USE OF HIERARCHICAL DECOMPOSITION IN COMPUTER SYSTEMS DESIGN

by

STEVEN JOSEPH DIFRANCO

B.S., Case Institute of Technology
(1970)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1977

Signature of Author.....
Alfred P. Sloan School of Management
May 12, 1977

Certified by.....
Thesis Supervisor

Accepted by.....
Chairman, Dept. Committee on Graduate Students

USE OF HIERARCHICAL DECOMPOSITION IN COMPUTER SYSTEMS DESIGN

by

STEVEN JOSEPH DIFRANCO

Submitted to the Alfred P. Sloan School of Management on May 12, 1977 in partial fulfillment of the requirements for the Degree of Master of Science.

ABSTRACT

Christopher Alexander has proposed a methodology for use in the general process of design. This paper investigates the application of this methodology, hierarchical decomposition, to the design of computer systems. A partial application of hierarchical decomposition to a real-world computer system, the Navy's Joint Uniform Military Pay System (JUMPS), is demonstrated. The paper concludes that while hierarchical decomposition has direct application to computer systems design, the practicality of the application appears limited to systems of rather small size.

Thesis Supervisor: Jeffrey A. Meldman,
Assistant Professor of Management Science

TABLE OF CONTENTS

I Introduction

The rapid progress in all areas of automated data processing technology over the past two decades has provided the potential for the realization of increasingly sophisticated and varied applications. The computer-based system designer has until recently been relatively successful in approaching the problem of design informally and intuitively because both the scope and complexity of applications have been limited by the technology available for the implementation of a design. As this technology has become less restrictive, computer applications have grown in size, and the design requirements, which define the nature of the desired interface between the system and its environment, have multiplied rapidly. This growth in design requirements has increased the complexity of the design process to the point where an intuitive, informal approach to problem definition no longer suffices. A major constraint to the successful implementation of computer application systems today appears to be not the underlying technology, but the natural limits of the systems designer's intuitive processes. At the same time, this problem is aggrevated by the limits of the language, or verbal concepts, in which design requirements are stated. The further removed from traditional data processing new applications become, the less adequate these verbal concepts are in aiding the designer in identifying

and understanding the particular design problems with which he is confronted. Some problems can only be partially addressed; others may be completely ignored because of the lack of a suitable language by which they can be identified and included for consideration in the design process.

Christopher Alexander[1] proposes a methodology for use in the general process of design: hierarchical decomposition. The methodology allows first for the identification of design requirements unencumbered by the traditional languages of the designer's disciplines, and, second, for the partitioning of a set of requirements into subsets sufficiently small to allow for separate, intuitive evaluation. His approach is to treat design requirements as a set of binary stochastic variables, some of which are dependent; by imposing sufficient conditions on this set, he demonstrates that it is possible to decompose the set into subsets so that the dependencies between subsets is minimized. His proposition is that such subsets of requirements facilitate the process of design by defining design problems at a level of complexity which can be approached intuitively.

The hypothesis of this paper is that the approach and techniques proposed by Alexander have application to the design of computer-based systems. The objective of this paper is to evaluate this methodology from the perspective of a computer-based system designer. The first chapter below discussed Alexander's proposals and methodology. The

second chapter proposes some mechanisms by which hierarchical decomposition can be applied to computer-based systems design. This application is illustrated by considering the design problem of a real-world computer-based system, the Navy's Joint Uniform Military Pay System (JUMPS).

II Hierarchical Decomposition--An Overview

The purpose of this chapter is to summarize the methodology of hierarchical decomposition as presented by Alexander, and to define the terminology he employs in its exposition.

a. The design problem. All designers face the same problem: that of constrained creation. That is the task of design is the invention of purposeful form which meets certain requirements imposed by the environment in which the form will be employed. These requirements are generally interdependent; an attempt to construct a form to meet one requirement may facilitate or make more difficult the satisfaction of another. The complexity of a design problem depends on both the number of requirements and their degree of interdependence. As the complexity of a design problem increases, the innate cognitive ability of the designer becomes less capable of integrating the competing requirements he must simultaneously consider. The designer is, of necessity, forced to simplify these requirements, to reduce them in some way so that he need only consider a mentally manageable number at one time. Hierarchical decomposition is the methodology proposed by Alexander to achieve this simplification.

Alexander defines form as that part of the world over

which the designer has control. The environment, or that part of it outside the control of the designer which defines the problem (and which is the source of the design requirements) is the context. The form and context together compose an ensemble. The context places demands on the form, which we articulate as requirements, or specifications which the form must meet to be acceptable. If the form does not satisfy a particular demand, misfit is said to occur with respect to that demand. The objective of design can be restated as the achievement of fitness between form and context.

b. The design process. Alexander studies in some detail the process of design, the process by which the designer attempts to construct a form to achieve fitness with a given context. Alexander examines two cultural archetypes, a selfconscious (advanced, such as our own) cultural, and an unselfconscious (primitive) culture, to compare the differences in their approach to design. The difference. he contends, is principally in the degree of separation between the designer and his form. The unselfconscious designer deals directly with form. The selfconscious designer is concerned with an abstract representation of the form, a picture drawn with verbal concepts. He is concerned then not directly with the form to be designed, but with the "economics" or "acoustics" of the

form. These concepts he manipulates by "theories of design" espoused by the disciplines of his endeavor, such as architecture, in whose particular languages these concepts are couched. The unselfconscious designer, on the other hand, avails himself of no such concepts or theories. Construction of form in unselfconscious cultures is most often guided by traditions or rituals, which have evolved over time as generations of form-makers have adjusted their forms in minute ways to correct misfit. As these changes are made, if fit results, they are incorporated into tradition: a perscription for form-making. This approach is satisfactory in unselfconscious cultures because the contexts of their forms are stable, or at least change only very slowly. The unselfconscious designer, then, is seldom faced with a new problem, or context, into which his form must be fit. As a result, he is not concerned with why certain forms work in certain cases, and not in others. He is aware only of a right way or a wrong way to construct form, as steeped in the tradition and rituals of his craft.

Alexander suggests that the unselfconscious designer often produces more successful forms because his view of the design problem at hand is not distorted by the verbal disciplinary concepts employed by his selfconscious counterpart. To remedy this situation in the selfconscious design process, the designer must create a further abstraction of the problem, a picture which describes only the structure,

and excludes the biases of the verbal concepts with which the designer normally views the problem.

c. Formal representation of the design problem. To accomplish this, Alexander proposes the following. Consider a point at which the form interacts with the context in any way to be a binary variable. If the form designed meets the demand placed on it by the context at that point, we consider that fit occurs, and misfit otherwise. The collection of all such points defines a set of "misfit variables". The misfit variables are not all independent, in the sense that construction of form to achieve fit at one misfit variable may make it easier or harder to achieve fit at another. The interdependencies between any two of these misfit variables we call "misfit variable interactions".

Formally, this is represented as an undirected graph G(M,L). The set M is the set of misfit variables; the value of the variable is 1 if misfit occurs, and 0 otherwise. The set L is a set of coefficients describing the interactions between the misfit variables; their sign and magnitude reflect the direction and strength of the dependency. The set M can be viewed as the vertices of the graph G, and L as the links betwen them.

G(M,L) is therefore an abstract but formal picture of the design problem; it reflects the designer's best estimate of the structure of the design problem. In the process of

design, the designer attempts to divide this problem into subproblems of smaller scope. The designer would like these subproblems to be both as internally cohesive and as independent from each other as possible. In terms of G(M,L), the subproblems are defined by partitioning M into subsets of misfit variables. Because some of the members of M are interdependent, as described by the set L, not all decompositions of M into subsets are equally beneficial to the designer. The most sensible decomposition of M would produce subsets of misfit variables which are highly "clustered"--in which the density of the links between variables in a subset is as high, and the density of links between members of different subsets as low, as possible. Alexander provides one method of achieving such a "sensible" decomposition of the set M, which he calls hierarchical decomposition.

The algorithm employed by Alexander to achieve this is, very generally, as follows. Each misfit variable is considered to be a stochastic variable, each chosen so that the condition of misfit is equally probable for all variables. The set L (after suitable normalization) is taken to estimate the pairwise correlation coefficients between these variables. From these, the probability of any distribution of misfit and fit among the elements of M can be calculated. These probabilities determine the information content, H(M), of the set M, taken as a whole. In a similar fashion, the information content of any subset of M, say S_i, can be

determined by calculating the probability distribution of its possible states.

Consider the partitioning of M into two subsets, S_1 and S_2. The dependence between these two subsets is given as R(p), which is equal to $H(M)-(H(S_1)+H(S_2))$, where p identifies the particular partition. What we want to do is to select S_1 and S_2 such that R(p) is minimized (i.e., such that the subsets are as independent as possible). Algorithmically, an iterative, "hill-climbing" procedure is employed to achieve this. The set M is partitioned arbitrarily, and R(p) evaluated. Each misfit variable is then moved, one at a time, between S_1 and S_2 until a minimum value for R(p) is achieved. The algorithm terminates with the identification of the two most independent subsets of M. The process is then repeated by considering each of the subsets in turn and decomposing them independently, until the "best" partitions are single element subsets. The decomposition thus takes on the appearance of a binary tree or hierarchy, whose nodes at any level represent the best decomposition of the immediately preceeding subset.

d. Solution of the design problem. The designer at this point is faced with a number of individual sets of misfit variables, each dense in internal interactions, but relatively independent of every other such set. They thus represent fairly isolated groups of requirements--the

designer can consider the nature of the form required to avoid misfit in one of the subsets without significantly affecting fitness between form and aspects of the context defined by other subsets of misfit variables. In this hierarchical decomposition, the complexity of the design problem has been reduced not by grouping requirements under abstract verbal concepts, but by uncovering the structure of the context. The solution to the original problem is attained by the composition of forms constructed for each of the subproblems suggested by the decomposition--the synthesis of form.

III Application of Hierarchical Decomposition to Computer-Based Systems Design

The preceeding chapter provided an overview of Alexander's view of the general process of design, the process by which forms are constructed, and outlined the methodology of hierarchical decomposition. The thesis of this paper is that this methodology has a direct application to the design of computer-based systems. This chapter discusses this application. The first section below examines some definitional issues, and the remaining sections investigate the specific mechanisms by which the methodology of hierarchical decomposition can be applied to the design of computer-based systems.

a. Definitions. Before attempting to apply Alexander's methodology to computer-based systems design, two definitional issues must be addressed. First, we will wish to equate "computer-based system" with Alexander's notion of "form", and "computer-based system environment" with "context". To do this, we must define, for the purposes of this paper, the scope of a computer-based system--which elements or components are part of the form, and which are part of the context. Second, we recognize that the "process of design" is not, in itself, a single well-defined activity, but is rather a concept describing a composition of activities in which the

designer participates or undertakes. We must define the activities relevant to computer-based systems design in which hierarchical decomposition has application.

(1) Computer-based systems scope. The scope of a computer-based system is a central issue in the study of computer-based systems design. A computer-based system implementation can be viewed as the insertion of an artificial form into some larger context. It is the extent of the system, the scope of the form being constructed, which defines the "systems interface" between form and context. As Alexander suggests,[2]

> every design problem begins with an effort to achieve fitness between two entities: the form in question and its context.

"Fitness" is both a statement of the objective of the design process, and a (desirable) condition of the interface between form and context. It is essential that the design process center, at least initially, on the accurate identification of this interface, which Alexander refers to as the "form-context boundary". In the first computer-based systems application, this boundary was fairly well defined. These applications dealt primarily with the automation of manual clerical functions, which themselves were well defined. Since these traditional data processing applications were designed for the replacement of existing forms, usually on a one-for-one basis, the form-context boundary could be

easily defined. As both technology and experience in computer-based systems grew, new systems were developed not simply to replace existing (manual) forms, but to integrate existing functions and to perform functions not previously existing. Form-context boundaries became increasingly hard to identify. As the "true" interface between a computer-based system and its context became more obscurred, the proper focus of the design process, the achievement of fitness along this interface, was lost. The design process, necessarily proceeding from some statement of requirements or function, selected system interfaces that could be most easily described. This phenomena gave rise to the "narrow" definition of the scope of a computer-based system. Many large systems were, and continue to be, designed based only on quantitative specification of input and output across the most visible interface--the computer room door. This strategy, while simplifying the design process, ignores the achievement of fitness along the broader, and more appropriate, system interface. Only after a system thus designed is implemented (operationalized), does fit or misfit along this boundary, the actual form-context boundary, become apparent.

For the purposes of this paper, then, we will assume a very broad, when compared to the traditional, definition of computer-based system. We will assume that not only those components within the confines of the computer room are included in our definition of form, but also all those

necessary to achieve or support the system's entire function. These may include manual or automated correction facilities, archival storage and library components, and data transmission and receipt facilities. We will also include in our definition of form the operating procedures developed for use by organizational components of the context which are required for system support.

(2) The design process in computer-based systems. Many authors have explored the nature of the design process in computer-based systems, and have proposed frameworks to describe the activities involved in this process. Murdick[3] reviews the work of 17 contributors in this area, and demonstrates the similarity in approach, if not language, in identifying and structuring design-related activities. He suggests the following stages for the design process as a composite of these separate proposals:[4]

> Some problems of nomenclature arise, but an examination...seems to indicate the following synonyms:
>
> 1. Investigation=preliminary survey=problem definition=define need or mission objective= analyze the present system.
>
> 2. Feasibility study=conceptual design=establishment of performance specifications= gross design=Phase I design.
>
> 3. Detailed design=develop system operating specifications=systems definition=analysis and synthesis=systems acquisition=Phase II design.

