Purdue University
Purdue e-Pubs

Open Access Dissertations | Theses and Dissertations

Spring 4-9-1976

A Methodology for The Determination and Communication of Requirements for an Information Processing System

Carl Allen Singer

Follow this and additional works at: https://docs.lib.purdue.edu/open_access_dissertations

 Part of the Computer Sciences Commons

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106

77-1777

SINGER, Carl Allen, 1946-
A METHODOLOGY FOR THE DETERMINATION AND COMMUNICATION OF REQUIREMENTS FOR AN INFORMATION PROCESSING SYSTEM.

Purdue University, Ph.D., 1976
Computer Science

Xerox University Microfilms, Ann Arbor, Michigan 48106

A METHODOLOGY FOR THE DETERMINATION AND COMMUNICATION OF REQUIREMENTS FOR AN INFORMATION PROCESSING SYSTEM

A Thesis

Submitted to the Faculty

of

Purdue University

by

Carl Allen Singer

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 1976

Grad. School
Form No. 9
Revised

PURDUE UNIVERSITY

Graduate School

This is to certify that the thesis prepared

By CARL ALLEN SINGER

Entitled A METHODOLOGY FOR THE DETERMINATION AND COMMUNICATION OF REQUIREMENTS FOR AN INFORMATION PROCESSING SYSTEM

Complies with the University regulations and that it meets the accepted standards of the Graduate School with respect to originality and quality

For the degree of:

DOCTOR OF PHILOSOPHY

Signed by the final examining committee:

Andrew V. Hunt, chairman

J. F. Kinnamon Jr.

Gordon B. Wright

L. Mokowitz

Approved by the head of school or department:

April 9 1976 Jay W Wiley

To the librarian:

This thesis is not to be regarded as confidential

Andrew V. Hunt
Professor in charge of the thesis

DEDICATED TO THE MEMORY OF

BERL DOV SINGER, ztl.

ACKNOWLEDGEMENTS

I offer thanks to Professors J.F. Nunamker, Jr. and Gordon P. Wright for their friendship, guidance and encouragement; to Professor Daniel Teichroew for first introducing me to the discipline of Management Information Systems; to Professors Andrew B. Whinston and Herbert Moskowitz for their suggestions and to my mother for her constant support.

TABLE OF CONTENTS

Table of Contents (Cont.)

LIST OF FIGURES

List of Figures (Cont.)

List of Figures (Cont.)

List of Figures (Cont.)

Appendix E

ABSTRACT

Singer, Carl Allen,Ph.D.,Purdue University,May 1976.A Methodology for the Determination and Communication of Requirements for an Information Processing System. Major Professor Andrew B. Whinston.

A Requirements Statement Methodology is developed and coupled with a solution to the data base design problem. The need for the Requirements Statement Methodology is discussed from the viewpoint of management texts and systems design guidelines. The methodology is developed using forms, computer generated forms, a data dictionary and interactive dialogue. A detailed explanation of PSL (the Problem Statement Language) and PSA (Problem Statement Analyzer) in the context of the Requirements Statement Methodology is presented. A formal discussion of data base design, specifically record and set design, appears. A theoretical model to solve record design is developed and a hueristic and an algorithmic approach to record design are implemented, tested and discussed.

CHAPTER I - INTRODUCTION

The Need for a Requirements Statement Methodology (RSM)

The growing complexity, cost and power of computer systems has put a premium on well designed and properly implemented information processing systems (IPS). To this end, a growing body of research and literature has addressed the problem of designing information processing systems. This body of knowledge includes formal languages for describing certain aspects of the target system (logical system description, data structure, file structure, hardware configuration, etc.) and techniques or models for systems design. To a great extent design procedures, techniques and models have driven or defined the needs for formal languages which either provide data to them or communicate their output.

A broader approach is to consider the design of an IPS to be, itself, a systems design problem. The purpose of this approach is not to stress a recursiveness of definition, but to provide a sound framework for determining the information required at all stages of the system design process. Furthermore, emphasis will be on the gathering, determination and communication of this information. A requirements statement methodology will be developed from this framework.

A Review of Relevant Literature

Before developing an approach to the design of an IPS, a review of current thoughts and applications in this area is in order. The first

general source will be management textbooks. These texts set the tone with which a new generation of managers will review the systems design process.

Management Texts

Kast[1] in "Organization and Management" emphasizes the need for information to make decisions. "The object [of systems design] is not optimization of data processing systems; rather, the objective is development of better information-decision systems for management." Kast advocates the use of graphic flow charts to obtain a picture of the current information flow. "Preliminary designs spell out in rough form the requirements of the system under study. Considerable detail must be included, such as the timing of information needs, alternative routings, and types of equipment that might be utilized in implementing the system." Kast continues, "...the interface between managers and information system designers is critical, and mutual understanding should be fostered in order to maximize returns from design efforts." Kast defines three stages in the "continuous process of design and implementation for computerized information systems:" Systems Specification, Data-Processing Implementation and Programming. In keeping with modern thoughts, Kast continues, "Specification work should be delegated to operating people who will use the system. If decisions and information flow form the basis for the system, operating decision makers will be in a better position to identify current and future needs. Specialists can 'get in the act' in the second phase, when the feasibility of implementing the specified system is investigated."

Rosenblatt[2] in "Modern Business: A Systems Approach" says:

> Installing a computer system requires careful planning which may take as long as a year. Present and anticipated information needs must be carefully studied. Computer equipment capabilities vary greatly, and components must be ordered months in advance. All supporting systems of paper movement and personnel must be developed. Forms and computer instructions must be designed and written. Even when all these things are carefully thought out and accomplished, the day when the computer is actually delivered can be chaotic. There are always bugs that must be discovered and worked out, and the transition may actually come to a standstill while it waits for the computer to begin working correctly. For example a major bank in a large city recently went for four months without sending out statements on loans because of the problems in switching to a new computer system.
>
> Computer technology and business needs change so rapidly that most computer systems are in a constant state of revision or expansion. The system is never really set, and these frequent changes are likely to cause dissatisfaction among managers and employees. Mistakes may be blamed on the computer system which is "never" right. Customers may become irate because of the impersonal mistakes the computer makes. Managers must anticipate and deal constructively with these problems, or they may encounter a great deal of ill will.

The Kast textbook, like many newer texts[3,4,5,6,7], defines an objective, presents an approach and then provides some guidelines for systems design. Each seems to re-define the design process or borrows a multi-step definition from another source. Requirements Statement is defined as an early step in the design process, and there is more emphasis on user statement of these requirements. The focus of the Rosenblatt book is more general. It provides colorful anecdotes, etc. No formal procedure is defined, but management is cautioned as to the complexities and pitfalls of the design process.

Systems Designers

Against this background, consider the systems design process as seen by systems designers. Teichroew and Peters[8] state "every firm must have an information system to satisfy legal requirements, to provide communication with other organizations, to provide data for management decision making, control and planning." Five activities are defined to meet these requirements:

1. The recording of data describing the events and transactions that occur.
2. The processing of these transactions.
3. The production of documents that are necessary for internal and external communication.
4. The preparation of reports to satisfy legal requirements and for management.
5. The maintenance of files.

The steps in the design process are defined via an analogy with the design of a production plant. These steps are the perception of need, feasibility study, design, construction, test system, operation and modifications. The communication required is shown graphically via charts. This is the emphasis on the determination and gathering of requirements.

In 1967, Stieger[9] formally classified the needs for communication. He observes, "In large systems design projects it is not uncommon for the analysts to develop their own Information System with forms, files, collection and ordering procedures to aid them in the Study for an Information System." Stieger categorizes communication as association (man and his memory), dialogue (man and other man) and monologue

(man and machine). He then defines the following techniques:

Graphic Techniques - Flowcharts

Grids, Arrays, and Matrices

Linear Techniques - Languages
Programming Languages
Procedural Languages
Non-Procedural Languages
Data Management Languages and Man On-Line Executive Languages

He then suggests the following:

1. Improvement of the statement of problems by application of . . . theoretical work . . . and supported by techniques in the use of matrices . . . .

2. As a part of the problem statement language the specification of data relations should be developed with the minimum of imposed structure required for human appreciation of the content of the data base.

3. As a part of the problem statement language the facility to enter data relations and precedence as they are discovered and the analysis of global ordering through network techniques.

4. The display of analysis results by means of tables for the convenient consumption of humans.

Teichroew[10] defines the objectives of analysis as, "to determine, and record, the information needs of the organization and the individuals in it." Teichroew reviews seven approaches to requirements statement: Young and Kent[11], Information Algebra[12], Langefors[13,14], Lombardi's Algebraic Data System[15,16], ADS (Accurately Defined Systems)[17,18], TAG (Time Automated Grid)[19,20] and Systematics [21--26]. Teichroew begins by quoting "the authors' definition of data processing systems and their approach to analysis and design."

INFORMATION ALGEBRA[12]

An information system deals with objects and events in a real world that are of interest. These real objects and events, called "entities" are represented in the system by data. The data processing system contains information from which the desired outputs can be extracted through processing. Information about a particular entity is in the form of "values" which describe quantitatively or qualitatively a set of attributes or "properties" that have significance in the system. Data processing is the activity of maintaining and processing data to accomplish certain objectives.

LANGEFORS[13,14]

There are some basic propositions made here in connection with the systematic approach advocated, which appear to be in contradiction to present practices or assumptions. One is the hypothesis that in most cases it is possible to isolate and define the relevant organization functions in a separate operation to be performed before the actual design of the system is attempted. It is thus assumed that these functions are defined from the basic goals of the organization and therefore will not need to await the detailed construction of the system. The other hypothesis is that it is possible to define all input information necessary to produce a desired output. The basic assumption here is that actually any information can only be defined in terms of more elementary information, which will then occur as input parameters. Therefore, once a class of information is defined then it is known what input information is required for its production. The point here is that it should not be necessary to work out formulas or programs for an entity where important variables are missing, so that starting by programming is no safeguard against ignoring important data.

YOUNG AND KENT[11]

The content of our analysis is that the objectives of the data processing system have been stated in terms of the required outputs; these outputs are not considered as subject to revision. On the other hand, although the inputs may be organized in any desired fashion, it appears necessary or at least convenient, to state one of the possible input organizations from which any equivalent one can be derived. It should be noted that the input may supply any one of a number of equivalent pieces of

information, e.g., either customer's name to be copied directly onto an output or an identification number from which the name can be looked up.

LOMBARDI[13,14]

The common denominator of file processes is the production of output files as functions of input files.

ADS[17,18]

The starting point is the definition of reports--what output information is required. Once the reports are defined, the next step is to find out what information is immediately available. This is followed by laying out the information system in between the output and input. The origin of all information needs to be specified. The outputs of this system are always looked at in terms of inputs.

TAG[19,20]

The technique requires initially only output requirements of a present or future system. These requirements are analyzed automatically [by a computer program] and a definition is provided of what inputs are required at the data level.

SYSTEMATICS[21--26]

SYSTEMATICS is a language solely concerned with techniques and concepts useful to systems analsts in designing information models to meet user's requirements .... It is a tool for specifying solutions to information systems problems. More important, it is also a tool for developing such solutions.

Teichroew's succinct discussion of a Requirements Statement Language (RSL) is presented as Appendix A.

Nunamaker[27,28] developed a Problem (requirement) Statement Language (now termed SODA/PSL) as a necessary input for Systems Optimization and Design Algorithm (SODA). He asserts that optimization is

usually limited to equipment selection for a given application. Optimization of design, etc., is overlooked. As a solution to the problem, Nunamaker proposes automation of the systems design process. Figure 1 is an overview of SODA. The decisions needed by SODA are shown in Figure 2. It is clear that considerable input data is needed to operate SODA. In addition to obtaining this data, the problem of communicating it to the SODA models has to be solved. This lead to the earliest version of PSL. It is important to emphasize that the PSL was the result of a felt need.

Ho[29,30] has developed a formal model and definition for requirements statement languages and requirements statement analysis. Ho discusses software correctness as a motivating factor for a formal model of an RSL. Correctness is defined as the production of the output specified by the user for a given input. An approach towards software correctness is the *a priori* construction of a correct program. This approach differs from the other means of assuring program correctness, testing or proving a program is correct (*a posteriori*), in that it requires both a method for recording user requirements and a method of analyzing these requirements and developing the software.

In a recent paper Teichroew[31] discusses the system life cycle. He begins, "Many professionals engaged in analysis, design, implementation and operations of systems are not satisfied with the progress that has been made--systems take too long to build, they cost much more than predicted and do not work as promised when installed." He defines the "major task" as the "structuring of subsystems (applications software, data base, computer systems and non-computerized procedures)." One

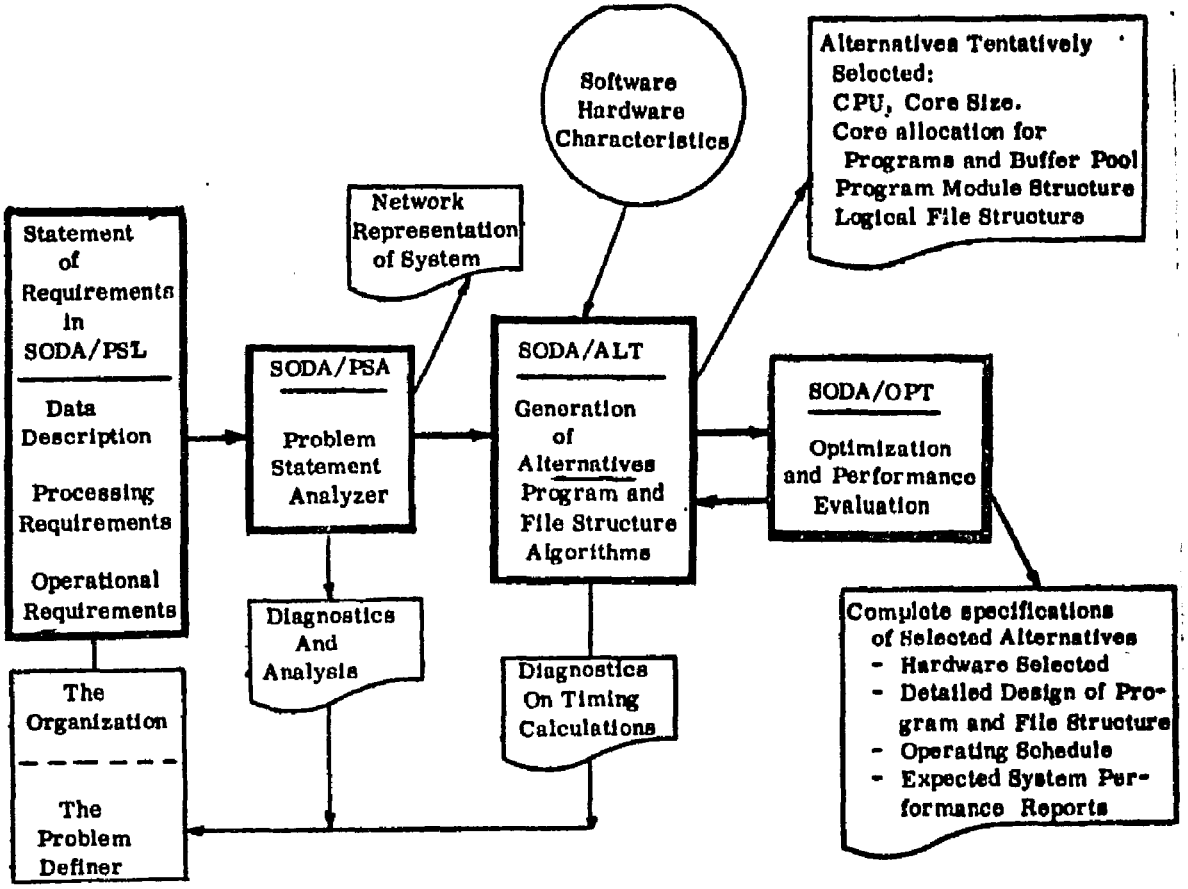


Figure 1. SODA: Systems Optimization and Design Algorithm [28]

Level	Decision Variables	Objective	Alternatives	Constraints	Optimization Techniques
			SODA/ALT		
1	CPU	Select the minimum rental cost CPU class	list of available CPU classes	Time available for processing	search for ordered CPU classes
2	Core Size	Select the minimum rental cost core size	list of available core sizes	-Times available for processing -CPU class	search of ordered core sizes
3	Program Module	Select the set of modules with the minimum transport volume	feasible grouping arranged in tree structure	-Core Size available for modules	Network analysis and branch and bound search over feasible alternatives
4	Data Structures	Select Files such that the maximum number of I/O for any module is a minimum	feasible grouping arranged in tree structure	-Program Modules	Network analysis and branch and bound search over feasible alternatives
			SODA/OPT		
1	Storage Structure	Minimize the number of inter block gaps for tape or the number of accesses for a disc	1 char $\leq$ block size $\leq$ upper limit 1 $\leq$ accesses $\leq$ upper limit	-Program Modules -Data structure	Non-linear programming model
2	Number and type of auxiliary memories	Minimize the variable reading and writing time	list of available devices	-CPU and Core Size -Program Modules -Storage structures -Data structure	Integer Programming Model

Figure 2. Decision Levels of SODA [28]

approach to systems design has been procedures manuals. Teichroew continues:

> These procedures manuals normally include: (1) a set of activities which must be carried out, frequently presented as a PERT diagram, (2) a set of forms to be completed at various stages of the development, which compromise "documentation," and (3) a project management system, sometimes including a computerized recording and reporting system to measure progress against the plan. These procedures manuals have grown out of practical experience and have received little formal analysis.

It is important that the properties of a system be defined. Teichroew proposes three classes of properties (Figure 3).

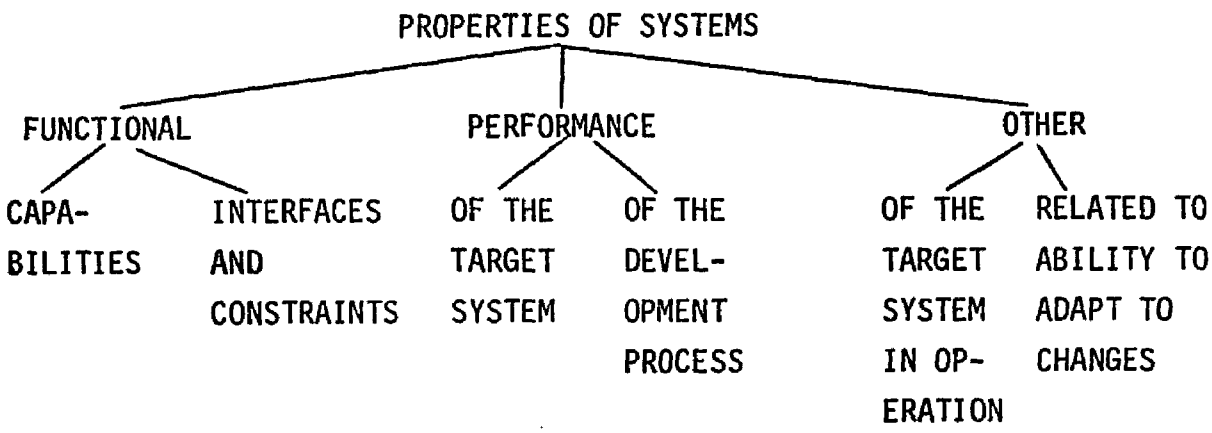


Figure 3. Taxonomy for Properties of Systems [31]

The "other" category includes such characteristics as reliability, flexibility, etc. Teichroew then points out the recent emphasis on the data base subsystem. This will be discussed later as a motivation for the RSM implementation.

There are many approaches to systems design. Most design efforts are a combination of these approaches. The importance of this analysis is that the different approaches have different information requirements.

Teichroew defines these as "(i) Rules of thumb, guidelines and principles, (ii) Successive approximation and (iii) Normative approaches." Figure 4 indicates how these three approaches are used in the system design process. The RSM must be able to gather, analyze and communicate the information expressed in Figure 4.

LEVEL OF TECHNIQUE	I PERCEPTION OF NEED	II LOGICAL SYSTEM DESIGN	III PHYSICAL SYSTEM DESIGN	IV CONSTRUCTION	V TEST AND CONVERSTION	VI OPERATION	VII MODIFICATION & MAINTENANCE
Present Methods	Not formally recognized	Narrative, tables, charts	Manual benchmarks	Flowcharts, programming languages	Ad hoc	Operating systems	No formal procedure
Improved Methods	Capital investment review procedure	Documentation standards	Standards for system design	Programming standards: modular, decision tables	System test standards Emulators	Manual scheduling	Change control procedure
Computer Aids (stand alone)	Investment analysis programs	Problem statement languages	Simulators	Flow charters COBOL aids DBMS	Test generators	Computer-aided scheduling	Monitors Computer-aided analysis

Figure 4. Techniques Used in System Life Cycle [31]

In this section the need for a RSM to meet the various needs of systems design has been explored. An approach to determining more specifically what a RSM should be and an implementation follow.

An Approach to Building a Requirements Determination and Communication Model

The design process is, of course, frequently characterized by its output, the designed system. The focus on the resultant (target) system has two major shortcomings. The process of designing a system is often neglected as simply a means towards an end. Secondly, the requirements for the target system are not fully specified. The ultimate success of a system is measured by how effectively and efficiently it meets user requirements (expressed, implied or hidden), not by how sophisticated or elegant the design. The systems design process consists of two primary questions: the first asking, "What are the requirements for the Information Processing System(IPS)?" and the second, "How

can these requirements best be met?" To begin, we must determine requirements, communicate these requirements and finally design to meet these requirements (Figure 5).

Figure 5. An Overview of Requirements Statement

The systems design process should also be considered as a systems design exercise. The primary input to the systems design process is the user requirements, however, the systems design process needs more than just user requirements if it is to result in an effective system. Objectives, constraints and as yet unidentified information are added input to the systems design process. This information can be characterized as either information which directly effects the target system or that which relates to the design process. The Design Input Data Base (DID) is defined as the set of all information required to design an information processing system. The determination of what is contained in the DID and how to gather and communicate this information is the major thrust of the model presented in chapters two and three.

It would be prudent to note, at this point, that the input needed for systems design is not necessarily the "natural" output of the user's and analysts' requirements definition process. So, too, much of the information in the DID needs to be transformed before it is useful to the systems design process. Since the description of the logical

system is only a subset of the DID, the other information required must also be determined, analyzed and communicated. Before this takes place the other information must be identified. A "backwards" approach, similar to that which led to the earliest PSL, is advocated. Each phase of the systems design process is identified and the data needed for each phase is identified. This data is then traced back through other phases of systems design to the DID. As the phases of the systems design process change or as these phases must consider different types of alternatives, the DID will change. Therefore added justification for a modelling approach stems from the growing complexity of the systems design process. Multi-programming and real-time applications lead to different considerations in systems design than, say, batch processing. Similar changes in the design process may arise from designing a transaction-oriented versus non-transaction-oriented system, or from re-designing an existing system as opposed to the initial design of a new system. As more, different techniques are needed to successfully accomplish systems design, a comprehensive model is needed to identify the contents of the DID and how to best gather, analyze and communicate this information.

Overview

This work introduces the concept of a Requirements Statement Methodology(RSM) and couples it with a solution to the data base design problem.

The Requirements Statement Methodology and its contents are discussed in detail in the second chapter. Various design and implementation considerations such as the use of forms, computer generated

forms and data dictionaries are discussed. A discussion of Accurately Defined Systems(ADS) provides the motivation for forms usage. The Design Input Data Base(DID) is developed in conjunction iwth the RSM. The various steps to systems design are mapped into sections of the DID and discussed at length. The impact of additional considerations such as designing for new versus existing systems, query oriented languages and a customized query language are also discussed.

Chapter three provides a detailed explanation of the Problem Statement Language(PSL) and the Problem Statement Analyzer(PSA) in the context of the RSM. This extensive discussion is necessary to clearly explain the RSM and to highlight the determination of data structure which is required for data base design. This presentation also highlights the contributions of RSM in helping the systems definition and design process. Of particular interest is the use of an interactive requirements statement technique. Appendix C provides additional PSL syntax information viewed from logical breakdown of structure, document flow, data structure, process/data linkage, timing and conditional statements and PSL completeness checks.

Chapter four is a formal discussion of data base design. An overview of the data base concept is followed by formal discussion of record design and set design. An example is used to help illustrate these problems.

Chapter five presents a model to solve the record design problem. The use of the RSM and PSA reports are explored through the use of an example. A hueristic is developed and discussed in detail. A clustering algorithm is developed and implemented. Appendix E contains the

computer program and sample outputs for the record design algorithm. An analysis of these outputs is contained in the conclusion of chapter five.

Chapter six discusses possible extensions to the RSM.

CHAPTER II - THE REQUIREMENTS STATEMENT METHODOLOGY AND ITS CONTENTS

This chapter discusses the development of the Requirements Statement Language(RSM) and the various design considerations which are involved. A data dictionary is developed. A forms oriented RSM is discussed as is the use of computer generated forms. The Design Input Data Base(DID) is then developed.

The Requirements Statement Methodology

Various considerations have led to the design decisions which result in the current RSM. The RSM combines features of many other requirement (problem) statement techniques with informal methods which have been successful in various systems design efforts. The RSM incorporates the ease and acceptability of forms (ADS), the precision and analysis available from formal languages (PSL) and the ingenuity of many systems designers who have had to build their own design tools. A broad overview of the RSM concept is presented in Figure 6.

Chronologically, the first problem with an RSM is user acceptance and understanding. Tools must be used and used correctly. Factors which effect user acceptance include ease of use, clarity and foreseen results. All methods promise results; RSM can be no different here. Experience with ADS indicates that a forms-oriented system which requires a minimum of user training, which clearly details what information is required of the user and which allows the user to speak his own "natural" language quickly gains user acceptance. On the other hand, results suffer from the inexactness or ambiguities of users' "natural" language

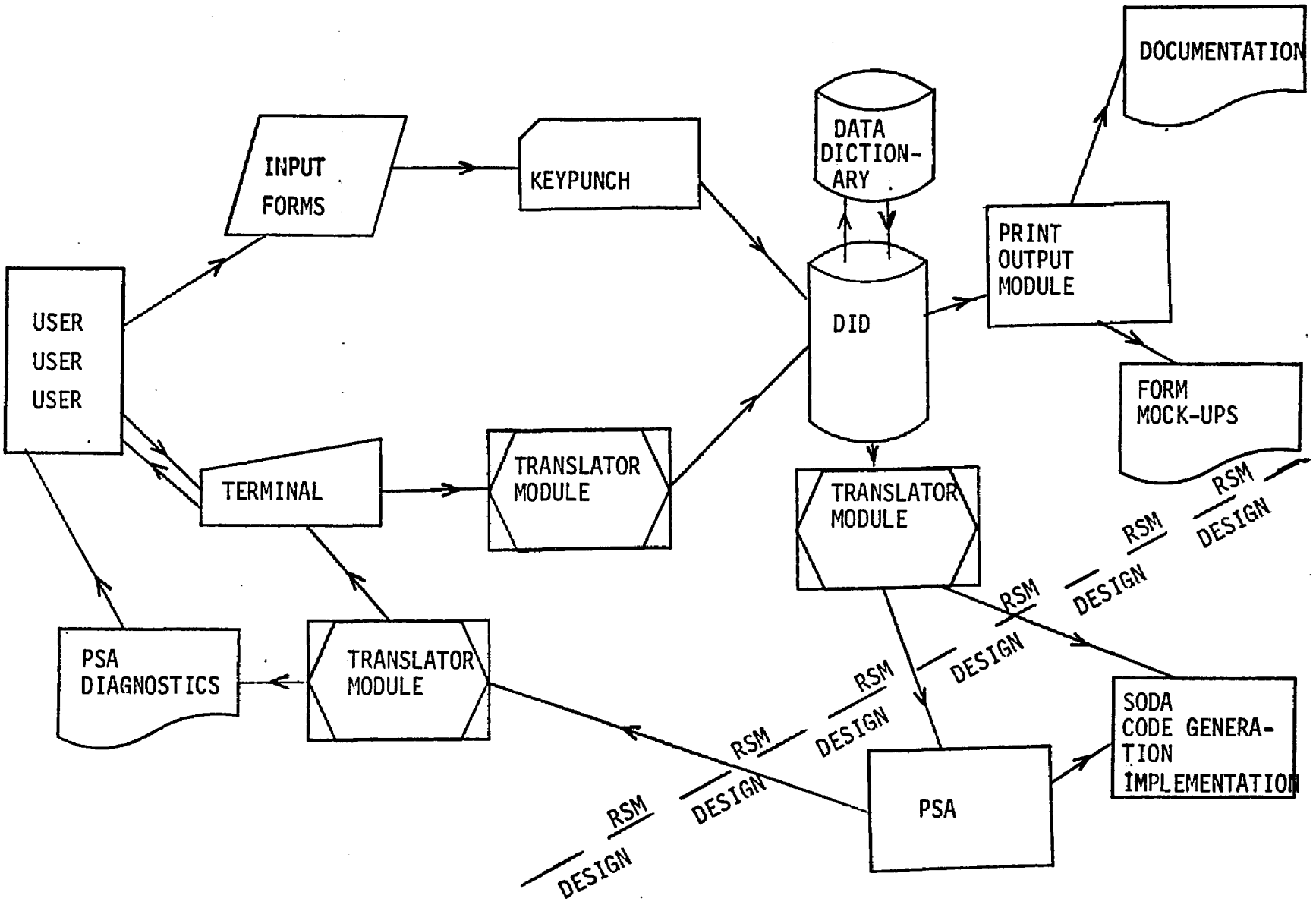


Figure 6. Requirements Statement Methodology (RSM)

statements or from the narrow band of information which the forms ask for. Finally, there is a problem of translation from the user's "natural" language to the languages of system design input.

Frequently every time the user wishes to express himself he must mentally translate his thoughts into a form (or language) compatible either with some requirements statement language or with a given input form. Everytime the user must make this translation he is prone to error. Even more importantly, he may have a misconception as to the translation required and he thus systematically enters incorrect information into an entire portion of the requirements statement. If there are many users, each must perform this translation effort in the identical manner, if not, more sources of error arise. Finally, if a concept is changed, expanded or clarified, the raw data still exists and only the translation module need be changed. Examples of this concept may be useful:

A water pollution control agency requires that certain firms conduct weekly tests of effluent levels at various plant locations. This same agency requires that all firms report their test results on a monthly basis. (This is the system as the user sees it.)

