengthen the validity of the model was, "Is there a correspondence between the proposed model of the software acquisition management process and the system itself?" The final test of this question will come at a time when inputs, which have been made to improve system behavior, have had time to act and their results can be analyzed. However, until such time as actual data is available for analysis, "we feel the best test of confidence [validity] is the knowledge that the model has been carefully built up in conjunction with management [9:184]." Because of this relationship with the management people of AFSC, AFLC, and AFALD, the researchers are confident

that there is indeed a correspondence between the model and the system it represents.

Additional validation questions will have to be addressed as the conceptual model is further developed into the recommended computerized simulation model.

The conceptual model that is presented in this research effort was developed in order for the manager to know what management changes to make to the software acquisition process to improve its behavior, and for that manager to gain an understanding of the complex interrelationships that exist throughout the software acquisition system. This model meets these purposes and toward these purposes the model is valid.

CHAPTER V

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

We are at our human finest, dancing with our minds, when there are more choices than two. Sometimes there are ten, even twenty different ways to go, all but one bound to be wrong, and the richness of selection in such situations can lift us onto totally new ground. This process is called exploration and is based on human fallibility.

— *Lewis Thomas [31:39]*

This chapter summarizes the research objectives and the research questions, and presents the conclusions and recommendations of this research effort.

Research Objectives

The research objectives established for this research project were to:

1. Investigate the current software acquisition management process with emphasis on the elements that are creating problems in the areas of employment and control.

2. Identify and define the variables (activities) which must be included in a conceptual model of the software acquisition management process.

3. Develop a conceptual model of a viable software acquisition management process which explains the system's behavior in terms of the activities and their interrelationships.

Research Objective One

As a result of this research effort, this research team now has a much better appreciation for the complexity of the current Air Force software acquisition process. This research objective was established to analyze that complexity through an in-depth literature review and a series of personal interviews with the people in AFSC, AFLC, and AFALD that are directly involved in the software acquisition management process. This research objective provided the base from which the remainder of the research project was built.

Research Objective Two

The intention of research objective two was to use the information collected as a result of the literature reviews and interviews to identify and define the variables that were to be included in the final conceptual model of the software acquisition management process. The final list of variables and their definitions are a culmination of the information gathered, the researchers' own inputs and numerous return trips to those interviewed for additional feedback as the list was being developed. This research objective was accomplished in Chapter IV under the heading Conceptual Model.

Research Objective Three

The results of objectives one and two were combined and applied to research objective three. Once an understanding of the current software acquisition management process was developed and the necessary variables needed for the conceptual model were identified and defined, the research team felt confident that a conceptual model--the first step toward a computerized simulation model--of the process could be developed. This objective was achieved in Chapter IV--Development of a Conceptual Model of the Software Acquisition Management Process.

Research Questions

This research effort was conducted to apply the principles of management cybernetics to the software acquisition management process. The research was initiated with the awareness that these principles are not being applied either conceptually or practically. It is hoped that by presenting a working example of these principles, they will be applied to management problems to a great extent in the future.

The research questions that guided this research effort were:

1. What are the elements within the software acquisition management process, as currently employed, which are contributing to the problems that now exist?

2. What are the variables (activities) that must be included in a model of the software acquisition management process?

3. Can a conceptual model be developed that can accurately portray the dynamic behavior of the software acquisition management process?

Research Question One

One of the key elements within the software acquisition management process that is contributing to the current problems is the sharing of information among the various sections of the acquisition system. This element was found repeatedly in literature and personal interviews. One ESD technical report summarized the problem this way:

> Management "lessons learned" are not captured and disseminated regularly enough to benefit many new software acquisitions . . . , so that innovation for a new program may really be a repeat of the mistakes of a prior program [11:10].

Another element that is contributing to the current problems is the fact that AFLC is not brought into the acquisition cycle early enough to have an impact on reducing the life cycle cost of a system (26). The software acquisition manager in the SPO needs to know what resources the AFLC units will need to support the software once it is put in the field. These needs must be addressed and planned for early even though adequate funds may not be immediately available for their acquisition.

A third element that is contributing to the current problems is a lack of careful planning for future reprogramming needs. If the future need for reprogramming is not addressed, inadequate core size, insufficient documentation and excess life cycle costs will result which will greatly increase the problems of reprogramming in the future when the need occurs.

Research Question Two

The variables that were included in the conceptual model in response to the second research question are identified in Chapter IV under the heading Conceptual Model, and will not be reiterated here. These variables were selected as a result of the combined efforts of this research team, the personal interviews that were conducted, and the many volumes of literature that were reviewed in the early stages of this research effort.

The final list of elements was not the result of setting a goal to identify and define a predetermined number of variables, but rather was the result of a great deal of in-depth analysis and study of the software acquisition management process. The initial list contained twenty elements, some of which were combined into a single element; one was eliminated as being irrelevant and four others were added for the first time. Additional reviews were conducted before the list was finalized.

The importance of this method of building the list of elements to be included in the conceptual model is that a synergism took place between numerous people each with a special insight into the process.

Research Question Three

The answer to the third research question is, "Yes, a conceptual model can be developed that accurately portrays the dynamic behavior of the software acquisition management process." This model was developed in Chapter IV under the heading Conceptual Model.

This conceptual model portrays the relationships that exist within the software acquisition management process. It provides the manager with a tool to develop an understanding of the nature of the system and ways to develop stability within the process. This model can, and should be, applied during all phases of a system's life cycle.

The relationships portrayed in the model provide a mechanism that the manager can use to evaluate actual achievements and relate to the overall acquisition strategy.

The importance of this research is that it presents a conceptual model of the relationships of those elements that are necessary to achieve management control of the software acquisition management process, and it provides the conceptual base that is fundamental to the development of a mathematical control model of the process.

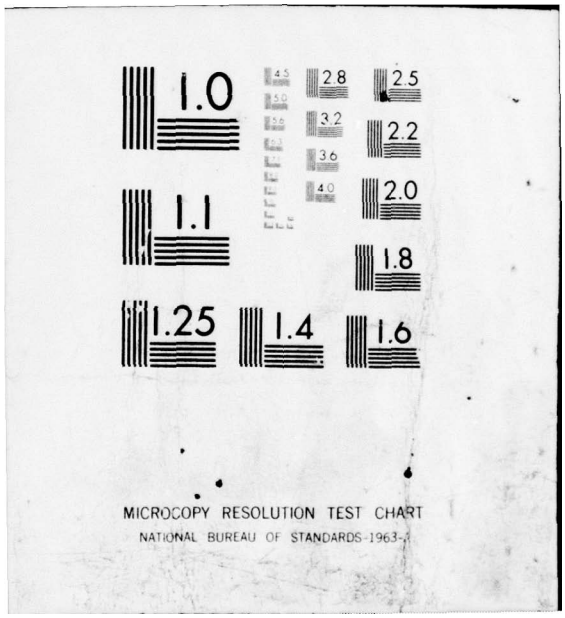
AD-A076 946 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL--ETC F/G 5/2
MANAGEMENT CYBERNETICS: AN APPLICATION TO THE DEVELOPMENT OF A --ETC(U)
SEP 79 R E PESCHKE , M L SHERRILL

UNCLASSIFIED AFIT-LSSR-2-79B NL

2 OF 2
AD A076946



END
DATE FILMED
12-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
10.0
11.2
12.5
14.0
16.0
18.0
20.0
22.4
25.0
28.0
32.0
36.0
40.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Conclusions

The conclusions reached by this research are:

1. That the principles of management cybernetics can be applied to a complex process such as software acquisition management within the United States Air Force.