4. Implementation=installation=systems construction.

He further points out that[5]

> a step-by-step description is not really appropriate. Many activities are carried out in parallel and there is much iteration or recycling to refine the design.

Regardless of the sequence in which these activities are actually accomplished, it is suggested that they fall, conceptually, into two categories. The first, which we will call the "design phase", encompasses the activities of the first three steps in Murdick's framework. The second, or "implementation phase", corresponds to the fourth step.

The distinction between these two categories is the nature of the activities of which they are composed. Activities in the design phase can be viewed as processes of abstract conceptualization. Beginning with some notion of function(s) which the system is expected to perform, abstract logical components are assembled to translate some input to required output. By contrast, activities in the second category, the implementation phase, require construction of physical components which display the external characteristics of their logical counterparts.

The distinction between the nature of the activities in these two categories is underlined by the growing disparity in formal design aids, or methodologies, available for each category of activity. On the implementation side, programmers and systems designers have available an increasing number of more effective tools to assist them in principal

implementation activities such as code construction and hardware selection and configuration. For example, numerous algorithms for the performance of standard data processing functions have been developed, cataloged, and analyzed. Simulation packages are available to assist in evaluating alternatively configured systems' performance. While the implementation phase of the design process has not been reduced (or elevated, depending on point of view) to the realm of "science", there is no question that the computer-based systems implementor has a increasing variety of rational resources besides his imagination upon which to draw.

On the design side, however, comparatively little progress has been made. Until recently, there were almost no tools available to assist the designer in his task of developing an integrated logical form which displays fitness with its context. Davis observes that[6]

> It is one of the anomalies of information systems that this field, which is applying technology to information processing at such a rapid rate, has not applied technology to any significant degree to its own analysis and design process.

Recent work by Myers,[7], Rockart,[8] and King and Cleland[9] have begun to provide the systems designer with a model-based framework for design and evaluation of computer-based systems. Use of these model-based techniques, however, do not as yet appear to have achieved widespread practical application.

As the failure of many computer-based systems development efforts is increasingly attributed to design rather than

implementation activities, significant attention has been directed toward development of formal design-aiding methodologies. The most successful of these have found application is the development of detailed design specifications and in the initial organization of a computer-based system by logical processing modules. These methodologies, including "composite design"[10] and "structured programming"[11] have primary application in the activities of step 3 of Murdick's framework. But all of these methodologies proceed from explicit statements of general systems requirements and at least a basic definition of the proposed system's internal architecture--products of Murdick's step 1 and 2 activities. While there have been several attempts at providing a formal methodology for use in these activities (such as Tiechroew's Problem Statement Analyzer[12]), these have not achieved any practical degree of acceptance.

In most computer-based systems development efforts, designers continue to rely on experince gained through work on similar applications. While experience may be the best teacher in some endeavors, it is insufficient by itself in computer-based systems design for two reasons:

First, most designers lack an acceptable framework for analysis of existing forms. That is, the designer has not developed, or been presented with, a robust set of factors which explain the viability of a system. Failures and successes are explained ex post, but these explanations

provide the designer little insight from which he can benefit in the design of new forms. The best that he can achieve is a set of heuristics, specifying necessary design activities without explaining the structure of the problem which makes these activities advisable. Thus experience in computer-based systems design is often ineffective in guiding future design due to the lack of a comprehensible framework in which to evaluate it.

Second, the contexts into which computer-based systems are inserted vary significantly from application to application, even when the applications (the purpose or function of the systems) appear to be identical. The recent development of standard software packages has been made possible only through the stringent specification of requirements along the "narrow" definition of the form-context boundary. The decision to employ such a package and the selection of one from amoung those available is an activity of the implementation rather than the design phase, which is more properly concerned with fitness along the broader form-context interface. The process of form design, if it is to be based solely on experience or tradition (as in Alexander's unselfconscious cultures) must necessarily be concerned with a stable, or at least only very slowly changing context. The diversity among the organizational entities in which computer-based systems are employed evidences this lack of stable context in the process of their design.

What the above suggests is that experience alone is not sufficient in computer-based systems design. At the same time, no design-aiding technologies enjoy popular application. As will be suggested below, however, Alexander's methodoloy of hierarchical decomposition has application in assisting the designer in these initial design activities (summarized by Murdick above as Step 1 and Step 2 activities).

b. The form-context boundary. The application of hierarchical decomposition begins with the division of the ensemble into form and context. Alexander defines "form" and "context" only in a very general way:[13]

> The form is a part of thw world over which we have control, and which we decide to shape while leaving the rest of the world as it is. The context is that part of the world which puts demands on this form; anything in the world that makes demands of the form is context.

The purpose of this section is three-fold: First, to define in greater detail the concepts of "form" and "context" by examining the nature of the boundary between these from a computer-based system perspective; second, to outline an approach to selection of the form-context boundary most useful to the computer-based systems designer; and third, to illustrate this approach through examination of the Joint Uniform Military Pay System (JUMPS).

(1) Nature of the form-context boundary. It is first

necessary to recognize that computer-based systems are generally implemented in a formal organization environment, usually a business or government organization. These organizations themselves are composed of numerous organizational components, usually assembled in a hierarchical fashion, and each assigned particular functional responsibilities. In accomplishing their assigned function, organizational components seldom act individually, in isolation from each other, but rather depend on other organizational components. A computer-based system (the form to be designed) whose purpose is the displacement of function will interact similarly with at least some existing organizational components: It will place demands on some in support of its operation, and will have demands placed on it by those components to which it provides functional support.

We will consider these demands as "interactions" between the system and identifiable components of the organization in which it is implemented, and categorize these interactions as follows:

Information interactions: Computer-based systems are primarily concerned with the processing of information. Many functions such systems displace from organizational components are therefore those functions which are likewise information-centered. We will consider those points at which information is provided to the system as input, or produced by the system as output, to be points of information inter-

actions.

Control interactions: In addition to providing or utilizing the information which a computer-based system processes, organizational components may interact with a system by controlling its operation in some way, or vice versa. In a system designed for the retrieval and display of data, a command language, or systems interface, must be provided for the user to specify the data to be retrieved or the operations to be performed on the data. Other systems may be constructed so as to provide output relating to the execution of the system, perhaps error messages indicating that certain actions must be taken by the operator or user.

Financial interactions: Some organization components interact with the form from a financial perspective. Budget offices typically allocate funds or set ceilings on the resources available to the prospective system, in terms of development, operation, and maintenance.

Other interactions: Finally, there are other interactions which take place between organizational components and the form to be designed which are not information, control, or financial in nature. These include those interactions which specify directly certain requirements which the system must meet, and which are common to all systems in a designated class. Standard requirements for documentation of a system developed within a particular organization is an example of such an interaction. Requirements for precautions to be taken

to insure data base integrity or security of information are others.

We will define the context of the design problem to be comprised of all organizational components which will interact with the system in one of the ways just mentioned. The boundary between the system (form) and these organizational components (context) is described by the specific interactions themselves.

(2) Selection of the form-context boundary. We can consider two separate but related steps in the process of selection of the form-context boundary. The first, a "general positioning" of the boundary, identifies those components outside the control of the designer, but which will interact with the form to be designed. The second step, "specific positioning", identifies the specific interactions between the form and the organizational components of the context.

Three cases need to be considered in general positioning of the form-context boundary:

Displacement of existing function: The simplest computer-based system is one which displaces a single function performed by an existing organizational component. This organizational component itself can be thought of as a form which the system to be designed will, in part, replace. The designer can identify those organizational components which interact with

this current form through implicit or explicit construction of a descriptive model of the functional process. The components thus identified will occupy the context of the design problem.

Displacement and integration of function: Computer-based systems are increasingly implemented to integrate functions which may previously have been performed by several organizational components. The context of the design problem in this case comprises all organizational components which interact with the existing forms whose functions the system to be designed will displace. These can perhaps best be identified, as in the above case, by construction of a descriptive model of each of the current functional processes.

Performance of new function: As suggested in the introduction to this paper, the expanding technological basis for computer-based systems has made possible the implementation of new function, previously not achievable by existing forms. Descriptive models are, of course, ineffective in locating the context of the design problem in this case. Here the best the designer can do is anticipate the organizational components which will interact with the proposed system: those which either will support the system's operation, or which will depend on the system for their own functional support.

With regard to the first two cases just mentioned, it should be noted that construction of descriptive models of

the current functional process(es) which the system will displace is not, in general, sufficient to identify all components of the context. The system to be implemented may require operational support other than that currently available. For example, no facility may currently be in place for correction of erroneous input to the proposed system. As in the third case above, the designer's task is to anticipate in which organizational components these new "systems support facilities" will reside, and use this to augment his descriptive model.

After the context of the design problem has been generally defined, the designer must specifically position the boundary between form and context. This is accomplished by identifying the specific information, control, financial, and other interactions between the components of the context and the form to be designed. The most salient interactions will be those which contribute significantly to stress, or misfit, between the existing form(s) and their context(s). The motivation for development of the proposed system is often the elimination of stress in specific form-context interactions. It is important for the designer to realize, however, that all interactions must be considered in definition of the form-context boundary, not only those interactions currently contributing to this stress. The purpose of the form-context boundary selection process is to identify an interface between the system and its environment along which it is possible for points of misfit to occur. The designer must include in the

form-context boundary those interactions which are currently satisfactory because he has no guarantee that the form suggested by the process of hierarchical decomposition will maintain this fitness with those components. The process of general and specific form-context boundary positioning is illustrated by the following example.

(3) JUMPS application. To clarify the issues involved in the selection of an appropriate form-context boundary in computer-based systems design, we will consider a real-world system recently implemented within the Navy--the Joint Uniform Military Pay System (JUMPS).

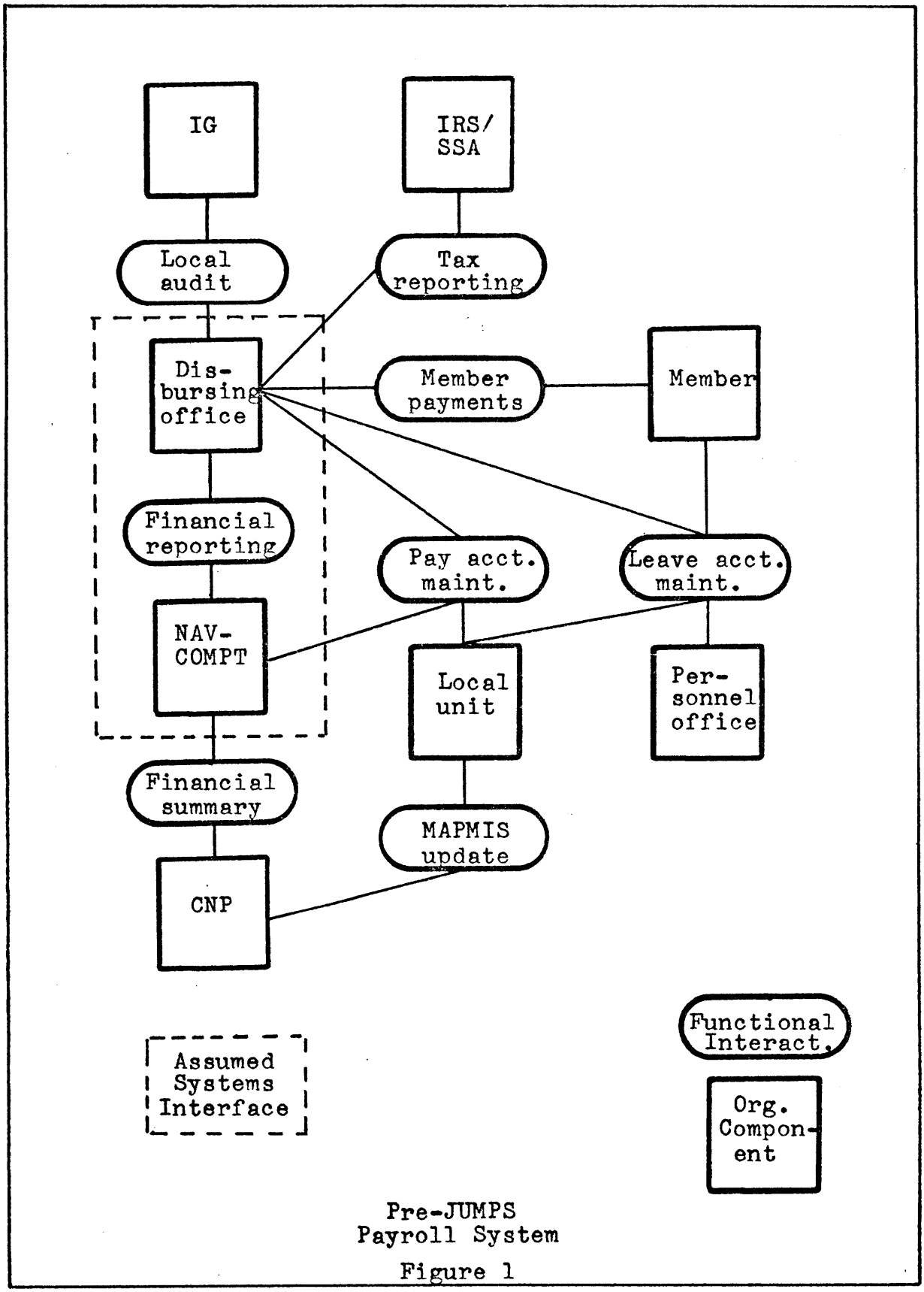
(a) JUMPS overview. JUMPS was originally proposed in 1966 as an integrated, automated, centralized replacement for the Navy's manual payroll and leave accounting system. The primary objectives of JUMPS were:[14]

> First, to improve the administration of the Military Personnel Appropriations through establishment of a central financial reporting system, based on the principles of accrual accounting, for the reporting of obligations and expenditures from these appropriations; second, to take advantage of the increased availability and effectiveness of automatic data processing (ADP) equipment and supporting software in the area of military pay.