A pollution test frequency criteria is established as a requirement. The implemented system (designed to meet user requirements as translated into a requirements statement language and interpreted by systems designers and programmers) looks at the monthly report and compares the number of tests conducted with the number of tests required. By this criterion a weekly test (say every Monday) was translated to a required frequency of 1/7 (one test per seven days). The monthly report

compares the required frequency of tests (1/7) to the actual. For a month with four Mondays, 4/30 is compared with 1/7 and the system incorrectly finds a firm as having a violation.

The same system had requirements (from another government agency) to indicate violators. Thus the firm with the above "violation" was put in the same violation report as a firm which dumped tons of raw sewage into a river. Similarly no requirement to distinguish between gross violators and others was established. Should a firm which is .03% over its goal for one reporting period, yet averages well below its goal be grouped with one which is consistently 5 times its goal?

In the area of data structure, the constructs of a formal data description language like DDL[32] or DL/I may confuse the users. Perhaps a simple tree structure drawing may suffice the users. Similarly a plex or cross-reference structure may be difficult for the user to envision or express. An interactive system may prompt the user to establish the correct structure. A translation module might interpret the complex relationship or translate from diagram to formal data description language.

In stating hardware characteristics, the user (in this case a hardware specialist) should be able to input the relevant data for any type of hardware without comparing 8-bit words versus 64-bit words, etc. Similarly, the operating characteristics of a push down stack machine cannot be directly compared with other machines. An input form geared for one hardware configuration may be awkward for gathering data for another. Thus two different input forms may be needed, each with its own translator module.

The use of forms to spell out the information required may be enhanced by the use of temporary forms to meet specialized needs. For example, a form capturing potential queries or query sequences would be better than a "report" form for capturing query data (the content of the query, response characteristics, volume, frequency, sequences of queries, etc.). Again a translation module is needed for the temporary form. When working with existing documentation, a translation module can save much work (and possible sources of error) by taking existing data and translating it into PSL. A common example of this is to take an existing data dictionary or data element listing and convert it into the appropriate set, entity, element notation of PSL.

The mechanics of inputting large amounts of data has led to designing the RSM to accommodate forms with keypunch masks for primary input of large amounts of data, with terminals for limited data entry, error correction and user feedback. Ideally the analysis of the requirements statement will provide "intelligent" and useable feedback for the user. Again, a translation module oriented towards the user may be needed to translate PSA diagnostics into narrative and instructions for correcting the requirements statement.

The last feature of the RSM is a data dictionary. Experience, again, dictates that such a dictionary is a most useful tool. Current data dictionaries[33,34,35] have provided a menu of possible items to be included in a data dictionary. In addition, the ADS effort saw the informal development (by Rick Stell of the Navy Material Command Support Activity) of a DERF (Data Element Reference File) which was simply a short alphanumeric code to be used as a synonym for the formal data

element name. The convenience and wide-spread acceptance of the DERF feature has prompted a similar standard for this implementation. The data dictionary will contain the following items:

1. DATA ITEM NAME is the formal name for the data item.
2. DERF (see above) is a unique mnemonic for the data item name, its length and composition (alphanumeric) can be adjusted to best suit an application, i.e. an organization with 15 distinct departments may use a one letter prefix to the DERF to identify originating department, etc.
3. SYNONYMS
4. FORTRAN SYNONYM is used if FORTRAN will be required for implementation, a 6-letter word (meeting FORTRAN requirements).
5. COBOL SYNONYM (if COBOL is used).
6. FORMAT/PICTURE is like either a FORTRAN or COBOL statement.
7. TYPE is either alphanumeric, integer, etc.
8. JUSTIFICATION--left, right or centered.
9. VOLUME--minimum, maximum, mean. If volume is dependent on another data element (say volume equals number-of-students).
10. RANGE--a minimum and maximum are provided.
11. VOLATILITY
12. VALIDITY RULES
13. SECURITY CATEGORY
14. DATA SET INFORMATION includes set membership, percent occurrence, etc.
15. NARRATIVE is encouraged to express additional information.

The outputs from the RSM are shown in Figure 7. Although a flexibility in forms design and usage is a feature of the RSM, certain common input forms have been designed. These are the Input/Output

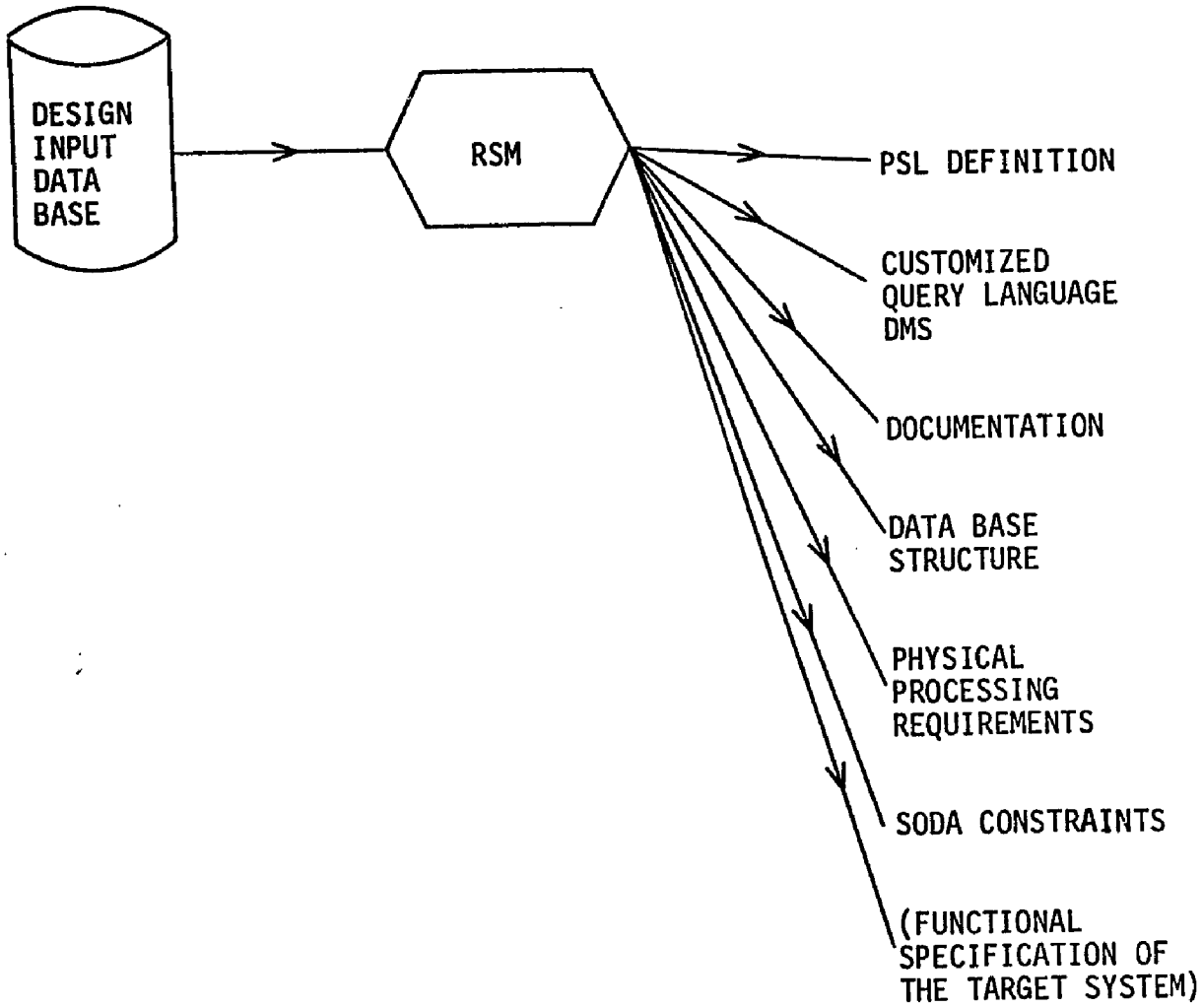


Figure 7. RSM Outputs

Form, the Data Dictionary Form and the Process Definition Form. The Input/Output (or layout) form is designed for use with a layout sheet to depict any given input (input form, card input, tape, terminal) or output (report, terminal output, punched card, tape). The form itself gathers and communicates the information which is transmitted via the layout sheet. The layout sheet is chosen appropriate to the media being used. This form evolved from portions of the ADS input and report forms. The Data Dictionary Form is designed to conveniently provide the data dictionary information (described above). The Process Definition is a hybrid of the logic and computation forms from ADS and general decision tables. Figures 8, 9, and 10 are samples of these forms. Appendix B contains detailed descriptions of how to use the forms.

Flexible forms developed with the aid of a form generator which produces the forms and generates appropriate input formats for reading data punched from the forms have been experimented with. The concept has not been tested enough to be evaluated. Figures 11 and 12 show a sample generated form and the corresponding formats. The sample form in Figure 11 illustrates two concepts, first form generation and second a "temporary" form to gather information in a "natural" language for specific users. The data gathered is for a data dictionary form.

The form is generated interactively. First a heading is requested, it is entered and the choice of justification is made. The page, name and date block are fixed for all forms. The rest of the form is done in blocks. Each block represents a given line or group of lines. The lines may consist of narrative, narrative followed by input, narrative as a block heading followed by lines of input or input only. Input may

FORM NUMBER: L-_____ PAGE _____
1 2 4 5 6

USER INITIALS: _____
9 11

DATE: ____-____-____
12-13 14-15 16-17

TITLE: _____
18 58

FORM TYPE: (1) input (2) output (3) both

MEDIUM: (1) card (2)tape (3)disc (4)printer (5) crt

FORM FREQUENCY:

(1) _____ times per ___ (1=year 2=quarter 3=month 4=week 5=day 6=weekday 7=hour 8=minute 9=second)
60 62 65 66

(2) _____ times per (WEEK/DAY) ___
61 65 66 67 68 69 70 71

(3) _____ times per (MONTH/DAY) ___ ___ ___ ___ ___ ___
61 65 66-67 68-69 70-71 72-73 74-75 76-77

(4) depends on DIN D-_____ (times _____)
61 65 68 70

(5) triggered by process P-_____ Page ___
62 64 65 66

DATA LINE	LOCATION			DIN	PICTURE FORMAT	VALIDATION RULE RANGE or PROCESS		PERCENT OCCURANCE
01.								
02.								
03.								
04.								
05.								
06.								
07.								
08.								
09.								
10.								
11.								
12.								
13.								
14.								
15.								
16.								
17.								
18.								
19.								
20.								
21.								
22.								
23.								
24.								
25.								
26.								
27.								
28.								
29.								
30.								
	7-8	9-11	12-14	15-17	18 21	22 30	31 32 36	37 41 42 44

Figure 8. Input/Output Form

(FOR) DATA ITEM NUMBER (DIN): D-_____ (1 2 5)

USER INITIALS:_____ (6 8)

DATA NAME SECTION

1. Data item name _____ (10)
2. Synonyms _____
3. Fortran Synonym _____ (7 12) (6 letters max.)
4. C6bol Synonym _____ (13) (24 max.)

DATA DESCRIPTION SECTION

FORMAT/PICTURE _____ (7 17) TYPE _____ (18) JUSTIFICATION _____ (19)

VOLUME (MINIMUM) _____ (20 29) (MAXIMUM) _____ (30 39) (MEAN) _____ (40 49)

VOLUME DEPENDS ON (DIN) D-_____ (50 53) (times _____ (54 58))

RANGE _____ (59 63) - _____ (64 68) VOLATILITY _____ (69 73) / 6 (74)

VALIDITY RULES (INPUT)___ (OUTPUT)___ (check if used)

SECURITY CATEGORY _____ (77 80)

DATA SET INFORMATION

BELONGS TO(DIN) D-_____ (6 9) PERCENT OCCURANCE _____ % (10 12)

CONTAINS (DIN) D-_____ (13 16) D-_____ (17 20) D-_____ (21 24) D-_____ (25 28) D-_____ (29 32)

NARRATIVE

1. _____
2. _____
3. _____
4. _____
5. _____

Figure 9. Data Dictionary Form

FORM NUMBER: P-______ (1 2–4) PAGE ______ (5–6)

USER INITIALS: ______ (9–11)

DATE: ____-____-____ (12-13, 14-15, 16-17)

TITLE: ____________________ (18–80)

LINE	TYPE (9)	TAB (10–60)	GRID 61	2	3	4	5	6	7	8	9	70	1	2	3	4	75
01.																	
02.																	
03.																	
04.																	
05.																	
06.																	
07.																	
08.																	
09.																	
10.																	
11.																	
12.																	
13.																	
14.																	
15.																	
16.																	
17.																	
18.																	
19.																	
20.																	
21.																	
22.																	
23.																	
24.																	
25.																	
26.																	
27.																	
28.																	
29.																	
30.																	

COMMENTS:

Figure 10. Process Definition Form

STREAM POLLUTION CONTROL AGENCY

DATA DIRECTORY INPUT FORM (PROVISIONAL)

FORM NUMBER 11 PAGE:.... NAME:..... DATE: .../.../..

PURPOSE: THIS FORM IS DESIGNED TO GATHER INFORMATION ABOUT THE DIFFERENT DATA ELEMENTS WHICH ARE RELEVANT TO STREAM POLLUTION MEAUREMENT AND CONTROL. PLEASE FILL OUT AS MUCH OF THIS FORM AS YOU CAN. ADDRESS ANY QUESTIONS TO CARL SINGER AT PURDUE UNIVERSITY (317/49-44437).

1. NAME OF DATA ELEMENT: .....
2. IDENTIFYING NUMBER ASSOCIATED WITH THIS DATA ELEMENT (DEN): .....

NARRATIVE DESCRIBING THIS DATA ELEMENT (OPTIONAL):

3. .....
4. .....
5. .....

OTHER NAMES THAT THIS DATA ELEMENT IS CALLED BY (OPTIONAL):

6. ....
7. ....
8. ....

WHAT OTHER DATA ELEMENTS (ENTER NAME OR NUMBER) DOES THIS DATA ELEMENT EITHER DESCRIBE OR BELONG TO --EXAMPLES: "REACH" BELONGS TO "RIVER", "DEPTH" BELONGS TO "RIVER"/ ENTER "RIVER" FOR DATA ELEMENTS "REACH" AND "DEPTH".

9. ....
10. ....
11. ....

IF THE DATA ELEMENT IS A NUMBER OR MEASURE, WHAT UNITS IS IT MEASURED IN?
12. EXAMPLES -- TONS, PARTS/MILLION, DOLLARS .....

WHAT IS THE NORMAL RANGE FOR THIS DATA ELEMENT -- EXAMPLE, FOR DATA ELEMENT EMPLOYEE-PAY-RATE, NORMAL RANGE WOULD BE "$2.25/HR" TO "$12.50/HR", THUS
13. "$.456/HR" OR "$37.50/HR" ARE WRONG .....

WHAT IS THE APPROXIMATE VOLUME OF THIS DATA ELEMENT, OR ON WHAT IS IT BASED-- EXAMPLES: "200" CITIES IN THE SYSTEM, ABOUT "20" REACHES/RIVER, --OR-- THERE ARE "NUMBER-OF-FACTORIES" (A DATA ELEMENT) FACTORIES ALONG THE RIVER.
14. ENTER NUMBER -OR- DATA ELEMENT NAME .....

15. COMMENTS .....
16. .....
17. .....
18. .....
19. .....

THANK YOU.

Figure 11. Generated Form

*HELP COMMAND:

.You will be asked to specify one block of the form at a time; each block is a line of the form repeated a given number of times. Each line will have the same format but different narrative. After specifying the line a sample line will be printed and you will then supply the narrative, one line at a time.

.Max line length is 77 characters [terminal dependent]

.The primary input must be alphanumeric

*OPERATING COMMANDS:

.Enter length of narrative

.Enter length of primary input

.Enter number of secondary inputs

.Enter length of secondary inputs

.Enter type (I,R,A)

[The generator builds the line with appropriate spacing.]

.Enter narrative, one line at a time

*FORMATS GENERATED:

[Sample for "other names this data element is called by".]

[to print blank line]	(I3,'. '2A10,2(6X,2A10))
[to print data line]	same as above
[to read data]	(I2,2I3,3(2A10))

[The above includes provisions for form and line number.]

Figure 12. Generated Formats with Commands

be alpha or numeric and may include more than one item across the page. The formats generated provide FORTRAN format statements which will be used to (1) output blank forms, (2) output completed forms and (3) to read packed input for the given forms. The forms generator takes care of both horizontal and vertical spacing, asking the user how many spaces he desires, centering text and warning of the page ending.

Figure 12 includes both the dialogue between terminal and user and the formats which correspond with the form printed as Figure 11. A full evaluation of the interactive form generation will require not only more users, but those with a poorer grasp of this concept--the designer of a technique cannot qualify as either an unbiased or an uninformed user. Serious questions as to the cost/benefit of such an effort have been voiced. Obviously, the projected number of forms, etc., will govern the desirability of implementation. To better understand the motivation of temporary or special purpose forms within the RSM, a discussion of ADS and its use follows. The use of PSL and the PSA in conjunction with the RSM appears in the next chapter.

Computer-aided ADS combined with SODA in actual use was found to have major advantages over manual methods, but also to have certain disadvantages. Nunamaker, et al[36] discuss an actual use of this technique as does HO[29,30]. Both discuss the design and development of an integrated financial management system for the Navy Material Command Support Activity (NMCSA). This author was ADS Coordinator for NMCSA during much of this project. Positive feedback from the project effort includes:

1. The use of ADS provides clear documentation and a record of specifications as the users stated them.

2. User acceptance of ADS (ease of use, clarity, etc.) helps "sell" the technique both to management and to staff.

3. ADS feedback (syntax checking, and consistency and completeness checking) are most useful for reducing mechanical errors.

4. Data directory facility is useful for problem statement and manual analysis.

5. ADS Analyzer feedback is easy to understand and useful.

6. ADS saves considerable time in requirements statement.

7. ADS is less subject to ambiguities and omissions than narrative.

8. ADS instructions are clear and provide a guideline for form use.

Negative feedback (or ADS shortcomings) include:

1. ADS is limited in scope (to only part of the logical system specification)

2. ADS is inadequate in expressing time and volume data.

3. ADS allows too much leeway in problem statement. It is therefore more difficult to parse and often inexact in problem definition.

4. ADS computation and logic forms are not clear and convenient.

5. ADS is inadequate in expressing data structures.

6. ADS forms are not oriented towards computerized transcription and analysis.

The Design Input Data Base

The "backwards" approach, one which first determines the steps of the design process then proceeds backwards to determine the information required to perform those steps will be used to define the Design Input Data Base(DID). The methodology used to perform each step of the design process will effect both the content and detail of the information

required, as will certain design constraints such as batch processing only, etc. Also, added features or techniques such as a customized query language or automatic code generation will also change the contents of the DID.

The DID will be developed first following the steps of a "normal" systems design effort and then considering additional features and their impact. To fully appreciate the complexity of the design process we must also consider the determination (gathering), analysis (verification) and communication of information in the DID. The primary task in formulating this model centers around the DID. Since the emphasis of this effort is not to redefine the systems design process, but to model the input to it; the systems design process will be considered a black box which is a "sink" requiring information from the DID. The DID has the following major sections:

1. Objective/Constraint Specification for the Design Process.
2. Objective/Constraint Specification for the (target) Information Processing System.
3. Hardware and Systems Software Characteristics Specification.
4. Application Systems Environment Specification.
5. Logical System Specification.

The information contained in each major section of the DID is now derived.

Section 1. Objective/Constraint Specification for the Design Process

System design objectives and constraints have, to date, seldom been clearly specified; only alluded to. Objectives and constraints include implementation lead time, design and development cost ceilings, manpower

and other resource limitations, and such qualitative specifications as "full participation by all organizational elements." Implementation lead time may be expressed as a deadline, a series of phase completion dates or in great detail, perhaps to the extent of using a PERT-type schedule and control system. Similarly cost ceilings or targets can be specified for the project as a whole or for phases of the project. Trade-offs between time and cost may be considered. For example, would a 90% efficient system (containing less features) which could be delivered in three months at a cost of 1.4 million dollars be preferred to a complete system (with the features) but costing 1.8 million dollars with a nine month lead time. Manpower resources and such resources as computer test time, consulting expenditures, etc. may be specified. Other quantitative and qualitative considerations may appear in the DID.

Section 2. Objective/Constraint Specification for the (Target) Information Processing System

For the sake of clarity we can, by loosely applying duality, consider objectives and constraints to be two sides of the same coin, i.e. the objective that a system provide timely response can be re-formulated as a constraint that system response time be less than X during peak usage. The concept of duality comes from mathematical programming. When maximizing (or minimizing) a concave function over a convex constraint space, the optimal solution may be determined either by searching for that feasible point which has the optimal value of the objective function or by finding that value of the objective function which is both best and feasible. In general terms, this principle allows a trade-off between objective function and constraint space. For example,

a constraint that the processing time for a given transaction average only 4 seconds during peak busy period, with a given cost objective of minimum cost; may be re-written as a constraint of a given cost (say $100) with an objective function of minimum average transaction time during peak busy period. This type of analysis involving "sensitivity analysis" may show that by relaxing the first constraint from 4 seconds to 5 seconds, the resulting cost may be reduced significantly.

Among the criteria for a "good" system are the following objectives and constraints:

1. Compatibility. The system must be compatible with other information processing systems in the organization. It must interface with other computer systems and/or data-bases and must also fit within the organization (communi-cation) structure.

2. Changeability. Changes in procedures, algorithms, pro-cessing requirements, etc., must be accommodated. Another source of change, growth, also is accommodated within good system design.

3. Security and Recoverability. Criteria for back-up, privacy, security, etc., must be established.

4. Accuracy. What, if any, errors are allowable? Standards for accuracy may be set or redundant procedures established to assure accuracy must be established. Accuracy in mea-surement and the use of approximations must also be specified.

5. Ease of Implementation and Maintenance. Standards for these areas must also be determined.

Section 3. Hardware and Systems Software Characteristic Specification

The models and algorithms which are used in systems design need detailed measures of computer system performance. Measures include core constraints, storage requirements and constraints, read/write times and basic instructio.. times. These measures allow an algorithm to

determine which processes (see Section 5) can be combined and how long it takes to access data and perform computations for a process.

Section 4. Application Systems Environment Specification

Systems often must co-exist with other systems. It is important to know the operating characteristics of each of the systems which will share the computer environment with the target system. This information includes measures of computation-boundness, input/output boundness and storage media utilization.

The approach to modelling the first two sections of the DID has not yet been developed in detail. Essentially, a search of literature and relevant systems design procedures manuals coupled with the approach which looks at the systems design process as (itself) a system design problem will identify the relevant information to be determined and communicated. Data for Sections 3 and 4 are dependent on the choice of models and algorithms used by the design process. Data is required to drive the models and algorithms; this data must come from the DID. After identifying the information required for the DID, procedures for gathering, verifying and communicating this information must be established. In the discussion of Section 5, Logical Systems Specification, such an approach is presented in detail.

Section 5. Logical Systems Specification

Logical system specification is the determination and communication of the user's information requirements. The logical system specification is non-procedural, telling what not how. For example, a non-procedural statement would define a process resulting in the computation

of net-pay as the difference between gross-pay and deducations; a procedural statement would say, "first compute deductions, then compute gross-pay, then subtract deductions from gross-pay." Even this trivial example shows that a procedural statement may govern physical system design. Although the network considerations place the computation of net-pay after those of gross-pay and deductions, there is nothing which constrains deductions to be computed before gross-pay. This ordering by the user is undesirable. If, for example, deductions includes a tax which is computed as a percentage of gross-pay, the above ordering is incorrect. There is a need to communicate the logical systems specification among users, analysts, designers, programmers, etc. Each has his own "language" which is concerned with different portions of the design and implementation process. The specification procedures must both help in the gathering and determination of the logical system specifications and provide a means for communicating throughout the design process.

One approach which is useful in this regard is a network oriented method coupled with a formal problem statement language. The addition of feedback (analysis of the logical system) and graphic techniques to this basic approach results in a powerful tool for systems design. The full scope of this approach is illustrated when determining the logical system for a complex application such as a non-report oriented, new, interactive information processing system. Changing these parameters to, say, batch, report-oriented, existing system, etc., will change the design process and in turn the DID.

In essence, the question to be answered is what is to be done--from step one--in dealing with logical systems specifications. A good place

to begin is with a description of the organization. This is often presented as an organizational chart--a hierarchical tree with nodes corresponding to offices and branches corresponding to lines of command. For this basic structure, the tasks and responsibilities of each node are identified. These tasks are usually specified at a gross level, for example, perform the employee management function, or manufacture an end product. This gross identification of tasks can be broken down to include, recruit, train, promote, pay employees, etc. At this level the tasks can be broken down into decisions and reports. From an information network flow, a decision is a (psuedo) report in the sense that it produces information (an output decision) and needs input to do so. In an extreme case, a pseudo-report may involve the monitoring of data where information is gathered and nothing is done with it. To allow for this anomaly an "output" is created even if it isn't used.

Having identified or defined these output or pseudo-outputs of information, the next step is to determine how these outputs are created. First the term "process" will be formally defined: A process is that action which takes one or more items of information (as input) and produces some other item of information (an output) as a result of some transformation on the inputted information. To avoid procedurality we can further restrict the output of a process to be a single item of information (Figure 13). This action which produces two items as output can be broken down into two distinct actions and processes--systems design will determine if the process will occur simultaneously, etc., (Figure 14). Next, the definition of output is relaxed to mean the output of any process, as opposed to a final output. Finally, it follows

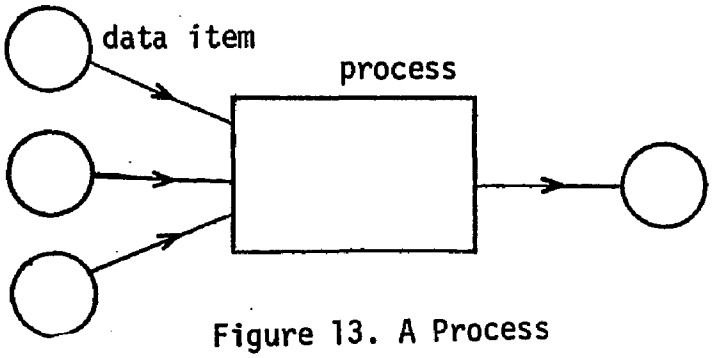
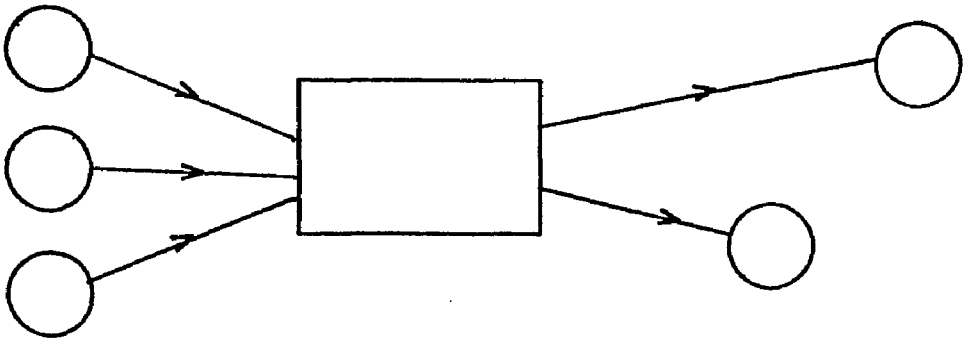


Figure 13. A Process



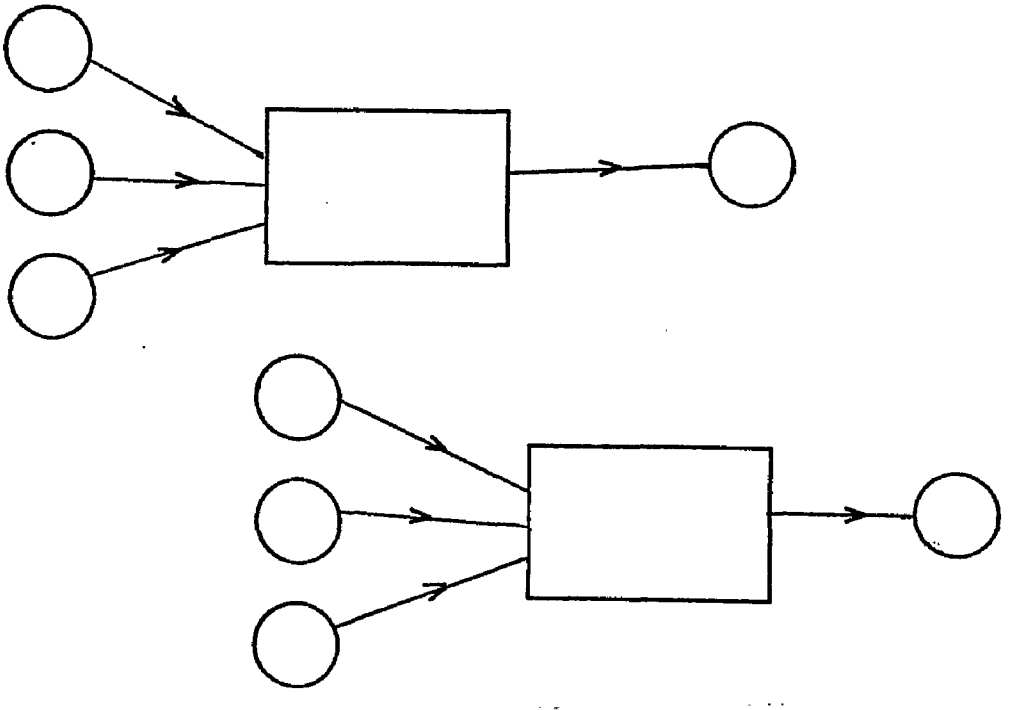


Figure 14. Two Processes

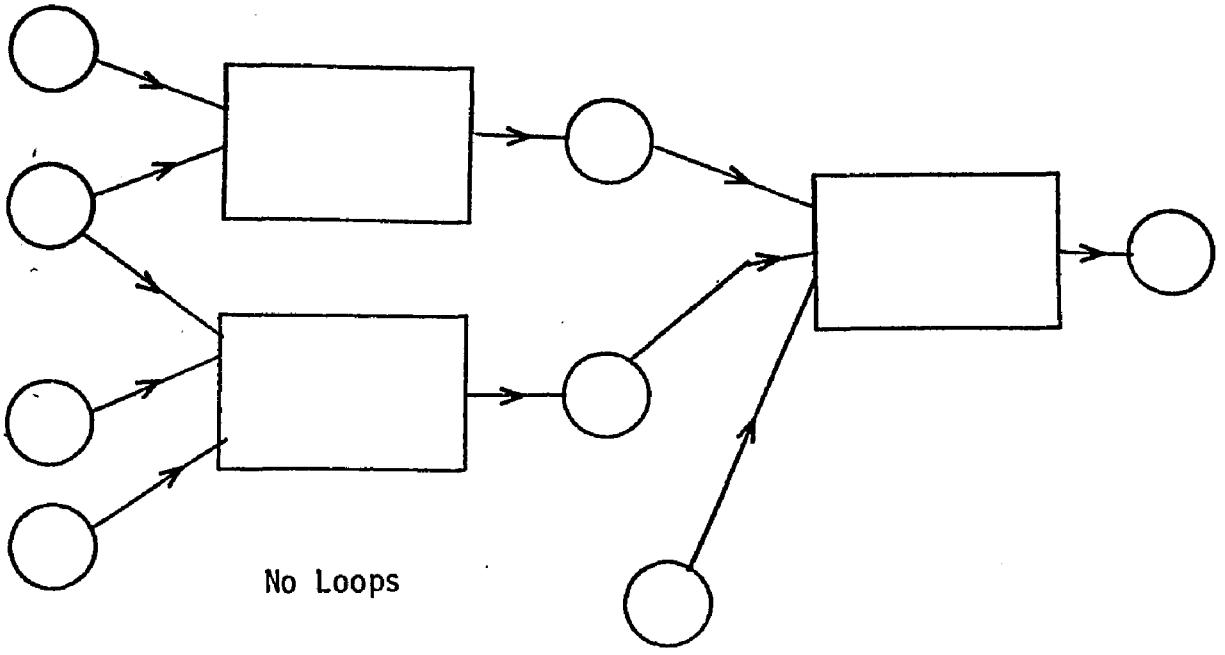
that the output of one process may be an input to another process. Thus a network is created (Figure 15). The use of a network oriented approach to logical systems specification lends itself well to many computer-aided and automated techniques for information systems design. A detailed discussion is presented by Nunamaker[27].