2. That the application of management cybernetics is a significant step toward the development of an overall perspective or architecture to the software acquisition management discipline.

3. That life cycle cost technology is not being adequately applied to software acquisition. Many instances were discovered, in discussions with AFLC and AFALD personnel, that affirmed this conclusion. Their general feeling is that if a program is fairly easily maintained during its operational life, it is more in the nature of luck than detailed planning.

Granted, life cycle cost techniques, as used in the acquisition of aircraft tires, do not directly transfer to embedded computer software. However, the theory and main objectives of life cycle costing could be applied more than they are now. The conceptual model presented here has purposely included those activities that should help to reduce, or at least identify, areas where costs can be reduced for the life of a software program.

4. That the existing mathematical knowledge and computer technology is adequate to develop the mathematical model necessary for complete program implementation.

5. That the final computerized model must possess the capability to produce specific outputs dependent upon the type of software being acquired. This does not mean that there needs to be a model for each of the five basic types of embedded computer software (i.e., Automatic Test Equipment (ATE), Operational Flight Programs (OFP), Aircrew Training Devices (ATD), Electronic Warfare (EW), and Communication, Electronics, and Meteorology (CEM)) (26). Rather the basic model, conceptually presented here, must be able to be modified very easily to present the output required by those managers involved in acquisition of any of the five categories. This is foreseen, at this time, as possibly involving changing the scaling factors of certain variables to relate to their importance to a specific category. It also may entail breaking these activities down further into their contributing variables in order to accurately represent their relationship to one of the five categories.

In any case, one model will not accurately reflect the variations found in the acquisition of the various types of embedded computer software programs, nor are five separate models necessary since the acquisition processes

are not significantly different enough to require five separate models.

6. That the inclusion of AFLC software personnel earlier in the acquisition cycle will help to reduce total life cycle costs. The personnel at WR-ALC concluded that this could be accomplished by as little as one systems analyst for each Air Logistics Center that will eventually be involved in maintenance of the software acquired.

Life cycle costs will be reduced even if the software is not developed with maintainability as a priority since problem solving and implementing changes is easier when some person in the maintenance operation is aware of the logic involved in the design of the program. It is also very helpful to have someone familiar with the modifications a program has gone through when a new change is required. It will be incumbent upon AFLC to adequately compensate these personnel to insure they remain with the program as long as feasibly possible.

Recommendations

The recommendations of this research study are:

1. That a study be conducted to establish quantitative parameters for the variables that were identified and defined. This effort should be applied toward the goal of developing a mathematical model based on the conceptualization developed in this research study.

2. That a study be conducted to develop and validate the computer programs necessary to implement a control model of the software acquisition management process. This model must, however, be easy to use, be credible within the management discipline, and be able to produce useful, timely information.

3. That any efforts to develop a mathematical control model involve the inputs of the AFSC/AFLC/AFALD personnel who will use the model once it has been developed.

4. That a study be conducted to investigate the approval process of the Data Automation Requests (DARs).

One of the problems discussed by many of those interviewed was the trouble they have in getting approval of a DAR. Although the DAR is generally used for hardware acquisition approval, it can have an eventual impact on software development in that the hardware to be used will dictate the software requirements. Besides the specific technical requirements the hardware generates, the delay in its acquisition, brought on by the lengthy DAR process, can delay development of the software (15).

This also should be investigated from the standpoint of an overall view of data processing within the Air Force and DOD. One example was found in AFLC that points out the overall problem. The managers of the automated test equipment at Warner-Robins ALC can purchase exotic electronics equipment without any external approval. But

to replace or add a microprocessor, which costs as little as 25 percent of some other equipment in the same test bed, they must submit a DAR that has to be approved by the AFLC Comptroller organization. This approval route does not make sense. In other cases outside of AFLC, the General Accounting Office and the General Services Administration can get involved and even have the power to veto. Since the cost of hardware is becoming less and less of the total program cost for data processing, this type of approval route needs to be researched thoroughly to determine its impact on the costs of software development.

The conclusions and recommendations, and the entire research effort represented by this thesis can be summed up by this quote from Stafford Beer,

> The identification of mechanisms led to concepts of law; and we have just seen how the laws of cybernetics govern our view of modelling. But models are useless unless they are applied; we go into action in a managerial situation armed with a model which--hopefully--embodies the laws and applies the mechanisms [8:115-116].

APPENDICES

APPENDIX A

INTRODUCING THE CORPORATE PARADIGM

The Five-Tier Hierarchy of Control

The cybernetic paradigm, as proposed by Stafford Beer, is shown in Figure A-1. This figure is for the entire firm or organization and does not show that this firm is also a division of a higher niveau such as the industry to which it belongs. Each one of the divisions shown in this figure is equivalent to the board level of the firm, but for the systems below (4:199-212). This is the law of recursiveness that is discussed in Chapter II. In that each niveau of an organization is cybernetically the same as those above and below, the niveau that contains the specific problem at hand can be identified and modeled or researched fairly easily. The paradigm in Figure A-1 was derived by Beer through an analogy of the control functions of any viable organization and the control physiology of the human nervous system.

Through the process of homomorphic modelling, as discussed in Chapter II, Beer used the five levels of control in the human nervous system to map onto the control functions needed in managing an organization (4:117-134). By studying the information flow to and from the brain, through these five physiological levels, the five systems of the corporate paradigm were developed.