While payroll systems are normally considered to be among the most simple of computer-based systems to design and implement, design and development of JUMPS proved to be both lengthy (10 years) and expensive (approximately $70 million).

There are several reasons for this, including the sheer size of the application (pay accounts are maintained for 600,000 members), the complexity of the military compensation structure (which supports over 100 different types of pay and allowances, each with its own particular set of authorizing conditions), and the mobility and dispersion of Navy members (who may be assigned to any of more than 1000 units capable of world-wide deployment). The most significant reason, however, appears in retrospect to have been the designers' lack of appreciation for the true scope of the system which they were designing.

Figure 1 is a general overview of the Navy's manual payroll system as it existed before JUMPS. Navy members are assigned to one of some 4000 commands, or local units. Disbursing support for a local unit (or several geographically proximate units) was provided by a local disbursing office. These offices maintained member pay accounts according to pay procedures issued by the Comptroller of the Navy (NAVCOMPT), and based on pay-related transactions and member change requests issued by the local unit. Payments to members (expenditures) and amounts due members but not paid (obligations) were reported by local disbursing offices, summarized by NAVCOMPT, and provided to the Chief of Naval Personnel (CNP), the Military Personnel, Navy (MPN) Appropriation manager. CNP also received personnel-related transactions directly from the local unit, from which a data base



Pre-JUMPS
Payroll System

Figure 1

providing Navy strength and compsition information was derived. This information, together with the obligation and expenditure reports, assisted CNP in the program, planning, and budget execution activities relating to the management of the MPN appropriation. Reports of wages and of withholding of Federal Income Tax and Social Security payments from members' pay were provided to the Internal Revenue Service (IRS) and the Social Security Administration (SSA). The Navy Inspector General (IG) was charged with conducting periodic formal on-site audits of local disbursing office operations.

This describes briefly the enviornment in which JUMPS was to be implemented. JUMPS was expected to perform the seven basic functions listed in figure 2 and described more fully below.

Pay account maintenance: The official pay account for each Navy member was to be maintained in automated form at the central site (the Navy Finance Center). The local disbursing office structure was to be maintained, but their function would be limited to transmission of pay-related transactions to the central site (also see below under "member payments"). These transactions would be initially generated by the personnel office of the local unit to which the member was attached. These transactions would report any change in the member's status (promotion, unauthorized absence, detachment, etc.) or the unit's status (deployed, entered dry dock,

Function to be performed	Currently Responsible Component	Current Interactions
Pay account maintenance	Disbursing office	Member/NAVCOMPT/ local unit
Member payment	Disbursing office	Member
Tax reporting	Disbursing office	IRS/SSA
Leave account maintenance	Personnel office	Member/local unit/ disbursing office
Obligation and expenditure reporting	Disbursing office/ NAVCOMPT	CNP
Personnel-pay account reconciliation	-- new function --	
Account status	-- new function --	

Proposed JUMPS Functions and Current Responsible Components

Figure 2

entered combat zone, etc.) which could affect the member's pay account by altering his set of "current entitlements". The central site would receive these transactions and attempt to update members' accounts based on the input transaction, automated logic (reflecting actual pay procedures then in effect), and the current status of the members' accounts. When JUMPS was fully implemented in January 1977, an average of 350,000 transactions were received per month. The major difficulty encountered in the design of the system was the complexity of the military pay compensation structure. The specifications for processing of transactions (implemented in the form of decision logic tables) required identification and handling of more than 16,000 separate cases. While some were trivial, some required almost a complete rewrite of the member's account, with attendant requirements for updating history files, audit trails, and tax and expenditure reports. In the end, it was decided not to implement some 600 cases in the automated update logic[15] but to accomplish the necessary action manually in an off-line mode at the central site.

Member payment: Payments to members were to be accomplished in one of two ways. First, some members could elect to participate in a "Net-check-to-bank" program, under which their net pay due (on the 15th and 30th of each month) was deposited directly in a financial institution via the Treasury's Composite Check Program.[16] Composite checks (listing all members and amounts to be deposited at a parti-

cular bank) were to be prepared at the central site. Second, the member could elect to receive his pay directly, in either cash or check. These disbursements were to be made by the local disbursing offices, based on pay account status provided by the central site (see below under "Account status"). The disbursing office would prepare a report of payments actually made for submission to the central site so that member pay accounts could be debited.

Tax reporting: The central site would prepare all reports required by the IRS, SSA, and the States, based on the member pay accounts.

Leave account maintenance: JUMPS was also to provide a vehicle for leave account maintenance, a function which has traditionally resided in the local personnel office. Because the military compensation structure allows a member to "sell" unused leave under certain circumstances, and because many pay-related transactions also affected a member's leave status, an integrated accounting system for both pay and leave seemed appropriate.

Obligation and Expenditure reporting. The central site was to prepare monthly reports of obligations and expenditures to CNP, based on the actual member pay accounts. These reports were broken down by numerous categories, and included approximately $400 million in disbursements each month.

Personnel account-pay account reconciliation. As mentioned above, CNP maintained an internal data base system, the Manpower and Personnel Management Information System

(MAPMIS) which provided Navy strength and composition information, and which was maintained by personnel-related transactions prepared by local personnel offices. Because many of the data elements in both JUMPS and MAPMIS were identical, a periodic reconciliation between the two systems to insure integrity of data was desirable. Although this function had not previously been performed, it was not expected to be significant, because input to both systems was prepared at the same units, the local personnel offices.

Account Status: The second new function to be perfomed by JUMPS was the provision of pay and leave account status to the member, the local unit to which he was attached, and to the local disbursing office. The statement of account status was to include, for each member, all credits and debits, with annotations of any changes since the last statement. The disbursing office was to use this statement to determine the amount of pay due each member paid locally, which was taken to be one-half the difference between accumulated credits and debits for each bi-monthly payday. The local disbursing officer could adjust this amount based on transactions input to the central site but not reflected on the current statement of account status.

The general systems specifications developed for JUMPS[17] provide for the accomplishment of these functions, and specify requirements to achieve these within the systems interface outlined in figure 1. As will be discussed below, this inter-

face embraced only a "narrow" definition of the system's scope, which contributed significantly to the subsequent problems encountered in JUMPS development.

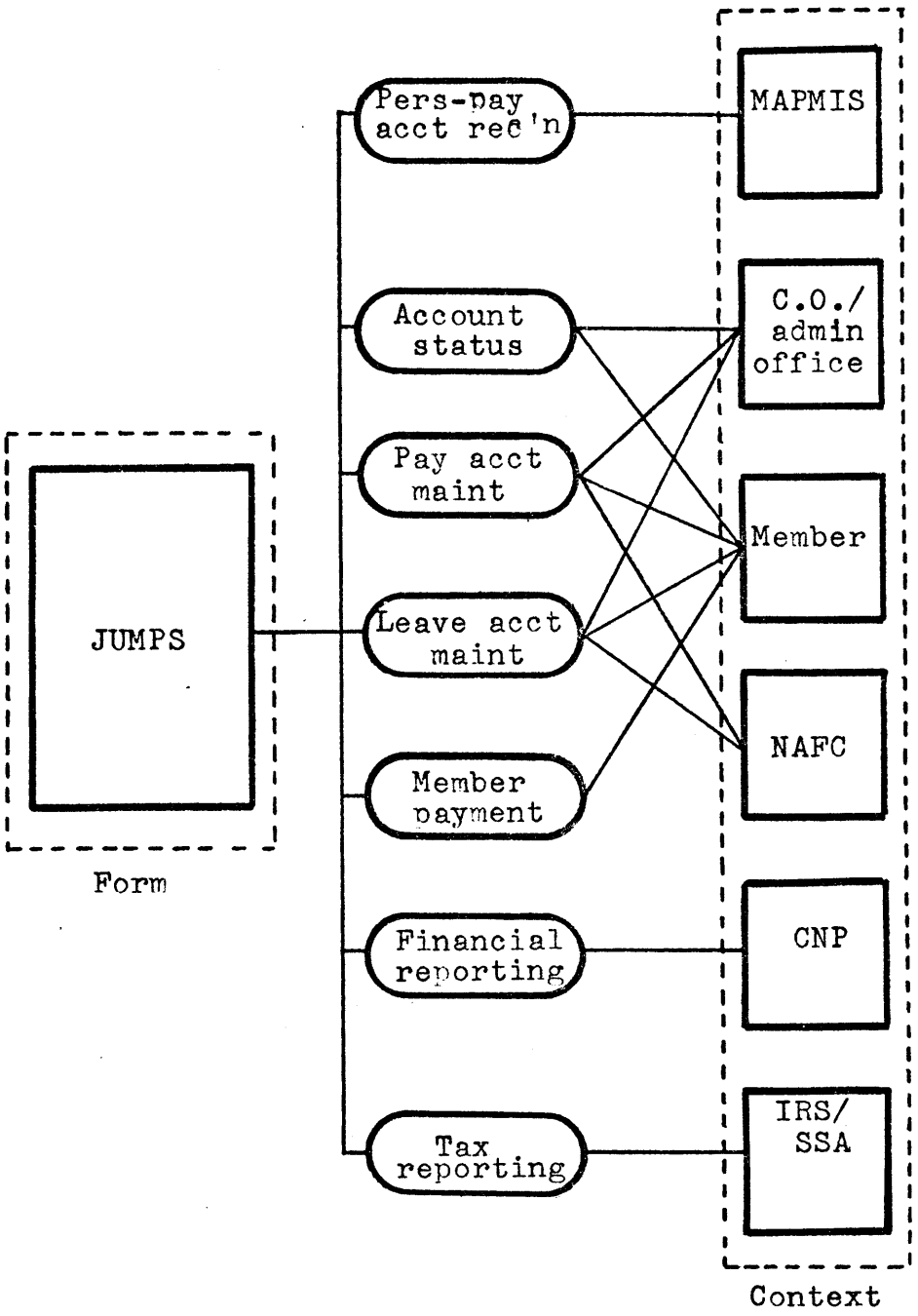
(b) General positioning of the form-context boundary. General positioning of the form-context boundary requires that the system's proposed functions be stated, the organizational components responsible for these be identified (if any), and the organizational components with which these currently interact be specified, as in figure 2. In considering "functions", we include those currently residing in a single organizational component, those shared among several, and new functions not performed by any existing organizational component.

As Figure 2 indicated, JUMPS was both to displace and integrate functions performed by several existing components and to perform new functions not previously existing. Figure 3 depicts the actual context of the JUMPS design problem, based on those organizational components with which the proposed system will interact to accomplish its designated functions. It should be noted that the context of figure 3 does not exactly coincide with the JUMPS environment determined by the system's interface indicated in figure 1. There are three major reasons why the assumed system's interface was inappropriate for the JUMPS design problem, which are visually evident from the differences between figure 1 and figure 3.

by JUMPS. By including it within the system, no requirements were developed to interface NAFC with the rest of JUMPS. Significant problems exist in the operation of the system today due to the lack of flexibility within JUMPS to accommodate new interpretations of pay-related legislation. By neglecting to consider potential misfit along the NAFC-JUMPS interface, significant stress is evident.

Third, no explicit recognition was given to the interactions required to support the proposed system function of personnel and pay account reconciliation. To accomplish this, the CNP data base system, MAPMIS, would need to interact with JUMPS to a significant degree. Because MAPMIS was not identified initially as a component of the form-context boundary, no requirements were developed to define this interface. This problem was belatedly recognized, but only after JUMPS had to a large degree been designed. As a result, only limited fit between JUMPS and MAPMIS was achievable.

(c) Specific positioning of the form-context boundary. After the organizational components which directly interact with the system have been identified and tentatively included as part of the context of the design problem, the portions of the boundary relevant to the design problem are specified by identifying the specific interactions across the boundary. Figure 4 considers a portion of the boundary selected above by enummerating the possible types of interactions



JUMPS Form-context Division

Figure 3

which can occur. (For simplicity, only information and control interactions between JUMPS and the local administrative office and JUMPS and the member are shown.) Some of these interactions currently exist: pay authorizations, member change requests, and special pay requests. Several new interactions required to support or use the new system's functions are also included: request for and provision of account status to either the member or the Commanding Officer/administrative office, an option to reschedule regular paydays by the Commanding Officer (to meet local unit operational constraints), and an error notice interaction to allow for correction of erroneous input to the system.

The specifications of form-context interactions (extended to include all portions of the form-context boundary) will completely position the boundary between the system and its context.

The above JUMPS example serves to illustrate the extension of Alexander's definition of form and context to a computer-based systems environment:

First, those organizational components responsible for function which will be displaced by the proposed system must conceptually be viewed as part of the form which the system will replace, and interactions between them and their "contexts" be considered in defining the form-context boundary for the new system.

Second, those organizational components whose function will not be displaced by the proposed system must be considered as part of the context, so that interactions between these and the form to be designed can be determined.

Third, organizational components with which the proposed system will interact in connection with new function must be explicitly included in the context to allow for estimation of required (but currently non-existant) interactions.

These considerations will aid the designer in assigning the elements of the ensemble (organizational components) to form or context in the manner most useful for design of computer-based systems. Definition of form and context allows the specific interactions between these to be investigated. Once these, which define the form-context boundary, have been identified, the designer can proceed to isolate the specific points along the boundary at which misfit can occur--the process of selection of misfit variables.

c. Misfit variables. At this point, the designer is concerned with developing a set of statements which reflect the possible ways in which misfit might occure between form and context. This section considers the language, or the articulation, of misfit in computer-based systems.