Other Sources of Need for the DID

In addition to the sections of the DID already outlined, added constraints, features or design techniques will be reflected as changes in the DID. A sample of these is presented below.

New Versus Existing Systems, A Dichotomy. Many approaches to design ignore the existence of previous or current systems which were meant to meet a similar set of requirements. With the possible exception of the creation of a new organizational entity, most systems design takes place in an environment which has had an information processing system--formal or ad hoc. Consider the sources of change to an existing information processing system: (1) additional requirements, (2) changes to existing requirements, (3) changes in organizational requirements, (4) inefficiencies or inadequacies in responding to existing requirements or (5) changes to the computer environment. A decision must be made of whether or not to re-design or to modify the existing system.

Methods must be developed to capture the logical systems specification either as documented by formal documentation methods or as represented by software--especially application programs. The latter approach, going from existing programs to a requirements statement, is analagous to decompilation. A further advantage of this approach is



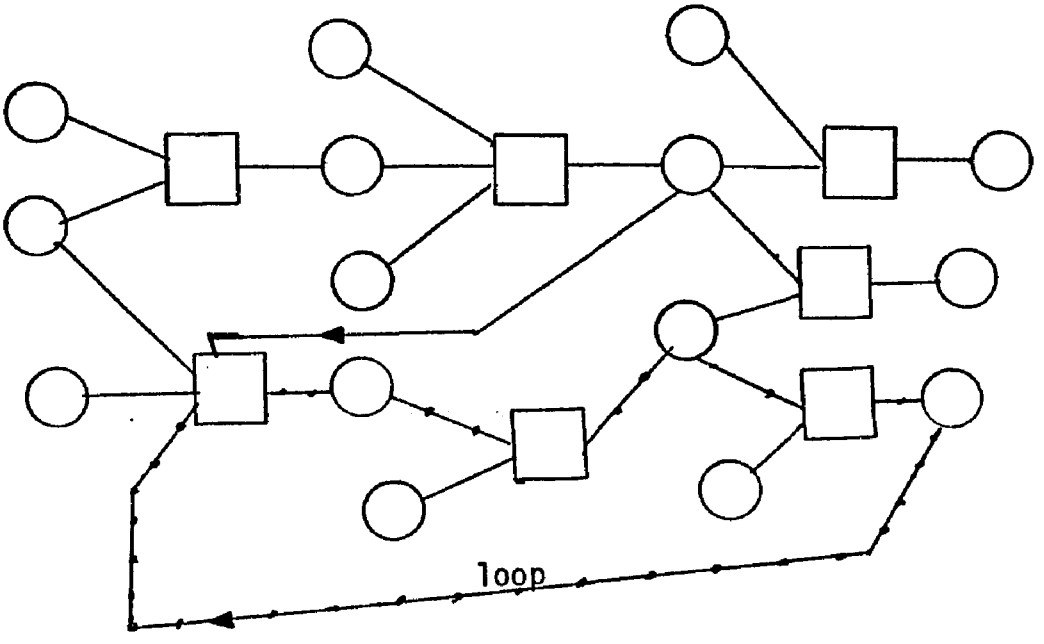


Figure 15. A Network

that it allows a comparison of existing application programs with new (or modified) requirements.

A more formal approach to the re-organization and/or re-design of the data base, query language and processing macros of an existing information system (the k'th iteration problem) follows. It is one (limited) application in the general area of design for existing systems, but provides useful insight and an approach within this area of interest.

Changes to Interactive, Query Oriented, Systems. The significant advance of this decade in data processing has been the development of systems which allow the user to interactively ask questions of the computer. These questions are, primarily, queries of the data base. This development involves both the use of random-access Input/Output and the development of query languages which allow the formal expression of questions. Figure 16 shows the structure of such a system.

The query language usually is similar to natural thinking and speech patterns. A query is frequently of the type:

FIND (list of data elements)

or:

FIND (list of data elements) SUCH-THAT (a conditional clause involving data elements and constrants)

"FIND" may be replaced by "LIST" (which really means find then list), "PLOT", etc.

The Query Language Analyzer then parses (decodes or separates) the query. Key words such as "FIND" and delimiters such as "SUCH-THAT" are located. The appropriate data elements are located and the correct commands to the data management system to retrieve occurrences of these elements are issued. Although most implementations of interactive

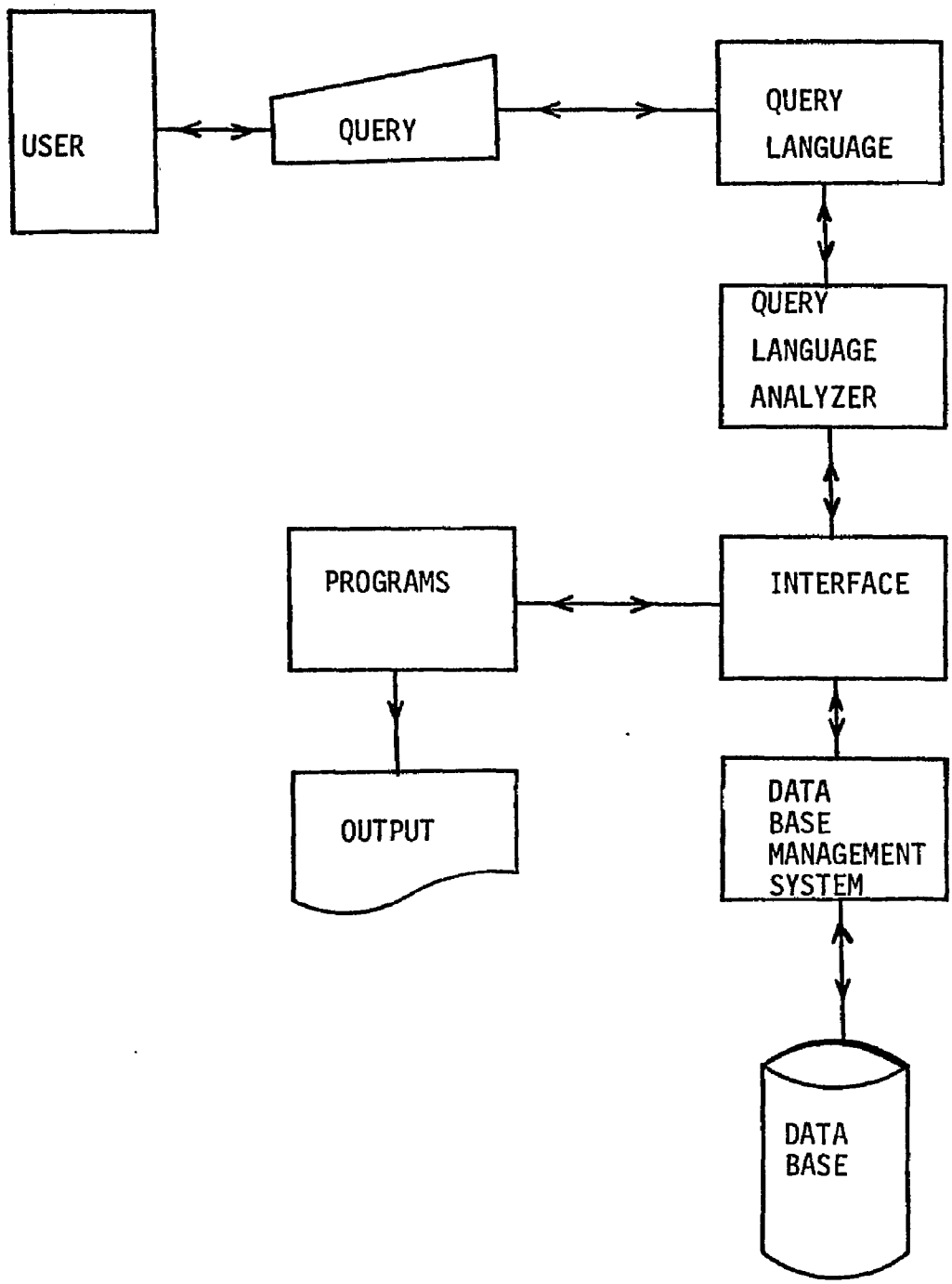


Figure 16. An Interactive, Query-Oriented System

systems will work strictly within the framework of a fixed query language, the following discussion which involves changing both the data base and the query language will be useful to show both the interaction and interdependence of these two, and implications on the RSM.

ASSUMPTIONS:

1. Data base exists and is described via a data description language[32].
2. Language exists to query this data base.
3. Usage (monitor) trails have been gathered for this iteration.
4. The PSL is current for the data base and the query language.
5. Performance measures exist[37].

DEFINITIONS:

1. Data Gathering.
   a. EXTERNAL - user specified suggestion or tasks in PSL.
   b. INTERNAL - trails of current queries are kept.
2. Change Types.
   a. SYSTEM (system visible/user transparent)-changes which primarily improve the efficiency of the internal handling of a query.
   b. USER (user visible/system visible)-changes which primarily improve the way a user asks a question, but essentially do not improve the internal handling of the queries.
   c. HYBRID-changes which effect system performance and the methods of user questions.

Data base query systems are unique in that they (1) are flexible and can accomplish (with efficiency dependent on structure of the data base and form of query language) any query of the data base, and (2)

changes to the data base and query language can be effected with relatively low (as compared to hardware changes or reprogramming) cost and can greatly effect the performance of the system. Because of the two above characteristics of data base query systems many changes to such a system represent improved methods or an improved environment for asking a query, or series of queries, to perform a given task. External data gathering then consists of PSL (or RSM generated PSL) which requests that additional features be implemented (such as graphing, etc.) or that current features or commands be modified or combined. These are similar to their corresponding PSL for a new system except that these changes may refer to existing processes (macro's, features or commands) whereas PSL for new systems deals only with data elements and their relation with each other (macro's, etc., don't exist at the requirements specification stage of a new system). Internal gathering of data consists primarily of determining usage characteristics and patterns of query systems. The data is then (somehow) analyzed for patterns or flow, to determine if (1) changes in data base structure, (2) changes in query languages or (3) changes in query handling should be candidates for implementation (after feasibility study, cost/benefit analysis, etc.).

Consider the following data base structure and query sequence:

1. LIST A B C D E such that e greater than 10, f ≠ "update", g less than 40

   Response: there are 1000 such record occurrences, list will be offline.

2. LIST A B C D E such that e greater than 18, f ≠ "update", g less than 40

   Response: there are 400 such record occurrences, list will be offline.

3. LIST A B C D E such that e greater than 24, f ≠ "update", g less than 40

   Response: 30 times listed.

4. LIST A B C D E such that e greater than 28, f ≠ "update", g less than 40

   Response: no such record occurrences.

5. LIST A B C D E such that e greater than 27, f ≠ "update", g less than 40

   Response: 2 such record occurrences (listed).

END OF QUERY:

The above sequence of queries lends itself to various kinds of analysis, data gathering and feedback:

1. User supplied suggestion: allow me to use symbols or abbreviations, such as "gt" for greater than, etc.

2. User supplied suggestion: allow me to establish a group of items to be listed, i.e., Group (name = N) = A B C D E, and change queries to LIST (N) .....

3. User supplied suggestion: allow me to repeat a query with a change parameter. Query line 2, above, becomes "ABOVE (e gt 10) ch (e gt 18)" or ch "10" to "18" REPEAT.

4. User supplied suggestion: provide me a macro to find the n greatest value(s) of an item. LIST A B C D E such that n(MAX e), f ≠ "update", g less than 40.

5. Internally originated improvement: save all internal pointers so the 2nd through 5th queries are handled more efficiently.

6. Internally originated improvement: change paging so the above query, which is frequently asked, doesn't cause excessive page accesses.

Analysis of the suggested changes:

1. The first two suggestions most probably would be implemented in such a way as to create a small additional overhead to the system but provide for easier entry of queries. These are "USER" changes.

2. The third suggestion could be implemented as a "USER" change, but if the "ABOVE" keyword causes the system to save pointers internally, it would be a "HYBRID" change.

3. The fourth also can be implemented as either a "USER" or "HYBRID" change, depending on methods employed.

4. The fifth is a "SYSTEM" change because the only thing the user might notice is better response.

5. The sixth is also a "SYSTEM" change.

Customized Query Language. A Customized Query Language(CQL) is a more generalized result of this type of analysis. The data base structure and the structure of the query language are input into an analysis which generates the CQL. The CQL rather than parsing any general query completely then searching the data base for the appropriate data to answer the query, or determine that the query is faulty, combines its knowledge of the structure to enhance both the parsing and answering of the query. For example:

(INPUT) List all cities which have a pollution level greater than 100 parts/million.

Analysis by a general query language:

(SYNTAX CHECK) "List" a correct command (continues)
"cities" is a correct item type (continues)
"which have" is a correct "verb" (continues)
"pollution level" is a correct item type (continues)
"100 parts/million" is a correct "number" (continues)

(calls data base)

finds record type "cities" (continues)
finds first occurrence of record type "cities" (continues)
finds item type "pollution level" ***ERROR***

Returns with response,
"pollution level" is not contained in record type named "cities".

Analysis by a customized query language:

(SYNTAX AND STRUCTURE CHECK)

"list", "cities", "which have" check as above.
"pollution level" a correct "item type" (continues)

data base structure check finds item type "pollution level"
is not contained in record type "cities". ***ERROR***
response as above.

(The correct query would have been "list all cities which are along rivers which have a pollution level greater than 100 parts/million.)

The CQL thus responds to this invalid query with considerably less processing, never having tried to access data, etc. Similarly, it responds to valid queries quickly as it contains the appropriate knowledge of the data base structure.

Real-Time Versus Batch, A Continuum. An aspect which is often neglected in the requirements determination process is an accurate determination of the time requirements. Although there are systems which are clearly batch, or real-time; or are constrained to be one or the other, the more general case is where the user specification of requirements determines where along this continuum the system response should be. Cost/benefit analysis may interface with this determination.

Clearly faster response is costlier, but to what extent is it better. For example, in a reservation system (such as an airlines reservation processing system) is instantaneous (say, one second) response any better than ten-second response--consider that the ticket agent must first enter the appropriate query, then read the response, decipher it, then courteously explain it to the customer, etc. Would the customer even notice the difference between one-and ten-second response. On the other hand, ten-second versus two-minute response may

be noticeably different in this situation. A "Competitive Edge" is gained and the cost of slow response may be lost customers, etc. Figure 17 shows the time continuum. The determination of cost versus response time may be most difficult in that it may involve designing and costing many completely different systems to evaluate the different alternatives. The "benefits" of quicker response or the sensitivity of the user to different response times must also be determined and quantified.

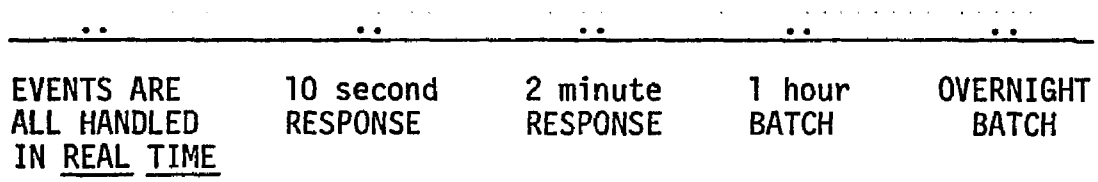
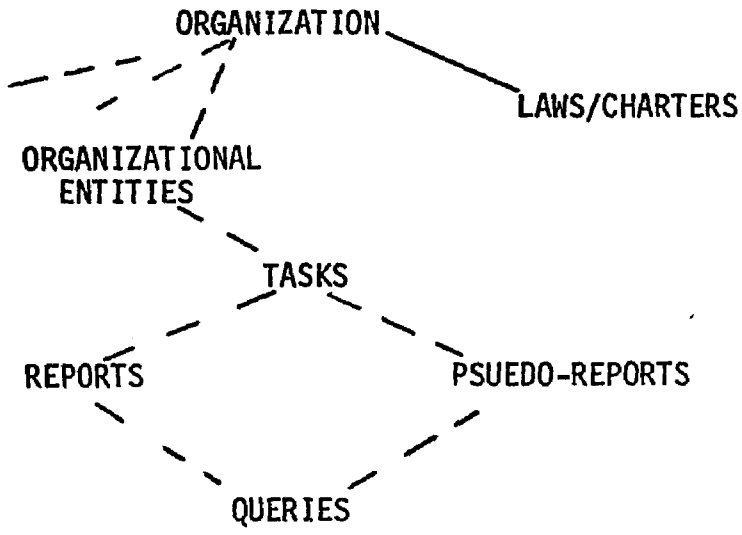


Figure 17. Workload Profile, Time Continuum

Contents of the DID

Having identified steps towards systems design and special considerations which impact on the DID, the task remains to list the contents of the DID. No such list can be complete as not all approaches to design have (or can) be enumerated and studied. Similarly not all special features or techniques can be considered. The DID approach should provide a guideline for evaluating future design techniques and future system features, and their impact on the information requirements of systems design. Figure 18 is a summary of the information in the DID.

This chapter has discussed the RSM concept, a method for gathering information for the DID and communicating it to the design process. The output of the RSM, if properly done, can be considered a "functional definition" of the target system. The DID is then defined and developed. An approach to systems design is defined and the required information is



- SYSTEM DESIGN OBJECTIVES
- SYSTEM DESIGN CONSTRAINTS
- COST/BENEFIT DATA
- MEASURES OF EFFECTIVENESS
- DESCRIPTION OF ENVIRONMENT
- DESCRIPTION OF EXISTING COMPUTER SYSTEMS
- DESCRIPTION OF OTHER IPS
- OTHER DOCUMENTATION
- TIMING DATA
- HARDWARE/SYSTEM SOFTWARE DATA
- CQL DATA
- K'TH ITERATION DATA
- DESIGN MODEL INPUT

Figure 18. Design Input Data Base

determined. A network oriented approach to logical systems specification is discussed. The dichotomy between designing for new versus existing systems is discussed as it impacts the DID. The impact of query oriented languages and customized query languages is discussed. Finally, the contents of the DID is listed. The following chapters will detail the use of PSL and PSA within the RSM.

CHAPTER III - USING PSL AND PSA

This chapter explores the Problem Statement Language(PSL) and the Problem Statement Analyzer(PSA) in great detail. The syntax of the PSL is described; the various PSA aids and reports are discussed; the use of PSL and PSA in the content of the RSM is developed; and an example, Company Z, is used to illustrate, test and evaluate the RSM.

History and Documents

The PSL and PSA have evolved from Nunamaker's work[27]; Stieger and Teichroew[38] introduced the original PSL Preliminary User's Manual in 1968. The structures of the PSL were expanded to include Problem Statement Units(PSU's) in the following PSL manual[39], additional changes to handle such things as growth rate (of volume) were detailed by Koch[40]. The current version of PSL is described by Teichroew, et al[41]; the PSL reference manuals[42,43] give an exact syntax of the current implementation. PSA commands are presented by Berg, Hershey and Bastarache[44] and Bastarache[45]. The growth of PSL and PSA has followed the increased complexity of systems and systems design, and the feedback from numerous users of earlier versions of PSL and PSA. The following discussion assumes knowledge of PSL version 3.0.

The PSL language manual provides the user (problem definer) with the correct syntax for using the PSL to express given information. It does not provide an extensive guide for a Requirements Statement Methodology.

The following review of PSL and PSA should provide a useful guide to understanding PSL and PSA within the framework of an RSM.

Usage

The PSL statements have been divided into seven categories to help demonstrate the structure of PSL and for clarity:

1. STRUCTURE STATEMENTS - These statements provide for structure within the REAL-WORLD-ENTITIES, OUTPUTS, INPUTS and SETS. Each of these "objects" can be structured. For example, an OUTPUT can consist of many parts (each of which is also an OUTPUT), each of these parts can, in turn, consist of still other parts (again, OUTPUTS). PSL restricts the structure of REAL-WORLD-ENTITIES, OUTPUTS and INPUTS to tree structure, that is each "object" can only be part of one other "object". The SET can be a SUBSET of many other SETS, thus allowing a network representation.

2. DOCUMENT FLOW STATEMENTS - The flow of documents, INPUTS and OUTPUTS, between the Information Processing System and REAL-WORLD-ENTITIES is expressed via GENERATES and RECEIVES statements.

3. DATA STRUCTURE STATEMENTS - The most complex structuring expressed by the PSL is data structure. The two primary statements in PSL for this purpose are CONSISTS and CONTAINED--for example, A CONSISTS(OF) B, C, D; and B (IS)CONTAINED(IN) A. Figure 19 shows the combinations of objects which can consist of, or contain other objects. Additional data structure is provided by the SET structure statements (SUBSET, see above) and the RESPONSIBLE-REAL-WORLD-ENTITY which links a REAL-WORLD-ENTITY with SETS.

4. PROCESS/DATA LINKAGE - The flow of data through the system is documented by various statements which both indicate the PROCESSES involved and the data used by these PROCESSES. The statements which communicate this information are: DERIVED (BY), DERIVES; GENERATED(BY), GENERATES; UPDATED(BY), UPDATES; USED(BY), USES; UTILIZED(BY), UTILIZES; RECEIVES. Figure 20 shows these structures being used.

5. "OTHER" STATEMENTS - SUBSETTING-CRITERION and RELATIONS are two special conditions which are identified by the PSL. ELEMENTS and GROUPS provide SUBSETTING-CRITERIA

for SETS. ELEMENTS and GROUPS also are associated with RELATION names. ENTITIES are RELATED (TO) other ENTITIES VIA RELATION names. PROCESSES MAINTAIN RELATION names and SUBSETTING-CRITERIA.

6. TIMING and CONDITIONAL STATEMENTS - To define frequency, INTERVALS are established and OUTPUTS, INPUTS and PROCESSES can HAPPEN a given number of times per INTERVAL. EVENTS are defined as a CONDITION becoming true (or false) or the INCEPTION or TERMINATION of a PROCESS. PROCESSES, in turn, are TRIGGERED (BY) EVENTS.

7. DESCRIPTIVE STATEMENTS - The statements which have been grouped into this category include the IDENTIFIES statement (ELEMENTS and GROUPS IDENTIFY ENTITIES); the VALUES (ARE) statement, which gives valid ranges for ELEMENTS; CARDINALITY and VOLATILITY for SETS and ENTITIES, and PROCEDURE statements which provide narrative for PROCESSES.

OBJECT	CONSISTS(OF)	OBJECT
OUTPUT		GROUP ELEMENT
INPUT		GROUP ELEMENT
ELEMENT		----
GROUP		GROUP ELEMENT
SET		INPUT OUTPUT ENTITY
		(other SETS are SUBSET)
ENTITY		

OBJECT	CONTAINED(IN)	OBJECT
OUTPUT		SET
INPUT		SET
ELEMENT		GROUP EITITY INPUT OUTPUT
GROUP		GROUP ENTITY INPUT OUTPUT
SET		---- (SUBSET OF other SETS)
ENTITY		SETS

Figure 19. Data Structure Statements

OBJECT	STATEMENT	OBJECT
OUTPUT ELEMENT GROUP SET ENTITY	DERIVED(BY) USING	PROCESS SET INPUT ENTITY GROUP ELEMENT
OUTPUT	GENERATED(BY)	PROCESS (only one)
ELEMENT GROUP SET ENTITY	UPDATED(BY) USING	PROCESS SET INPUT ENTITY GROUP ELEMENT
ELEMENT GROUP ENTITY	USED(BY) (TO)DERIVE/ UPDATE	PROCESS SET ENTITY GROUP ELEMENT
PROCESS	UTILIZES	PROCESS

Figure 20. Process/Data Linkage

The PSL is composed of sixteen sections, each beginning with a section header. The sections contain statements which build the description. The section dealing with organization, etc., is the REAL-WORLD-ENTITY (RWE) section. REAL-WORLD-ENTITIES are structured via a tree structure with other RWE's. They generate INPUTS and receive OUTPUTS from the IPS. Sections describing INPUTS and OUTPUTS are also sections in the PSL. Data Structure involves ELEMENTS, ENTITIES and SETS. A GROUP section is used to group ELEMENTS and GROUPS into GROUPS. A RELATION section defines relations among ENTITIES. The PROCESS section defines PROCESSES and information flow. The CONDITION section defines conditions which lead to EVENTS which are also a section. An INTERVAL section defines intervals for timing purposes. The remaining sections enhance the PSL description. They are: PROBLEM-DEFINER which

identifies the "user" who is stating parts of the system description; DESIGNATE which is used to establish SYNONYMS: DEFINES which works to give values such as ATTRIBUTE-VALUE and KEYWORD to names in the PSL. The MEMO section establishes a narrative file which is referenced by the SEE-MEMO statement. Figure 21 shows the use of these sections. A summary of PSL and PSA sections appears in Appendix C.

CONDITION	OBJECTS (SECTIONS)
STATIONS (ORGANIZATIONAL ENTITIES)	REAL-WORLD-ENTITIES
DOCUMENTS	INPUTS OUTPUTS
FILES	SETS ENTITIES RELATIONS
DATA DEFINITION	GROUPS ELEMENTS
PROCESSING DEFINITION	PROCESSES
DYNAMIC BEHAVIOR	EVENT CONDITION (TIME) INTERVALS

Figure 21. PSL Sections

PSL, PSA and the RSM

The PSL language manual provides the user with the correct syntax for using the PSL to express information. This discussion will provide an approach to using the tools of PSL and PSA within the framework of the RSM. As in all of the following discussions, the emphasis will be on using the language constructs. It may frequently be easier to use a "temporary" form to gather much of this information and then translate

it into PSL. Often, some information exists in previous documentation or in the form of a previous system. Since most design efforts do not begin in a vacuum, an initial effort may be made to translate the already existing documentation into PSL. If this information already exists in machine readable form, it may be advantageous to program an editor to translate the data. The SOURCE statement can be added to indicate the origin of this data. The task of translating existing documentation into PSL is similar to the use of "temporary" forms and translator modules; these efforts may often overlap.

The requirements statement process, as limited by the current PSL, is essentially the logical system definition process. Although the PSL may be expanded, primarily via the ATTRIBUTE statement, this discussion will focus on logical systems specification using PSL and the RSM. The overall logical system specification will begin with a description of the organization, data flow and finally process definitions linking data into a network and a system. Figure 22 shows an overall view of the target system being described.