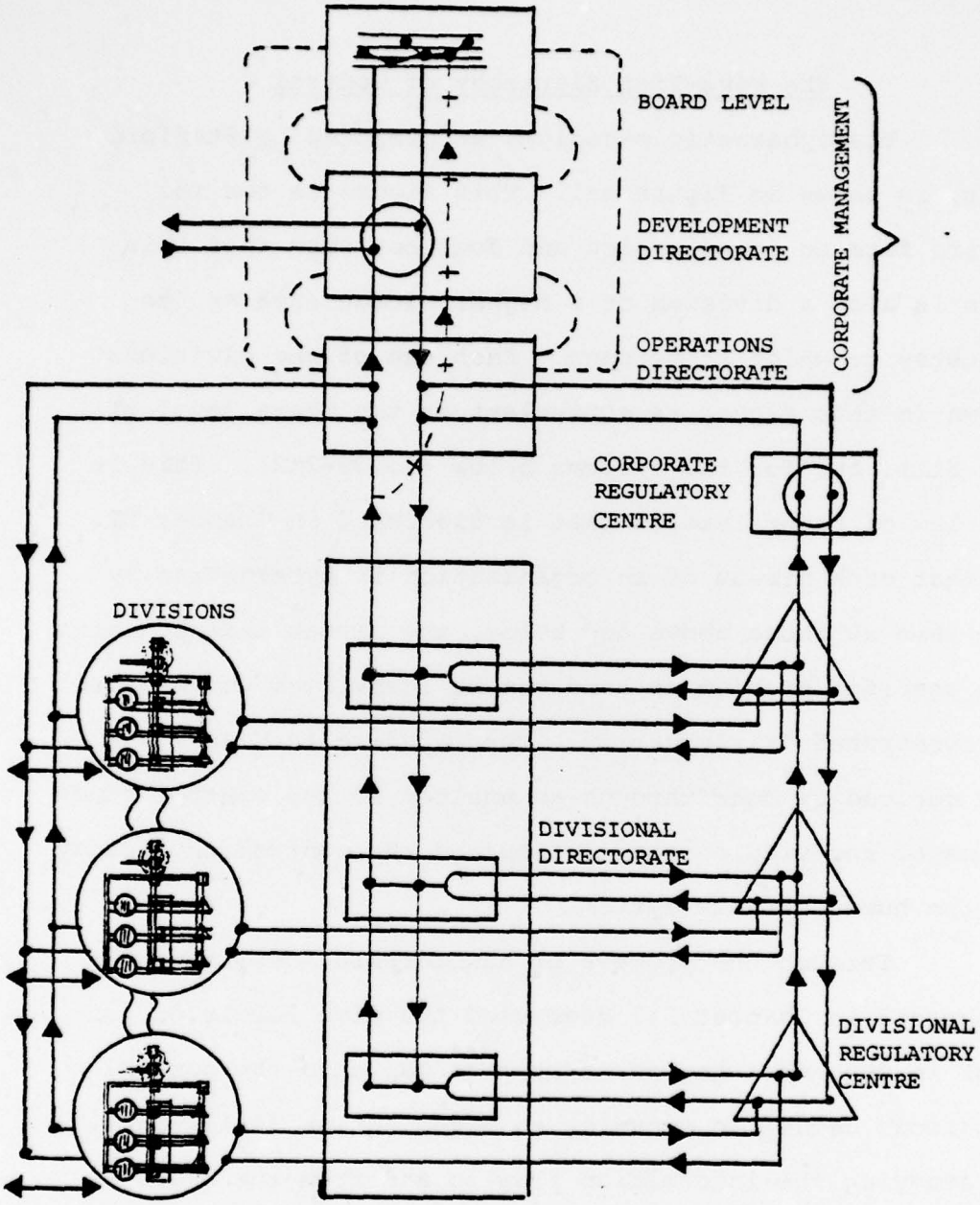


Fig. A-1. The Corporate Paradigm (4:199)

Figure A-2 shows an exploded view of the brain and how it can be divided into five control echelons. Although five echelons is an arbitrary choice, that number of levels identifies the major functional differences involved and still keeps the classifications from being too complex (4:129). The horizontal axis from the nerve endings through to the spinal vertebral level is the same as the horizontal axis shown in Figure A-1. Here is where the production of an organization is done and is where each organ in the body performs its unique function. The information flow through the synapse and the spinal vertebral levels is collected in the spinal column. System II of the corporate paradigm performs the same function. This information is sent to the lowest portion of the brain, shown in Figure A-2 as Control Echelon III, which regulates the autonomous internal functions of the body and yet it acts as an information source to the rest of the brain as well. System III serves the same dual function within the corporation. The portion of the brain in Control Echelon IV is connected directly to the environment through the sensors of the eyes and ears. System IV, the staff level function of the corporation, performs the same functions in that organization. Finally, in the human body there is the cerebral cortex, the portion of the brain that handles foresight, recall, pattern-making, the powers of association, and the thinking process in general. The cerebral cortex is not

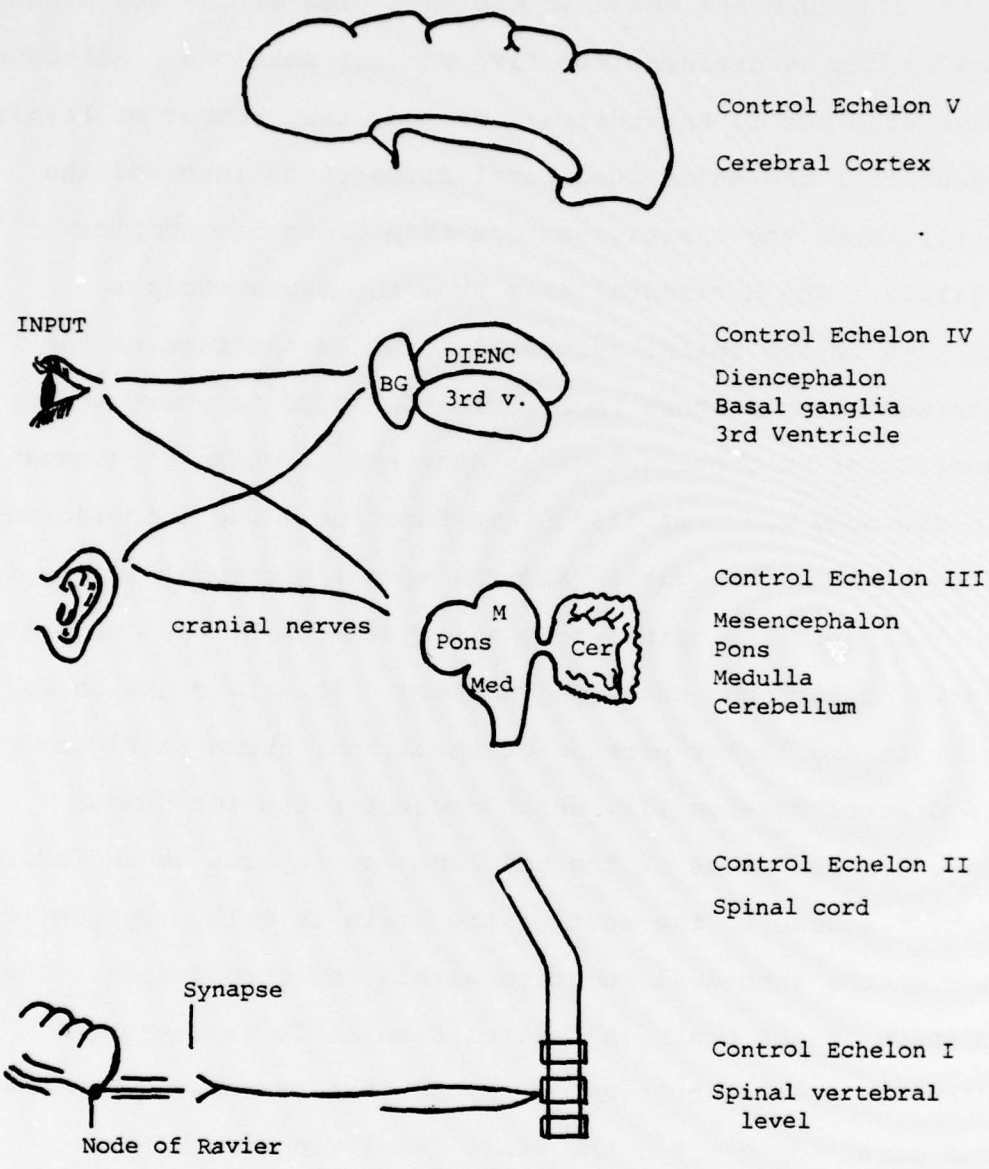


Fig. A-2. Exploded diagram of the brain showing classification as a five-tier hierarchy (4:129)

directly connected to the outside world. The same type of functions within the corporation are performed by the board of directors at the System V level. While they are not physically separated from the world, as is the cortex of the brain, they do provide the insight and intellect for the rest of the organization (4:117-134).

System I: Divisional Control

Figure A-3 shows the basic elements of the division level components. The relevant external world (REW) has an impact on the business operation (0) which uses a System One controller to maintain control (5:36). This level of the organization is looked on as almost totally autonomous by the firm's upper level management, but it is a complex day-to-day operation for the people involved.