(1) The nature of "misfit". We have some general notion of "misfit" as a lack of harmony, or a discordant

condition, but a formal definition of the concept is elusive. But we do know, as discussed above, the conditions under which a misfit can occur: at those points of interaction between form and context. We know, in a computer-based systems environment, something of the nature of these interactions. Form-context interactions, or the demands placed on one by the other, occur when information is passed across the form-context boundary, when control is exercised, or when financial or other requirements must be met. The question then becomes one of examining how misfit might occur in these four types of interactions.

(a) Information interaction misfits. We can consider two different aspects of the information exchanged across the form-context boundary at any given point: its content, or subject matter, and its characteristics. The characteristics of the information include such qualities as accuracy, currency or timeliness, frequency, and level of aggregation.[18] We could expect misfit to occur if either the content or characteristics of the information exchanged fail to meet the requirements of the recipient (form or context). Moreover, we could expect these requirements to vary from context to context, and from point to point along a single form-context boundary. This boundary has been positioned by identifying those organizational components which will interact with the form. These components may be differ-

entiated both by area of activity (marketing, production, accounting), and by the level of this activity (strategic, tactical, or operational[19]). The area of activity will generally determine the requirements for information content. The characteristics of the information required are largely a function of the level of activity.[20] With regard to information interactions between form and context, we therefore face possible misfit along a variety of dimensions at any given point on the form-context boundary.

(b) Control interaction misfits. While significant attention has been given to analyzing the informational requirements of differing organizational components, substantially less has been directed towards understanding control interactions in computer-based systems. This is understandable in that until recently, the technological basis for computer-based systems did not support any significant degree of control interaction between system and user. Now that such facilities as timesharing and on-line data base systems are possible, there exists the latitude for considerable interaction between system and user which must be considered in systems design. Little proposes that there exist general requirements which must be met at points of control interaction on the user-system interface.[21] These include communicability, ease of control, and robustness. These are obviously broad categories of requirements, rather

than specific statements of potential misfit, but they do provide some insight into the nature of misfit in control interactions. Stress may occur along a "communicability" dimension for a variety of reasons: the language of control may be foreign to the context; the mechanism for control (lightpen, keyboard) may be too time consuming or too restrictive; error messages or requests for input may be unintelligible to the context. Similarly, on the "robustness" dimension, misfit may be evident if the system lacks the ability to handle commonly mispelled systems commands. But a key point here again is that misfit is a function of both form and context, the system and its users. A control language perfectly compatible between form and context at one point of the interface may result in misfit at another. A data base manager, for example, might find that the English-like, robust language developed for the interface between the system and the non-technical user to be cumbersome, ineffective, or not sufficiently specific for his purposes. Little's general requirements, therefore, give some insight into the nature of potential misfit in control interactions.

(c) Financial interaction misfits. Of all possible misfits between form and context, those resulting from financial interactions are the easiest to understand. The most common and most visible misfit evident after implementation of a computer-based system has traditionally been

the misfit between actual accrued development cost and the anticipated budget.

(d) Other interaction misfits. Interactions other than those described above have become increasingly numerous over time as a growing number of organizational components begin to be placed at the form-context boundary. This growth is attributable in part to the growing public interest in the design and operation of computer-based systems. The impact of recent legislation (in the areas of privacy and security of information) is just beginning to be understood. Specific requirements, in terms of the actual design of computer-based systems which would insure fit between the system and applicable statutes have certainly not been determined. In addition to legislation, there exists an increasing tendency for organizations to attempt to standardize their computer-based systems and to establish in some cases specific regulations for their design. All systems for the disbursement of public funds, for example, must comply with General Accounting Office regulations;[22] systems developed in the Department of Defense which support military operations (such as JUMPS) must include provisions for mobilization or deployment contingencies.[23] It is obvious that this proliferation of requirements provides a fertile field for potential misfit.

With these general ideas of the nature of misfit which can occur between a computer-bases system and its environment in mind, we will define misfit variable as: A statement of a condition relative to interactions between form and context which if not met will result in stress, or misfit, in the ensemble.

(2) Traditional approaches to avoiding misfit. It may be argued that computer-based systems designers have always been concerned with avoiding misfit between the systems they design and the context of their application. The traditional mechanism employed to this end is a (more or less) carefully developed set of specifications, or requirements, which the system must meet to be "acceptable". These specifications are usually formulated based on prospective user input and external regulations with which the system must comply. The result is a collection of statements, some very specific, and some general, upon which the system design is based. This collection, for example, will generally include edit requirements for input data, algorithms to be used in calculating elements of output, specifications of the frequency of output and its format, and the budget or resources allocated for design, development, and operation of the system. Since many successful computer-based systems have been designed and implemented on this basis, this traditional approach appears to be at least satisfactory for some applications. On the

other hand, a substantial number of computer-based systems have failed to achieve acceptance following the same approach. Why does the design process produce successful forms in one instance and not in others, when the same basic (traditional) methodology is employed, perhaps even by the same designers?

The process of computer-based systems design is, recalling Alexander's terminology, a "selfconscious" one. That is, the designer is not concerned directly with the particular context of the design problem at hand, but rather with an abstraction of the context. This is articulated, in the traditional approach, through a language which comprises those verbal disciplinary concepts peculiar to computer-based systems design. As Alexander suggests, use of such a language can bias the designer's picture of the design problem. Bias is introduced primarily by the "preclustering" of points of potential misfit into a single verbal concept, such as "response time" or "audit trail". In some design problems, this preclustering may be appropriate--verbal disciplinary concepts may be adequate to describe the structure of the context to which the form will be fit. In other, perhaps superficially similar design problems, however, the same preclustering of misfit variables may turn out to be inappropriate for independent design choices--a bias has been introduced. The following examples from JUMPS illustrate this point.

(a) JUMPS systems specifications. The basic

document providing for the development of JUMPS[24] contains more than 150 separate specifications for design of the system. Approximately 23 of these can be considered as pertinent to the portion of the JUMPS-context boundary described above and depicted in figure 4: the JUMPS-Commanding Officer/Administrative Office and the JUMPS-member interfaces. Two representative examples of these specifications are:[25]

> When local input to the centralized site is released from the local level, unaccompanied by supporting documentation, it will be suspended to ensure transmittal of the required documentation.
>
> Transactions common to both military pay and personnel systems will be input using single source, source data automation techniques and will be entered in both systems on the same basis, whenever practicable.

The language of these specifications is clear, as is the intent. The first presumable will insure that unsupported transactions are not allowed to update internal system's files. The intent of the second is two-fold: to reduce clerical workload at the local level, and to achieve compatibility between the personnel (MAPMIS) and JUMPS data bases. The important observation here is that both these specifications are included to avoid potential problems, or misfits, in the operation of JUMPS, which the designers realized through their experience could occur. The language of the specifications, while meaningful in the disciplines of data processing or computer science, is not specific to the actual design problem at hand. The specifications attempt to specify how misfit can be avoided without examination of the nature

of misfit in this particular implementation.

Considering the first specification, we may ask why some input might be unsupported. What sort of misfits might be created by insisting on such documentation? Is the documentation difficult to prepare? Are the timing requirements for the input such that supporting documentation cannot be prepared in the same time frame? With regard to the second specification, we might expect misfit to occur for a variety of reasons: Is the organization of the local office such that responsibility for preparation of both personnel- and pay-related transactions reside with the same person? Can local units support or maintain the source data automation equipment required? Are the time frames for input to both MAPMIS and JUMPS similar? The point here is that while a system can be designed to meet these specifications, the structure of the actual design problem (the point of potential misfit specific to this application) is not apparent from the specifications. The specific points of misfit are precluster-ed by these design specifications. Even if both of the specifications are met, several misfits could result, as suggested above. The point is that the structure of the design problem has been distorted by employment of tradition-al verbal concepts in its description.

The 23 JUMPS specifications pertinent to the portion of the JUMPS-context boudary mentioned above are listed in Appendix 1. These specifications will be used as the basis

for the development of misfit variables along the JUMPS-context boundary of figure 4. It will be remembered that the selected systems interface for JUMPS (upon which the specifications in Appendix 1 were predicated) differs from the form-context boundary defined above, and are therefore incomplete.

Traditional systems specification definition may result in a successful form if by accident no stress between form and context results. What the above JUMPS example attempts to convey is that the language employed in this traditional approach to avoiding misfit is too general to provide an adequate understanding of the structure of the problem facing the designer, the generality due to the implicit "precluster-ing" of misfit variables. If a system is successfully designed following this approach, the designer can take credit only for meeting specifications, not necessarily for understanding the structure of the actual design problem itself. The language of specification and design requirements offer the designer an abstract, but biased picture of the problem. What we need, suggests Alexander,[26]

> is to make a further abstract picture of our first picture of the problem, which eradicates its bias and retains only its abstract structural features; this second picture may then be examined according to precisely defined operations, in a way not subject to the bias of language and experience.

This "further abstract picture" is based on the selection of a set of misfit variables appropriate to the context and form

to be designed. The "precisely defined operations" are those of a formal decomposition algorithm operating on this set.

(3) Selection of misfit variables. We have developed above the idea of how a computer-based system interacts with its context and the ways in which these interactions might result in a stressful, or misfit, condition between form and context. The question facing the designer at this point is the articulation of the set of such points of potential misfit, the misfit variables. Let us consider the qualifications which the statements of misfit variables must meet.

(a) Well-understood. For a misfit variable to be useful to the designer, he must have a clear understanding of what conditions must be met to avoid misfit. That is, the designer must be able to envision, for any given misfit variable, a form which would achieve fit at that point. (There are two issues here--estimation of the condition necessary to avoid misfit, and a statement of this condition. Different designers may have different estimates of the required condition. But given that the condition has been estimated, the issue here is that the statement of the condition must be well-understood.) For some misfit variables, the condition required to achieve fit may be stated unambiguously. These are those variables with which a quantitative

performance standard can be associated. This will be the case for every misfit variable that exhibits continuous variation along a well-defined scale. As an example of such a misfit variable, we can consider a possible misfit in JUMPS associated with the variability in the amounts paid a member over several paydays. The designer here recognizes that misfit will occur if a member's paycheck varies drastically from payday to payday (when no transactions have occurred against the member's pay account). (This situation is possible in a military pay environment because pay due a member is often composed of several items of pay, some of which are based on daily rates and others on a standard monthly rate.) If the designer can specify the degree of variability allowable without misfit, he can construct a well-understood misfit variable. For example:

> Payments to a member cannot vary more than $1.00 between paydays, excepting variations attributable to new transactions against the member's account.

Here "variability" is dependent on the particular system designed, and it exhibits "continuous variation along a well-defined scale", namely, a dollar (numerical) scale. The variable is a useful one because the condition necessary to achieve fit is well-understood--in this case, because a performance standard can be specified.

This is not to suggest that the only allowable misfit variables are those with which a performance standard can be associated. Some of the most significant misfits (and those

most often neglected in traditional systems design) are not quantifiable. Consider another example from JUMPS: One of the functions of the system is to provide each member with the status of his pay and leave account (a Leave and Earnings Statement, or LES). The designer realizes that misfit can occur if the LES is not understandable to the member. There exists no well-defined scale along which to measure the "understandability" of the LES, but this makes the potential misfit no less significant. In this case the designer must attempt to develop, in a commonsense language, what LET format and content are required to avoid misfit. This may be accomplished by demonstration, or by determining some of the characteristics the LES need posses to render it understandable: only common abbreviations used, all transactions and payments identified by date and type, taxable and non-taxable items of pay clearly separated. The point here is that the designer must attempt to specify, as completely as possible, the conditions to be met to avoid misfit. The fact that for a qualitative variable such as this the conditions cannot be quantitatively stated does not imply that they cannot be well-understood. In Alexander's words, it is necessary that[27]

> each variable be specific enough and clearly enough defined, so that actual design can be classified unambiguously as fit or misfit.

(b) Form-independent. It is necessary that misfit variables selected be stated in such a way that they do not,

a priori, determine the design of the form. As an example of a misfit variable which is not form-independent, consider a JUMPS design specification for the production of the Leave and Earnings Statement:[28]

> The Leave and Earnings Statement will be produced solely from data contained in and controlled by the pay account maintained by the central site.

While this statement is well-understood, admission of it as a misfit variable may unnecessarily constrain the design. If the designer is to have the flexibility in design required to maximize fit along the many points of interaction on the form-context boundary, the misfit variables he considers must be insofar as possible, form-independent. That is, the structure of the system to be designed should not be directly motivated by the statement of a particular misfit variable. In this example, the designer must attempt to restate the specification by isolating the misfit it attempts to avoid, in a more form-independent manner. One possibility would be:

> The Leave and Earnings Statement should reflect the actual status of the member's pay account maintained at the central site.

This clearly provides the designer with greater latitude. Some information maintained at the local level might be included on the LES, or the LES might be produced locally based on information provided by the central site. (It may turn out, however, that the form suggested by the original specification is the one which best satisfies this and other misfit variables.)

Given these qualifications, the designer can begin to select misfit variables. The starting point in this process is the considerations of the existing form(s) which the system will replace (or from which it will displace function). Organizational components of the context are studied to isolate both points of misfit and fit with the current form for those interactions supporting existing functions which the system will assume. The designer will thus proceed around the selected form-context boundary, examining all information, control, financial, and other interactions "in place". At those points of misfit, the designer must attempt to make a specific statement of the misfit variable, one that is both well-understook and form-independent. In information interaction, for example, the designer must isolate the specific source of any current apparent stress: Does the misfit arise due to content, or is the problem due to incompatability between (existing) form(s) and context along some information characteristic dimension (timeliness, accuracy, frequency, level of aggregation)? The current points of fit at each interaction must also be evaluated, and a statement of potential misfit developed. For these existing interactions, the performance standard associated with the misfit variables is directly available from observation of the performance of the existing form(s).