Using PSL and PSA still requires "good" systems design procedure. PSL is not well suited for certain policy oriented stages of systems design, but the use of narrative can be accommodated and PSL can provide some structure beyond that of simple narrative. The first step is to identify the users or problem definers using the PROBLEM-DEFINER section. The KEYWORD statement may define the overall area of responsibility such as a section of the DID. Similarly the KEYWORD may be used to outline detailed areas of responsibility such as RETAIL-ACCOUNTS-RECEIVABLE. The RESPONSIBLE statement will also link the problem definer with any

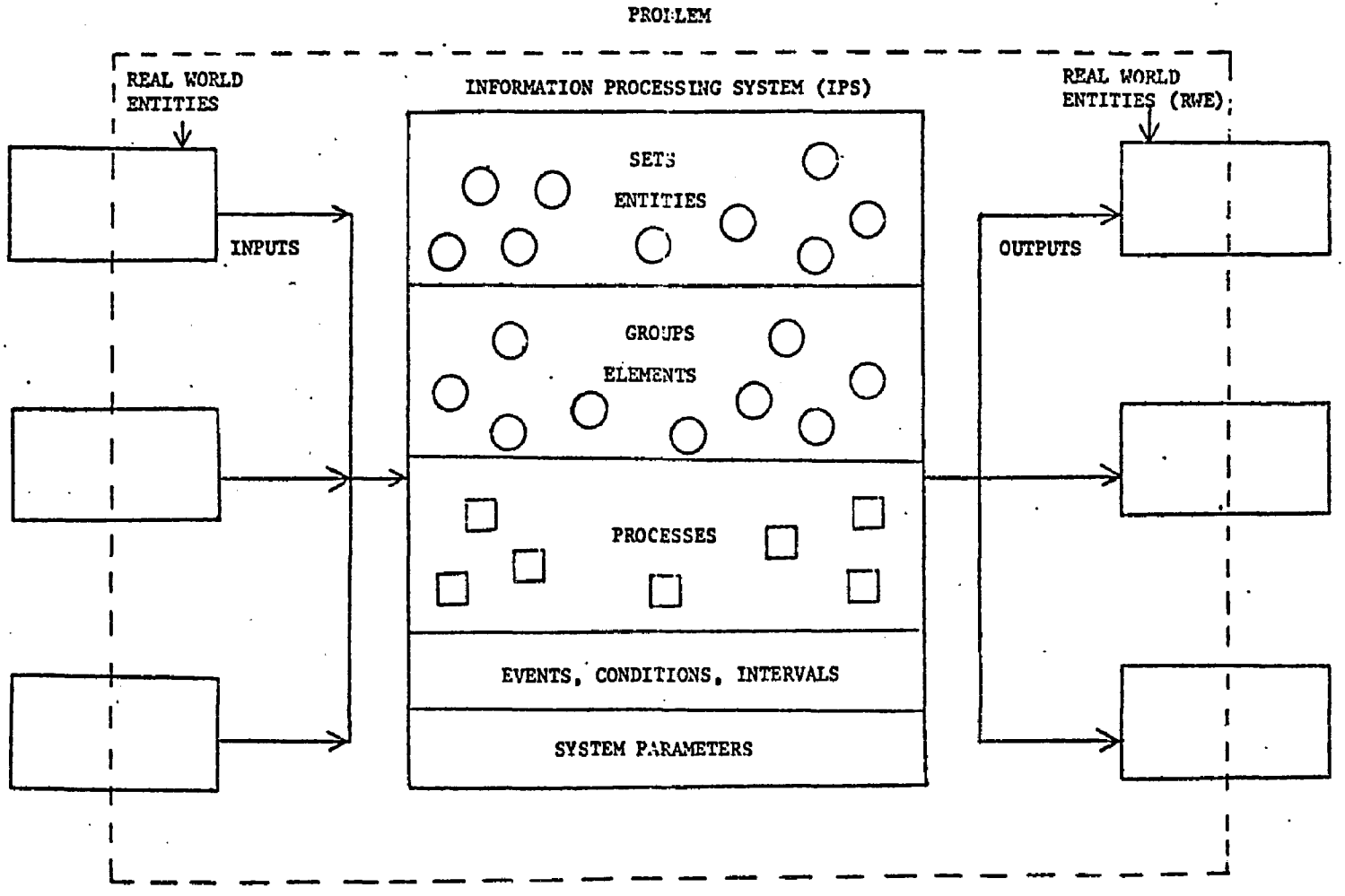


Figure 22. Model of the Target Systems Being Described in PSL Showing the Object Being Described[41].

(PSL) sections which he is responsible for. A MAILBOX may be defined for the problem definer. The SECURITY statement will limit different parts of the PSL to different problem definers. Little can go wrong with the PSL statements to this point except typographical errors and mistakes in the PSL syntax. As the design effort continues, conflicting areas of responsibility may occur.

Referring back to the DID, the first major section is the Objective/Constraint Specification for the Design Process. The current PSL has not, as yet, "hard-wired" any sections or statements for the expression and documentation of this portion of the DID. A facility for documenting and (later) managing a PERT-type control network would be a useful addition for the time and money considerations of the design effort. REAL-WORLD-ENTITIES may need to be defined as this early stage but the detailed description will wait for the Logical Systems Specification section below. Current PSL documentation is most easily limited to using MEMOs, but the PSL timing sections (objects), EVENT, CONDITION and INTERVAL may be used to describe the design process itself. Clearly this will require some "understanding" as to the expanded role of PSL.

The second section of the DID is Objective/Constraint Specification for the (Target) Information Processing System. As above, PSL is not yet tailored to easily specify this part of the DID. The use of PSL is limited as above.

Hardware and Systems Software Characteristic Specification is the third section of the DID. Again PSL does not yet have sections specifically designed to accommodate this DID section. A dynamic generation of PSL ATTRIBUTES would provide a useful guide for documenting this section.

Thus core constraints, storage requirements, read/write times, etc, could be defined as attributes for certain types of REAL-WORLD-ENTITIES and thus specifying characteristics as ATTRIBUTES would require the documentation of these within the PSL. Section four of the DID, Applications Systems Environment Specification, like the previous three sections does not lend itself to description by the current PSL. Again MEMOs could be used to store the narrative.

The current PSL is limited to the description of the logical system. A good base for the logical system description is a description of the organization involved. This is done via the REAL-WORLD-ENTITY section. The hierarchy of the organization is described using the PART and SUB-PARTS statement. A tree structure is then formed. It is likely that narrative will be required to explain different parts of the organization. The DESCRIPTION statement allows for this. If information describing many REAL-WORLD-ENTITIES or distinguishing between them is required, the narrative should appear in a MEMO to be referrenced by a SEE-MEMO statement. An ATTRIBUTES statement will be useful to add standard characteristics to each REAL-WORLD-ENTITY. Such ATTRIBUTES as number of employees, physical location and tasks may be required to fully describe a REAL-WORLD-ENTITY. The current implementation of the ATTRIBUTE statement allows only a list structure but future implementations may allow tree structures. Similarly if usage warrants, future implementations may "hard-wire" certain ATTRIBUTES into the PSL--that is establish statements and syntax for use. The use of KEYWORDS may be helpful here to show which offices (organizational entities, divisions, departments or REAL-WORLD-ENTITIES) handle which types of tasks. Such KEYWORDS as TAX,

PAYROLL, ACCOUNTING, BILLING, etc., may be useful for identifying appropriate REAL-WORLD-ENTITIES to be referenced by PSA commands (see below).

In most applications there will be many users (PROBLEM-DEFINERS) each describing his own portion of the organization. The most likely error at this point is a redundancy of names or conflicting areas of responsibility. In the first case, a more complete name such as "small-parts-inventory-control-section" may be needed to avoid the redundancy caused by a name like "inventory-control-section." In the second case a determination that there is no redundancy in names but that two different problem definers believe that they are responsible for the same area must be resolved. Errors may be made in specifying the organizational structure. The tree has been incorrectly envisioned or mis-specified. The PSA would be useful at this point to check the information already specified. The DICTIONARY command will provide a list of all names used by the system. A KWIC INDEX will provide groupings of similar names. A NAME-GEN on KEYWORD will divide the system into parts based on KEYWORD. To get a good look at the structure the STRUCTURE command for REAL-WORLD-ENTITIES or the PICTURE command (to draw the tree) will be useful. The FORMATTED-PROBLEM-STATEMENT will repeat the inputted information and link all information to sections as appropriate. If a PROBLEM-DEFINER section statement gives a RESPONSIBLE(FOR) a REAL-WORLD-ENTITY; the FORMATTED-PROBLEM-STATEMENT will also show the appropriate PROBLEM-DEFINER in the description of the REAL-WORLD-ENTITY. A chief advantage of the FORMATTED-PROBLEM-STATEMENT is that it does provide the complementary statements as appropriate.

When satisfied with the description of the organization, the tasks assigned to each part of the organization should be reviewed. The task assigned to top management is that of strategic planning. Strategic planning determines the objectives of the firm and which resources are to be applied to meet these objectives. Computer modelling, risk analysis, and forecasting techniques are frequently used to help with the task. But for the most part this task is staff-oriented, non-repetitive and variable in its requirements. As such it is difficult to document and frequently difficult to apply computer techniques to help meet these requirements. It may be advisable for these limitations to be expressed to top management. The strategic planning task is both critical and visable (especially since it involves top management) and unsuccessful attempts to "computerize" this function can set a bad tone for future IPS efforts. The allocation of resources may also place the strategic planning function outside the realm of a systems study or systems design effort. Cost/benefit analysis may determine that strategic planning is too costly to computerize or limit the design effort to providing models and forecasts. These may be documented in PSL in a method similar to the specification of reports (below).

The next level of tasks is defined as management control. At this level resources are allocated, performance is measured and rules are made. Much of the activity at this level is periodic (weekly, quarterly, etc.) with summaries and exception reports being the primary input. It is at this level, too, that inquiries of the data base and other non-repetitive applications abound. Whereas strategic planning outputs policies, objectives and constraints; the control level of an IPS outputs

associated with each task. The RSM should help both the problem definers and managers in making this determination.

Decisions and reports should be identified as OUTPUTS (required). If possible, the contents of each OUTPUT should be specified. Usually this is done by naming GROUPS of data and eventually breaking these down into ELEMENTS. SETS of data can be attached to REAL-WORLD-ENTITIES using the RESPONSIBLE(FOR) statement. INPUTS and OUTPUTS are attached to REAL-WORLD-ENTITIES via RECEIVES and GENERATES statements. At this stage in the logical system design, certain required OUTPUTS and available INPUTS can be identified. Complex OUTPUTS and INPUTS can be structured into parts, etc. Also at this point, data structure begins to take shape, either from previous documentation or from the breaking down of OUTPUTS and INPUTS to their contents. Chapters four and five will explore data base design using the data structure provided by the above.

The last level of tasks within an IPS is operational control. Tasks within the operational control level involve following the rules and procedures established by management. Everything at this level is formal and involves fixed procedures, thus lending itself well to the use of computers. The inputs are transactions or periodic events (such as END-OF-MONTH) and the outputs are actions, frequently specified by reports such as PICKING-TICKET, WORK-ORDER, etc. Although complex decisions are not made at this level, situations requiring the use of fixed procedures do occur. These situations or their resulting decisions should be identified. A major issue at this point in the specification process is completeness. Every task should be identified within PSL. Each task can then be broken down into other tasks and sub-tasks. These can

loosely be called PROCESSES. As additional problem definers look at the processes they may break them down into more and more processes (tasks and sub-tasks). This is a general feature of PSL, the ability to go top down, from the whole to a breakdown of its parts. PSL also allows a bottom up approach when needed. Confusing or unexplained items can be named as ELEMENTS or ENTITIES. PSA will continuously identify these items as being without a source or use thus prompting the problem definer to eventually define the PROCESSES involved.

Data flow now enters into the logical system specification. PROCESSES are formed to produce the required OUTPUTS (reports or decisions). The information required for these PROCESSES is then identified and structured. PROCESSES are then defined to provide for the newly defined information (data) and more data and more PROCESSES are defined. This builds a directed network depicting information flow. No one, set approach to problem definition will be "best" for all situations, but a progression from tasks to OUTPUTS to INPUTS frequently is useful. The PSA is useful in many ways to interact with this portion of the logical system specification. First, the FORMATTED-PROBLEM-STATEMENT, puts all the documentation together in an easily readable package. The use of the NAME-GEN can separate PSA reports into usable pieces so each problem definer can focus on his own area of concern. The PSA PICTURE command provides a graphic view of the processes, their connection with each other and the flow of data through the system. The PROCESS-INPUT-OUTPUT command also helps here. The DATA PROCESS INTERACTION MATRIX shows which data objects are input, output or updated by a given process. This may be useful in grouping data elements into groups, entities and sets. The

The PROCESS INTERACTION MATRIX shows which processes interact with each other. This is similar to the SODA precedence matrix and can be used for the grouping of Processes into Modules. Diagnostics at this point include lists of elements which are used but which have no source and data which is input but not used. With existing documentation and data dictionaries, a focus on ELEMENTS sections can yield required processes and data flow. Data structure is reviewed via the CONTENTS report. This report gives the contents of sets, inputs, outputs, entities and groups. The CONSISTS-COMPARISON shows the similarity in contents of different data groupings. The logical (not physical) data base design is greatly enhanced by using these reports.

Using PSL and PSA Within the Scope of the RSM

The RSM technique would have forms or an interactive terminal to prompt the user. Certain diagnostics should appear as immediate feedback (for example, redundancy warnings). When the user feels that he has completed a section of the PSA a "completeness-check" command should interactively prompt the user for information which is still lacking (for example, "PROCESS WEEKLY-PAY-GEN does not have any timing information, please designate HAPPENS/TIMES or specify EVENT and CONDITION information.") and allow its immediate entry. The interactive prompting of the user will not only assure completeness, but may also provide guidance and direction for the statement of the logical systems specification. The PSA list of completeness checks by Teichroew[46] is a basis for this discussion. Appendix C contains a draft of the completeness checks. The following are samples of the messages an interactive RSM will provide the user (PSL problem definer) with: (It should be remembered that the RSM

will use forms and keypunch for large bulks of data and the terminal for additions and modifications.)

```
(TERMINAL-RSM INTERACTIVE MESSAGE)
REAL-WORLD-ENTITY  NEW-EMPLOYEE-PROCESSING-OFFICE which is
part of REAL-WORLD-ENTITY  EMPLOYEE-PROCESSING-OFFICE does
not generate any input nor receive any output.  Please
specify input, output or enter KWIK to search for similar
names.

(USER RESPONSE)
KWIK INPUT

(RSM MESSAGE)
The following INPUTS were found:
            1.EMPLOYEE-OUT-PROCESS-FORM
            2.EMPLOYEE-WEEKLY-PAY-PROCESS-MODIFICATION-FORM
            3.NEW-EMPLOYEE-INITIATION-DATA-FORM
            4.EMPLOYEE-NORMAL-PROCESS-SALARY-CHANGE
Please enter INPUT name, number from above list, or NOTYET

(USER RESPONSE)
3

(RSM MESSAGE)
Thank you, the INPUT (3) NEW-EMPLOYEE-INITIATION-DATA-FORM  is
now generated by REAL-WORLD-ENTITY  EMPLOYEE-PROCESSING-OFFICE.

(RSM MESSAGE)
OUTPUT  YEAR-TO-DATE-ABSENCE-BY-DEPARTMENT-REPORT  which is
generated by  YEAR-TO-DATA-ABSENCE-PROCESS  which is a sub-
part of YEAR-TO-DATE-PROCESSING-PERSONNEL  is not received
by any REAL-WORLD-ENTITY.  Please specify appropriate REAL-
WORLD-ENTITY, or NOTYET.

(USER RESPONSE)
NOTYET

(RSM MESSAGE)
The NOTYET command has been entered, this diagnostic will
be held in abeyance for the remainder of this session and
printed on session summary report.

(RSM MESSAGE)
The following ELEMENTS are not contained in any inputs,
outputs or processes.  This listing is ordered by incidence:
            1.OUTPUT PAY-CHECK-HOURLY
                  ELEMENTS  1.EMPLYEE-NAME  2.HOURS-OVERTIME
                        3.TIME-IN  4.TIME-OUT  5.TOTAL-PAY
            2.PROCESS PAY-CHECK-HOURLY-PROCESS
                  ELEMENTS  1.TIME-CARD-GROUP
```

(USER RESPONSE)
Change 1.1 to EMPLOYEE-NAME delete
1.3 and 1.4 are part of GROUP TIME-CARD-GROUP
KWIK 1.2, 1.5

(RSM MESSAGE)
Change 1.1 to EMPLOYEE-NAME eleminates problem
There is no GROUP TIME-CARD-GROUP, TIME-CARD-GROUP is an ELEMENT
KWIK 1.2 HOURS-OVERTIME/a.HOURS-WORKED-OVERTIME/b.TOTAL-HOURS-WORKED
KWIK 1.5 TOTAL-PAY/a.TOTAL-PAY-HOURLY/b.TOTAL-PAY-SALARIED/c.YEAR-TO-DATE-TOTAL-PAY.

(USER RESPONSE)
TIME-CARD-GROUP should be a GROUP/ CONSISTS OF TIME-OUT and TIME-IN.
1.2a delete
1.5a save

(RSM MESSAGE)
GROUP TIME-CARD-GROUP established as requested.
1.2 is now HOURS-WORKED-OVERTIME
1.5 is now TOTAL-PAY-HOURLY
TOTAL-PAY is not contained in any inputs, outputs or processes.

(USER RESPONSE)
NOTYET

RSM MESSAGE)
The NOTYET command has been entered ....

The above sample of interactive PSL definition via an RSM has been simplified to show the types of response available. A complex system might "speak" in terms of some other requirements statement language or an English language and internally translate to PSL. Such terms as PROCESS or ELEMENT might never appear, nor would the user need to know that PSL has defined these terms. The sophistication, background and training of the different problem definers would govern this portion of the implementation.

Conflicts in use of PSL syntax (such as UPDATE versus MAINTAINED) might be avoided in similar fashion if an "intelligent" PSA would choose

which of these is the correct terminology for a given circumstance. The following chapter discusses data structure within the context of the RSM. This process would also be greatly enhanced if a user were prompted (i.e. asked the right questions at the right time) by an interactive PSA type analyzer.

An Example

An example was chosen to test and evaluate RSM concepts, to generate ideas for the RSM and, now, to help communicate these ideas. It would be impractical to choose a real world application so Company Z[47] was chosen. Company Z has been developed to closely simulate the information which a team of systems analysts would have available during a systems design effort. Appendix D contains excerpts of Company Z. The emphasis of Company Z is the logical systems specification and this exercise will emphasize that aspect of the RSM.

The project assignments given to various groups working with Company Z in effect replace Section 1 of the DID, Objective/Constraint Specification for the Design Process. The Introduction to the Company Z problem is an abbreviated Section 2, Objective/Constraint Specification for the (Target) Information Processing System. The logical systems specification begins with the description of the organization; the tasks are then described and the document flow determined. The documents are then broken down into data elements and the processes which yield these elements are defined.

Beginning with the description of the organization, the following PSL statements might be used:

REAL-WORLD-ENTITY COMPANY-Z;

SUBPARTS ARE SALES-DEPARTMENT, ACCOUNTING-DEPARTMENT,
SHIPPING-DEPARTMENT,...,PERSONNEL-DEPARTMENT;

REAL-WORLD-ENTITY SALES-DEPARTMENT;

DESCRIPTION;
The function of the Sales Department is to ....;

GENERATES CUSTOMER-ORDERS;

RECEIVES SALES-REPORTS;

This same information might be more easily communicated via an RSM. An interactive RSM together with an expanded RSL might produce the following (man-machine) monologue: (NOTE: * indicates RSM message, # indicates user entry; @ indicates return-key, end of line.)

*LOGON,JOHN PHILLIP USER,3X5Y@

(NOTE: 3X5Y@ is user password.)

#HELLO, WHAT DO YOU WANT TODAY?

*ENTER NEW ORGANIZATION DESCRIPTION@

#PLEASE NAME THE ORGANIZATION.

*COMPANY-Z@

#PLEASE NAME THE SUBPARTS OF COMPANY-Z.

*+DEPARTMENT: SALES,ACCOUNTING,SHIPPING,...,PERSONNEL@

(NOTE: The "+DEPARTMENT" entry will add "DEPARTMENT" as a suffix to all of the above names. RSM will also generate statements in PSL establishing these as REAL-WORLD-ENTITIES.)

#PLEASE ENTER DESCRIPTION AND COMMENTS OR NARRATIVE FOR SALES-DEPARTMENT.

*The function of the Sales Department is to ... @

#PLEASE ENTER TASKS ASSIGNED TO SALES-DEPARTMENT.

*CUSTOMER-ORDER-TASK, MARKETING-FUNCTION@

(NOTE: An expanded PSL would define TASK as a "hard-wired" attribute of REAL-WORLD-ENTITIES. Tasks will eventually break down to PROCESSES.)

#PLEASE NAME THE SUBPARTS OF CUSTOMER-ORDER-TASK.

*NOTYET,ALL@

(NOTE: RSM allows the user to skip parts of the definition and enter these parts later. The "ALL" signifies to skip the SUBPARTS for all of the current items, thus no request is made by RSM to break MARKETING-FUNCTION into SUBPARTS at this time.)

#PLEASE NAME THE INPUTS WHICH ARE GENERATED BY SALES-DEPARTMENT.

*CUSTOMER-ORDER;

#PLEASE NAME THE OUTPUTS WHICH ARE RECEIVED BY SALES-DEPARTMENT.

*SALES-REPORTS@

The PSL generated by the RSM from the above would be the same as that directly expressed by the user. Although the RSM entry involves more overhead and much more prompting, it assumes less knowledge of PSL and would be much easier to input. At his convenience, the user can enter additional information. The simplest types of statements involve such declarations as synonyms:

*SYNONYM: DEPT=DEPARTMENT@

*SYNONYM: CUST=CUSTOMER@

At this point RSM and PSA will guide the user as to further information to be specified. Similarly, when the user wishes to enter information which was previously skipped, he may do so:

*CUST-ORDER:DETAIL@

(NOTE: "DETAIL" commands RSM to request all information which is appropriate to this type of entity--in this case a TASK.)

#CUSTOMER-ORDER IS A TASK.

#CUSTOMER-ORDER IS PERFORMED BY (REAL-WORLD-ENTITY) SALES-DEPARTMENT.

```
*@
...
(NOTE:  The user indicates agreement by hitting return, if he
 wishes to make changes he enters them.)

#PLEASE NAME THE SUBPARTS OF CUSTOMER-ORDER.

*DETERMINE-CREDIT, PROCESS-ORDER@

#PLEASE NAME THE SUBPARTS OF DETERMINE-CREDIT.

*@

#PLEASE NAME THE SUBPARTS OF PROCESS-ORDER.

*+SHIP,+BILL,+BACKORDER@

(NOTE:  Here the + indicates that the name is augmented by the
 name of the "parent", i.e. PROCESS-ORDER-SHIP,PROCESS-ORDER-
 BILL, etc.)

#PLEASE NAME THE SUBPARTS OF PROCESS-ORDER-SHIP.

*NOALL@

(NOTE:  The NOALL statement ends the search for more subparts.)

#PLEASE NAME THE REPORTS (INPUTS) GENERATED BY CUSTOMER-ORDER.

*@

#PLEASE NAME THE REPORTS (INPUTS) GENERATED BY DETERMINE-CREDIT.

*+REPORT:UNDETERMINED-CREDIT,NEW-CUSTOMER,BAD-CREDIT,CREDIT-
   CEILING,CREDIT-OK@.

(NOTE:  "+REPORT:" places the suffix "-REPORT" on all above names.)

#PLEASE NAME THE REPORTS (INPUTS) GENERATED BY PROCESS-ORDER.

*@

#PLEASE NAME THE REPORTS (INPUTS) GENERATED BY PROCESS-ORDER-SHIP.

*SHIPPING-NOTICE,PRODUCT-PACKING-SLIP,CUSTOMER-INVOICE.

*CHANGE PROCESS-ORDER-SHIP:PROCESS-ORDER-SHIP-AND-BILL.

*DELETE PROCESS-ORDER-BILL. @
```

(NOTE: In answering the above question the user decided that shipping and billing were actually part of the same TASK and thus made the necessary changes.)

.
.
.

To this point the RSM has acted only as a text editor to ease the process of inputting PSL. Now RSM and PSA combine both to serve as documentation and to guide the user in completing the logical systems specification. The first command given by the user is a request for a formatted problem statement. This essentially asks for a look at the PSL which has been generated thus far:

FORMATTED PROBLEM STATEMENT

REAL-WORLD-ENTITY COMPANY-Z;

SUBPARTS ARE SALES-DEPARTMENT, ACCOUNTING-DEPARTMENT ... ;

REAL-WORLD-ENTITY SALES-DEPARTMENT;

PART OF COMPANY-Z;

TASKS ARE CUSTOMER-TASK, MARKETING-FUNCTION;

TASK CUSTOMER-ORDER-TASK;

PERFORMED BY SALES-DEPARTMENT;

SUBPARTS ARE DETERMINE-CREDIT, PROCESS-ORDER;

TASK DETERMINE-CREDIT;

PERFORMED BY SALES-DEPARTMENT;

(NOTE: This is a default because SALES-DEPARTMENT has no SUBPARTS.)

PART OF CUSTOMER-ORDER-TASK.

TASK PROCESS-ORDER.

PERFORMED BY SALES-DEPARTMENT;

PART OF CUSTOMER-ORDER-TASK;

SUBPARTS ARE PROCESS-ORDER-SHIP-AND-BILL, PROCESS-ORDER-BACKORDER;

```
TASK PROCESS-ORDER-SHIP-AND-BILL;

PERFORMED BY SALES-DEPARTMENT;

PART OF PROCESS-ORDER;

GENERATES SHIPPING-NOTICE, PRODUCT-PACKING-SLIP, CUSTOMER-INVOICE;

INPUT SHIPPING-NOTICE;

GENERATED BY PROCESS-ORDER-SHIP-AND-BILL;

(NOTE:  The extended PSL allows a TASK to GENERATE an INPUT, as
 opposed to a REAL-WORLD-ENTITY GENERATING that INPUT.)

     ...NO STTUCTURE STATEMENTS

     ...NO DATA STRUCTURE STATEMENTS

     ...NO TIMING STATEMENTS

(NOTE:  The above are diagnostics.)
.
.
.
```

The formatted problem statement has regurgitated the information which the user provided and has added both cross-references and diagnostics. Should the user wish to remedy those diagnostics which speak to omission of information, he may procede as follows:

```
*COMPLETE:SHIPPING-NOTICE@

#NAME THE INPUT WHICH SHIPPING-NOTICE IS PART OF.

*@

#NAME THE SUBPARTS OF SHIPPING-NOTICE.

*@

#WHAT GROUPS AND/OR ELEMENTS DOES SHIPPING NOTICE CONSIST OF.

*GROUP:CUSTOMER-NAME-ADD,ELEMENT: ITEM-NUMBER,ITEM-QUANTITY-
   SHIPPED@
.
.
.
```

#WHAT REAL-WORLD-ENTITY OR TASK RECEIVES SHIPPING-NOTICE?

*RWE:SHIPPING-DEPARTMENT.

In addition to these features the many PSA reports allow the user to look at various aspects of the system which he is defining. The first order of business may be to look at the organization as he has defined it. By requesting the PICTURE COMMAND for REAL-WORLD-ENTITIES (via a NAMELIST) an organizational chart is drawn. By requesting this same PICTURE COMMAND for TASKS, he can then see how the TASKS are structured and he can then request a pictorial view of which TASKS are performed by which REAL-WORLD-ENTITIES. Figures 23 and 24 show some of these PSA-type visual aids. Reviewing, this first step in logical systems specification has been to define the organization, determine tasks associated with RWE's and determine which documents (INPUTS) are generated by these tasks. The RSM forms and special, one time forms may be used in addition to interactive monologue to gather much of this information.

The next steps involve determining the data structure and flow. The PSL INPUTS (to the data process system) are actually outputs from the given organizational entities. The information contained in each of these documents must be determined and traced back to their sources, either within that RWE or elsewhere. Processes which are required in the information flow are determined and data structure is also described. Again Company Z will serve as the information source. This example will focus on the various credit reports. The TASK is DETERMINE-CREDIT. The first problem encountered is that the SALES-DEPARTMENT does not perform this task. The necessary changes to the logical system specification

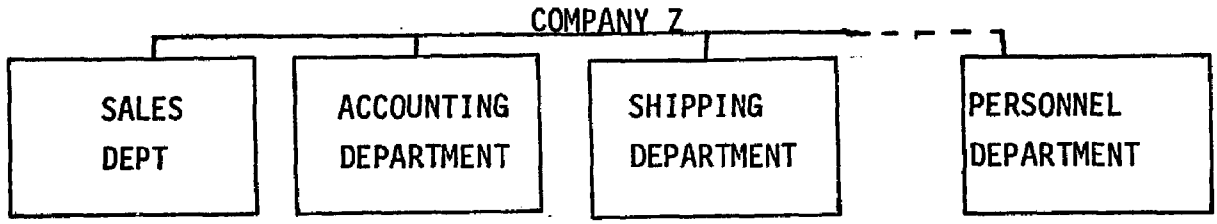


Figure 23. Real World Entity Picture

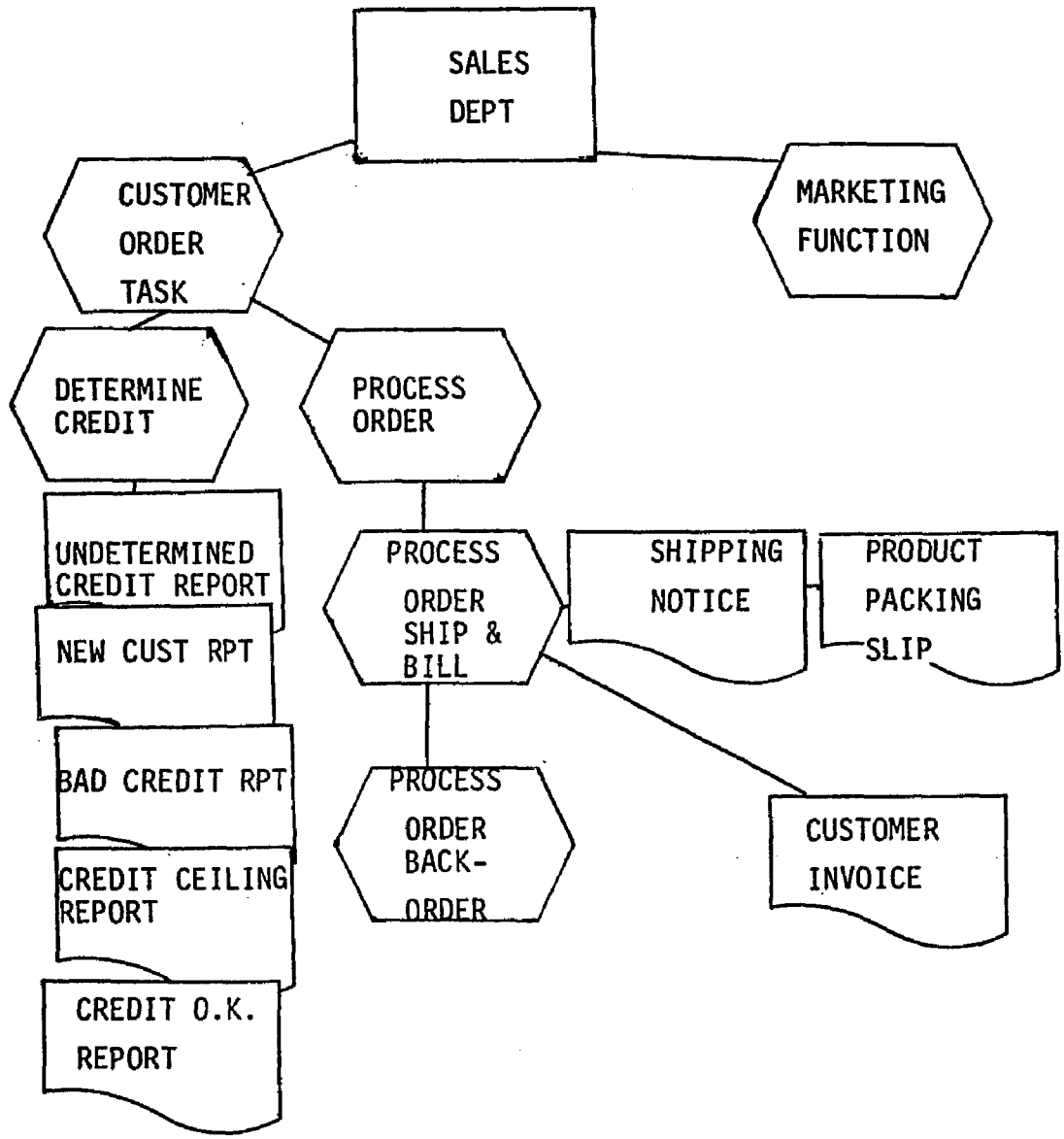


Figure 24. RWE-TASK-REPORT Picture

will require that a REAL-WORLD-ENTITY (say, CREDIT-DETERMINATION-OFFICE) performs this task and where this RWE fits.