This operation is normally where the output of the firm is produced. The System I manager is concerned with allocation of various resources among a range of required tasks and the calculation of the risk of some event happening. What is critical is the determination of what information is needed by the controller, which many times happens to be a computer, to effectively solve the problems of his division (5:34-36; 4:201-203).

System II: Integral Control

The division of Figure A-3 was shown to have inputs from its environment and its controller. Figure A-4 shows

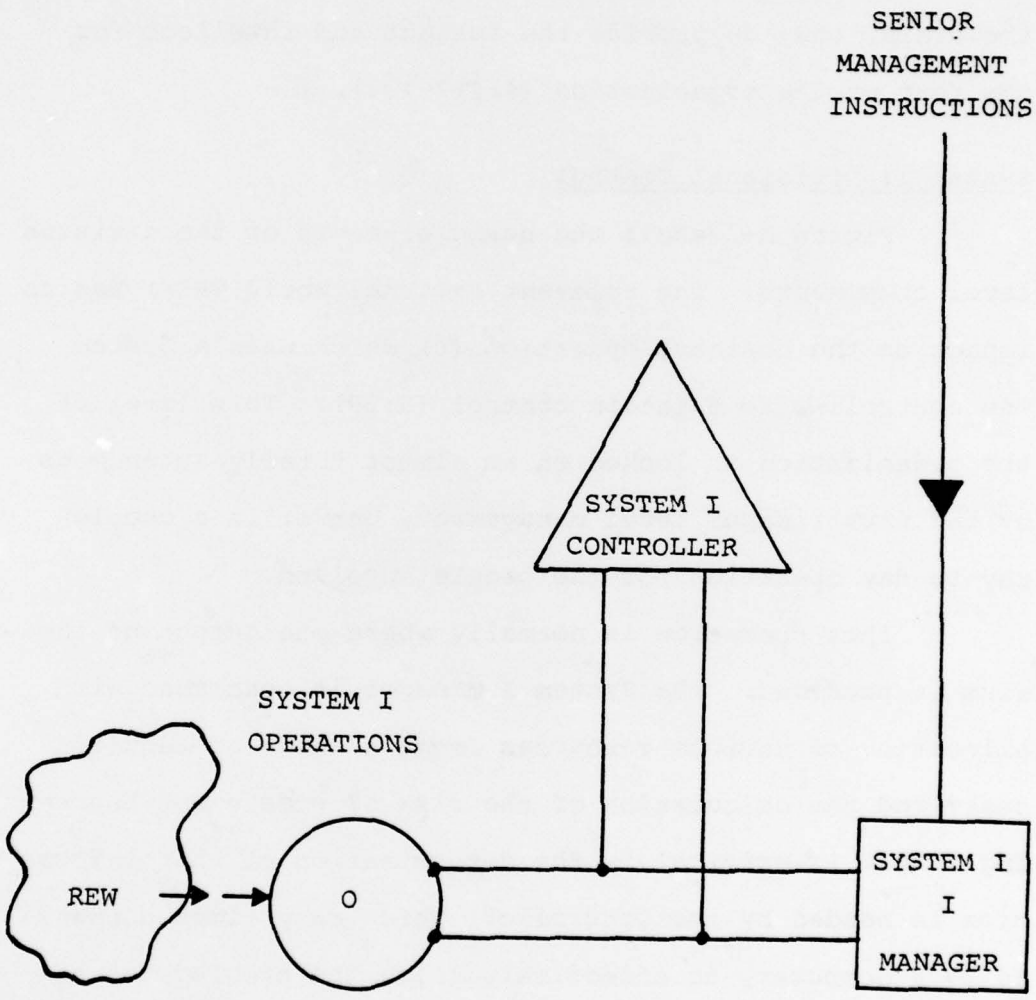


Fig. A-3. System I (5:36)

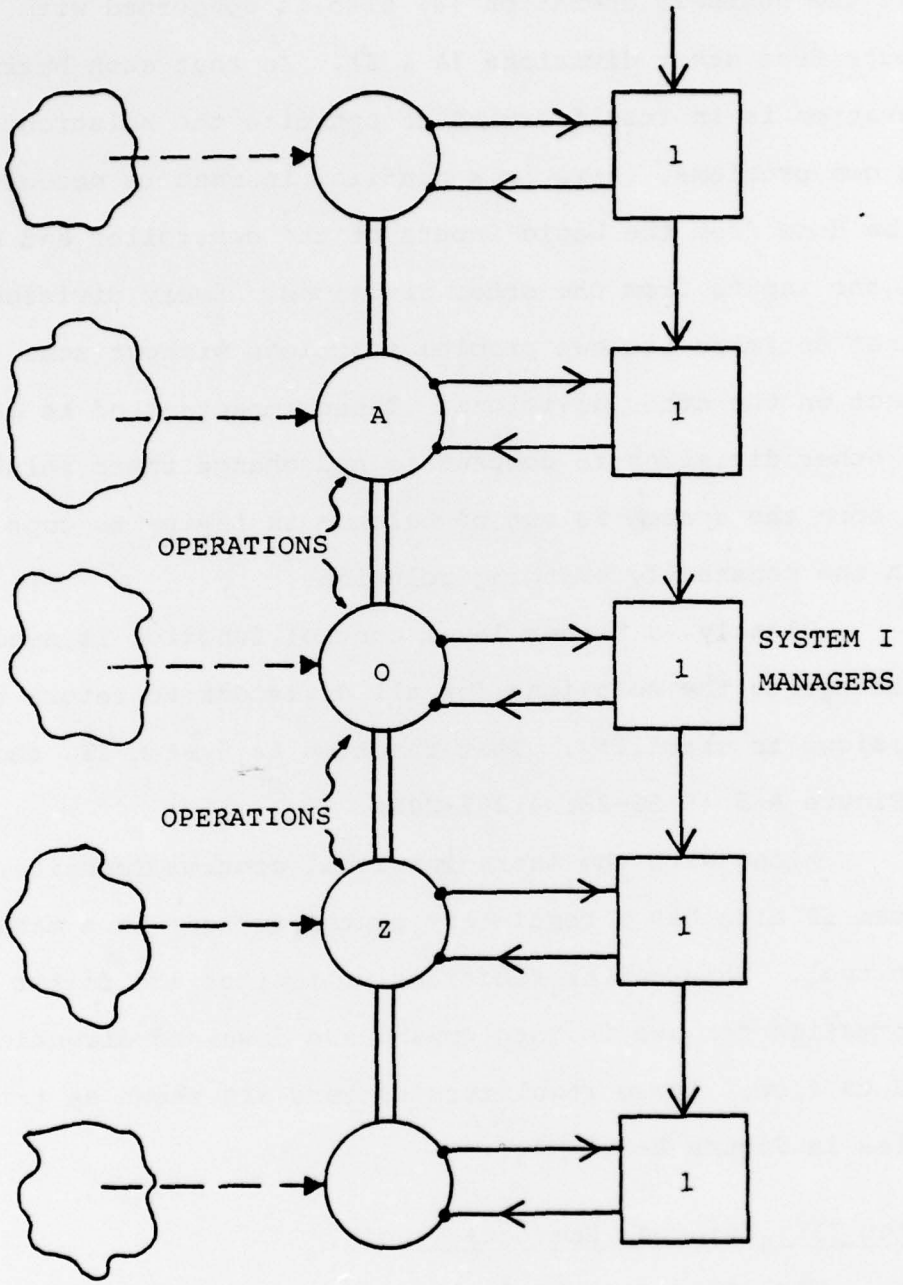


Fig. A-4. Activities of the Firm (5:36)

that the business operation (0) also is concerned with inputs from other divisions (A & Z). In that each business operation is in itself trying to optimize the solutions to its own problems, there is a conflict in what is perceived to be done from the basic inputs of the controller and REW, and the inputs from the other divisions. Every division cannot optimize its own problem solutions without some impact on the other divisions. These impacts tend to cause the other divisions to compensate and change their solutions and soon the system is out of balance in trying to cope with the constantly changing solutions.