For those functions which currently exist and will be displaced by the system to be implemented, misfit variables

can be selected based solely on conditions observable at the selected form-context boundary. But for new systems functions or for new systems support functions, the designer has no such facility. Alexander's approach in this case is less than precise:[29]

> if we try to design something for an entirely new purpose...the best we can do in stating the problem is to anticipate how it might possibly go wrong by scanning mentally all the ways in which other things have gone wrong in the past.

It can be argued that the major difficulty in computer-based systems design is in the specification of systems requirements for performance of new function, and that it is in this area that Alexander offers the least insight. We have, however, by extending Alexander's approach to general design to the specific design process in computer-based systems, provided some focus to his suggested search for potential misfit in new functional systems development. In selecting the form-context boundary, the designer has identified specific organizational components and specific types of interactions necessary to support the new function. He is aware of the ways in which misfit might occur in these interactions. In considering a new control interaction, for example, the organizational component (in the context) would be analyzed to determine the standards the system must meet in terms of such variables as robustness and communicability. A specific statement of the potential misfit must be made, together with the condition required to avoid misfit. In

addition to information and control, the designer is aware of and considers other interactions which have impact on the systems design--the legal or internal requirements the system must meet in performance of a new function.

What Alexander does provide, then, (as extended by the considerations developed above) is a framework which focuses the designer's attention on both the locality of potential misfit (the form-context boundary), and the types of potential misfit (the interactions between the form and context).

(4) JUMPS application. To illustrate the general approach just outlined, this section developes an initial set of misfit variables appropriate to the portion of the form-context boundary depicted in figure 4 above. Because a complete description of this boundary is not available, the misfit variables developed below are based on the JUMPS systems specifications listed in Appendix 1. As will be remembered, these specifications are not exactly aligned with the form-context boundary, but were based on the systems interface assumed in figure 1.

We can first consider the interactions between the Commanding Officer/Administrative Office and the manual payroll system which support existing functions which JUMPS will displace. We attempt to state, in as well-understood and form-independent manner as possible the conditions necess-

ary to avoid misfit in each interaction.

Pay authorizations: Pay authorizations are the means by which member's pay accounts are adjusted. In their preparation, two persons outside the administrative office may be involved: the Commanding Officer for certain pay authorization certifications (for those related to specialty or proficiency pay), and the member, when required to certify contract-related entitlements (such as those which may accrue to the member upon reenlistment). We include these "requirements" as the first two misfit variables:

1. Commanding Officers should certify certain pay authorizations.

2. Members should certify their elegibility for certain pay entitlements.

Another consideration is that pay authorizations must be easy to prepare in the administrative office. We make a statement of the potential misfit (but defer the "well-understood" qualification until later):

3. Authoriizing documents must be simple to prepare.

For reasons similar to these, other misfit variables pertinent to this interaction between JUMPS and its context can be stated:

4. Mail service should be employed as the mode of input from deployed units.

5. Pay authorization preparation should not be the highest priority work in the administrative office.

6. Administrative office personnel should not be required to receive formal training for pay authorization preparation.

7. The vehicle for pay authorizations must easily accommodate new or expanded reporting requirements (such as new items of pay).

8. Facility should be provided for reporting of pay authorizations applicable to all members at a given unit in a single transaction, when such authorizations are based on a change in the unit's status.

9. "Minimize" conditions must be observed in use of Naval messages.

10. Pay authorizations must be input(received by the central site) not later than three days after their effective date.

11. JUMPS should not require significant additional space in local administrative offices for records or equipment used in pay authorization preparation.

12. Sufficient documentation must be retained at the administrative office level to allow for audit and possible re-input of erroneous transactions.

13. Pay authorizations must be preparable based only on on-board information.

This first set of misfit variables relates only to the information interaction "Pay authorizations". Such an interaction exists in the current manual pay system between the Commanding Officer/Administrative office and the personnel office, the latter having now been included as part of the form to be designed. Although the interaction being considered is informational in nature, the content of the information is not given by the misfit variables developed above. Misfits related to content will be developed during consideration of the JUMPS-NAFC portion of the form-context boundary, since the NAFC determines the information required for each

item of pay and allowance, based on applicable legislation. These misfit variables relate generally to the conditions which must be satisfied to achieve fit in the operation of the administrative office. Some of the variables relate to the participants required for preparation of the pay authorizations (1,2,13); others to the modes and timing of the input (4,9,10); the allowable priority of the work and the capability of the personnel assigned to authorization preparation is addressed (3,5,6); the potential for misfit if the format of the input is not sufficiently flexible to allow changes in reporting requirements is covered (7); and other variables describe restrictions on space for equipment, documentation and forms (11), the need for local audit (12), and "unit authorization" (8).

This set of misfit variables represents the first attempt at identifying the potential misfit conditions relevant to an existing interaction. The next interaction considered, also informational in nature, supports a new system function--provision of member leave and pay account status to the Commanding Officer/administrative office.

Account status:

14. Leave and pay account status must be provided at least monthly.

15. Leave account status should include, by member, current leave balance, leave taken, and leave lost.

16. Pay account status must include, by member, pay and allowance items authorized, foreiture or checkages of pay, and all one-time entitlements (such as bonuses).

17. Information should be provided to verify all member receipts/detachments.

18. Reenlistment eligibility must be provided, by member.

19. Account status should be easy to interpret.

The content of the information provided can be directly specified here, as can its timing and aggregation, since this will not be specified at any other point on the form-context boundary. We consider next the control interactions (neither previously existing) between the Commanding Officer/ administrative office and JUMPS.

Payday schedule:

20. Commanding Officers have the option of rescheduling regular paydays based on operational unit schedules.

21. Rescheduling requests should be directed to the disbursing office providing support to local unit.

22. Rescheduling requests should include a unit identification and a requested date of payment.

Error notice:

23. Erroneous or questionable entitlement authorizations should be returned to the local unit at which correction can be accomplished.

24. Request for correction should be specific as to the condition for rejection by the central site.

25. Correction of erroneous input must be accomplished within the same time frames as new input.

26. Feedback should be provided to local units regarding the accuracy and timeliness of authorization preparation.

Having developed an initial set of misfit variables describing the Commanding Officer/administrative office-JUMPS interactions, we next consider interactions between JUMPS and the member:

Member change requests:

27. Members must have easy access to JUMPS to initiate pay account changes (allotments, withholding rates).

28. Members should need to have no formal training in military pay procedures.

Payments:

29. Members should be able to specify the mode of each payment (either cash or check).

30. Pay amounts should be predictable.

31. The Disbursing Officer under whose symbol payments are made should be solely accountable for their propriety.

32. Authorized special payments should be accomplished on the day of request.

33. A member should be able to designate a recipient of his pay or portion thereof.

34. Paydays should be regularly scheduled.

35. Members in transit (between duty stations) should be able to receive payment of any and all monies which may be due them.

36. A member should not be required to receive all pay due him on any payday.

37. Members should be able to request that their pay be deposited directly in a financial institution.

Account status:

38. Members must be provided with the status of their pay and leave accounts at least monthly.

39. Pay and leave account status must be sufficient to predict pay due and leave available for at least the month after issue.

40. Account status must be easy to understand.

41. Account status must be identical to that provided to the Commanding Officer/administrative office (for similar data items).

42. Distribution of account status to members cannot require a significant effort at the local unit.

Special pay requests:

43. Approval of special pay should be the perogative of the Commanding Officer.

44. Special payments must be limited to monies actually earned through date of payment.

45. Special payments should be normally discouraged.

These 45 misfit variables represent an initial estimate of potential conditions for misfit along a small portion of the form-context boundary. The number of misfit variables will increase significantly as interactions with other components on the boundary are considered, particularly those on the JUMPS-NAFC and JUMPS-MAPMIS portions of the boundary. When all portions of the boundary have been considered, the process of misfit variable selection ends with a collection of well-understood, form-independent statements of potential misfit. The collection will be considered the set of misfit variables for the design problem at hand. The structure of

the problem is in a sense estimated by this set of misfit variables, and, as will be seen, by the interdependencies among these.

d. Derivation of a "suitable set" of misfit variables. Before the dependencies between the selected misfit variables are estimated, it may be necessary to modify these to meet the restrictions imposed by the formal decomposition algorithm to be applied. To support the assumptions and restrictions of the algorithm employed by Alexander, the selected set of misfit variables must meet three conditions, which is achieved by manipulation of the initial set of misfit variables into a "suitable set".

(1) Equal scope. As described above, the algorithm for decomposition of the set M (the misfit variables) employed be Alexander treats each member of this set as a binary stochastic variable. With each variable, X_i, there is associated a probability of misfit ($P(X_i=0)=p_i$). It is clear, in a real-world context, that achievement of fit is not equally probable for all selected misfit variables. However,[30]

> If we allowed the p_i to be different for different variables X_i, we should have to bring this into the following analysis, which would lead to very complicated equations, and make it impossible to find a simple and general basis for decomposition.

Therefore we restrict p_i to be equal for all i. What this

implies is that, of all possible forms of which the designer can conceive, the fraction which achieves fit should be roughly equal for each selected misfit variable. The variables, in other words, need to be selected so that they are approximately equal in scope. Consider two JUMPS misfit variables included in the initial set developed above:

33. A member should be able to designate a recipient of his pay or a portion thereof.

37. Members should be able to request that their pay be deposited directly in a financial institution.

That these variables are not equal in scope can be demonstrated by considering the nature of the forms required to achieve fit at each misfit variable. Because the first is broader in scope than the second, any form resulting in fit with the first insures fit with the second: If a member can designate (any) recipient of (any) part of his pay, he can certainly have his pay deposited at a financial institution. The converse, however, is not true. To meet this restriction of "equal scope", the original misfit variables must be restated:

33a. A member should be able to allot any (fixed) portion of his pay on a continuing basis to a designated individual or institution.

37a. Members should be able to request that the balance of unalloted pay due be deposited directly in a financial institution on a continuing basis.

Restatement of misfit variables to achieve equality of scope is the first step in the manipulation of the initial set of misfit variables into a suitable set.

(2) Partial independence. We will be interested in estimating the degree of interaction between selected misfit variables. In Alexander's decomposition algorithm, these interaction estimates will be taken (after normalization) to represent the two-variable product moment correlation between pairs of stochastic variables. The mathematical treatment of decomposition requires that this correlation be suitably small. What this implies to the designer is that[31]

> we must be satisfied that the variables are as independent as we can get them to be.

For example, two of the initial JUMPS misfit variables are:

> 30. Pay amounts should be predictable.
>
> 39. Pay and leave account status must be sufficient to predict pay due and leave available for at least the month after issue.

As currently stated, the two misfit variables are highly dependent, although they were generated to accommodate fit along two independent dimensions: to insure against arbitrary variability in pay amounts, and to allow a member to predict the amount of pay he is due based on a statement of his account status. We might therfore restate the first variable:

> 30a. Payments to a member should not vary significantly between paydays, excepting variations attributable to new transactions against the member's account.

The second step in establishing a suitable set may require restatement of the initially selected variables so that they are at least partially independent.

(3) Specific and detailed. Implied in both the above conditions is the notion of specificity of the misfit variables. The statements of misfit must be as specific as possible for two reasons: First, Alexander's decomposition algorithm ignores any third- or higher-order interactions between variables. What this implies is that[32]

> the two-variable correlation for any pair of variables must be independent of the states of all other variables. Since the state of one variable is most likely to affect the correlation between other variables, if that one variable is wide in scope the best we can do in satisfying this is to make all the individual variables as specific and minute as possible.

Second, the broader the statement of misfit, the less removed the decomposition will be from the designer's initial "verbal concept" picture of the problem. The designer must question each misfit variable identified to see if a more specific statement (or several statements) of misfit can be made. Consider the JUMPS misfit variable:

3. Authorizing documents must be simple to prepare.

The variable is too general to provide much insight into the specific nature of the potential misfit. After further investigation into the demands of the context, we could expand this variable to detail more precisely the points of potential misfit:

3a. A common format for all pay authorizations should be utilized.

3b. An error recognized during pay authorization preparation should be correctable "in place".

3c. Abbreviations and codes used in pay authorization preparation should be easily understandable.

3d. Facility should be provided for one-time entry of identical data elements applicable to a series of pay authorizations which are prepared at the same time.

3e. Pay authorizations should be event-oriented, rather than entitlement-oriented.

This last misfit variable requires some explanation. In a military pay environment, an "event" is defined as a change in a member's status: a member reports to a new duty station, is promoted, or goes into an unauthorized absence status, for example. Several different entitlements (items of pay) may be affected by a change in the member's status. The event of "a member reports for duty aboard a submarine" will begin credit for Sea Duty Pay and Submarine Pay, and terminate credit for Subsistence Allowance. In general, the specific effects of an event on the set of entitlements due a member can only be determined by application of the complicated and detailed pay procedures then in effect (as published by the Navy Accounting and Finance Center). By requiring pay authorizations to report only events, rather than specific changes in entitlements generated by the event, the degree of training and expertise required in the administrative office is reduced, and the preparation of pay authorizations is simplified.

The designer's last task in establishing a suitable set of misfit variables is to review those variables initially

identified and restate these in the most specific and detailed manner possible:[33]

> The more specific and detailed we make the variables, the less constrained G(M,L) will be by previous conceptions, and the more open to detailed and unbiased examination of its casual structure.