The Data Dictionary Form (Figure 9) outlines the information gathered for each individual data element. Although this information can be entered and updated interactively via the RSM, the forms are more convenient. The expanded PSL will conform with this same information requirement. Similarly, the Input/Output Form (Figure 8) may be used in lieu of the interactive RSM. When dealing with an existing document the clerical effort of detailing the information contained on that form may be enhanced by using the Input/Output Form as a guide for gathering this required information. Since the form only requests frequency and data structure information, the RSM must be used to provide the structure, document flow, process/data linkage information. The RSM may also be used, especially when the process definition phase involves new processes and documents. The following sample shows the RSM helping the user with process definition:

```
*DEFINE:  PROCESS-ORDER@
#PROCESS-ORDER is a TASK which is PART OF CUSTOMER-ORDER-TASK.
#PROCESS-ORDER has no SUBPARTS, please begin definition.
*SYNONYM:  ORD=ORDER, QUANT=QUANTITY@, PROD=PRODUCT@
*IF:  CUST-ORD-QUANT GT PROD-QUANT-ON-HAND
*THEN:  CUST-QUANT-SHIPPED  EQ  PROD-QUANT-ON-HAND
*   CUST-BACKORDER-QUANT  EQ  CUST-ORD-QUANT - CUST-QUANT-SHIPPED
*   PROD-BACKORDER-QUANT  EQ  PROD-BACKORDER-QUANT + CUST-BACKORDER-
    QUANT
*GENERATE:  CUSTOMER-BACKORDER-MESSAGE
*ELSE:  CUST-QUANT-SHIPPED  EQ  CUST-ORD-QUANT@
```

```
#PLEASE DEFINE CUSTOMER-BACKORDER MESSAGE.
*NOTYET, CONTINUE@
*GENERATE:  CUSTOMER-INVOICE, SHIPPING-NOTICE, PRODUCT-PACKING-SLIP
*PROD-QUANT-SHIPPED  EQ  PROD-QUANT-SHIPPED  +  CUST-QUANT-SHIPPED@
#PLEASE DEFINE CUSTOMER-INVOICE.
.
.
.
```

A PSA picture command may be requested to provide a flowchart of the above process. The RSM will request the appropriate timing information. An EVENT CUSTOMER-ORDER-RECEIVED is defined; it triggers the CUSTOMER-ORDER-TASK.

Completeness Checks

To aid in the determination of requirements and their statement in PSL, the PSA provides analysis to the user. The completeness checks indicate what the PSL requires of the user.

These completeness checks serve to outline what information is required to describe PSL sections. The areas considered are:

Systems Flow
Structure
Data Contents
Processing
Size and Volume
System Dynamics

It should be pointed out that PSA does not provide checks for the quality or accuracy of the PSL.

This chapter has provided a detailed discussion of PSL, PSA and RSM. Features for a future RSM implementation have been detailed. The RSM provides the necessary input for data base design. The remainder of this paper will focus on aspects of the data base design problem.

CHAPTER IV - AN EXERCISE IN DATA BASE DESIGN

A problem of current interest is data base design. The logical systems specification provides a description of the data and data flow as each user (PROBLEM DEFINER in PSL) sees it. Design involves meeting the needs of the different users efficiently and effectively. An overview of the data base concept will introduce this topic.

Data Base, An Overview

There is no single definition of "data base" which will be acceptable to all. A data base is many things:

1. It is the foundation upon which information is built. (Information is knowledge derived from observations or from unorganized facts or data.)
2. The data base is a starting point for the development of a information processing system.
3. The data base is a broad foundation which stabilizes an information processing system.

The above definitions say, in effect, that a data base is not an end, but a means. A means for developing an information processing system from unorganized facts or data. A data base is a tool, not a product.

The key component of a data base must be the data it contains. Data may be defined as "something, actual or assumed, used as a basis of reckoning." Data is often looked at via a hierarchical concept of files, record and elements. An element (also called data item, field, item, elementary data item, data element) is the smallest piece or quanta of

logical comprehension. A record is a logical, defined collection or group of elements. (This is similar to the PSL ENTITY.) A file is a logical, defined collection or group of records. (This is similar to the PSL SET.) The adjectives, "logical" and "defined" above mean that the designer or originator of the record or set chooses what elements have the property of being members of that record or set. Similarly, the originator of the file has defined what records are logically members of the file--even if physically they are not part of the file. The term "information" is often used without defining its real meaning. Define information in terms of three elements: entities, attributes and values. (This is not to be confused with the PSL use of these terms.) Entities are (usually) objects. Entities possess certain characteristics or properties which distinguish them, uniquely, from all other entities. The distinguishing properties are identifiers. Entities possess other characteristics or properties known as descriptors. A descriptor is an attribute/value pair. An object (entity) is completely defined in terms of attributes and values.

EXAMPLE: Johnny's Bicycle

Attribute	Value
Model	Tricycle
Color	Red
Brand	Acme
Owner	Johnny Smith
Class of "thing"	Bicycle

Note that in this example a unique descriptor, an identifier, may be missing. Note also that many attributes may have no meaning for Johnny's bicycle--number of doors, rank, horsepower.

ANOTHER EXAMPLE: John Smith, Sr.

Attribute	Value
NAME	Smith, John Henry, Sr.
SSAN	299-40-3354
RANK	O-3, Captain
DEPENDENTS	3
UNIT	Company A

SSAN (social security number) is a unique identifier, above. This can be shown in three dimensions:

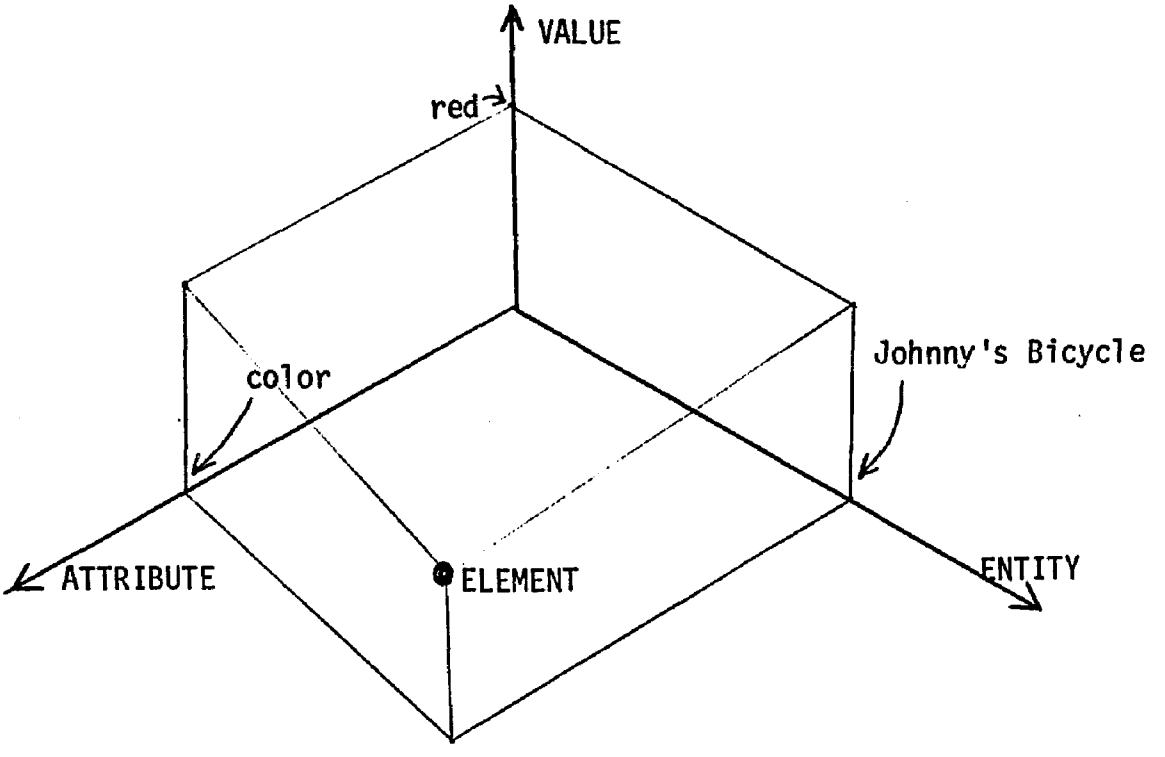


Figure 25. An Element

Although there may be many red bicycles, even many of Johnny's bicycles, a unique identifier such as SSAN will map to only one point:

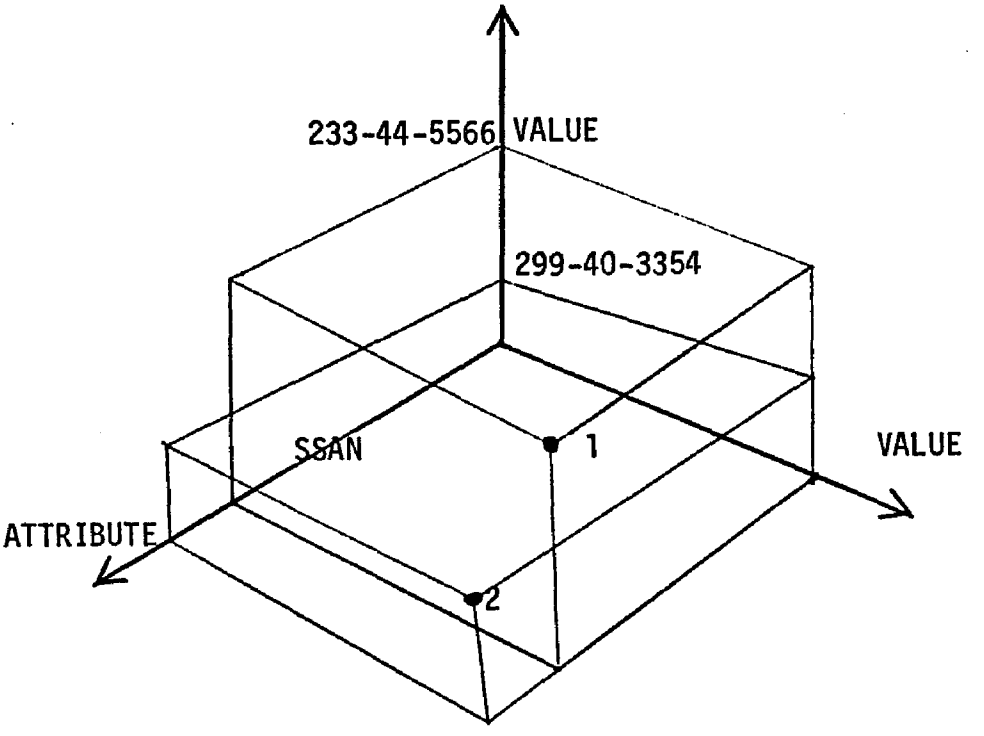


Figure 26. Two Elements

NOTE: Element 1 has SSAN 233-44-5566 name John Henry Smith, Sr.
Element 2 has SSAN 299-40-3354 name John Henry Smith, Sr.

The attribute SSAN is unique, the attribute name would map to the same point for both elements.

In light of the above descriptions, data base design may be defined as (1) defining a subset of the information space and (2) creating convenient paths between elements, i.e. the appropriate elements are defined (in terms of entity, attribute and value), an access method is chosen; and its relative location in the data base is determined. The data base design process will be considered in two stages, record design and set

design. Record design is the grouping of data elements into records. Set design is determining which records belong to what sets and how they can be accessed.

There are two major sources of potential record design. The user specifies information which is related to other information (in PSL via GROUP and ENTITY statements). The second source of record design is from an input/output analysis of the processes defined in the PSL. Basically, data items which are input or output together may be considered for grouping records.

The objective of record design is to minimize transport volume subject to storage restrictions. Transport volume is the amount of data which must be input and output to perform a given process. Each grouping of data items into a record has impact on the transport volume. One extreme of design would be to have each data item as a separate record. The disadvantage of this approach is the overhead associated with each record header. The other extreme, grouping all data items into one record, would minimize record overhead, but all data items would be input and output (with the record) even when they are not needed. Clearly a formal model and definition of transport volume and record design is needed.

Swenson[37] presents a formal approach to design evaluation. A formal approach for the design of record, i.e. the grouping of data items (ELEMENTS in PSL) into record types (ENTITIES in PSL) is presented below. A discussion of the impact of design techniques on the DID and RSM will follow.

Record Design

DATA ITEMS

Let D_i $i = 1 \ldots n$ be all the data items in the system.

Let DL_i be the length of each occurrence of D_i.

Let M_i be the number of times D_i appears in one occurrence. (This number is usually one. An average may be used if M_i varies.)

Let W_i be the overall number of occurrences (i.e. volume) of D_i.

Let U_i be the number of times each occurrence of D_i is updated during a given processing cycle.

RECORD TYPES

Let R_j $j = 1 \ldots k$ (k unknown) be the number of record types. (Any group of 1 to n data items is a record type.)

Let $DR_{ij} = \begin{cases} 0 & \text{if data item i is not in record type j.} \\ 1 & \text{if data item i is in record type j.} \end{cases}$

Let RL_j be the length of an occurrence of record type j.

$$RL_j = \underset{i=1,n}{SUM} \; DR_{ij} \, DL_i \, M_i + H + \tilde{P}t$$

Where H is the overhead for record header information (usually four words). $\tilde{P}t$ is an estimate of the overhead associated with set member/owner pointers. There are three words of storage for each member/owner pointer. Each member pointer consists of a word for previous member record, owner and next member record. Each owner pointer consists of a word for the first member record, last member record and number of member records. $\tilde{P}t$ is equal to three times the estimated number of sets that a given record type is owner/member of. Since set structure hasn't been determined at this point, $\tilde{P}t$ equal to 6 or 9 may be a useful estimate.

Let P_k be the K'th process

Let V_k be the volume of P_k (i.e. the number of times P_k occurs with a given cycle).

The Data Item/Process Incidence Matrix is defined to communicate the usage of data items (figure 27).

$$\text{Let } DP_{ik} = \begin{cases} 1 & \text{if } D_i \text{ is input to } P_k \\ -1 & \text{if } D_i \text{ is output from } P_k \\ 0 & \text{otherwise} \end{cases}$$

The Record Type/Process Incidence Matrix is derived from the above (Figure 28).

$$\text{Let } RP_{jk} = \begin{cases} 1 & \text{if there exists an } i \text{ such ahat } DP_{ik}DR_{ij} = 1 \\ -1 & \text{if there exists an } i \text{ such that } DP_{ik}DR_{ij} = -1 \\ 2 & \text{if both of the above conditions hold} \\ 0 & \text{otherwise} \end{cases}$$

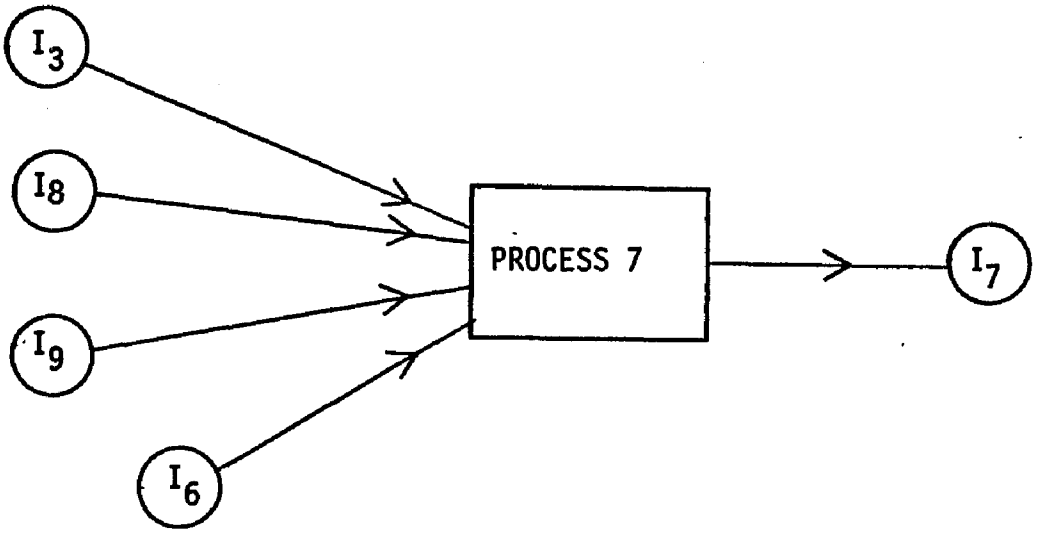
The first attempt at solution to this problem--find records for the K'th process to minimize transport volume (TV) would yield 1 input record containing all data items which are input to the process and one single output record containing all data items which are output. This approach fails because the interaction with other processes and redundant storage and updating are ignored.

To avoid this pitfall the following is added to the definitions:

Let I_m m = 1 ... p be identifiers to data items.

$$\text{Let } ID_{im} = \begin{cases} 1 & \text{if } I_m \text{ is an identifier for } D_i \\ 0 & \text{otherwise} \end{cases}$$

Let IL_m be the length of idenfifier I_m.



INCIDENCE MATRIX

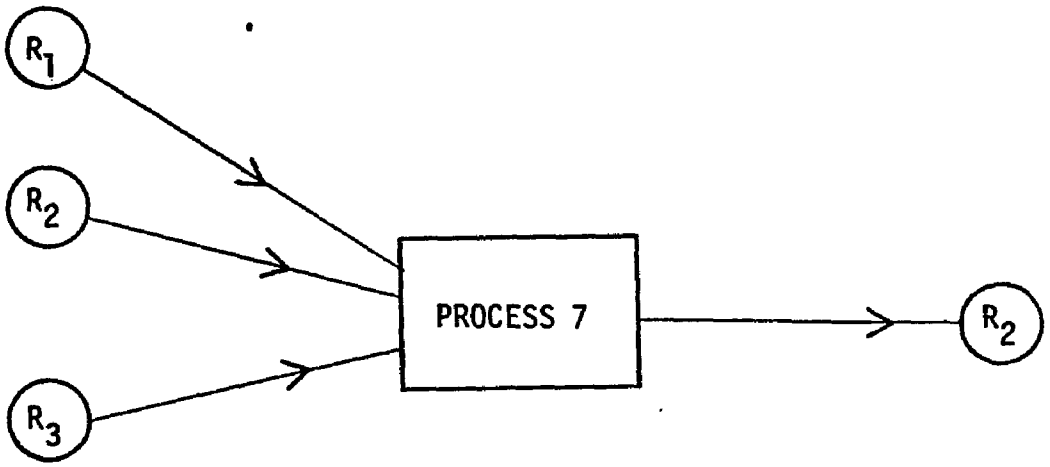
DATA ITEM	PROCESS 1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3							1					
4												
5												
6							1					
7							-1					
8							1					
9							1					
10												
11												

Figure 27. Data Item/Process Incidence

Record 1 consists of I_1 I_2 I_3

Record 2 consists of I_7 I_5 I_4

Record 3 consists of I_6 I_8 I_9 I_{10}



INCIDENCE MATRIX

	PROCESS											
	1	2	3	4	5	6	7	8	9	10	11	12
RECORD TYPE 1							1					
2							2					
3							1					

Figure 28. Record Type/Process Incidence

Establish a kronecker delta, d:

$$d_{jm} = \begin{cases} 1 & \text{if record type } j \text{ is identified by } I_m \\ 0 & \text{otherwise} \end{cases}$$

$$d_{jm} = 1 \quad \text{if} \underset{i=1,n}{\text{SUM}} \; DR_{ij} \; ID_{im} \neq 0.$$

Record length is then redefined as:

$$RL_j = \underset{i=1,n}{\text{SUM}} \; DR_{ij} \; DL_i \; M_i + H + Pt + \underset{i=1,m}{\text{SUM}} \; d_{jm} IL_m$$

We wish to group data items into records such that:

$$\text{minimize TV} = \underset{jk}{\text{SUM}} \; (\text{absolute value of}) \; RP_{jk} \; RL_j \; V_k + U_i W_i DR_{ij} RL_j$$

= processing volume + updating volume

Redundancy, a data item appearing in more than one record type is costly in that maintenance and updating must be performed on each (redundant) record. Assuming that redundancy will not be advantageous simplifies the record design problem.

$$\text{minimize TV} = \underset{jk}{\text{SUM}} \; (\text{absolute value of}) \; RP_{jk} RL_j V_k$$

$$\text{subject to } DR_{ij} = 1 \text{ (no redundancy)}$$

No closed form solution to this problem has been found. Certain simplifying assumptions may be useful before attacking a practical method for record design.

Let C_i be the cycle (time-cycle) associated with data item i. Cycles may be arbitrarily defined (twice/week, six times per month, bi-annually, etc.); their effect on record design is that no record can contain data items with more than one cycle. (The record is said to have the same cycle as the data items which it contains.) Thus a first partitioning of the data items may be made on cycles. The problem then becomes to find

the minimum transport volume for the data items, record types and processes within a given cycle. The total transport volume being the sum of the TV for each cycle.

The second partitioning may be done by identifier; all data items within a given record type must/should have at least one identifier in common. There may be pathological circumstances where the rule fails, but this assumption will be valid nearly all the time.

The PSL problem definer in specifying GROUPS, INPUTS, OUTPUTS and ENTITIES has provided candidates for record groupting. A sophisticated user may well define ENTITIES which map directly into record types. However, caution must be taken that the user specification is not binding on final design, only suggestive.

Let G_i be the GROUP (if any) that a data item belongs to.

Let E_i be the ENTITY (if any) that a data item belongs to.

Let S_i be the INPUT or OUTPUT (source or sink) that a data item belongs to. Since all of these can be structured into parts, the added restriction that only one INPUT or OUTPUT can go between a PROCESS and a REAL-WORLD-ENTITY. A list of potential candidates for inclusion into a given record is now created. Consider the WEEKLY-PAYROLL-HOURLY-PROCESS (and its subparts) which performs the weekly payroll processing for hourly (wage-rate) employees. First an exhaustive list of the objects (PSL INPUTS, OUTPUTS, ENTITIES, PROCESSES, GROUPS, SETS, ELEMENTS and REAL-WORLD-ENTITIES) in this part of the system is needed. PSA would provide this list via a simple command.

REAL-WORLD-ENTITIES

1. PAYROLL-PROCESSING-OFFICE
2. EMPLOYEE-BENEFITS-OFFICE
3. TAX-ACCOUNTING-OFFICE-EMPLOYEE-DIVISION
4. EMPLOYEE
5. UNION-LIAISON-OFFICE
6. FEDERAL-GOVERNMENT-SOC-SECURITY-OFFICE
7. FEDERAL-GOVERNMENT-INTERNAL-REVENUE-SERVICE
8. MANAGEMENT-INFORMATION-SERVICES-DATA-COLLECTION-OFFICE

PROCESSES

1. WEEKLY-PAYROLL-HOURLY-PROCESS
2. WEEKLY-PAYROLL-HOURLY-PAYCHECK-PROCESS
3. WEEKLY-PAYROLL-HOURLY-SOC-SEC-PROCESS
4. WEEKLY-PAYROLL-HOURLY-TAX-PROCESS
5. WEEKLY-PAYROLL-HOURLY-DEDUCTIONS-PROCESS
6. WEEKLY-PAYROLL-ALLOTMENTS-PROCESS
7. WEEKLY-PAYROLL-DISBURSEMENTS-NON-TAX-PROCESS
8. REG-HOURS-COMP
9. REG-PAY-RATE-COMP
10. OVT-PAY-RATE-COMP
11. OVT-HOURS-COMP
12. REG-PAY-COMP
13. OVT-PAY-COMP
14. GROSS-PAY-COMP
15. LOCAL-TAX-COMP
16. STATE-TAX-COMP
17. FED-TAX-COMP
18. SOC-SEC-COMP
19. ALLOTMENTS-COMP
20. HEALTH-BEN-COMP
21. RETIREMENT-COMP
22. LOCAL-TAX-YTD-COMP
23. STATE-TAX-YTD-COMP
24. FED-TAX-YTD-COMP
25. SOC-SEC-YTD-COMP
26. DEDUCTIONS-COMP
27. NET-PAY-COMP
28. ALLOTMENT-TOTAL-COMP
29. SICK-LEAVE-YTD-COMP

ENTITIES

1. YEAR-TO-DATE-TOTALS-EMPLOYEE
2. YEAR-TO-DATE-TOTALS-COMPANY
3. YEAR-TO-DATE-TOTALS-DEPARTMENT
4. EMPLOYEE-PERMANENT-INFORMATION
5. LOCAL-TAX-INFORMATION-EMPLOYEE
6. STATE-TAX-INFORMATION-EMPLOYEE

INPUTS

1. TIME-CARD-DOCUMENT
2. BONUS-INITIATION

OUTPUTS

1. PAYCHECK

GROUPS

1. TIME-CARD
2. EMPLOYEE-ADDRESS
3. STATE-TAX-DATA
4. LOCAL-TAX-DATA
5. EMPLOYEE-SOC-SEC-DATA
6. EMPLOYEE-TAX-INFORMATION
7. PAY-RATE-TABLES
8. EMPLOYEE-YEAR-TO-DATA-INFORMATION
9. EMPLOYEE-ALLOTMENTS-DATA
10. HEAL-BENEFIT-DATA
11. UNION-DATA
12. RETIREMENT-PLAN-DATA

ELEMENTS

1. TIME-IN
2. TIME-OUT
3. NORMAL-PAY-RATE-CODE
4. OVERTIME-PAY-RATE-CODE
5. SOC-SEC-YEAR-TO-DATE-EMP
6. HOURS-WORKED
7. SOC-SEC-FED-PERCENT
8. SOC-SEC-CEILING
9. LOCALITY-NAME
10. STATE-NAME
11. STATE-TAX-YEAR-TO-DATE-EMPLOYEE
12. STATE-CODE
13. STATE-TAX-AMOUNT
14. LOCAL-TAX-YEAR-TO-DATE
15. LOCAL-TAX-AMOUNT
16. BONUS
17. LOCALITY-CODE
18. EMPLOYEE-SSAN
19. EMPLOYEE-STREET-NUMBER
20. EMPLOYEE-CITY-STATE-ZIP
21. NUMBER-OF-DEPENDENTS
22. NUMBER-OF-ALLOTMENTS
23. ALLOTMENT-AMOUNT

24. OVERTIME-PAYRATE
25. OVERTIME-PAY
26. OVERTIME-HOURS
27. REGULAR-PAYRATE
28. REGULAR-HOURS
29. REGULAR-PAY
30. UNION-DUES
31. SUPERVISORS-SSAN
32. RETIREMENT-PLAN-CODE
33. RETIREMENT-DEDUCTION-AMOUNT
34. DEPARTMENT-NUMBER
35. DIVISION-NUMBER
36. HEALTH-BENEFIT-PLAN-CODE
37. HEAL-BENEFIT-PLAN-AMOUNT
38. JOB-SKILL-CODE-EMP
39. SICK-LEAVE-REMAINING
40. SICK-LEAVE-USED-THIS-PERIOD
41. GROSS-PAY
42. NET-PAY
43. DEDUCTIONS
44. FED-TAX
45. FED-TAX-YEAR-TO-DATE
46. EMP-PROMOTION-CLOCK-DATE
47. EMP-COMP-START-DATE
48. FED-TAX-RATE
49. SOC-SEC-DEDUCATION
50. NORMAL-HOURS
51. TOTAL-ALLOCATIONS
52. STATE-TAX-RATE
53. LOCAL-TAX-RATE

The first task is to better organize the information available. The REAL-WORLD-ENTITIES can be compared with the INPUTS and OUTPUTS. The INPUT TIME-CARD-DOCUMENT has no source. A REAL-WORLD-ENTITY (number 9) DEPARTMENT is established. Similarly Figure 29 shows that the majority of REAL-WORLD-ENTITIES have neither INPUTS nor OUTPUTS. This indicates that the PSL statement is incomplete. Continuing, the PROCESSES can be drawn to show the structure involved (Figure 30). The process flow yields the required INPUTS, OUTPUTS and/or ENTITIES. Figure 31 shows this flow and identifies newly created objects.