Clearly, a higher level control function is needed to integrate the solutions for all divisions to return the divisions to stability. That function is System II, shown in Figure A-5 (5:36-38; 4:203-204).

Along with the intradivisional control function, System II also has a regulatory center for use as a management tool. This center functions to monitor and filter information for use in both upward and downward directions of data flow. These regulatory centers are shown as triangles in Figure A-5.

System III: Internal Homeostasis

The title for the System III position or function is that of operations directorate. This is the first level that is really involved with all levels of the firm.

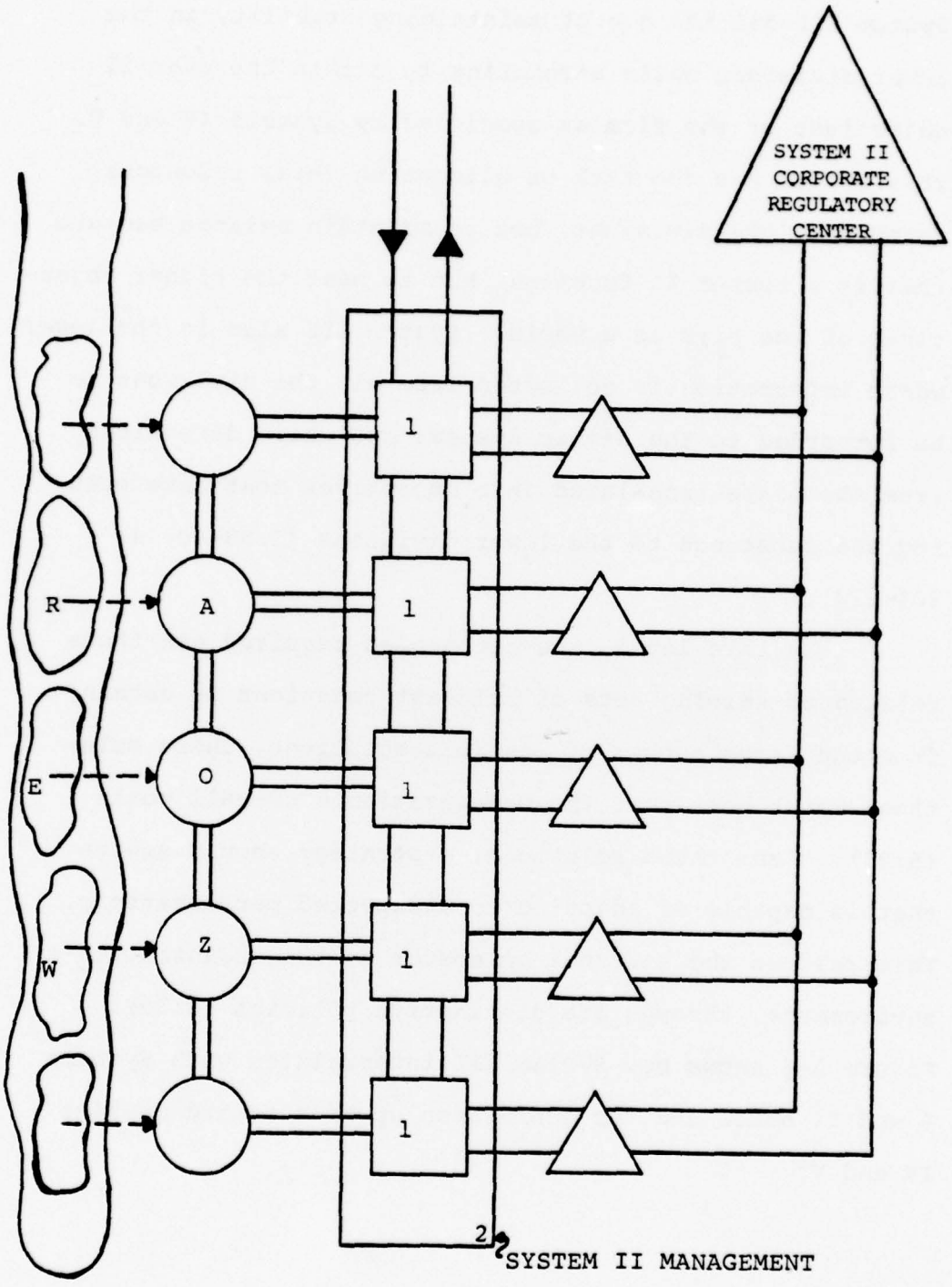


Fig. A-5. System II (5:36)

System III has the job of maintaining stability in the lower divisions while attempting to attain the overall objectives of the firm as specified by Systems IV and V. This system has the task of allocating total resources throughout the divisions, not to maintain balance because that is a System II function, but to meet the higher objectives of the firm as a whole. System III also is the level where information is collected from all the divisions to be forwarded to the higher systems and where directions from above are translated into objectives that have meaning and substance to the lower divisions (5:38-40; 4: 204-212).

At this level, the techniques required are those related to solving sets of relevant equations to obtain, from the large number of possible solutions, those solutions which best meet the organization's overall goals (5:39). From these solutions, a strategy should emerge that is capable of adapting to unexpected perturbations. This relates the internal homeostat to the organization's environment, through its distinctive policies (5:39). Figure A-6 shows how System III interrelates with Systems I and II below and the connection upwards toward Systems IV and V.