We have begun above to manipulate the initial set of JUMPS misfit variables into a suitable set, one which conforms to the constraints imposed by the algorithm to be employed for its decomposition. We can proceed in this fashion, modifying the initial set to meet the conditions of equality of scope, partial independence, and specificity required. In addition to the modifications made above, the following additional variables are derived:

19a. Account status provided to administrative offices should use only common abbreviations and codes.

19b. Automatic entitlement changes (pay and allowances related to longevity) must be clearly indicated.

19c. Extraordinary account status conditions should be highlighted (excess leave, overpaid status).

19d. A clear indication of the origin of a current change in the member's account should be provided.

19e. Summary or trend information regarding the rate at which members assigned to a unit use leave should be provided.

27a. Members should not need to be aware of the current status of their accounts to request allotment or withholding rate changes.

27b. Members should not be required to use special equipment to initiate pay account changes.

30a. Payments to a member should not vary significantly between paydays, excepting for variations attributable to new transactions against the members' account.

40a. Disposition of all amounts withheld from a member's pay must be clearly indicated (taxes, allotments, insurance, and other checkages).

40b. Codes and abbreviations used is describing account status should be easily understood or explained on the document containing the status.

40c. Explicit statement of all payments made (both locally and centrally) as to date and amount must be provided.

40d. Expiration date of limited entitlements must be indicated.

40e. "Balance brought forward" and "balance carried forward" amounts must agree on successive reports of pay account status.

The original 45 misfit variables have been expanded to the 68 now in the suitable set. This example illustrates that derivation of a suitable set of misfit variables can require significant manipulation of the originally selected variables. Moreover, the 68 misfit variables identified here only describe a small portion of the JUMPS-context interface. The JUMPS-NAFC interface, or boundary, will contain a much larger number of potential misfit variables, as it is across this portion of the form-context boundary that the system's fit with the legal requirements for certification and processing of all pay and allowance items is specified. While derivation of the complete set of misfit variables for the JUMPS application is beyond the scope of this paper, some idea of the potential number of these

is available from review of the JUMPS systems requirements. At the JUMPS-NAFC interface alone, approximately 16,000 seperate processing requirements would need to be considered.[34] While this number might be reduced somewhat by careful selection of misfit variables, the number of misfit variables in the complete set for this design problem is likely to exceed 20,000. Thus the designer's task in establishing a suitable set of misfit variables is likely to be a far more substantial undertaking than this abbreviated JUMPS example may indicate.

e. Misfit variable interactions. The most significant distinction betwen Alexander's proposals and traditional systems specification definition is the explicit recognition by the former of the interrelationships between design requirements. It is the articulation of these interralationships which define the set of links, or misfit variable interactions that in turn provides structure to the design problem and enables its subsequent decomposition. This section investigates the nature of these links in a computer-based systems context and suggests an approach for use in estimating the links between bariables in the suitable set.

(1) The nature of misfit variable interactions. The notion that achievement of fit at one point on the form-context boundary may make it easier or more difficult to achieve

fit at another is a common one to most designers. In computer-based systems, we can consider the concepts of "tradeoff" and "concurrence". For example, a common tradeoff facing a programmer is between execution time and memory requirement for a particular block of code. From a slightly broader perspective, he may be concerned with the tradeoff between efficiency of code and maintainability. A designer may consider two requirements: one to provide a user with a facility for accessing the "current time", and another to develop a mechanism to allow for charging of computer usage. The designer in this case may see a concurrence between the two requirements, in that a form which accomplishes the first (typically through a macroinstruction facility) will make it easier to accomplish the second (by basing usage on time and using the same facility to determine this).

The common thread of these examples is that tradeoffs and concurrences, or interactions between sources of potential misfit, exist because of the structure of the form employed. In other words, the designer, through experience, realizes that the logical or physical structure of the forms he has constructed in the past have been such that tradeoffs or concurrences between different requirements have been evident.

Interactions motivated by logical structure: Here the designer may consider interactions possible due to the nature

of the logical components chosen to perform a given function. The designer may be considering the tradeoff between "ease of use" and "flexibility" in a control interaction at some point between form and context. A system designed to support the first might be implemented as a small set of cammand variables, invoked by use of a lightpen on a video display screen. The conflict between this logical implementation and the requirement for flexibility is evident: the user's scope of control is limited to a specific set of interactions with the system. The conflict, or tradeoff, arises through choice of form. The designer is sware of requirement interaction because the logical structure of the forms available suggest such an interaction.

Interactions motivated by physical structure: Consider for example a systems implementation on a processor operating under a fixed partition, multiprogramming operating system. The designer may be considering two requirements: First, that an internal automated routine be provided for correction of erroneous transactions, and, second, that the system not require operator intervention for scheduling. The designer may see an interaction (in this case a tradeoff) between these two requirements with which the physical implementor (the programmer) will be concerned. The degree of automated correction achievable will depend in part on the size (in terms of memory requirements) of the error correction routines. Operator intervention will be required to reallocate partitions if the

program size exceeds the largest partition allocated in normal operations. A tradeoff is required here, but the necessity for it is based on the underlying technology, or physical structure, of the form. If the operating system employed an alternative memory management scheme (demand paging, for example), no operator intervention would be required regardless of the size of the error correction routines. In this case, there would be no interaction between these two requirements, although other interactions, appropriate to the technology of this form, may be evident.

These examples are insufficient to define formally the nature of misfit variable interaction in computer-based systems, but they do suggest that such interactions exist, and are articulated through the designer's experience with the technology and logical properties on which these systems are based. In Alexander's words,[35]

> We shall say that two variables interact if and only if the designer can find some reason (or conceptual model) which makes sense to him and tells him why they should do so.

(2) Estimation of misfit variable interactions. Having defined a suitable set of misfit variables, and being aware of the nature of the potential interactions between its members, the designer is in a position to estimate these, to establish the set of links L. Hierarchical decomposition allows the designer to consider both the direction and magnitude of

these links, by assigning a signed weighting to each interaction he sees possible. The strength of interaction can be viewed as the potential for concurrence or conflict (tradeoff) in achievement of fit for two variables based on the nature of the forms available. If the designer considers that the interaction is significant for only a few types of possible form designs, he may consider the interaction a weak one. If almost every conceivable form results in a conflict or concurrence between the two variables, the interaction may be considered strong. It is of course theoretically possible to evaluate the strenght of interactions far more specifically. Alexander suggests, however, that[36]

> In practice we shall, at best, be able to distinguish two or three strengths of interactions.

We will consider only two strengths of interaction between misfit variables of the suitable set: "weak" and "strong". This distinction is an arbitrary one, but one which intuitively may be the easiest to implement. It is also noted that the direction of the interactions (conflict or concurrence) need not be specified for decomposition by the algorithm employed by Alexander.[37] While the reason for this is embedded in the mathematics of the decomposition algorithm, the result is intuitively appealing, in that we would expect the decomposition to depend only on the magnitude of the interaction between variables, rather than on the direction of interaction.

(3) JUMPS misfit variable interactions. We estimate the interactions between the variables in the suitable set by considering each variable's potential for concurrence or conflict with every other variable. To illustrate this, we will consider only the interactions between variables in a subset of the JUMPS suitable set of misfit variables selected above. This particular subset was chosen because the density of the links between variables is such that the set is amenable to decomposition. This subset will include the variables:

3a. A common format for all pay authorizations should be utilized.

3e. Pay authorizations should be even-oriented, rather than entitlement-oriented.

4. Mail service or Naval messages should be employed as the mode of input from deployed units.

6. Administrative office personnel should not be required to receive formal training for pay authorization preparation.

7. The vehicle for pay authorizations must easily accommodate new or expanded reporting requirements (such as new items of pay).

8. Facility should be provided for reporting of pay authorizations applicable to all members at a given unit in a single transaction, when such suthorizations are based on a change in the unit's status (such as upon entering a combat zone).

9. "Minimize" conditions must be observed in use of Naval messages.

13. Pay authorizations must be preparable based only on on-board information.

23. Erroneous or questionable entitlement authorizations should be directed to the local unit at which correction can be accomplished.

24. Request for correction should be specific as to the condition for rejection.

25. Correction of erroneous input must be accomplished within the same time frames as new input.

We begin by considering potential interactions between misfit variable 3a, the need to employ a common format for pay authorization input, and each other variable in the set. This is accomplished by construction, if possible, of a conceptual model in which avoidance of misfit at variable 3a would make avoidance of misfit at another variable easier or more difficult to achieve. Such a model can be constructed which "links" variable 3a with variable 4, the need to utilize mail or Naval messages as the mode of input from deployed units, motivated by the underlying physical structure of components of the design problem. The "link" is based on the fact that Naval message transmission equipment can be programmed easily to accommodate standard format messages, reducing message processing time and message transmission errors. In this case, the link between variables is a concurrence. We estimate the strength of the link to be strong, since this concurrence is evident in all possible forms, or possible systems designs which could be employed. Variable 6, the need to avoid requiring formal training of administrative office personnel, concurs with variable 3a. This link is based on an intuitive assumption that preparation of input in a single

format can be more easily learned than use of multple-formatted input documents. The strength of this link is again estimated to be strong, as this concurrence will be evident in almost all possible forms. There is also a weak conflict between variable 3a and variable 7, the need for flexibility in the vehicle selected for pay authorization input. This conflict is evident based on the logical structure of the design problem: We reason that use of a single format for input will render the future inclusion of new or expanded input requirements more difficult. But because there are techniques available to achieve fit in both variables simultaneously (perhaps through multiple-use, variable length fields on input records), this conflict is evident in only some possible forms. The link is therefore weak. There appear to be no interactions between variable 3a and the other misfit variables in the set. Consider for example the potential for interaction between variable 3a and variable 13, the need to prepare pay authorizations based only on on-board information. There seems to be no conceptual model, either logically or physically based, in which achievement of fit at one variable would make achievement of fit at the other either more or less difficult. The need for a single format for pay authorization input and the need to prepare this input based only on on-board information appear to be totally independent, so no link can be established. (A point of note here is that this independence is based on

the underlying assumption that each personnel/administrative office has the same on-board information and will use the same single format for input. These assumptions are, in this case, valid, but one could easily imagine a situation--a design problem--in which these two variables could interact). From this point on, we will not consider "non-links", but confine the discussion to only those misfit variables which can, in the JUMPS design problem, be linked through some conceptual model. Variable 3a interactions can be summarized as:

3a interacts with 4(+S), 6(+S), and 7(-W).

We next consider the potential interactions with the second misfit variable in the set: 3e, the need for pay authorizations to be event-oriented. The first interaction is with variable 6 (no formal training). Based on logical considerations, we can reason that, since reporting of events (relatively few in number) is easier than interpretation of the effect of the event on members' sets of entitlements (which are numerous and complex), the two variables concur. Since this reasoning would apply to all possible forms, the interaction is strong. A weak concurrence between variable 3e and variable 7 (flexibility in input) is posited: Reporting requirements (authorizations) for entitlements are subject to change more often than reporting requirements relating to events. (This is based on the fact that there exists a

relatively fixed number of events--member reports for duty, is detached from duty, moves into government quarters, etc. Changes in pay procedures in general modify only the "mapping" of events into entitlements. It is therefore possible to modify the entitlement structure completely and retain the same reporting of events. The event-entitlement "mapping" is achieved in an automated fashion at the central site, rather than at the local level.) Thus the need to allow for modifications in reporting requirements is facilitated by employment of even-oriented input. The interaction is weak in that future pay procedure changes could include definition of new events. We also estimate there to be a strong concurrence between varialbe 3e and variable 8, the need to provide a facility for pay authorization input based on a change in a unit's status. Because a unit status change is interpretted as an event, this condition is most compatible with an even-oriented reporting system. Variable 3e also concurs with variable 13, the need to prepare pay authorization input based only on on-board information. Since events are well-defined and independent of a member's set of current entitlements and previous status, no reference to a member's account status need be made to prepare input. The interaction however, is weak is that such information could be made available to the local administrative office. Variable 3e interactions can be summarized as:

3e interacts with 6(+S), 7(+W), 8(+S), and 13(+W).

Considering next variable 4, the need to utilize mail or message input from deployed units, we estimate an interaction between it and variable 9, the need to observe "Minimize" conditions. ("Minimize" is a condition of radio silence employed in war time circumstances or in exercises designed to simulate these. Varying degrees of "minimize" exist, all of which prohibit the transmission of routine administrative "traffic", which includes pay and personnel related messages.) There is an obvious conflict between these variables, but its strength is weak, in that mail-based input is available as an alternative during "minimize". There exists a strong conflict, however, between variables 4 and 25, the need to accomplish erroneous input turnaround within three days. Mail service to deployed units is notoriously poor (except for those units with a high degree of air support, such as aircraft carriers). Variable 4 interactions can then be given as:

4 interacts with 3a(+S), 9(-W), and 25(-S).

Turning to variable 6 (no formal training), we posit a concurrence with variable 8 (facility for reporting unit status changes). The conceptual model on which this interaction is founded is based on the structure of the pay procedures. Some unit status changes affect entitlements for only a portion of the members assigned to a unit. (For example, when a ship enters dry dock for a period in excess of 90 days, Sea

Duty Pay is terminated for all enlisted members, but officers, who are never entitled to Sea Duty Pay, are unaffected.) By allowing for a single report of a change in unit status, administrative office personnel are not required to determine which members of the crew underwent an "event", which reduces requirement for formal training. The concurrence is weak in that a system which allowed for separate input for all members assigned could be achieved (with "event applicability" determined at the central site). Variable 6 interactions are:

6 interacts with 3a(+S), 3e(+S), and 7(+W).

All variable 7 (flexibility in input) interactions have already been identified:

7 interacts with 3a(-W) and 3e(+W).