REAL-WORLD-ENTITIES/INPUT MATRIX

		RWE								
		1	2	3	4	5	6	7	8	9
INPUT	1									X
	2	X								

REAL-WORLD-ENTITIES/OUTPUT MATRIX

		RWE								
		1	2	3	4	5	6	7	8	9
OUTPUT	1				X					

Figure 29. RWE/INPUT, RWE/OUTPUT MATRICES

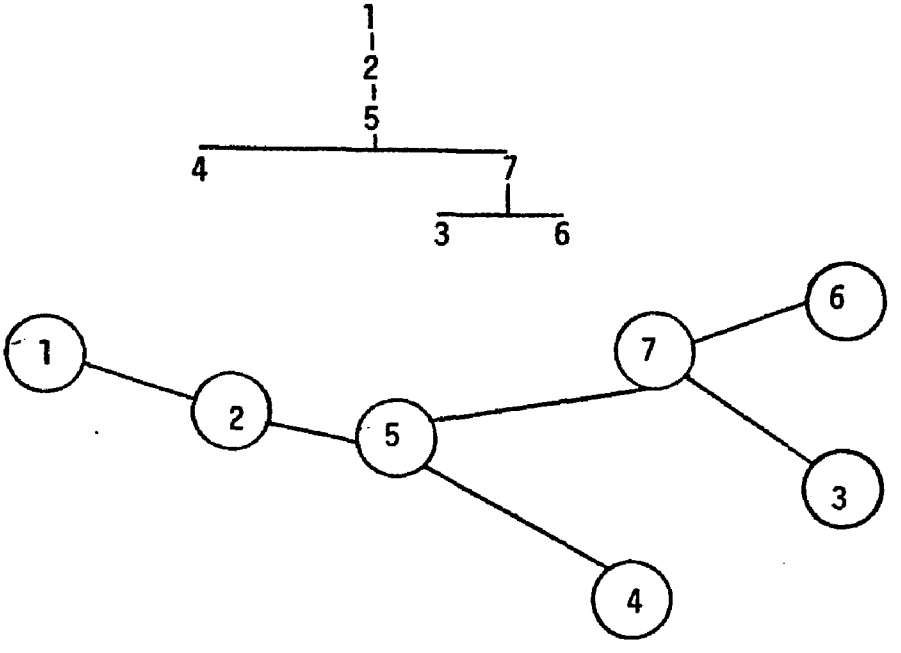


Figure 30. Process Structure

The process flow and the associated data flow in Figure 31 begin to point out certain deficiencies in the PSL, both in contents and in form. First of all ENTITIES and GROUPS (and SETS if they were used) are defined in a rather arbitrary fashion by the problem definer. Analysis of the data flow requires that the data be as unstructured as possible. Although GROUPS are useful concepts for describing the information, the grouping which they apply is arbitrary (user defined) and must not force the final data structure (as output from system design). Similarly, ENTITIES and SETS, although frequently specified by the user and left unchanged by the design process, should not force design. The example also shows problems with expressing arrays and tables (such as GROUP 7, PAY-RATE-TABLES); this is both a shortcoming in the current PSL implementation and a possible matter involving the understanding of PSL by different users. The RSM should choose a single method for describing an array or table then provide forms oriented towards describing the array or table and finally a translator module which will best express this structure in the PSL.

Before beginning a detailed attempt at record design, the process structure and flow should be considered. The structure implied is that the seven processes originally specified identified tasks or groups of processes, as in an overview. Processes 8 thru 29 are the breakdown into more elementary processes. PROCESS 14, GROSS-PAY-COMP (computation) is both visually and via precedence analysis central to the overall process. ELEMENT 41, GROSS-PAY, the output of this process is also very critical. The user specification of document (INPUT or OUTPUT) contents becomes an important consideration here. If the PAYCHECK is required to

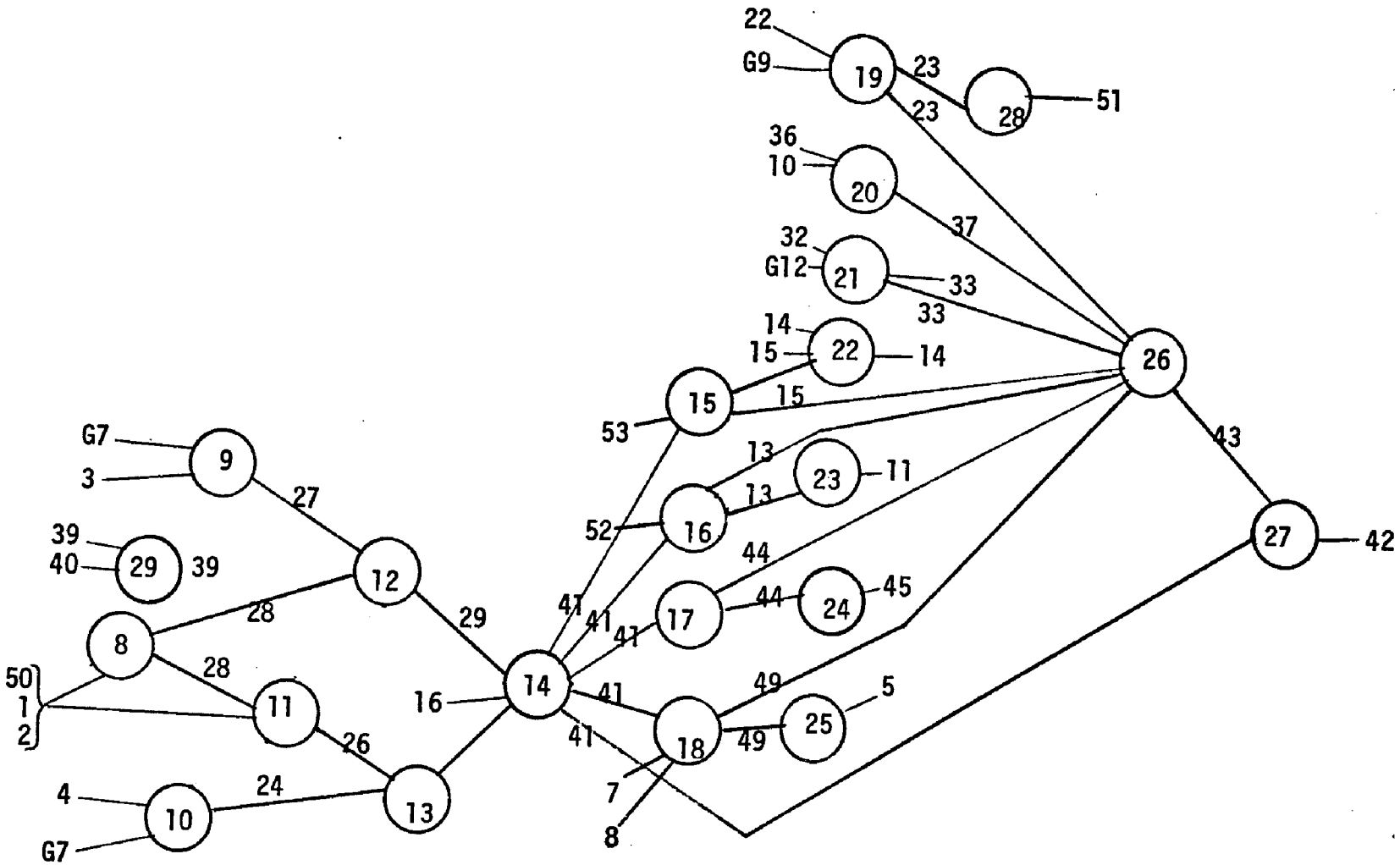


Figure 31. Process Flow

show GROSS-PAY in addition to NET-PAY, GROSS-PAY becomes an important consideration here. If the PAYCHECK is required to show GROSS-PAY in addition to NET-PAY, GROSS-PAY becomes part of an output. On the other hand if a PAYCHECK showing only NET-PAY is acceptable, GROSS-PAY would be an internally used ELEMENT which is never output or input. The effects of document contents on the target system is via data structure and grouping (record design). For example, if a report describing a PART required the pay classification of the employee who packed it, a PART oriented record (or set) might have to add employee data to it, and updates of employee pay classification would have to be posted to the part oriented record(s) in addition to the employee oriented file.

Another form of analysis for record design is REAL-WORLD-ENTITY(RWE) source/destination analysis of the ELEMENTS. Figure 32 attempts to show the relation of RWE's to ELEMENTS in matrix form. It can be seen that RWE 1, PAYROLL-PROCESSING-OFFICE, is the source of nearly all the elements and over half of the outputs from the system also go to this office. Thus an analysis based on this flow may be hindered by poor input data or input data which leaves little basis for discrimination or grouping.

An analysis of the flow of elements into and out of process using incidence matrices may also be a useful approach. Nunamaker[28,29] discusses this in addition to the grouping of processes based on data flow, etc. Ho[30] emphasizes the use of history (ADS history elements are equivalent to master files in a physical representation) elements for record design. He states, "Logical data base design examines the set of history relational structures used by each program module [group of

ELEMENT/REAL-WORLD-ENTITY	1	2	3	4	5	6	7	8	9
1									I
2									I
3	I								
4	I								
5	I								
6									I
7						I			
8						I			
9	I								
10	I								
11	I/O								
12	I								
13	I								
14	I/O								
15	I								
16	I								
17	I								
18	I								
19	I								
20	I								
21	I								
22	I								
23	I								
24	I								
25	I								
26	I								
27	I								
28	I								
29	I								
30					I				
31	I								
32	I								
33	I								
34	I								
35	I								
36	I								
37		I							
38	I								
39	I								
40	0								
41	0								
42	0								
43	0								
44							0		
45	I/O								
46	I								
47	I								
48							I		
49						0			
50	I								
51	0								
52			I						
53			I						

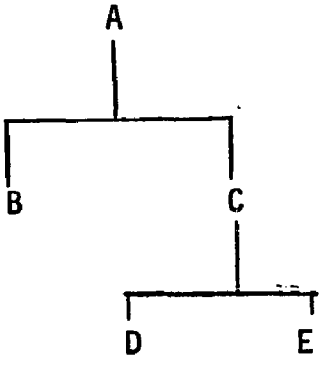
Figure 32. Real World Entity/Element Matrix

processes] in order to derive the logical relationships among the structures that would be represented in the data base. For each program module, the history relational structures that are used by the module are partitioned into classes characterized by identifier sets of and by the processing cycle of the relational structures belonging to the class. Then, each partition class is analyzed for logical relationships characterized by the identifier sets of the structures belonging to the class." Although following these and similar concepts may frequently produce a "good" design, no closed form solution to the record design problem has been found. An approach analagous to the clustering methods of AID, THAID and MNA may be a useful search tool to search through and evaluate various alternative groupings of elements into records. Chapter five explores this in greater depth. Emphasis on the RSM approach is enhanced when feedback from various design methods indicates possible "costs" associated with given record design based on user (problem definer) stated requirements for outputs, etc.

Set Design

With the use of a data base, the set structuring problem becomes one of satisfying each user in providing him with data structured as he sees it. In the language and concepts of the CODASYL committee, each user should be allowed to express the way he sees the system (thus developing sub-schemas) and the design process will then develop a schema which is satisfactory to all users. The restraint this puts on the user is that he be consistent (with his other pronouncements) and complete in his Data Description Language (DDL) specification of the sub-schema. It may be advisable to incorporate a cost/benefit approach into the

procedure for checking of sub-schema in relation to other sub-schemas. Is it worth the cost to allow a single user to view the data in a framework which is radically different from everyone else? The RSM will allow for feedback in this area. A general approach is to allow every user his own (consistent and complete) sub-schema. The first attempt at designing a schema is then the union (or combination) of the sub-schemas. Each sub-schema represents data structure via either tree or plex structures. Figure 33 shows a simple tree structure and the corresponding precedence matrix and reachability matrix representations.



PRECEDENCE MATRIX

	A	B	C	D	E
A		1	1		
B					
C				1	1
D					
E					

REACHABILITY MATRIX

	A	B	C	D	E
A		1	1	2	2
B					
C				1	1
D					
E					

Figure 33. Tree Structure and Associated Matrices

A second tree structure (from a second sub-schema) can then be added to the existing sub-schema in order to form a schema. Figure 34 shows such an addition. If we discount loops, the matrix representations are still useable. Figure 35 shows a representation which includes a loop. As the combination of sub-schemas yeilds more complex structures, plex and cross-reference structures may occur. A plex is a structure where more than one relation may link two objects. (In a tree the owner/member relation is the only link.) Figure 36 shows a cross-reference structure. Figure 37 shows a plex structure and the equivalent tree representations.

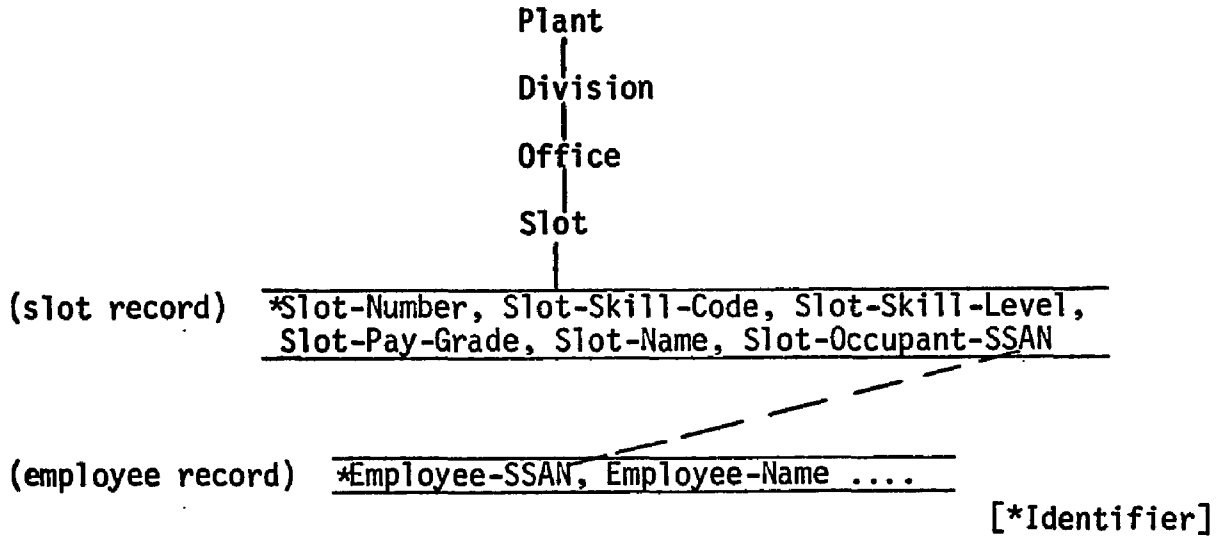
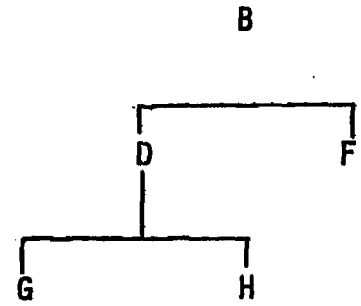


Figure 36. Cross-Reference Structure

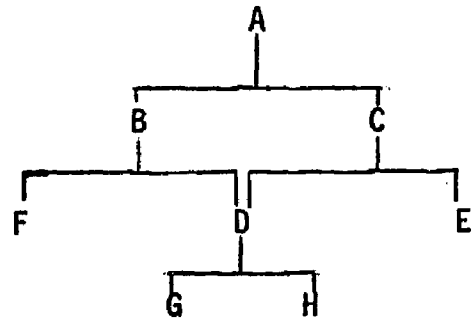
The set design process must resolve loops (conflicts among sub-schema) and accommodate plex, cross-reference and other complex structures. The resolution of a loop such as that in Figure 35 is



Sub-Schema 2

PRECEDENCE MATRIX

	B	D	F	G	H
B		1	1		
D				1	1
F					
G					
H					



Resulting Schema

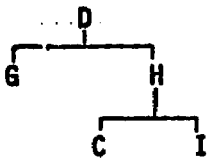
PRECEDENCE MATRIX

	A	B	C	D	E	F	G	H
A		1	1					
B				1		1		
C				1	1			
D							1	1
E								
F								
G								
H								

REACHABILITY MATRIX

	A	B	C	D	E	F	G	H
A	1	1	2	2	2	3	3	
B			1		1	2	2	
C			1	1		2	2	
D						1	1	
E								
F								
G								
H								

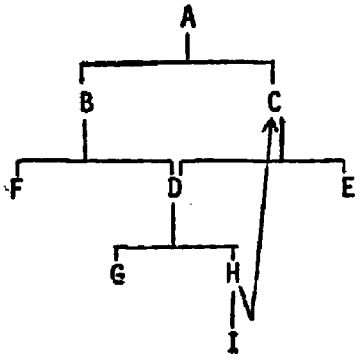
Figure 34. Combining Two Sub-Schemas.



Sub-Schema 3

PRECEDENCE MATRIX

	C	D	G	H	I
C					
D			1	1	
G					
H	1				1
I					



Resulting Schema

PRECEDENCE MATRIX

	A	B	C	D	E	F	G	H	I
A		1	1						
B				1		1			
C				1	1				
D							1	1	
E									
F									
G									
H			1						1
I									

REACHABILITY MATRIX

	A	B	C	D	E	F	G	H	I	
A	1	[1]	2	2	2	3	3	4		[4] via I
B		[3]	1		1	2	2	3		[3] via I
C			1	1		2	2	3		
D		[2]				1	1	2		[2] via I
E										
F										
G										LOOP C→H
H		[1]						1		H→C
I										

Figure 35. Schema With Loop

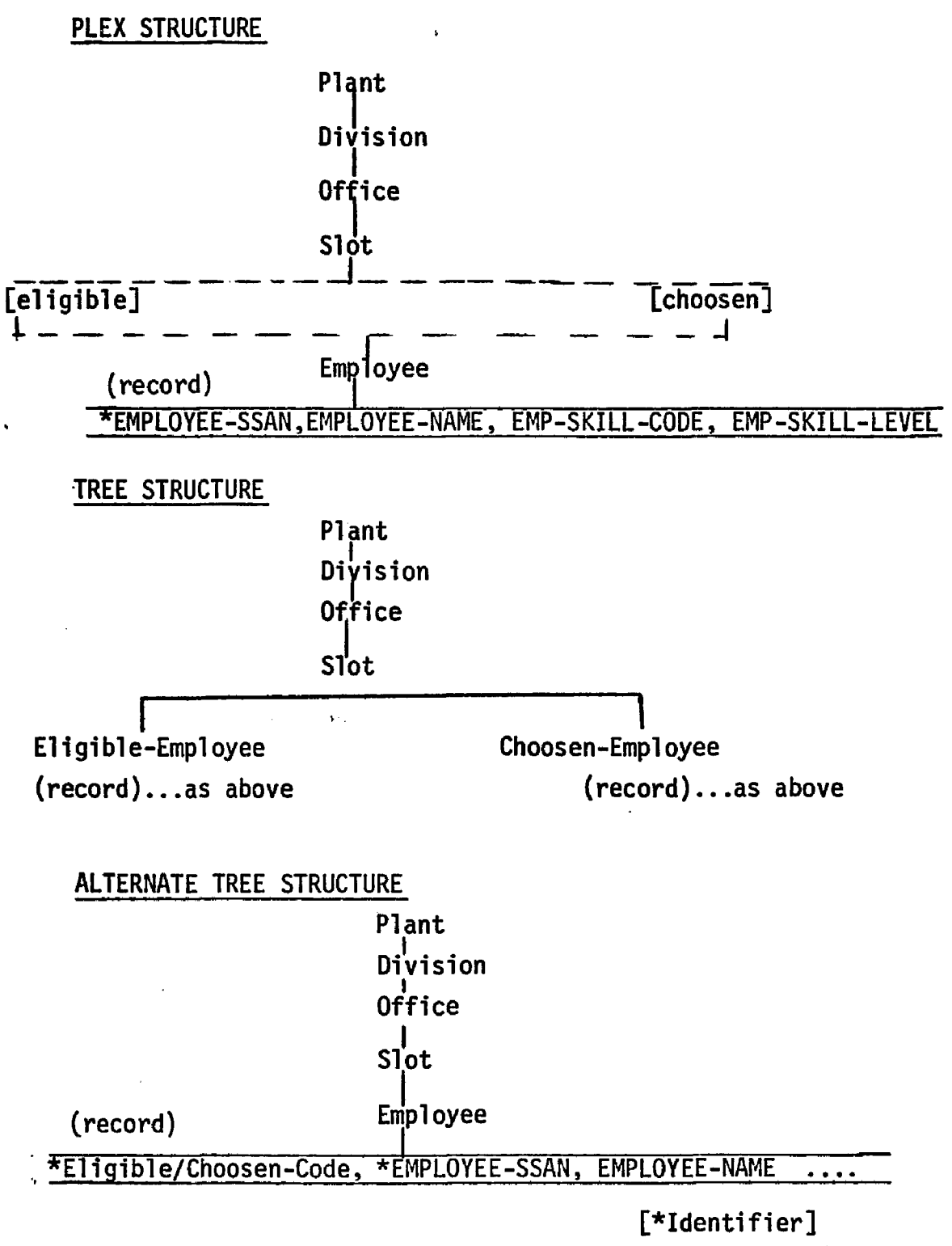


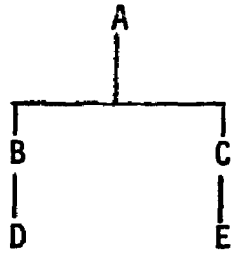
Figure 37. Plex Structure

accomplished by either having a second (copy) occurrence of the set C or a total reevaluation and restructuring of the data base schema. This will involve the RSM in asking the user to redefine the sub-schema or evaluating new sub-schema alternatives (based on other sub-schema and the schema). If, for example, sub-schema 1 was respecified so that set D was owned by B, not C, a new schema might be found (Figure 38).

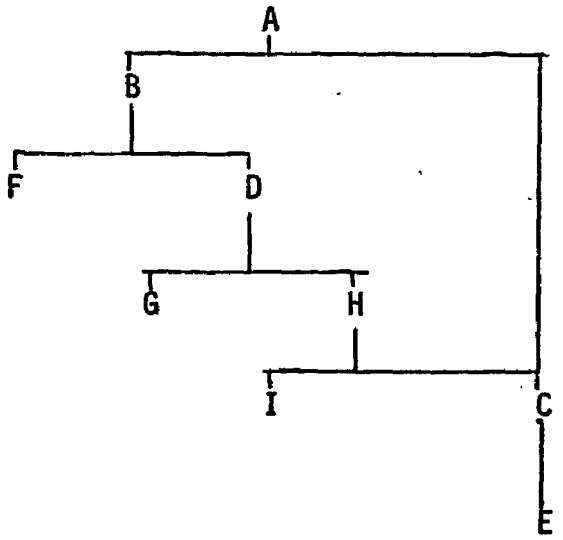
An important concept in data base schema design is the establishment of links among the sets. Figure 39 shows a network structure with various additional links. Should, for example, the link from U.S.A. directly directly to Individual be maintained in the schema. A formal approach to solving this important problem is now presented. This approach is (1) to find the minimum structure (2) determine the cost/benefits of alternative links and (3) to choose from among the possible links.

Finding the minimum structure involves reducing the network back to a tree (if possible). First the precedence matrix is obtained. The reachability matrix is then derived. The precedence matrix is then modified so no relation $P_{ij} = 1$ exists for any i and j where the $R_{ij} \neq 0$ (excluding the direct precedence itself). Figure 40 shows such a reduction. The determination of this minimum structure is done via a matrix operation similar to Nunamaker's[27] use of the Warshall algorithm[48] in determining reachability. A reachability matrix is formed using the maximum (as opposed to minimum) reachability. Then if $R'_{ij} \neq 1$ [R' is used to indicate this new reachability matrix.] and $P_{ij} = 1$; P_{ij} is an extraneous link and set equal to zero. This method will not reduce the network to a tree if "ties" exist. That is if there are two or more nodes i which directly precede node j--i.e. j has multiple owners.

Sub-Schema 1 (revised)



Revised Schema



PRECEDENCE MATRIX

	A	B	C	D	E	F	G	H	I
A		1	1						
B				1		1			
C					1				
D							1	1	
E									
F									
G									
H			1						1
I									

Figure 38. Revised Sub-Schema/Schema

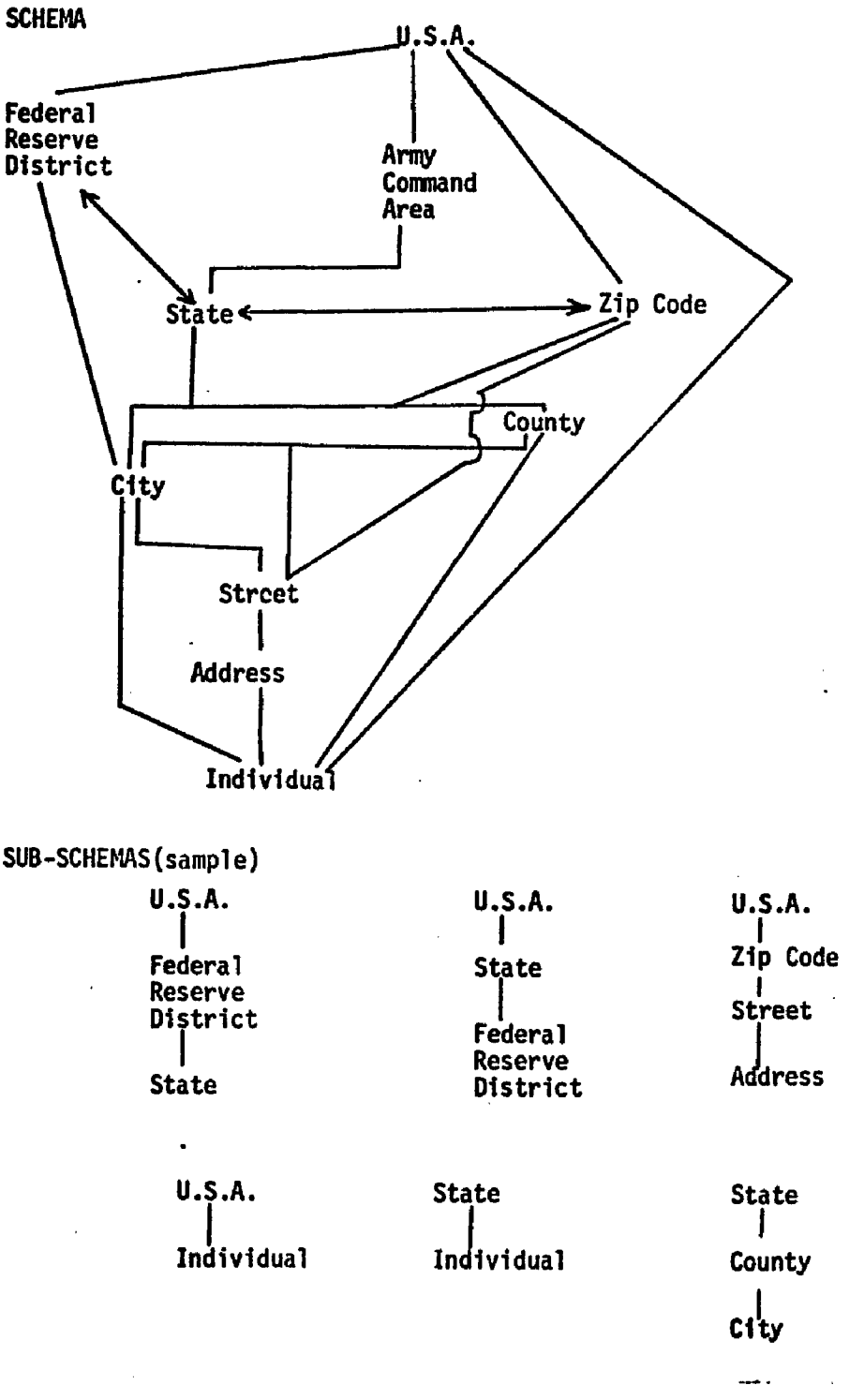
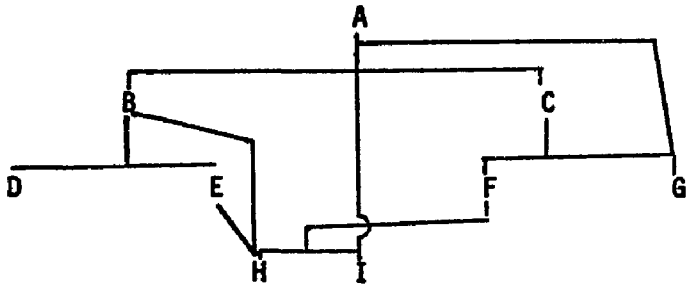


Figure 39. A Complex Network

ORIGINAL STRUCTURE



PRECEDENCE MATRIX

	A	B	C	D	E	F	G	H	I
A		1	1				1		
B				1	1			1	
C						1	1		
D									
E								1	
F								1	1
G									
H									
I									

R' MAX REACHABILITY MATRIX

	A	B	C	D	E	F	G	H	I
A		1	1	2	2	2	2	2	3
B				1	1	2	2	2	
C						1	1	2	2
D									
E								1	
F								1	1
G									
H									
I									

MINIMUM STRUCTURE

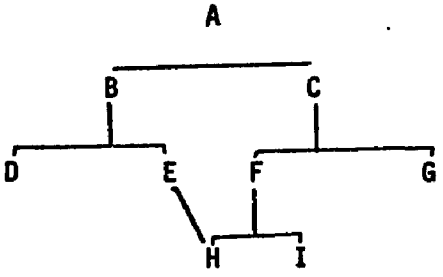


Figure 40. Reducing to Minimum Structure.

The cost of establishing (or re-establishing) an additional link (set ownership/membership) involves the pointers required to maintain this link and the associated storage and update costs. In general three such pointers are needed; a pointer to the owner record, one to the prior record and one to the next record. Swenson[37] shows this in Figure 41. The benefit associated with an additional link involves the speed and simplicity of access. This determination begins with the determination of the volume along any link.

Let V_{ij} be the volume (for a given period) of direct requests (query or program) for access to records in j within i--if i represents State and j represents City, requests for City within State.

Let V^*_{ij} be the total volume (for a given period) of requests (direct or otherwise) for access to records in j within i. Thus if $V_{ik} = 10$ and the path from i to k is via j, V^*_{ij} is incremented by 10.