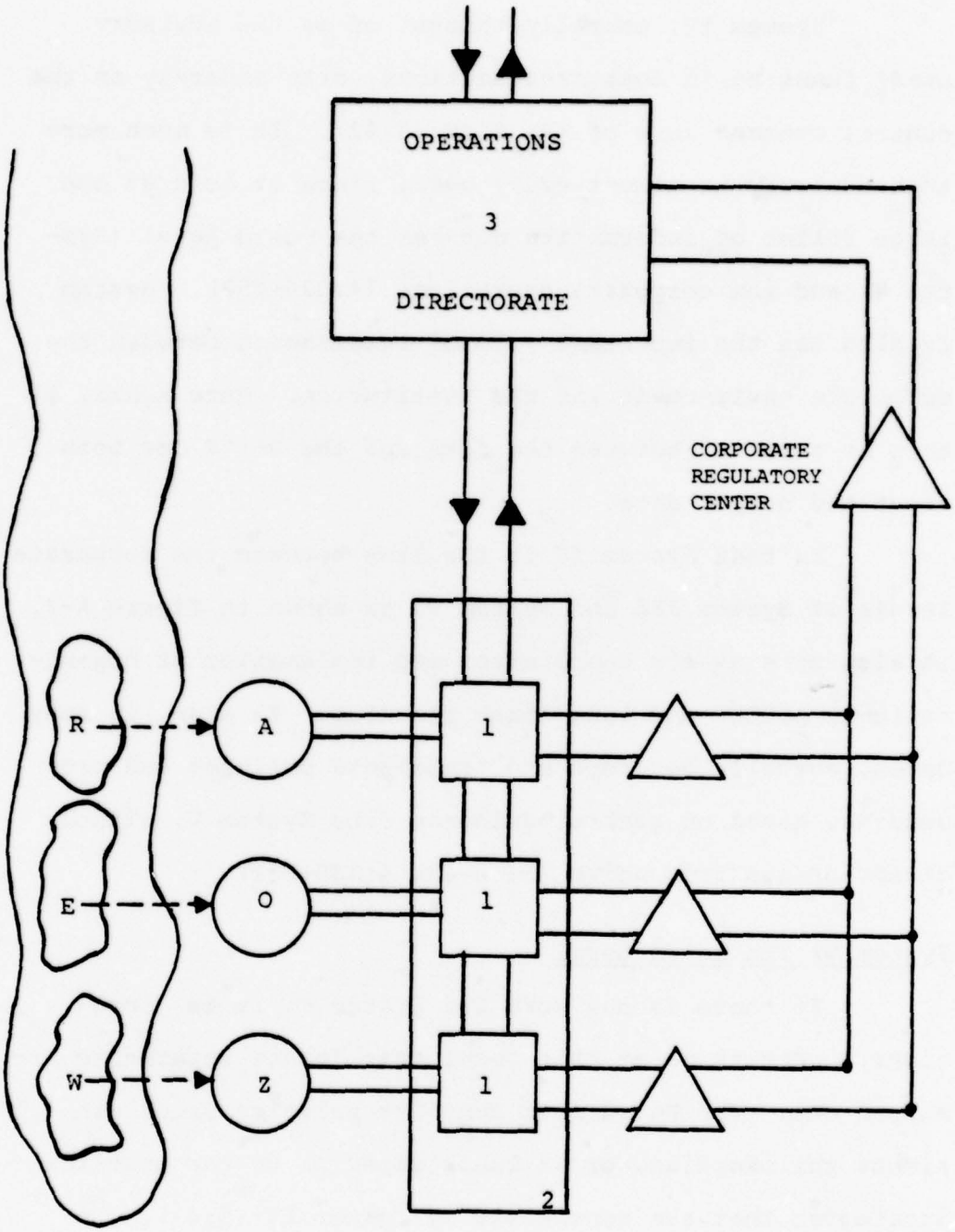


Fig. A-6. System III (5:40)

System IV: Development Directorate

System IV, normally thought of as the advisory staff function in most organizations, sits squarely on the control command axis of the firm (5:42). It is much more than advisory in almost every case, since it acts as one large filter of information between the board level (System V) and the corporate operations (4:230-252). System IV also has the important role of interfacing between the corporate environment and the institution. Here again, it acts as a filter between the firm and the world for both input and output data.

In that System IV is the link between the corporate levels of System III and System V, as shown in Figure A-7, it also acts as the coordinator and implementor of organizational policy and long-range planning. It also, in many cases, actually develops and implements policies and procedures, based on general guidance from System V, without direct orders from above (5:40-43; 4:230-252).

System V: The Board Level

If there is one word for System V, it is "foresight." For it is at this level that future strategies are mapped out. The Board must consider policies which are almost philosophies, or at least superior to the practical strategies that are considered by System IV (5:43). To accomplish this consideration, System V has direct inputs

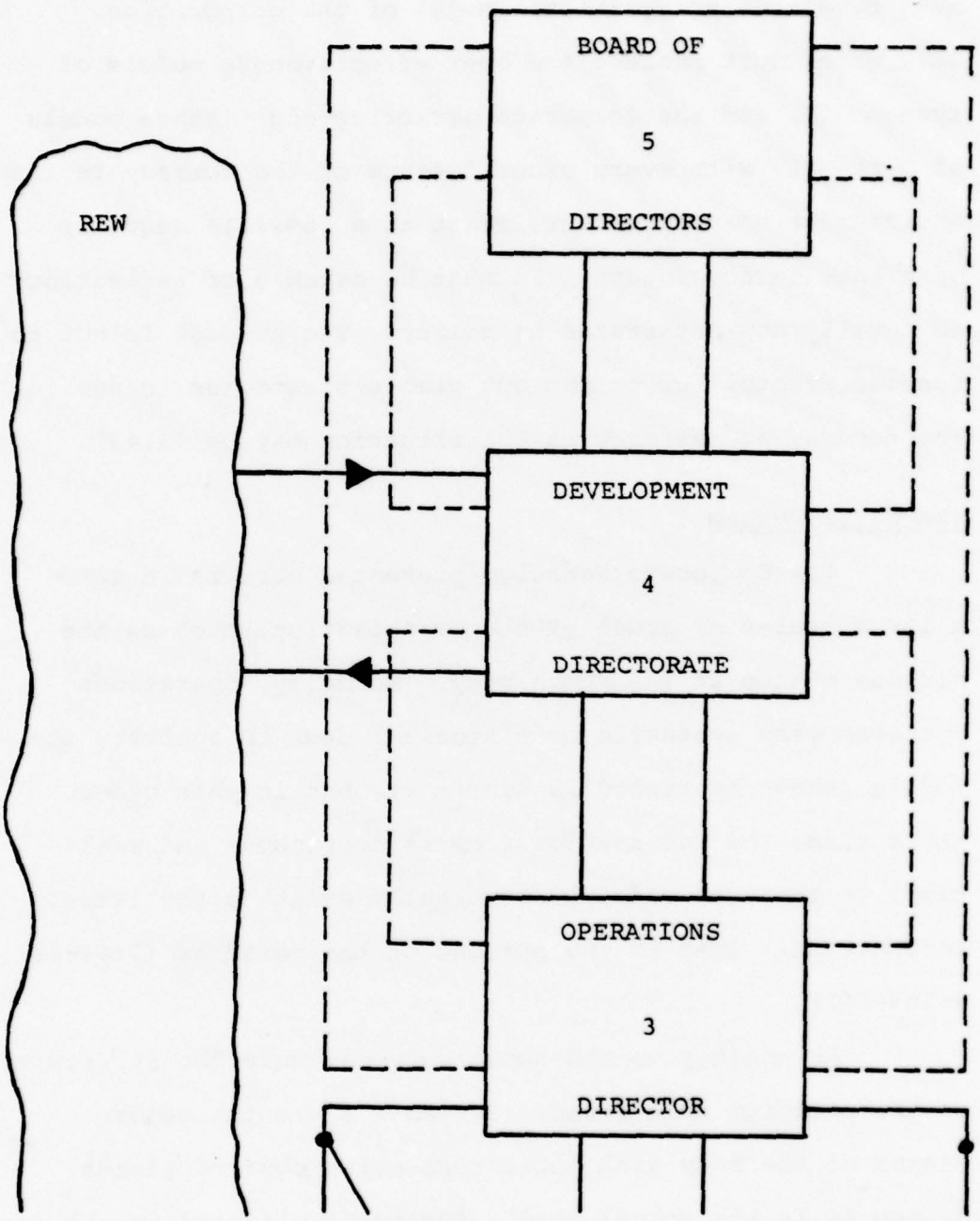


Fig. A-7. The Organizational Interface of System IV (5:41)

from System IV and some inputs from System III. It also must have some sort of total model of the corporation. This model must include the cost-effectiveness models of System III, and the corporate marketing and finance models of System IV with every other feature of the company in its environment that appears relevant to a possible sequence of events in the future. It must be capable of reflecting on totally new departures in policy. The purpose is not to foresee events, but to map out viable strategies to use in the future, no matter what the situation may be (5:43).