Variable 8 (facility for reporting unit status changes) interacts with variable 24, the need to identify the specific condition for input rejection. The model on which this interaction is based is less straightforward than those employed above (and in fact may only be evident from the problems actually encountered in the operation of JUMPS--a luxury not available to the original JUMPS systems designers). A common unit status change report is "Squadron S reported for duty aboard aircraft carrier on date D". In attempting to "post" such a transaction to each member's account at the central site, it may be determined that one member of Squadron S was already in a duty status aboard the same aircraft carrier on

the date of the "event". The transaction is rejected, but the cause of the rejection is not specific. The member may have reported early as part of the squadron vanguard and this event reported; the member may not have been properly detached from previous duty aboard the aircraft carrier; the member may have been part of an administrative or logistics unit left aboard the carrier between squadron deployments. In any case, use of unit status change input complicates the error resolution process significantly, and the interaction between these two variables is a conflicting one. The interaction is weak, because suitable modification to the "unit change" input (such as exception basis reporting) could alleviate this problem. Variable 8 interactions are:

8 interacts with 4(+S), 6(+W), and 24(-W).

Variable 9 ("minimize") interacts with variable 25 (error turnaround requirements), for reasons already stated. The interaction is a strong conflict because any system including message-based input is unlikely to meet the error turnaround requirements during "minimize" conditions.

9 interacts with 4(-W) and 25(-S).

Variable 13 (pay authorization preparation based only on on-board information) interactions have already been identified:

13 interacts with 3e(+W).

The need to return erroneous input to the local unit at which correction can be accomplished, variable 23, concurs

with variable 24 (identify specific reject condition). A system which can detect the reason for rejection specifically facilitates identification of the local unit at which correction can ba accomplished. The interaction is weak, because a system could be implemented which returned erroneous input to all local units which might possibly be able to correct the error. Variable 23 also concurs weakly with variable 25 (error turnaround requirements), in that this requirement will more likely be achieved if the appropriate local unit to which the error should be returned can be identified:

23 interacts with 24(+W) and 25(+W).

Finally, variable 24 (identify specific reject condition) concurs strongly with variable 25 (error turnaround requirements), for obvious reasons.

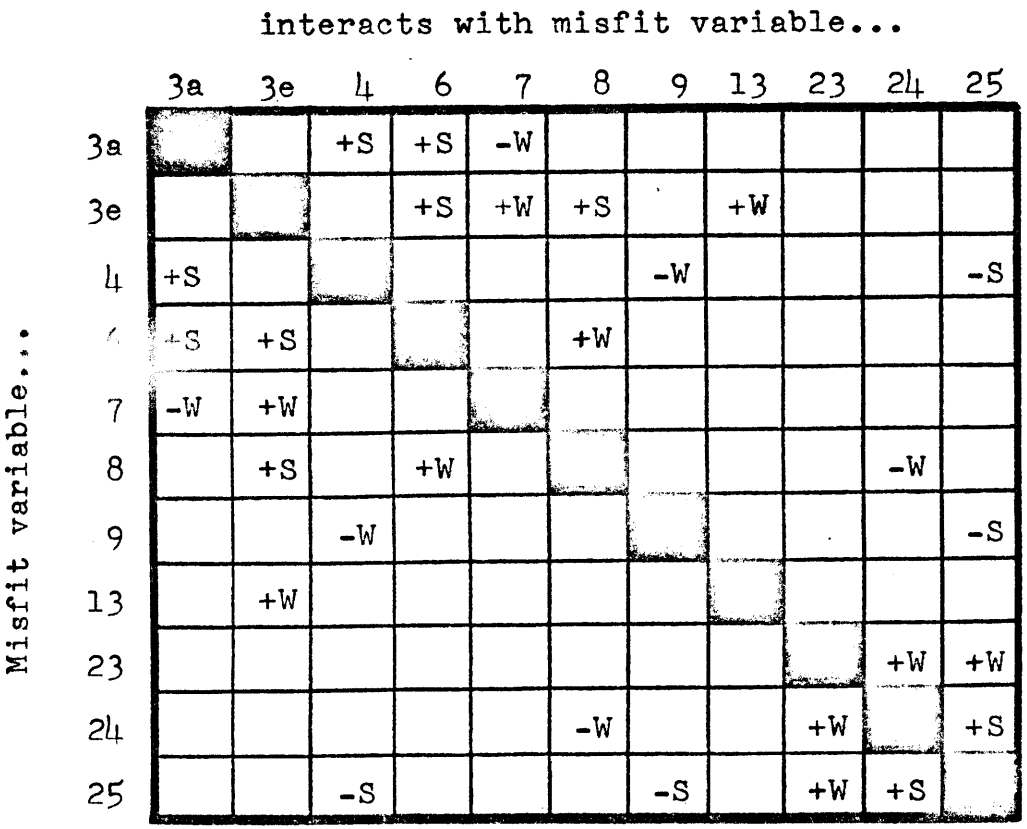
24 interacts with 8(-W), 23(+W), and 25(+S).

The interactions concerning variable 25 (error turnaround requirements) have already been identified:

25 interacts with 4(-S), 9(-S), 23(+W), and 24(+S).

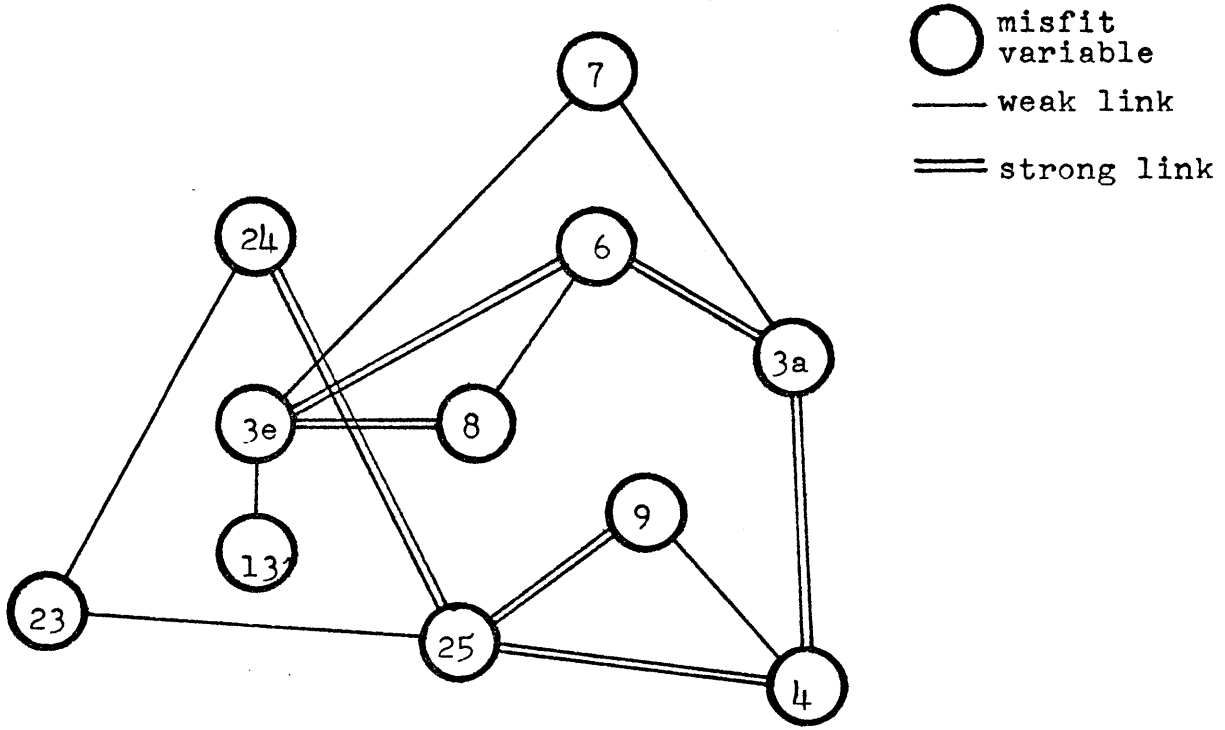
A summary of the interactions between variables of this subset of JUMPS misfit variables is provided in tabular form in figure 5.

Having identified both the elements of the set M, and the set of interactions between these, L, a graph of the design (sub)problem can be drawn, as in figure 6. The misfit variables are identified as nodes of the graph, and the inter-



JUMPS Misfit Variable Interactions

Figure 5

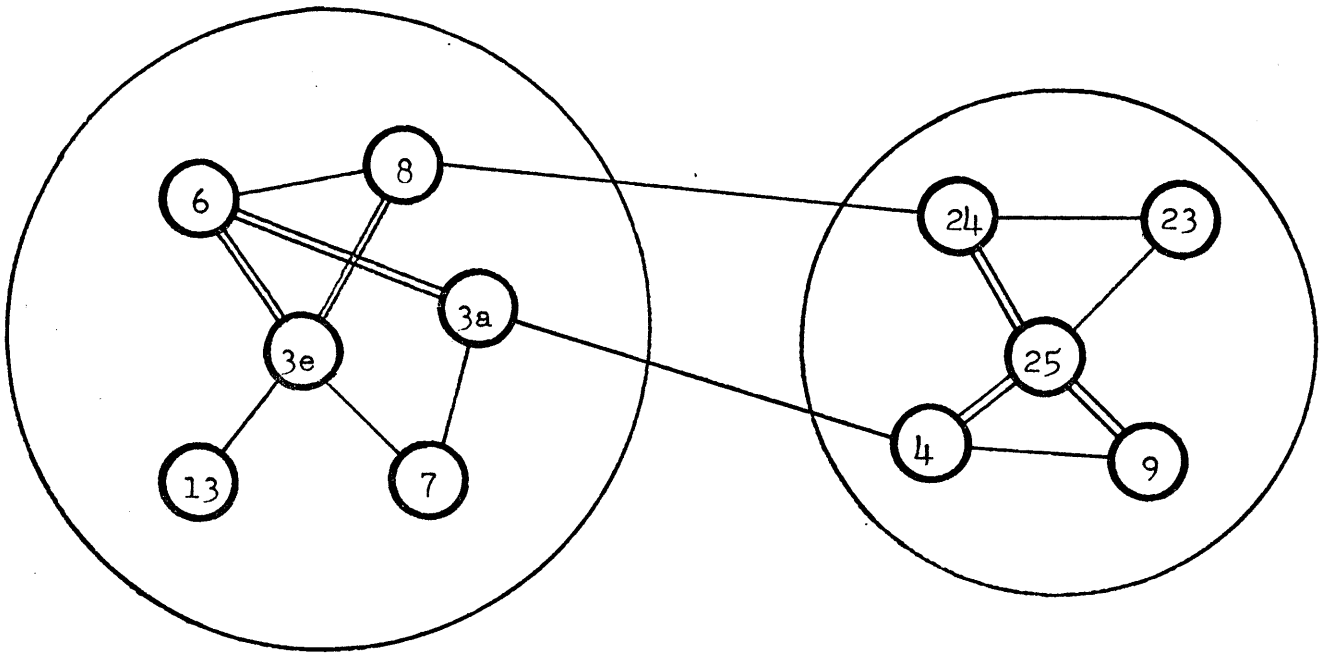


G(M,L) for JUMPS Design Problem

Figure 6

actions as the links between them. Figure 6 provides little insight into the structure of the design problem, but by redrawing the graph, as in figure 7, two clusters of misfit variables become evident. (This simple, first-level decomposition is accomplished informally, but by inspection it is evident that figure 7 represents the best two-way partitioning of the subset of JUMPS misfit variables. Application of Alexander's algorithm would yield identical results.)

Figure 7 therefore suggests that the designer can consider the two design problems independently, concerned only with designing form to achieve fit within each of the separate subgroups of misfit variables. It can be argued that this decomposition provides the designer with little additional insight, in that this particular segmentation of design requirements is obvious without the mechanics of misfit variable selection, interaction estimation and decomposition. The reason for this is that the subset of JUMPS misfit variables used in this example reflect only a small number of very localized misfit variables. Decomposition of the entire set of JUMPS misfit variables may result in variable clusterings which are far less intuitive than that indicated by this example. (It is interesting, even in this example, however, that the misfit variables regarding the mode of JUMPS input (4 and 9) are associated with input error variables rather than initial input variables.) This example is not sufficiently



G(M,L) for JUMPS Design Subproblem
Showing Best Two-way Partition

Figure 7

broad to demonstrate any significant deviation from the "preclustering" suggested by the systems specifications listed in Appendix 1.

One additional and significant point of note is the number of misfit variable interactions which the designer must consider in deriving the set of links, L, for the decomposition. This number is given by n(n-1)/2, where n is the number of misfit variables in the suitable set. For this 11 variable JUMPS example, 55 potential links were evaluated, of which 15 were considered significant and included in the set L. Alexander provides a larger example[37] which includes 141 misfit variables; 1394 links were included in the set L for this problem (of a potential total of 9870). For a vary large suitable set of misfit variables, the potential number of links which must be evaluated is very high. Based on an estimated 20,000 variables in the complete JUMPS suitable set, approximately 200 million links would need to be evaluated. While a large number of these might be rejected out of hand, the actual number of links established and included in the set L is still very large. In the above example, the "link density" (actual links in L/potential number of links) is 0,27. In Alexander's larger example, the link density is 0.14. These two examples alone are insufficient to establish a general relationship between link density and the size of the suitable set. However, even a link density as low as 0.005 (28 times

less dense than in Alexander's example) would result in approximately 1 million links in L for the entire JUMPS design problem. Even if only these 1 million links required active consideration by a JUMPS designer, and assuming they could be each evaluated at the rate of one per minute, the total task would require more than 16,000 man-hours. From even these rough estimates it is obvious that Alexander's methodology for hierarchical decomposition is, at best, awkward when applied to very large systems.