A procedure for the search and evaluation follows:

Step 1. Form the minimum structure representation (per above).

Step 2. Determine Precedence and Reachability matrices.

Step 3. Gather V_{ij} from documentation in RSM.

Step 4. Determine which nodes have no precedence. ($P_{ji} = 0$ for all j indicates that i has no precedence.) These are "lead" nodes.

Step 5. For each such node i loop through all nodes j with $R_{ij} = 2$. Determine V^*_{ij} = SUM (all k such that $R_{jk} \neq 0$). $V_{ik} + V_{ij}$.

Step 6. Determine a threshold volume $\bar{V}_{ij}$ (see below).

Step 7. If V^*_{ij} is greater than $\bar{V}_{ij}$ add link ij.

Step 8. Repeat Step 5 until no more lead nodes exist.

Step 9. Remove all lead nodes from the structure and procede from Step 2 for all remaining structures.

This procedure evaluates the cost/benefit of each additional link. If the minimum structure is not a tree (i.e. ties have caused a network representation) the determination of V* will be modified to include only those following nodes which are more easily obtained via the current node. Figure 42 shows this situation.

The determination of $\bar{V}$, the threshold, is a hueristic which involves the difficulty of going from i to j (when link ij does not exist). A formulation which may be acceptable is one which takes in account the difficulties at each point along the path from i to j, incorporates a cost for added storage and allow a constant for "fine tuning" this method:

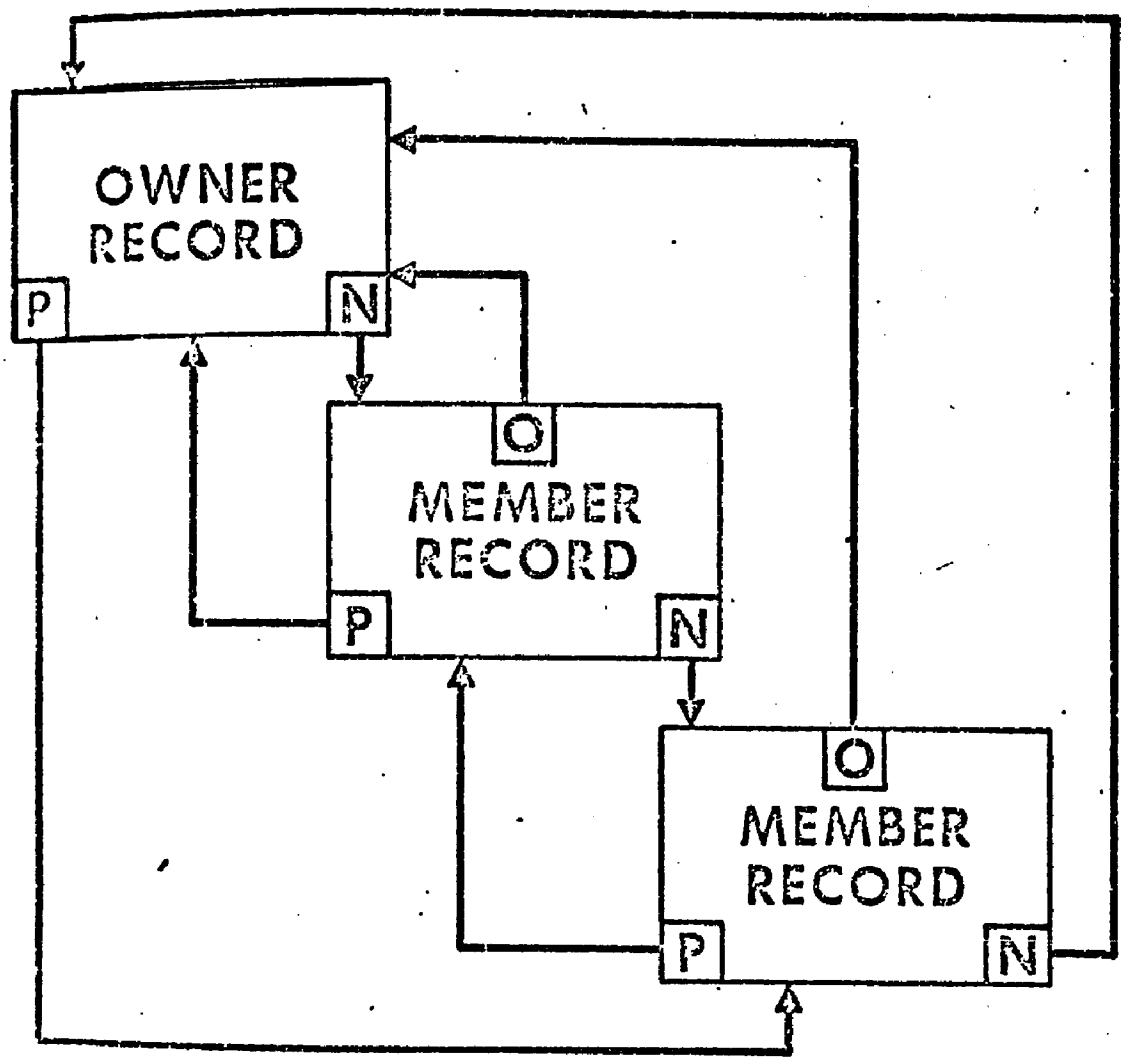
$$\bar{V}_{ij} = X/S \underset{k}{\text{SUM}} (D'_{ik} R_{ik} + D'_{kj} R_{kj})$$

X is a tuning constant.

S is a storage factor relating to the cost of pointers.

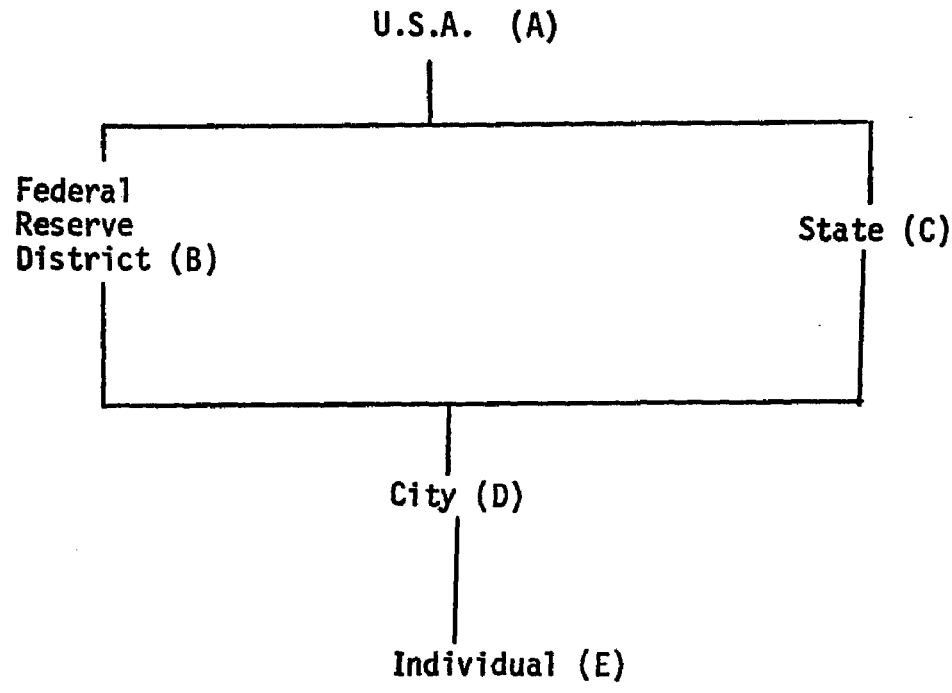
D'_{ik} is a difficulty factor for the access of k from i.

R_{kj} = 0 or 1 (since R_{ij} = 2)



N = NEXT POINTER
P = PRIOR POINTER
O = OWNER POINTER

Figure 41. Pointers [37]



given:

$V_{DE} = 10, \quad V_{AB} = 5, \quad V_{AC} = 4 \quad \text{and} \quad V_{AD} = 7.$

given: CD is an easier access path than BD.

$$V^*_{AC} = V_{AC} + V_{AD} + V_{AE}$$
$$= 5 + 4 + 0 = 9$$

$$V^*_{AB} = V_{AB} = 5$$

Figure 42. Determining V* in a Network.

CHAPTER V - USING RSM FOR DATA BASE DESIGN

Having used a "backwards" approach, developing the RSM to meet the information requirements for the systems design process, the true test of the RSM should be how well it serves the design process. The first area of evaluation concerns data quality, aptness of presentation and completeness. The RSM should yield an accurate picture of the requirements. The ability of the RSM to communicate to the system designer parallels the RSM's ability to provide feedback to the user (the system specifier). The PSA outputs are most significant here. The second area of interest is using the RSM in conjunction with SODA for making specific design decisions. This includes data base design, process design and grouping, and coding and implementation. Thus the RSM must be evaluated as both a documentation and design tool.

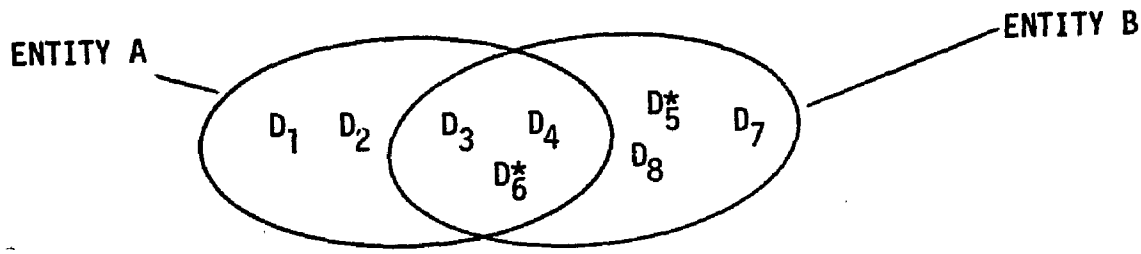
The usefulness of the RSM for communication and documentation is discussed in chapter two. The various picture reports, the formatted problem statement and the data dictionary are most useful. A complete discussion of the PSA outputs was previously cited [44,45]. The emphasis of RSM and PSA will focus on specifics, simply serving as a good communication media is not sufficient. Given a complete and consistent problem statement, the design process begins.

Record Design Overview

The first portion of design corresponds with Nunamaker's SODA approach[27,28]. The DID contains all of the information required for the precedence and incidence matrices. The reachability matrices follow as does the feasible process grouping matrix. A detailed explanation of this phase of the design process is provided by Nunamaker[28].

In applying the methods of the previous chapter to record design, the systems designer encounters a large combinatorial problem if he takes the problem statement and "decomposes" the information provided by the GROUP, ENTITY and SET constructs. Algorithms to solve this problem would automate the record design phase of systems design while beginning with all information at the DATA ITEM level--ignoring GROUP, ENTITY and SET. The RSM can help a hueristic designer make the corresponding transition from logical record design to physical system design--meeting user requirements but not allowing the user to make physical design decisions. An evaluation of the RSM helping in this transition follows.

The logical record design, primarily the specification of ENTITIES, frequently maps directly into physical record design. The problem definer has made decisions similar to that of the systems designer in choosing entities--similarities of useage, same identifier, etc. The viable candidates for hueristic record design are those entities with (1) identifiers in common and (2) DATA ITEMS in common (figure 43).



* identifiers

Figure 43. Intersecting Entities

In figure 43 ENTITY A contains DATA ITEMS 1,2,3,4 and 6; and is identified by 6. ENTITY B contains DATA ITEMS 3,4,5,6,7 and 8; and is identified by 5 and 6. Figure 44 shows some possible physical record designs which might be choosen.

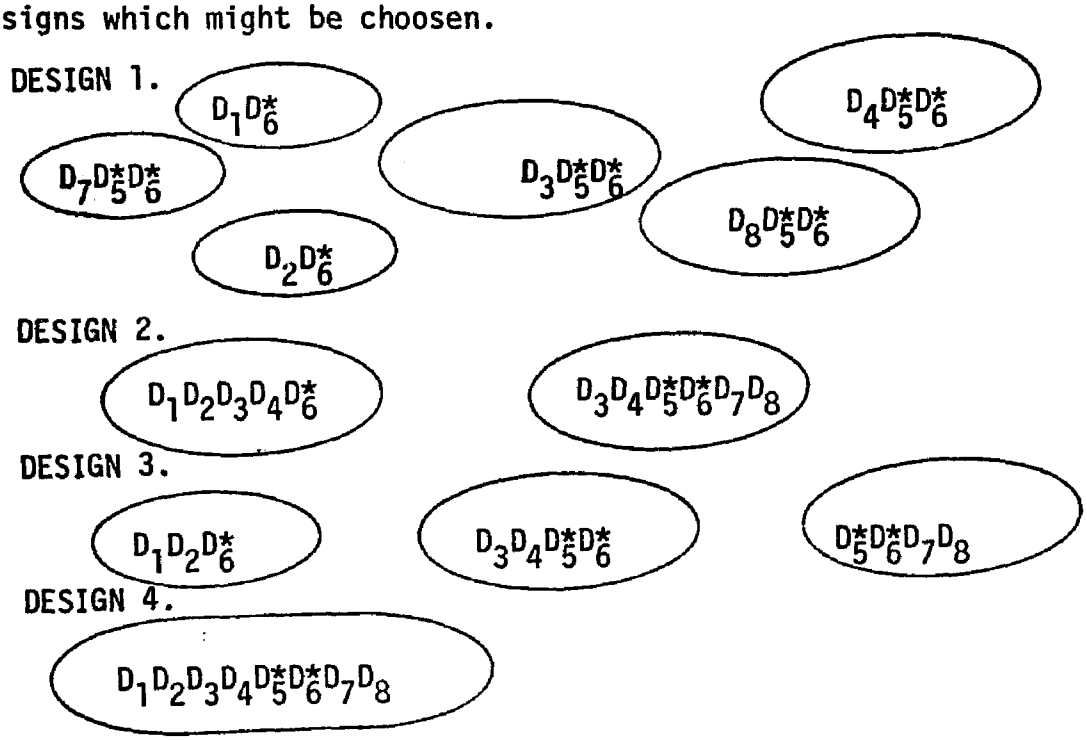


Figure 44. Record Design Alternatives

Design 1 has each data item isolated with its identifiers. This design would have minimum transport volume but maximum storeage requirements. Design 2 has transformed the logical specification directly into the physical record design. Design 3 isolates the inter-

section of the two user specified entities. Design 4 combines the two user specified entities into one record. Using data item/process incidence and process volume information, the transport volume associated with each of these four design alternatives can be computed.

The Company Z example used in chapter three and a PSL specification of a similar payroll system will be used for a working example. Figure 45 shows the PSA generated process pictures which give an overview of the processing being described. To aid in conceptualizing the system, the DATA ITEMS incident to each PROCESS can be added. The consists matrix report, figure 46, and the consists matrix, figure 47 provide a view of the data structure showing which ELEMENTS are part of which ENTITIES, GROUPS, INPUTS and OUTPUTS. Figure 48 shows this same information in list form. Figure 49 is a consists comparison report. This information gives the system designer a partial set of design alternatives. The current PSL does not provide sufficient identifier information as an integral part of the above report.

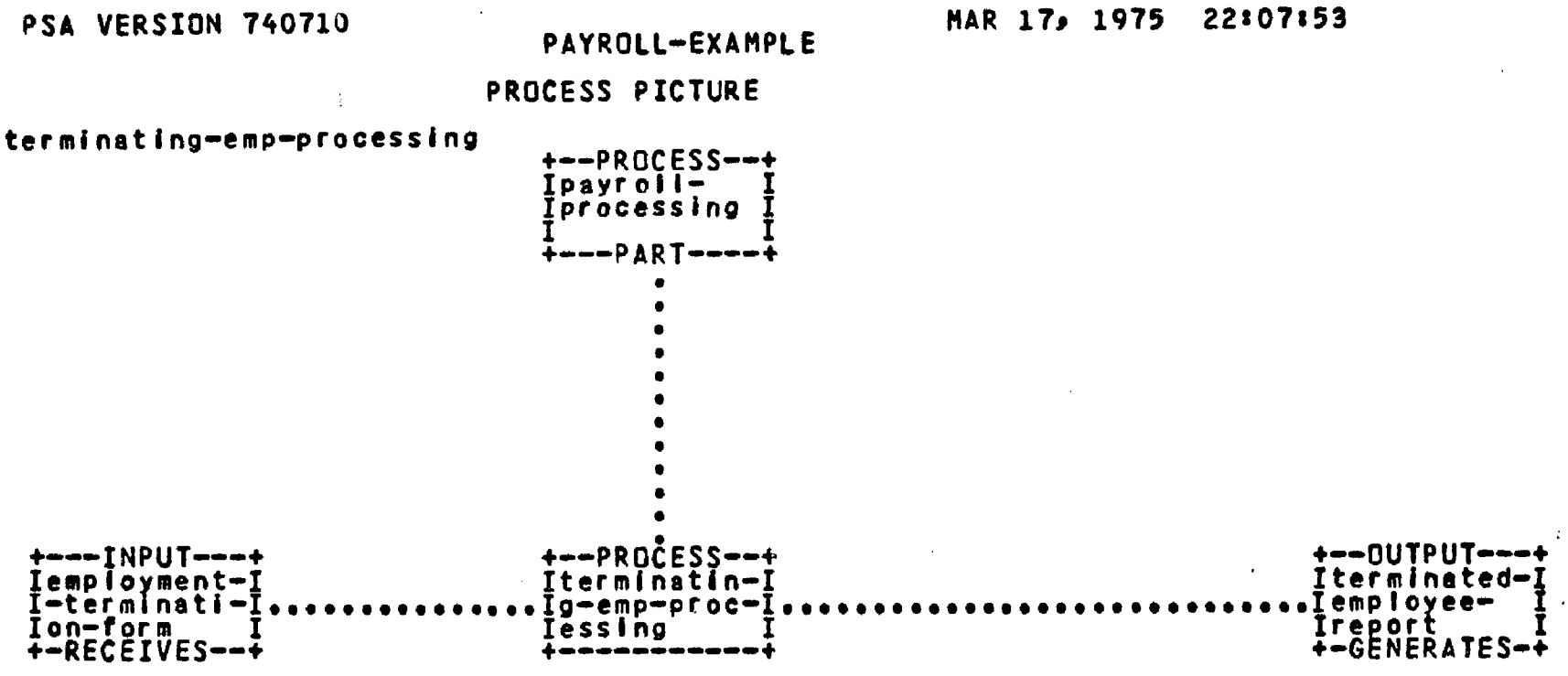


Figure 45. Payroll Process Picture

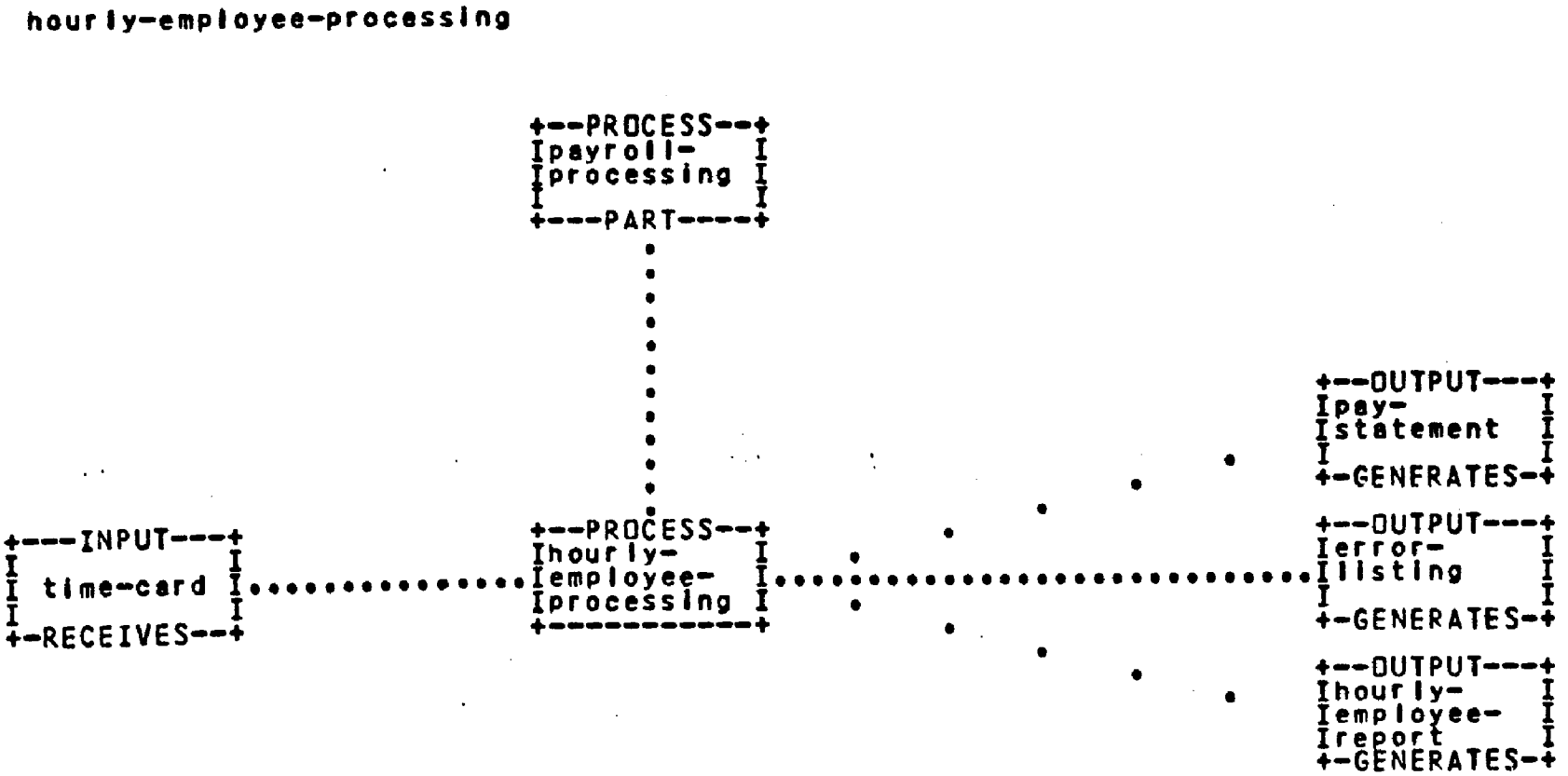


Figure 45, cont.

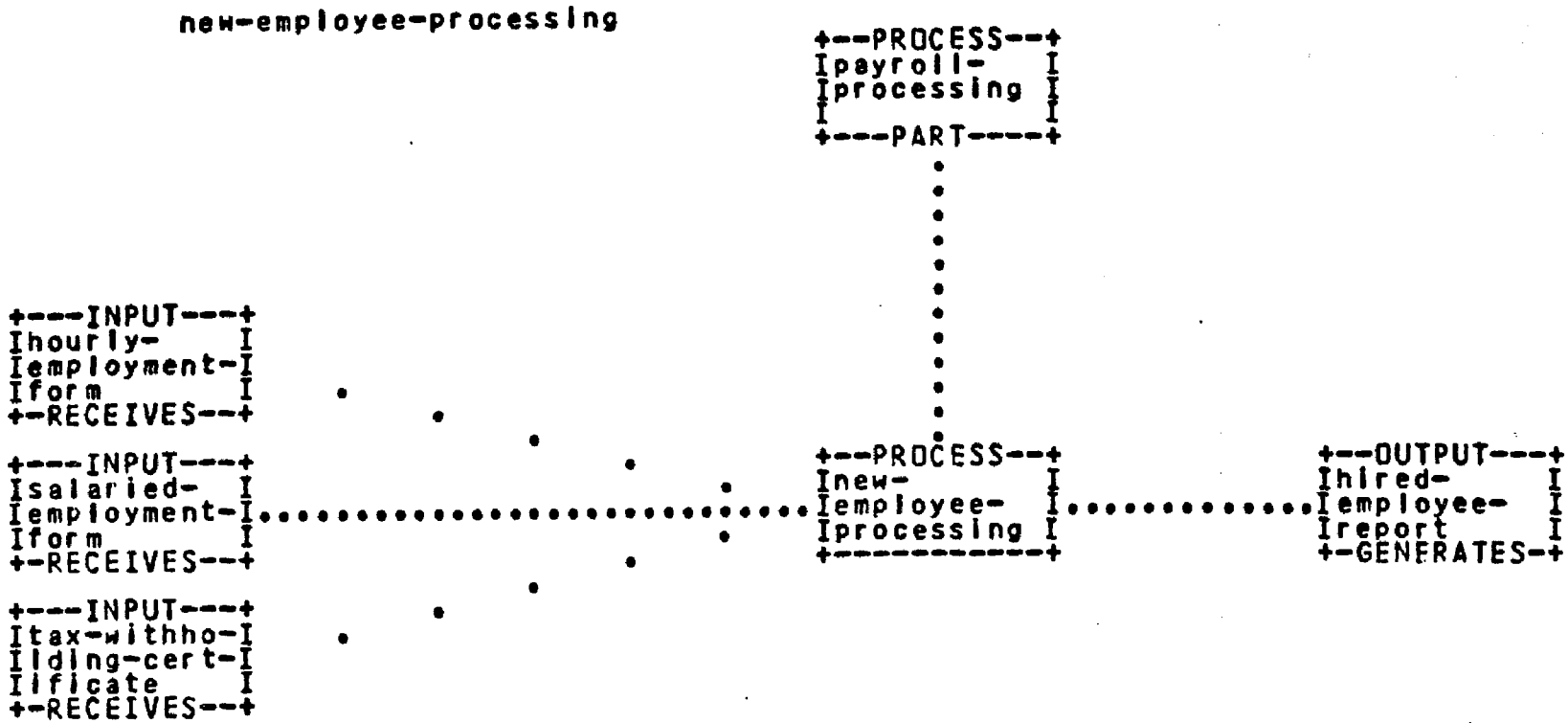


Figure 45, cont.

salaried-employee-processing

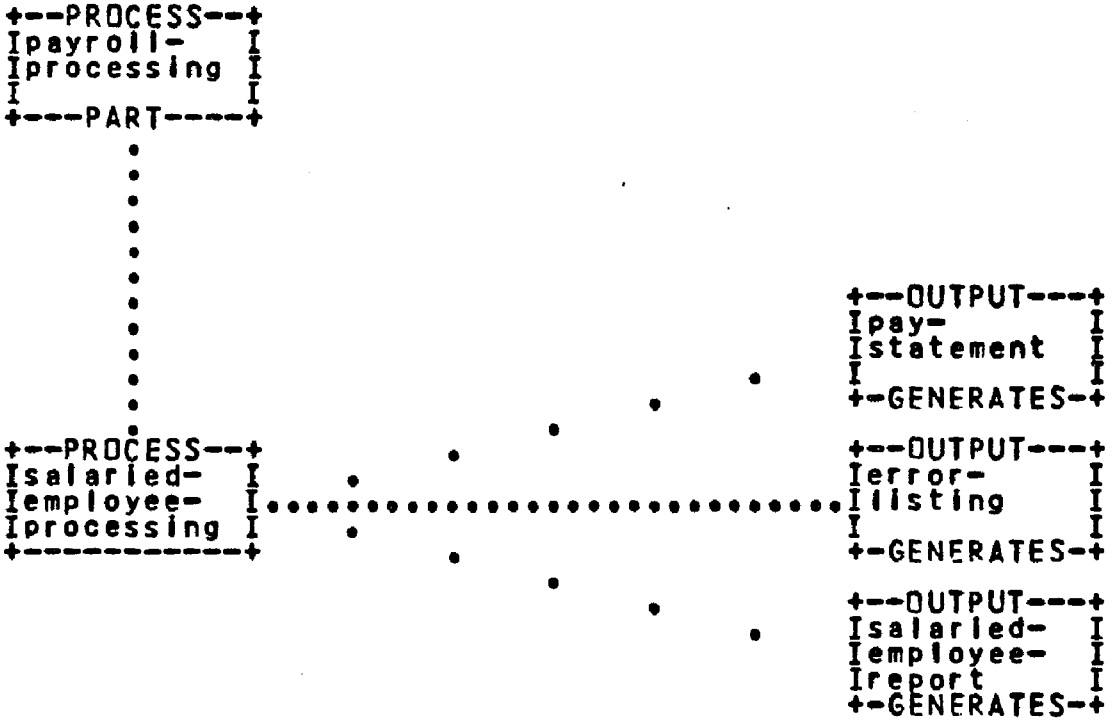


Figure 45, cont.