The Total System

The Corporate Paradigm presented here is in actuality a real-time model of the organization, much as the nervous system of the human body. Normally, operations research uses synthetic or historical data to activate its models (those mentioned in System V), but in this cybernetic firm, the information flow is continuous and real-time, so that the model is constantly updating the latest information. That is the purpose of the paradigm (5:44-46; 4:199-252).

An analogy to the human body may make the structure of the paradigm more clear. System I's are the major organs of the body with their respective control glands. System II is the spinal cord. System III located in the rear, lower portion of the brain is the autonomic nervous

system with its associated input and output systems of responses. System IV is the central portion of the brain that consolidates all the information from the rest of the body. System V is the cerebral cortex, the thinking center of the brain (5:44).

The argument, as stated by Stafford Beer, is then that, "Viable systems are organized like this whether they are physical, social, or economic [5:44-45]."

APPENDIX B

LIST OF INTERVIEW QUESTIONS

1. Do you agree that the spiraling cost of ECS is a major acquisition problem?
2. Which of the following would you say is the most pressing, urgent and significant when considering ECS acquisitions?
   a. Contracting methods
   b. Management techniques
   c. The state of the technical art
   d. Accurate statement of software requirements
3. When considering management techniques and methodologies, which of the following are problematic?
   a. Tracking the system's progress
   b. Defining the hardware and software requirements
   c. Understanding the hardware and software relationships at each phase of the system's acquisition cycle
   d. Defining the software product
   e. Validating and verifying the final product
   f. Defining and then implementing milestones for the ECS acquisition
4. Do you agree that if the problem of cost and management of ECS are to be solved a good place to start is with a clear understanding of the ECS acquisition process?

5. Would you characterize the preparation and training for ECS acquisition managers as extensive, or is it more of a learn-while-doing process?

6. Do you believe that life cycle considerations are normally included in the process of defining software requirements?

7. Meaningful management information is often unavailable when needed, because of a lack of consistent practices for feedback of software management information. Do you feel we could do better in this area if we required more exact reporting by the contractors?

8. Is hardware development and construction initiated so early in the program that software is often forced to accept changes (because of hardware problems) without appropriate engineering and design?

9. Do you believe that, in most cases, since software is uniquely different from hardware, the management schemes and procedures set up for hardware will not work for software?

10. It has been asserted that software as opposed to hardware lies on the critical path of most Embedded Computer Systems procurements. Do you believe it would be desirable to have the software analysis and design start earlier in the acquisition process than it does now?

11. Do you see the acquisition of an embedded computer system using a total system approach or is it generally treated separately?
12. Are the specifications normally drawn up so that a software package can be maintained organically without the help of a senior systems analyst intimately familiar with the program?
13. What variables do you see impacting on the costs and schedules of embedded computer software?

APPENDIX C

GLOSSARY OF TERMS

BIT--binary digits (BITs), the smallest unit of information understood by the computer, are the elements that reflect the states of the binary number system. Bits are organized into groups to represent symbols in the same fashion as dots and dashes in Morse Code (14:4).

CPU Time--the amount of time the Central Processing Unit (CPU) uses to complete each set of instructions. For example, it may take the machine only two seconds to execute a program, but, due to other operations the computer may be accomplishing at the same time, it takes it one minute to go from an input until the operator gets the required output (see Wall Time).

Cybernetics--"the science of communication and control in the animal and the machine. That is to say that cybernetics studies the flow of information round a system, and the way in which the information is used by the system as a means of controlling itself; it does this for animate and inanimate systems differently [6:254]."

Embedded Computer System--"An embedded computer system is a computer system that is integral to an electro-mechanical system such as a combat weapons system, aircraft, . . . , and the like. Embedded computer systems are considered different than Automatic Data Processing Systems (ADPS) primarily in the context of how they are developed, acquired and operated in a using system [22:4-8]."

Entropy--"the measure of a system's inexorable tendency to move from a less to a more probable state [4:306]." For living organisms entropy equates to death; for organizations maximum entropy equates to total disorder and a lack of all required information.

Homomorphic Model--a scientific model which involves a many-to-one correspondence onto which two different situations are mapped which defines the extent of structural identity within the situation without destroying necessary operational characteristics of the component elements.

Module--the smallest computer program unit that can be compiled or assembled (12:58).

Niveau--the term used to distinguish between levels of recursion within the organization structure in order to avoid possible confusion with the particular concept of "level" within a given niveau (5:82).

Paradigm--"An exemplar or pattern; a basic way of doing something recognizable beneath many superficial variations [4:307]."

PMRT--Program Management Responsibility Transfer. "The transfer of program management responsibility for a system (by series), or equipment (by designation) from the implementing command to the supporting command [13:58]."

Q-GERT--a method for graphically modelling systems in a manner that permits computer analysis. G-GERT augments GERT (Graphical Evaluation and Review Technique) with the addition of queueing and decision capabilities (24:vii).

Scientific Model--"a homomorphism onto which two different situations are mapped, and which actually defines the extent to which they are structurally identical [6:113]."

Software--"Software is the sum total of all programs, data, and routines. Frequently, software is defined to also include associated documentation, such as specifications, ICDs, manuals, etc. [2:105]." For the purposes of this research, the definition does include the development of associated documentation as well as the executable program instructions.

Support Software--"Support Sofware is any software designated to support the development and testing of other software. Thus, it is comprised of developmental software and test software [3:82]."

Top-Down Development--this development methodology starts at the level of the whole program to be developed and, through a series of decompositions of available specifications, ultimately arrives at the machine or programming language that will be employed.

Variety--"the total number of possible states of a system, or of an element of a system [4:307]."

Viable System--a viable system or discipline is one that has the capability to survive and grow harmonically in a dynamic environment (6:84).

Wall Time--the number of hours that the computer resources will be required to complete the software development effort. It is measured as the sum total from input to required output for all runs required.

SELECTED BIBLIOGRAPHY

A. REFERENCES CITED

1. Aeronautical Systems Division, Air Force Systems Command. Management Guide to Avionics Software Acquisition. Volume I: An Overview of Software Development and Management. ASD-TR-76-11, Volume I. Washington: Government Printing Office, June 1976.

2. _____. Management Guide to Avionics Software Acquisition. Volume II: Software Acquisition Process. ASD-TR-76-11, Volume II. Washington: Government Printing Office, June 1976.

3. _____. Management Guide to Avionics Software Acquisition. Volume IV: Technical Aspects Relative to Software Acquisition. ASD-TR-76-11, Volume IV. Washington: Government Printing Office, June 1976.

4. Beer, Stafford. Brain of the Firm. New York: Herdes and Herdes, 1972.

5. _____. "Concerning the Cybernetic Paradigm for Organizing the Firm." Unpublished research paper, undated.

6. _____. Decision and Control: The Meaning of Operational Research and Management Cybernetics. New York: John Wiley and Sons, 1978.

7. _____. Designing Freedom. London: John Wiley and Sons, 1974.

8. _____. Platform for Change. New York: John Wiley and Sons, 1975.

9. Coyle, R. G. Management System Dynamics. New York: John Wiley and Sons, 1977.

10. Davis, Ruth M. "Reducing Software Management Risks," Defense Systems Management Review, Vol. 1, No. 6. Washington: Government Printing Office, 1978.