IV Summary and Conclusions

This paper has demonstrated a partial application of hierarchical decomposition to the design of computer-based systems. Figure 8 summarizes the activities involved in the application, and their objectives and criteria. Many of the benefits to the designer of computer-based systems have already been discussed. There are two other benefits of this methodology, however, which deserve mention.

First, hierarchical decomposition supports, rather than replaces, the heuristic approaches to design already available. We have seen, for example, that the work of Little (regarding important dimensions in form-context control interactions) and Gorry and Scott Morton (important dimensions in information interactions) provide a starting point for the identification of misfit variables. It has also been suggested that the process of estimating links between misfit variables offers the designer a formal vehicle by which his experience can be productively brought to bear on the design problem at hand. What Alexander's methodology represents is not a new and independent approach to design, but rather a technique through which both existing heuristics and experience can be more formally expressed. As such, it can perhaps best be viewed as an effective facilitating vehicle for design.

Second, it was suggested above that the context of a

1. Functional Specification.

Objective: Identify the functions which the system will perform

Criteria: Consider:
- -displacement of existing function
- -performance of new function
- -new systems support function

2. General Boundary Positioning.

Objective: Identify organizational components composing the context of the design problem

Criteria: Context is formed by those organizational components outside the designer's control which:
- -currently interact with the organizational components whose function(s) the new system will displace
- -will interact with the new system either in performance of new function or for new systems support

3. Specific Boundary Positioning.

Objective: Identify specific interactions between form and context

Criteria: Interactions may be:
- -information
- -control
- -financial
- -other

Hierarchical Decomposition Summary

Figure 8

4. Misfit Variable Selection.

Objective: Identify points of potential misfit in each form-context interaction

Criteria: Misfit variable is a statement of a conditions which if not met results in stress in the ensemble. They must be:
-well-understood
-form-independent

5. Suitable Set Manipulation.

Objective: Restatement of selected misfit variables to conform to requirements of decomposition algorithm to be employed

Criteria: (For Alexander's decomposition algorithm):
-equal scope
-partial independence
-specific and detailed

6. Misfit Variable Interaction Estimation.

Objective: Estimation of dependencies between all pairs of variables in the suitable set

Criteria: -A pair of misfit variables is "linked" by some conceptual model available (introspectively) to the designer, based on the logical or physical problem structure
-The strength of a link is determined by the relative number of possible forms in which the conceptual model holds

Hierarchical Decomposition Summary

Figure 8 (continued)

7. Decomposition.

Objective: Partitioning of the suitable set of misfit variables into nearly independent subsets, each representing a design subproblem

Criteria: Dependent on the decomposition algorithm employed

Hierarchical Decomposition Summary

Figure 8 (continued)

computer-based system, rather than representing a stable environment, is subject to considerable change over time. It is now also recognized that computer-based systems have relatively long lives--that once implemented, they are expected to remain in place in an operational status for some time. These two observations suggest that a successfully designed form must be easily adaptable to its changing environment. As Myers observes,[39]

> we can say that programs never achieve stability. They never achieve freedom from bugs or from additions or from changes.

Simon has speculated[40] that successful adaptation, or evolution of form is facilitated if it is structured into fairly independent subsystems. Such a structure would allow for change to take place at a subsystem level, rather than require extensive adaptation by the entire form. The objective of hierarchical decomposition is the identification of nearly independent subsystems in the design problem. In a more specific statement applicable to computer-based systems, Myers has defined the dimensions along which such independence is desirable: independence in informational and control interactions between subsystems.[41] Since we have based definition of misfit variables in exactly these terms, we can expect that the resultant subsystems will be as independent as possible along these dimensions. What this implies for computer-based systems design is that the solution to the design problem has been constructed in such

a fashion so as to most readily allow for system maintenance (correction of misfits) and expansion (to accommodate change in the context).

While it has been demonstrated that hierarchical decomposition can be applied to computer-based systems design, it is apparent from the JUMPS example that this application may, in some cases, be too lengthy a process to be practical. The practicality of the application would appear to depend largely on the nature of the computer-based system to be designed. We can speculate that large systems, in which many specific processing requirements must be considered (such as JUMPS), are not amenable to Alexander's methodology, based on the sheer number of misfit variables and variable interactions which must be defined or estimated. In smaller systems, or ones in which such detailed requirements are not imposed, hierarchical decomposition may enjoy more practical application.

In summary, this paper has suggested the potential for the application of hierarchical decomposition to computer-based systems design; a complete demonstration of this application is, however, still wanting. Many questions and issues remain to be explored in this area, principal amoung them the characteristics of the design problem necessary for practical application of hierarchical decomposition. Efforts to answer these questions, could, however, result in the availability of a powerful design-aiding methodology for the

computer-based systems designer.

V Notes and References

[1]Christopher Alexander, Notes on the Synthesis of Form (Cambridge: Harvard University Press, 1967).

[2]Ibid., p. 15.

[3]Robert G. Murdick, "MIS Development Procedures," Journal of Systems Management, Dec. 1970, pp. 22-26.

[4]Ibid., p. 24.

[5]Ibid.

[6]Gordon B. Davis, Management Information Systems: Conceptual Foundations, Structure, and Development (McGraw-Hill, 1974), p. 425.

[7]Glenford J. Myers, Reliable Software Through Composite Design (Petrocelli/Charter, 1974).

[8]John F. Rockart, "Model Based Systems Analysis: A Methodology and Case Study," IMR (Sloan Management Review), Vol. 11, No. 2 (Winter 1970), pp. 1-14.

[9]William R. King and David I. Cleland, "The Design of Management Information Systems: An Information Analysis Approach," Management Science, Vol. 22, No. 3, pp. 286-297.

[10]Both Myers (7) and W. P. Stevens, et. al., "Structured Design," IBM Systems Journal, Vol. 13, No. 2 (Spring 1974), pp. 115-139.

[11]O. J. Dahl, et. al., Structured Programming (London: Academic Press, 1972).

[12]D. Teichroew, "Problem Statement Analysis: Requirements for the Problem Statement Analyzer (PSA)," ISDOS Working Paper No. 43, Dept. of Industrial Engineering, Univ. of Michigan, Ann Arbor, 1971.

[13]Alexander, p. 18.

[14]Requirements and Design for JUMPS in the 1980's," Military Pay Systems Department, Navy Accounting and Finance Center, Washington, D.C., 1976, p. 1.

[15] *Ibid.*, p. 98.

[16] Department of the Treasury Transmittal Letter No. 53, Sept. 28, 1970.

[17] Department of Defense Instruction No. 7330.4, Nov. 7, 1966.

[18] See for example G. Anthony Gorry and M. S. Scott Morton, "A Framework for Management Information Systems," *Sloan Management Review*, Vol. XX, No. X (Fall 1971) pp. 55-70, or Davis (6), p. 207.

[19] Framework proposed by R. N. Anthony, *Planning and Control Systems: A Framework for Analysis* (Boston: Harvard University Graduate School of Business Administion, 1965)

[20] Davis (6), p. 208.

[21] J.D.C. Little, "Models and Managers: The Concept of a Decision Calculus," *Management Science*, Vol. 16, No. 8 (April 1970), pp. B-466-B-485.

[22] "Policy and Procedures Manual for Guidance of Federal Agencies," General Accounting Office, Washington D.C., Titles 2 and 6.

[23] "Management of Automated Data Systems Development," Department of Defense Instruction 5010.27, July 8, 1970.

[24] Department of Defense Instruction 7330.3 (17), superseded by Department of Defense Instruction 7330.4a, July 1, 1977.

[25] *Ibid.*, pp. 2, 24.

[26] Alexander (1), p. 77.

[27] *Ibid.*, p. 99.

[28] DoD Instruction 7330.4, p. 7.

[29] Alexander (1), p. 102.

[30] *Ibid.*, p. 175.

[31] *Ibid.*, p. 114.

[32] *Ibid.*

[33]Ibid., p. 115.

[34]In fact, more than 16,000 separate processing requirements were considered. Documentation for these requirements is in the form of decision-logic tables, in the "JUMPS Design and Requirements Manual," Navy Finance Center, Cleveland, Ohio, 1976.

[35]Alexander (1), p. 109.

[36]Ibid., p. 111.

[37]Ibid., p. 186

[38]Ibid., pp. 136-173.

[39]Myers (7), p. 4.

[40]H.A. Simon, The Sciences of the Artificial (Cambridge: MIT Press, 1969).

[41]Myers (7), p. 33.

APPENDICES

Appendix 1

The JUMPS misfit variables selected in chapter 3 are based on the following systems specifications for JUMPS, which were originally contained in Department of Defense Instruction 7330.4, "Requirements for Development, Test, Evaluation and Installation of the Joint Uniform Military Pay System (JUMPS)" dated November 7, 1966. The parenthesized number after each specification refers to the numbering of these in the original document.

1. Individual members of Military Services will be paid on regularly scheduled paydays for two pay periods each month. The pay periods will end as of the 15th and the last day of each month. (A.1)

2. Time between any cut-off of input processing for a pay period and date of payment to individual members must be long enough for accurate preparation of the payroll, including the application of suitable control procedures and the correction and adjustment of errors. (A.3)

3. Payroll payments will be made by check unless there are obvious benefits in cash payments. The "composite check" procedure prescribed in Treasury Transmittal Letter No. 53, September 28, 1970, will be applied to the maximum extent feasible. (A.4)

4. Systems development will be aimed at reducing the manual and clerical workload of operational military units and organizations, substituting centralized and computerized processing, wherever feasible. (A.10)

5. Transactions common to military pay and personnel systems will be input using single source, source data automation techniques and will be recorded in both systems on the same basis, wherever feasible. (A.13)

6. In addition to periodic, formal internal audit of the operating system, responsible commanders will monitor critical input, processing, and output points, to assure security and integrity of the system. (A.14)

7. Accounting for leave earned, leave taken, and leave lost will be included in JUMPS. Data needed to review and approve individual requests for leave will be maintained at appropriate operational levels. (B.2)

8. For local administration of pay and leave operations, a personal financial record will be maintained. It will be used as a temporary file of input documents affecting members' pay, including allotment authorizations and will be transferred to new duty stations on PCS of the member. Reference will be made to this document by disbursing officers making payments to transients. (II.A.2)

9. Individual statements of account will be prepared for each member monthly. These statements will be called "Leave and Earnings Statements." They will be produced solely from data contained in and controlled by the pay account maintained by the central site. (II.A.4)

10. The Leave and Earnings Statement will contain, as a minimum, the following specific elements of information: Here follows a detailed listing of these elements of information, including, for example, date of preparation, social security number, name, all continuing and one-time entitlements and deductions. (II.A.4.a)

11. Status changes and actions affecting members' pay accounts may originate at the members' site and enter the pay system via the local finance or disbursing officer having custody of the member's personal financial record or may enter the system through other systems (e.g., the personnel system).

In either case, the input will be forwarded to the central site in machine-readable form where practicable, supported by human-readable documentation. Control procedures will be developed to ensure receipt by the central site of all data transmitted by the originator and the reject and suspense of unacceptable data. (II.B.1)

12. Input procedures and techniques will apply the principles of source data automation. Such applications may vary due to the nature of the input action and environmental conditions, and will be flexible, permitting continuing improvements in such techniques. (II.B.2)

13. Where certain special or incentive pay entitlements require reports of performance, in a certain time frame, or reports of duty at a particular location, following initial certification, these reports of performance will be on an exception basis (cases where entitlement requirements are not met), for those personnel normally authorized such payments. Authorities responsible for initial certifications and recurring exceptions reports will ensure maintenance of adequate and auditable records supporting such certifications and reports. (II.B.5)

14. The system must have a base of accurate, reliable, and timely input. Authorities inputting data to the military pay system are responsible for the propriety and accuracy of such inputs. Pecuniary accountability for improper payments will inhere in the finance or disbursing officer making payments, in accordance with applicable statutes. (II.B.6)

15. Where source documentation is read directly by EDP equipment (by scanning at an intermediate or centralized site, for example) it must be forwarded as it becomes available, consistent with available mail/courier service. In such a situation, copies of the documentation will be retained locally in the finance or other local office accessible to the disbursing officer until receipt of the Leave and Earnings Statements from the central site, at which time the locally retained documentation may be destroyed. (II.B.8)

16. To the extent of local capability, codes, transactions, status changes, and other input data will be pre-edited for validity before communication to the central site. (II.B.9)

17. For regular, recurring semi-monthly payments, Military Services will pay locally or centrally, or by a combination of these methods using Treasury check as the payment medium to the maximum extent. (II.D.2)

18. For members paid locally, the regular pay date may be rescheduled, when considered operationally desireable by the local commander. (II.D.3)

19. Special payments to individual members on a local basis, such as partial pay or casual pay, will be made as the need arises and will generally be limited to emergency or hardship cases and to special categories of personnel, such as recruits, in-transit personnel or personnel joining or being detached from a duty station or activity. (II.D.4)

20. Members will be given the option of having a portion of their net pay due carried forward as an unpaid item due in their accounts. (II.D.5)

21. Input to the pay accounts must be complete and accurate and adequate controls to assure correct processing must be established for accurate and complete accounting and reporting. Care will be taken to program a complete system of checks and balances from input through final output. (II.H)

22. When input to the centralized site is released from the local level, unaccompanied by supporting documentation, it will be suspended to ensure transmittal of the required documentation. Such suspense file will be a subject for internal reviews and audits. (II.H.1)

23. Actions rejected by the system will be controlled so that appropriate follow-up action can be taken. (II.H.3)