11. Electronic Systems Division, Air Force Systems Command. A Review of Software Cost Estimation Methods. ESD-TR-76-271. Washington: Government Printing Office, August 1976.

12. _____. Software Acquisition Management Guidebook: Cost Estimation and Measurement. ESD-TR-78-140. Washington: Government Printing Office, March 1978.

13. _____. Software Acquisition Management Guidebook: Software Maintenance. ESD-TR-77-327. Washington: Government Printing Office, October 1977.

14. Friedman, Jehosua, Philip Greenberg, and Alan Hoffberg. Fortran IV. New York: John Wiley and Sons, Inc., 1975.

15. Jarrell, Lieutenant Colonel Thomas H., USAF. Software Project Manager, ASD/YYM, Wright-Patterson AFB OH. Personal interviews conducted intermittently from 1 September 1978 to 10 August 1979.

16. Johnson, Richard A., Fremont E. Kast, and James F. Rosenzweig. The Theory and Management of Systems. New York: McGraw-Hill Book Company, 1973.

17. Mangold, Eldon R. "Software Visibility and Management," Proceedings, TRW Symposium on Reliable, Cost-Effective, Security Software, March 1974, p. 13.

18. Marciniak, Lieutenant Colonel John J., USAF. Director, Computer Resource Development Policy and Planning, HQ AFSC, DSC/Development Plans, Andrews AFB MD. Telephone interview. 16 January 1979.

19. _____. "Software Acquisition Within Air Force Systems Command--A Management Approach," Defense Systems Management Review, Vol. 1, No. 6, Washington: Government Printing Office, 1978, pp. 32-39.

20. McChesney, Lieutenant Colonel Jack L., USAF. Assistant Professor of Logistics Management, AFIT/SL, Wright-Patterson AFB OH. AFIT Course FM 5.23, "Contracting and Acquisition Management," Class 79B. Lectures. 3 October 1978 through 15 December 1978.

21. Office of Management and Budget. A Discussion of the Application of OMB Circular No. A-109. Washington: Government Printing Office, 1975.

22. Oklahoma City Air Logistics Center. Embedded Computer System Integrated Support Plan. Volume I: Executive Summary. Tinker AFB OK, January 1979.

23. Poe, General Bryce, II, USAF, Commander, Air Force Logistics Command. Address to the Army Logistics Executive Development Course students, Fort Lee, Virginia, 3 April 1979.

24. Pritsker, A. Alan B. Modeling and Analysis Using Q-GERT Networks. New York: John Wiley and Sons, 1977.

25. Putnam, Lawrence H., and Ray W. Wolverton. Quantitative Management: Software Cost Estimating. New York: Institute of Electrical and Electronics Engineers, Inc., 1977.

26. Riley, Major James M., USAF. Chief, Embedded Computer/Software Group, AFALD, Wright-Patterson AFB OH. Personal interviews conducted intermittently from 16 January 1979 to 15 August 1979.

27. Rome Air Development Center. Software Cost Estimation Study. Volume II: Guidelines for Improved Software Cost Estimation. RADC-TR-77-220. Washington: Government Printing Office, August 1977.

28. _____. Software Data Collection Study. Volume II: Data Requirements for Productivity and Reliability Studies. RADC-TR-76-329. Washington: Government Printing Office, December 1976.

29. Schoderbek, Peter P., Asterios G. Kefalas, and Charles G. Schoderbek. Management Systems, Conceptual Considerations. Dallas: Business Publications, Inc., 1975.

30. "Software Improvement Plan Pushed," Aviation Week, April 5, 1976, p. 43.

31. Thomas, Lewis. The Medusa and the Snail. New York: The Viking Press, 1979.

32. U.S. Department of Defense. Acquisition and Support Procedures for Computer Resources in Systems. AFR 800-14, Vol. II. Washington: Government Printing Office, 1975.

33. _____. Annual Report Fiscal Year 1979. Washington: Government Printing Office, 1978.

34. _____. Management of Computer Resources in Systems. AFR 800-14, Vol. I. Washington: Government Printing Office, 1975.

B. RELATED SOURCES

Ackoff, Russell L. "Towards a System of Systems Concepts," in Systems Analysis Techniques, Robert W. Knapp and J. Daniel Couger, eds. New York: John Wiley and Sons, 1974.

Aeronautical Systems Division, Air Force Systems Command. Management Guide to Avionics Software Acquisition. Volume III: Summary of Software Related Standards and Regulations. ASD-TR-76-11, Volume III. Washington: Government Printing Office, June 1976.

Beer, Stafford. Cybernetics and Management. 2d ed. London: The English University Press, Ltd., 1971.

_____. "The Zaheer Lecture: Cybernetics of National Development: Evolved from Work in Chile," as presented to the Zaheer Science Foundation, New Delhi, on 5 December 1974.

Computer Systems Command, United States Army. Software Phenomenology: Working Papers of the Software Life Cycle Management Workshop. Fort Belvoir, Virginia, August 1977.

Electronic Systems Division, Air Force Systems Command. An Air Force Guide to the Computer Program Development Specification. ESD-TR-78-139. Washington: Government Printing Office, November 1977.

_____. Cost Reporting Elements and Activity Cost Trade-offs for Defense System Software (Executive Summary). ESD-TR-77-262, Volume II. Washington: Government Printing Office, May 1977.

_____. Cost Reporting Elements and Activity Cost Trade-offs for Defense System Software (Study Results). ESD-TR-77-262, Volume I. Washington: Government Printing Office, May 1977.

_____. Life Cycle Cost/Design-to-Cost Guidelines. ESD-TR-75-77. Washington: Government Printing Office, June 1975.

_____. Software Acquisition Management Guidebook: Life Cycle Events. ESD-TR-77-22. Washington: Government Printing Office, February 1977.

_____. Software Acquisition Management Guidebook: Series Overview. ESD-TR-78-141. Washington: Government Printing Office, March 1978.

Electronics Command, United States Army. Life Cycle Cost Model. ECOM-4338. Fort Monmouth, New Jersey, July 1975.

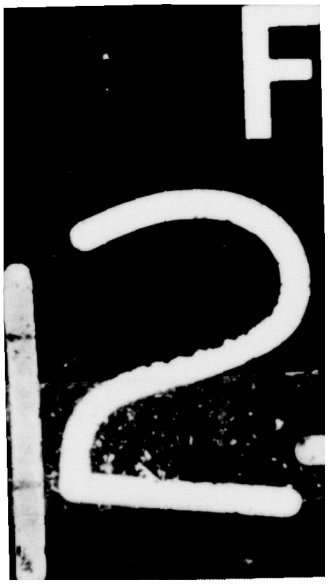
Forrester, Jay W. Industrial Dynamics. Cambridge MASS: The MIT Press, 1977.

Rome Air Development Center. Software Data Collection Study: Summary and Conclusions. RADC-TR-76-329, Volume I. Washington: Government Printing Office, December 1976.

Watson, Jerry Keith. "Acquisition of Embedded Computer Software: A Descriptive Model." Unpublished master's thesis, University of Missouri-Rolla, 1977.

Wigle, Captain Gary B., USAF. "Spare Memory and Timing Parameters in Avionics Computer System Requirements." Unpublished master's thesis. AFIT/GSM/SM/77D-30, AFIT/SL, Wright-Patterson AFB OH, December 1977. ADA056521.

Wooldridge, Susan. Softward Selection. Philadelphia: Auerbach Publishers, Inc., 1973.


F
12