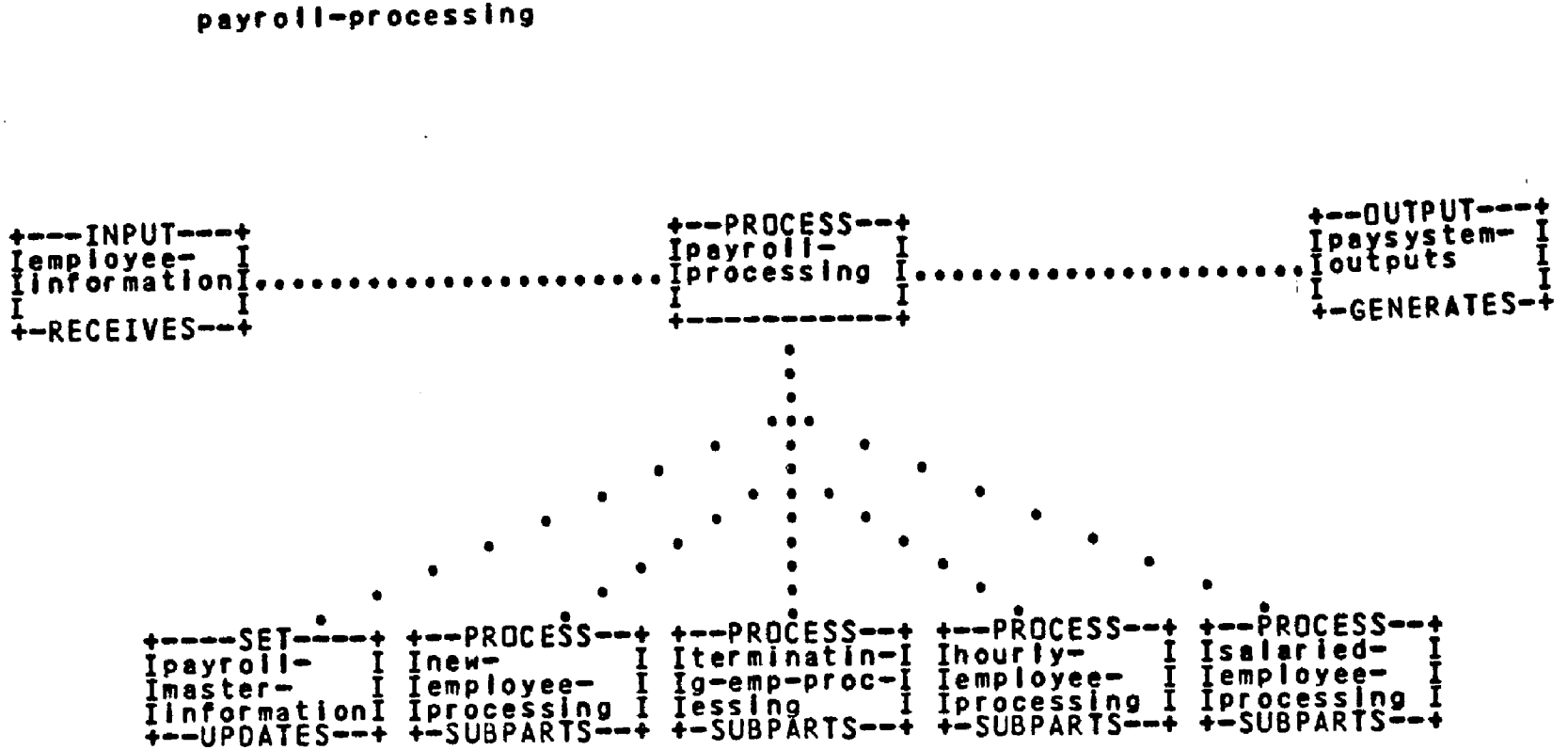


Figure 45, cont.

CONSISTS MATRIX REPORT

PARAMETERS FOR: CM

FILE CONTAINED

PSA311:CONROW: THE FOLLOWING ARE NOT CONTAINED IN ANYTHING:

last-department-change

ROW NAMES		COLUMN NAMES	
1 age	ELEMENT	1 hourly-employee-record	ENTITY
2 apartment-number	ELEMENT	2 salaried-employee-record	ENTITY
3 birthdate	ELEMENT	3 address	GROUP
4 check-number	ELEMENT	4 personal-data	GROUP
5 city	ELEMENT	5 check	GROUP
6 cumulative-federal-deductions	ELEMENT	6 pay-stub	GROUP
7 cumulative-fica-deductions	ELEMENT	7 term-report-entry	GROUP
8 cumulative-gross-pay	ELEMENT	8 hourly-job-data	GROUP
9 cumulative-hours	ELEMENT	9 tax-withholding-certificate	INPUT
10 cumulative-state-deductions	ELEMENT	10 salaried-job-data	GROUP
11 cumulative-tax-deductions	ELEMENT	11 department-record	ENTITY
12 current-date	ELEMENT	12 s-emp-report-entry	GROUP
13 department	ELEMENT	13 h-emp-report-entry	GROUP
14 employee-identification-number	ELEMENT	14 time-card	INPUT
15 employment-date	ELEMENT	15 hired-report-entry	GROUP
16 employment-status	ELEMENT	16 error-listing-entry	GROUP
17 error-code	ELEMENT	17 employment-termination-form	INPUT
18 federal-tax	ELEMENT	18 employee-name	GROUP
19 fica-tax	ELEMENT	19 pay-statement	OUTPUT
20 first-name	ELEMENT	20 error-listing	OUTPUT
21 gross-pay	ELEMENT	21 hourly-employee-report	OUTPUT
22 hours-per-day	ELEMENT	22 hired-employee-report	OUTPUT
23 house-number	ELEMENT	23 hourly-employment-form	INPUT
24 initial	ELEMENT	24 salaried-employment-form	INPUT
25 job-number	ELEMENT	25 salaried-employee-report	OUTPUT
26 job-title	ELEMENT	26 terminated-employee-report	OUTPUT
27 last-department-change	ELEMENT		
28 net-pay	ELEMENT		

Figure 46. Payroll Consists Matrix Report

	ROW NAMES	COLUMN NAMES
29	number-of-deductions	ELEMENT
30	number-of-employees	ELEMENT
31	overtime-hours-worked	ELEMENT
32	pay-date	ELEMENT
33	pay-grade-code	ELEMENT
34	pay-rate	ELEMENT
35	phone	ELEMENT
36	regular-hours-worked	ELEMENT
37	remaining-funds	ELEMENT
38	sex	ELEMENT
39	social-security-number	ELEMENT
40	state	ELEMENT
41	state-tax	ELEMENT
42	status-code	ELEMENT
43	street	ELEMENT
44	supervisor	ELEMENT
45	surname	ELEMENT
46	termination-date	ELEMENT
47	total-budget	ELEMENT
48	total-deductions	ELEMENT
49	total-hours	ELEMENT
50	zip-code	ELEMENT
51	address	GROUP
52	check	GROUP
53	employee-name	GROUP
54	error-listing-entry	GROUP
55	h-emp-report-entry	GROUP
56	hired-report-entry	GROUP
57	hourly-job-data	GROUP
58	pay-stub	GROUP
59	personal-data	GROUP
60	s-emp-report-entry	GROUP
61	salaried-job-data	GROUP
62	term-report-entry	GROUP

THE ROWS ARE CONTAINED IN THE COLUMNS WITH *S

Figure 46, cont.

CONSISTS MATRIX REPORT

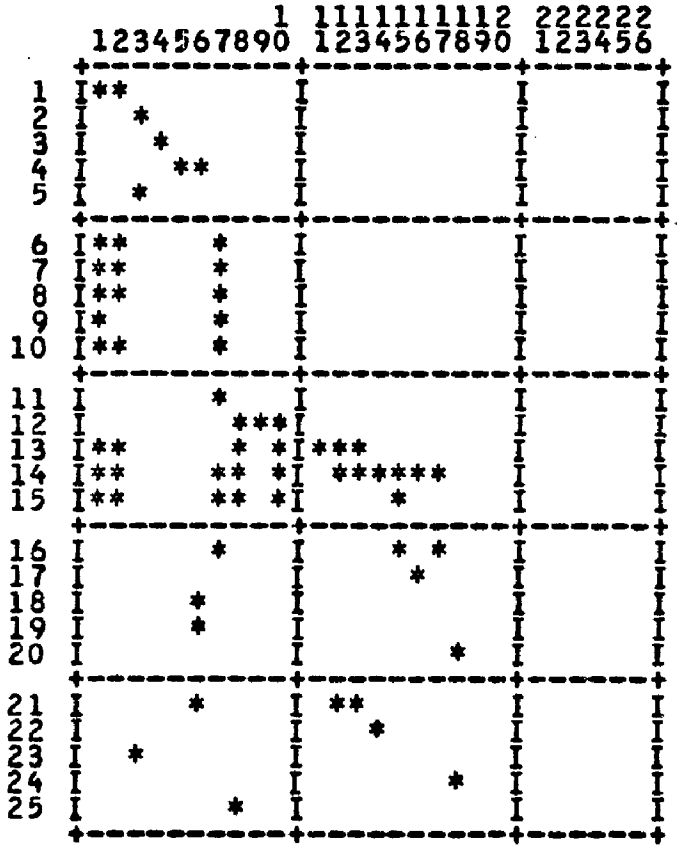


Figure 47. Payroll Consists Matrix

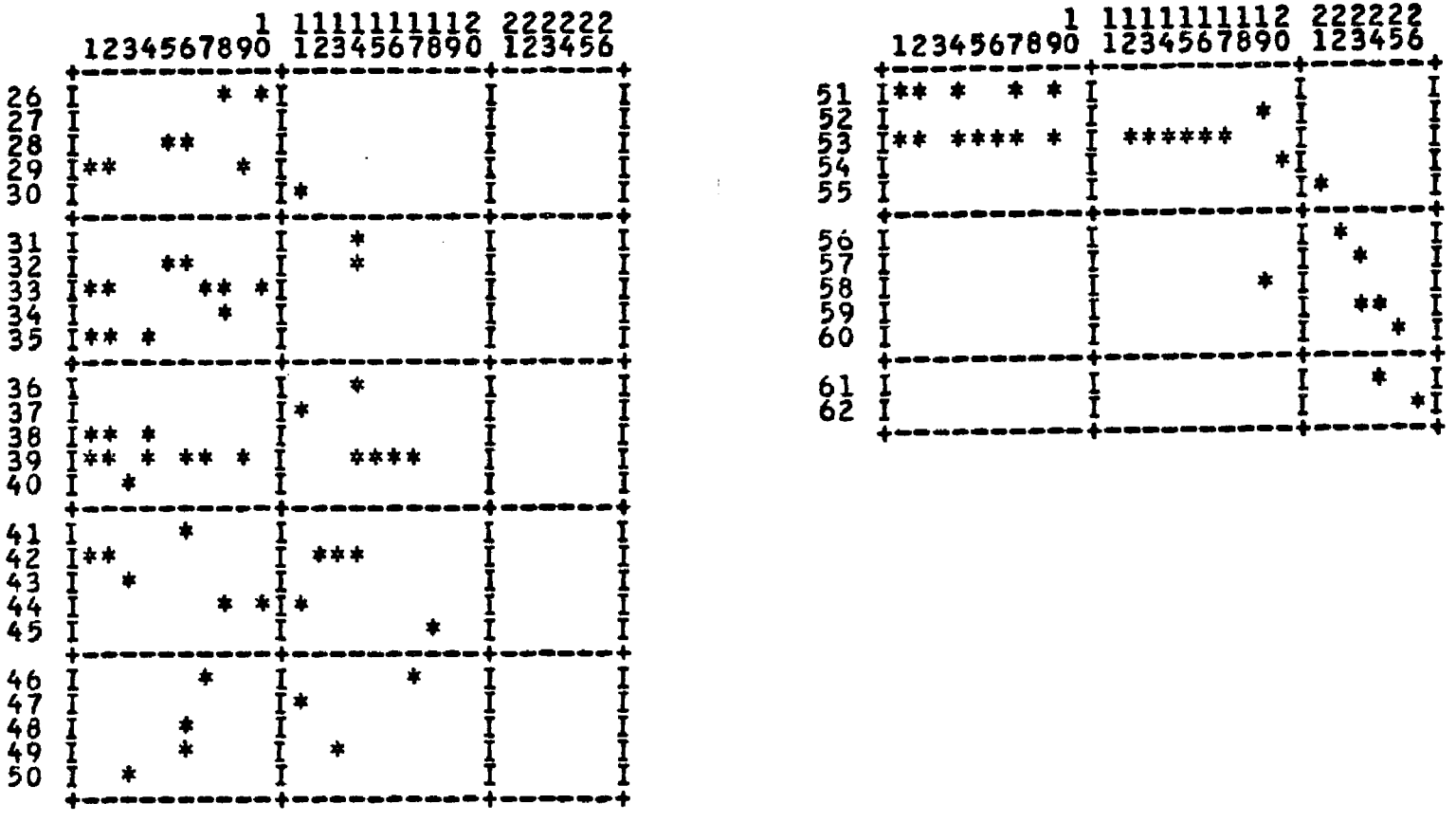


Figure 47, cont.

CONTENTS REPORT

```
1*    1 employee-information (INPUT)

2*    1 employment-termination-form (INPUT)
 1      2   employee-name (GROUP)
 2        3     surname (ELEMENT)
 3        3     initial (ELEMENT)
 4        3     first-name (ELEMENT)
 5      2   social-security-number (ELEMENT)
 6      2   termination-date (ELEMENT)
 7      2   employee-identification-number (ELEMENT)
 8      2   employment-status (ELEMENT)

3*    1 hourly-employment-form (INPUT)
 1      2   personal-data (GROUP)
 2        3     employee-name (GROUP)
 3          4       surname (ELEMENT)
 4          4       initial (ELEMENT)
 5          4       first-name (ELEMENT)
 6        3     social-security-number (ELEMENT)
 7        3     sex (ELEMENT)
 8        3     birthdate (ELEMENT)
 9        3     address (GROUP)
10          4       house-number (ELEMENT)
11          4       street (ELEMENT)
12          4       apartment-number (ELEMENT)
13          4       city (ELEMENT)
14          4       state (ELEMENT)
15          4       zip-code (ELEMENT)
16        3     phone (ELEMENT)
17      2   hourly-job-data (GROUP)
18        3     job-title (ELEMENT)
19        3     pay-rate (ELEMENT)
20        3     current-date (ELEMENT)
21        3     employment-date (ELEMENT)
22        3     job-number (ELEMENT)
23        3     pay-grade-code (ELEMENT)
24        3     supervisor (ELEMENT)
25        3     department (ELEMENT)
26        3     employee-identification-number (ELEMENT)

4*    1 salaried-employment-form (INPUT)
 1      2   personal-data (GROUP)
 2        3     employee-name (GROUP)
 3          4       surname (ELEMENT)
 4          4       initial (ELEMENT)
 5          4       first-name (ELEMENT)
 6        3     social-security-number (ELEMENT)
 7        3     sex (ELEMENT)
 8        3     birthdate (ELEMENT)
 9        3     address (GROUP)
10          4       house-number (ELEMENT)
11          4       street (ELEMENT)
12          4       apartment-number (ELEMENT)
13          4       city (ELEMENT)
14          4       state (ELEMENT)
15          4       zip-code (ELEMENT)
16        3     phone (ELEMENT)
```

Figure 48. Payroll Contents Report

```
17      2   salaried-job-data (GROUP)
18        3     job-title (ELEMENT)
19        3     pay-grade-code (ELEMENT)
20        3     current-date (ELEMENT)
21        3     employment-date (ELEMENT)
22        3     supervisor (ELEMENT)
23        3     department (ELEMENT)
24        3     employee-identification-number (ELEMENT)

5*    1 tax-withholding-certificate (INPUT)
 1      2   employee-name (GROUP)
 2        3     surname (ELEMENT)
 3        3     initial (ELEMENT)
 4        3     first-name (ELEMENT)
 5      2   address (GROUP)
 6        3     house-number (ELEMENT)
 7        3     street (ELEMENT)
 8        3     apartment-number (ELEMENT)
 9        3     city (ELEMENT)
10        3     state (ELEMENT)
11        3     zip-code (ELEMENT)
12      2   social-security-number (ELEMENT)
13      2   number-of-deductions (ELEMENT)
14      2   current-date (ELEMENT)

6*    1 time-card (INPUT)
 1      2   employee-name (GROUP)
 2        3     surname (ELEMENT)
 3        3     initial (ELEMENT)
 4        3     first-name (ELEMENT)
 5      2   social-security-number (ELEMENT)
 6      2   status-code (ELEMENT)
 7      2   pay-date (ELEMENT)
 8      2   regular-hours-worked (ELEMENT)
 9      2   overtime-hours-worked (ELEMENT)
10      2   hours-per-day (ELEMENT)
11      2   employee-identification-number (ELEMENT)

7*    1 error-listing (OUTPUT)
 1      2   error-listing-entry (GROUP)
 2        3     employee-name (GROUP)
 3          4       surname (ELEMENT)
 4          4       initial (ELEMENT)
 5          4       first-name (ELEMENT)
 6        3     employee-identification-number (ELEMENT)
 7        3     social-security-number (ELEMENT)
 8        3     error-code (ELEMENT)

8*    1 hired-employee-report (OUTPUT)
 1      2   hired-report-entry (GROUP)
 2        3     employee-identification-number (ELEMENT)
 3        3     employee-name (GROUP)
 4          4       surname (ELEMENT)
 5          4       initial (ELEMENT)
 6          4       first-name (ELEMENT)
 7        3     social-security-number (ELEMENT)
 8        3     employment-status (ELEMENT)
 9        3     employment-date (ELEMENT)
```

Figure 48, cont.

```
 9*    1 hourly-employee-report (OUTPUT)
 1       2   h-emp-report-entry (GROUP)
 2         3     employee-name (GROUP)
 3           4       surname (ELEMENT)
 4           4       initial (ELEMENT)
 5           4       first-name (ELEMENT)
 6         3     employee-identification-number (ELEMENT)
 7         3     department (ELEMENT)
 8         3     gross-pay (ELEMENT)
 9         3     status-code (ELEMENT)
10         3     total-hours (ELEMENT)

10*    1 pay-statement (OUTPUT)
 1       2   pay-stub (GROUP)
 2         3     employee-name (GROUP)
 3           4       surname (ELEMENT)
 4           4       initial (ELEMENT)
 5           4       first-name (ELEMENT)
 6         3     social-security-number (ELEMENT)
 7         3     pay-date (ELEMENT)
 8         3     check-number (ELEMENT)
 9         3     total-hours (ELEMENT)
10         3     gross-pay (ELEMENT)
11         3     total-deductions (ELEMENT)
12         3     net-pay (ELEMENT)
13         3     federal-tax (ELEMENT)
14         3     state-tax (ELEMENT)
15         3     fica-tax (ELEMENT)
16       2   check (GROUP)
17         3     employee-name (GROUP)
18           4       surname (ELEMENT)
19           4       initial (ELEMENT)
20           4       first-name (ELEMENT)
21         3     net-pay (ELEMENT)
22         3     check-number (ELEMENT)
23         3     pay-date (ELEMENT)

11*    1 paysystem-outputs (OUTPUT)

12*    1 salaried-employee-report (OUTPUT)
 1       2   s-emp-report-entry (GROUP)
 2         3     employee-name (GROUP)
 3           4       surname (ELEMENT)

 4           4       initial (ELEMENT)
 5           4       first-name (ELEMENT)
 6         3     employee-identification-number (ELEMENT)
 7         3     department (ELEMENT)
 8         3     gross-pay (ELEMENT)
 9         3     status-code (ELEMENT)
```

Figure 48, cont.

```
13*     1 terminated-employee-report (OUTPUT)
  1       2   term-report-entry (GROUP)
  2         3     employee-identification-number (ELEMENT)
  3         3     employee-name (GROUP)
  4           4       surname (ELEMENT)
  5           4       initial (ELEMENT)
  6           4       first-name (ELEMENT)
  7         3     social-security-number (ELEMENT)
  8         3     termination-date (ELEMENT)
  9         3     address (GROUP)
 10           4       house-number (ELEMENT)
 11           4       street (ELEMENT)
 12           4       apartment-number (ELEMENT)
 13           4       city (ELEMENT)
 14           4       state (ELEMENT)
 15           4       zip-code (ELEMENT)
 16         3     employment-date (ELEMENT)
 17         3     pay-grade-code (ELEMENT)
 18         3     cumulative-gross-pay (ELEMENT)
 19         3     cumulative-tax-deductions (ELEMENT)
 20         3     cumulative-fica-deductions (ELEMENT)
 21         3     cumulative-hours (ELEMENT)
 22         3     employment-status (ELEMENT)
 23         3     cumulative-state-deductions (ELEMENT)
 24         3     cumulative-federal-deductions (ELEMENT)
```

Figure 48, cont.

CONSISTS COMPARISON REPORT

BASIC CONTENTS MATRIX

THE ROWS ARE THE GIVEN INPUT NAMES.

THE COLUMNS ARE THE LOWEST LEVEL OBJECTS WHICH ARE CONTAINED IN THE ROWS, WITH INTERMEDIATE GROUPS IGNORED.

IF ANY COLUMNS ARE GROUP NAMES, THEN THE DEFINITION IS INCOMPLETE.

IF ANY COLUMNS ARE AMBIGUOUS NAMES, THEY ARE POSSIBLE ELEMENTS.

	ROW NAMES	
1	employment-termination-form	INPUT
2	hourly-employment-form	INPUT
3	salaried-employment-form	INPUT
4	tax-withholding-certificate	INPUT
5	time-card	INPUT
6	error-listing	OUTPUT
7	hired-employee-report	OUTPUT
8	hourly-employee-report	OUTPUT
9	pay-statement	OUTPUT
10	salaried-employee-report	OUTPUT
11	terminated-employee-report	OUTPUT

	COLUMN NAMES	
1	surname	ELEMENT
2	initial	ELEMENT
3	first-name	ELEMENT
4	social-security-number	ELEMENT
5	termination-date	ELEMENT
6	employee-identification-number	ELEMENT
7	employment-status	ELEMENT
8	sex	ELEMENT
9	birthdate	ELEMENT
10	house-number	ELEMENT
11	street	ELEMENT
12	apartment-number	ELEMENT
13	city	ELEMENT
14	state	ELEMENT
15	zip-code	ELEMENT
16	phone	ELEMENT
17	job-title	ELEMENT
18	pay-rate	ELEMENT
19	current-date	ELEMENT
20	employment-date	ELEMENT
21	job-number	ELEMENT
22	pay-grade-code	ELEMENT
23	supervisor	ELEMENT
24	department	ELEMENT

Figure 49. Payroll Consists Comparison Report

COLUMN NAMES

25	number-of-deductions	ELEMENT
26	status-code	ELEMENT
27	pay-date	ELEMENT
28	regular-hours-worked	ELEMENT
29	overtime-hours-worked	ELEMENT
30	hours-per-day	ELEMENT
31	error-code	ELEMENT
32	gross-pay	ELEMENT
33	total-hours	ELEMENT
34	check-number	ELEMENT
35	total-deductions	ELEMENT
36	net-pay	ELEMENT
37	federal-tax	ELEMENT
38	state-tax	ELEMENT
39	fica-tax	ELEMENT
40	cumulative-gross-pay	ELEMENT
41	cumulative-tax-deductions	ELEMENT
42	cumulative-fica-deductions	ELEMENT
43	cumulative-hours	ELEMENT
44	cumulative-state-deductions	ELEMENT
45	cumulative-federal-deductions	ELEMENT

Figure 49, cont.

BASIC CONTENTS MATRIX

AN * IN (I,J) MEANS THAT COLUMN J IS CONTAINED
DIRECTLY OR INDIRECTLY IN ROW I. THE COLUMNS
DO NOT CONSISTS OF ANYTHING FURTHER. INTERMEDIATE
GROUPS ARE IGNORED.

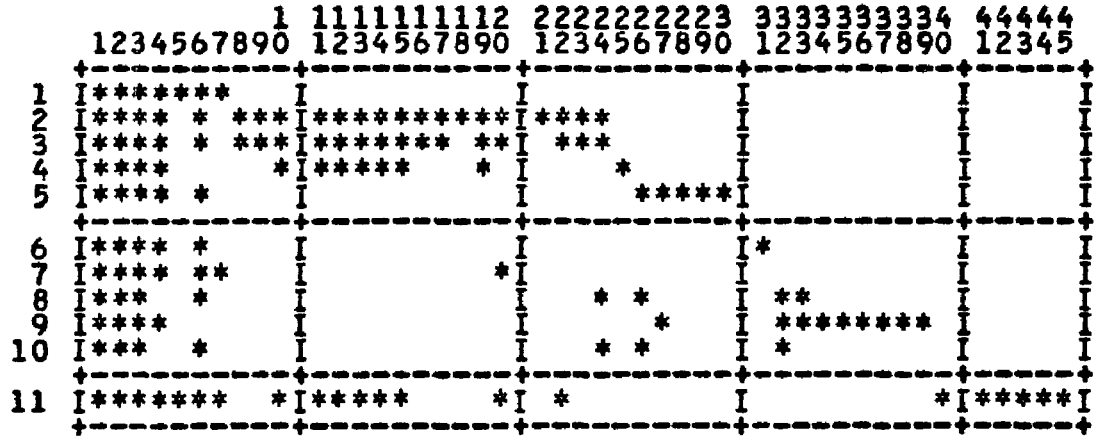


Figure 49, cont.

CONTENTS SIMILARITY MATRIX

THE NUMBER IN (I,I) IS THE NUMBER OF OBJECTS
AT THE LOWEST LEVEL CONTAINED IN ROW I FROM ABOVE.

THE NUMBER IN (I,J) (I NOT EQUAL J) IS
THE NUMBER OF OBJECTS AT THE LOWEST LEVEL IN
COMMON BETWEEN ROWS I AND J FROM ABOVE.

	1	2	3	4	5	6	7	8	9	10	11
1	7	5	5	4	5	5	6	4	4	4	7
2		22	20	11	5	5	6	5	4	5	13
3			20	11	5	5	6	5	4	5	13
4				12	4	4	4	3	4	3	10
5					10	5	5	5	5	5	5
6						6	5	4	4	4	5
7							7	4	4	4	7
8								8	5	7	4
9									13	4	4
10										7	4
11											21

CONTENTS SIMILARITY SUMMARY

ROW#	NAME		ROW#	NAME
1	employment-termination-form	IS A SUBSET OF	11	terminated-employee-report
3	salaried-employment-form	IS A SUBSET OF	2	hourly-employment-form
7	hired-employee-report	IS A SUBSET OF	11	terminated-employee-report
10	salaried-employee-report	IS A SUBSET OF	8	hourly-employee-report

Figure 49, cont.

Record Design Hueristic

The first task is to develop a heristic for record design based on the PSL, primarily using the consists matrix report (figure 47). The first portion of the procedure simplifies the matrix and searches for candidates for inclusion into records. This is an expansion of the method discussed earlier in this chapter.

Step 1. Remove all rows which correspond with GROUPS.

--In this example remove rows 51 through 62.

Step 2. Revise the consists matrix to retain the information lost by deleting the GROUPS in step 1. This yields a matrix with only DATA ELEMENTS in the rows. For each GROUP determine the row j and the column 1 which correspond with it. For any DATA ITEM represented by row i, the row/column incidence I_{ik} is expanded as follows:

$I_{ik} = 1$ if there exists j such that $I_{jk} = I_{il} = 1$.

NOTE: Since the only states of the matrix are 0 and 1, these states are denoted by a blank and an asterisk, respectively.

--In this example the DATA ITEM incidence with the INPUTS and OUTPUT is established. To see this compare the consists matrix (figure 47) with the revised consists matrix (figure 50).

Step 3. Note any columns which are identical and reorder the columns, grouping those which represent ENTITIES, GROUPS, INPUTS and OUTPUTS.

--In this example columns 6 and 19 are combined.

	Entities			Groups												Inputs					Outputs					
	1	2	11	3	4	5	6	7	8	10	12	13	15	16	18	9	14	17	23	24	19	20	21	22	25	26
1	x	x																	x	x						
3					x														x	x						
9	x							x																		x
12									x	x						x			x	x						
15								x	x	x			x						x	x				x		x
16							x						x								x			x		
21							x				x	x									x		x		x	
26									x	x									x	x						
28						x	x														x					
29	x	x														x										
32						x	x														x					
33								x	x	x									x	x						
39					x		x	x						x	x				x	x	x	x		x		x
42																										
44											x	x											x		x	
46								x										x								x
49							x					x									x		x			
TG's																										
1	x	x		x	x			x		x						x			x	x						x
2	x	x			x													x	x							
3	x	x						x																		x
4							x																			
5	x	x			x	x	x	x			x	x	x	x	x	x			x	x	x	x	x	x	x	x
6																	x									
7									x										x							
8			x																							
identifiers																										
4						x	x														x					
13			x						x	x	x	x							x	x			x		x	
14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 50. Revised Consists Matrix

Step 4. Locate possible identifiers and isolate these rows at the bottom of the matrix.

--The identifiers found are:

row 4, CHECK-NUMBER

row 13, DEPARTMENT

row 14, EMPLOYEE-IDENTIFICATION-NUMBER

row 39, SOCIAL-SECURITY-NUMBER

A design decision must be made as to whether EMPLOYEE-IDENTIFICATION-NUMBER or SOCIAL-SECURITY-NUMBER, or both, should be the identifier(s) for the various columns. This may cause a slight revision of the logical system specification and will have significant effect on the performance of the resulting system. The decision made is that EMPLOYEE-IDENTIFICATION-NUMBER be the identifier; SOCIAL-SECURITY-NUMBER will remain a DATA ITEM contained in various records.

Step 5. Find rows which are identical to other rows and combine these forming "temporary-groups" (TG's). If any of these TG's correspond with an existing columns (GROUP, ENTITY, etc.) be sure to note this. Name the TG's for clarity.

--The following TG's are formed:

TG 1. 2 5 23 40 43 50 [DATA ITEMS] Corresponds with ADDRESS GROUP.

TG 2. 35 38 To be called MISC-EMPLOYEE-DATA.

TG 3. 6 7 8 10 11 To be called CUMULATIVE-PAY-DATA.

TG 4. 18 19 41 48 To be called PAY-TG.

TG 5. 20 24 45 Corresponds with EMPLOYEE-NAME GROUP.

TG 6. 22 31 36 Corresponds with TIME-CARD GROUP.

TG 7. 25 34 To be called JOB-PAYRATE-TG.

TG 8. 30 37 47 Corresponds with DEPARTMENT-RECORD ENTITY.

Step 6. Form a revised consists matrix. Use the TG's instead of the individual DATA ITEMS. Give each DATA ITEM and TG the same column incidence as the GROUP(S) it is a member of, i.e. if row i belongs to column j and the corresponding row l in the original matrix belongs to column k, then row i now belongs to column k.

--Figure 50 is the resulting matrix after steps 1 through 6. Note: when column j (a GROUP) is a subset of column k (an INPUT, OUTPUT, ENTITY or GROUP) and DATA ITEM i is a member of this same GROUP j then DATA ITEM i is also contained in column k (per above).

Step 7. Partition the problem by identifiers and timing information forming an identifier incidence matrix.

--The matrix (Figure 51) shows that EMPLOYEE-IDENTIFICATION-NUMBER "dominates" this problem. In this example, little is gained by partitioning. In many examples timing information (weekly vs: annual reports, etc) may be most useful in partitioning a large record design problem. Historical data, usually in ENTITIES, is frequently seperated from more volatile data in this manner.

Step 8. Form a consists comparison matrix and determine which columns are subsets of others, also which are highly similar.

--Figure 52 is the matrix formed.

Because of the formulation of this example, groups which were represented by rows are subsets of INPUTS or OUTPUTS. The following is a listing of GROUPS followed by the INPUTS or OUTPUTS which they

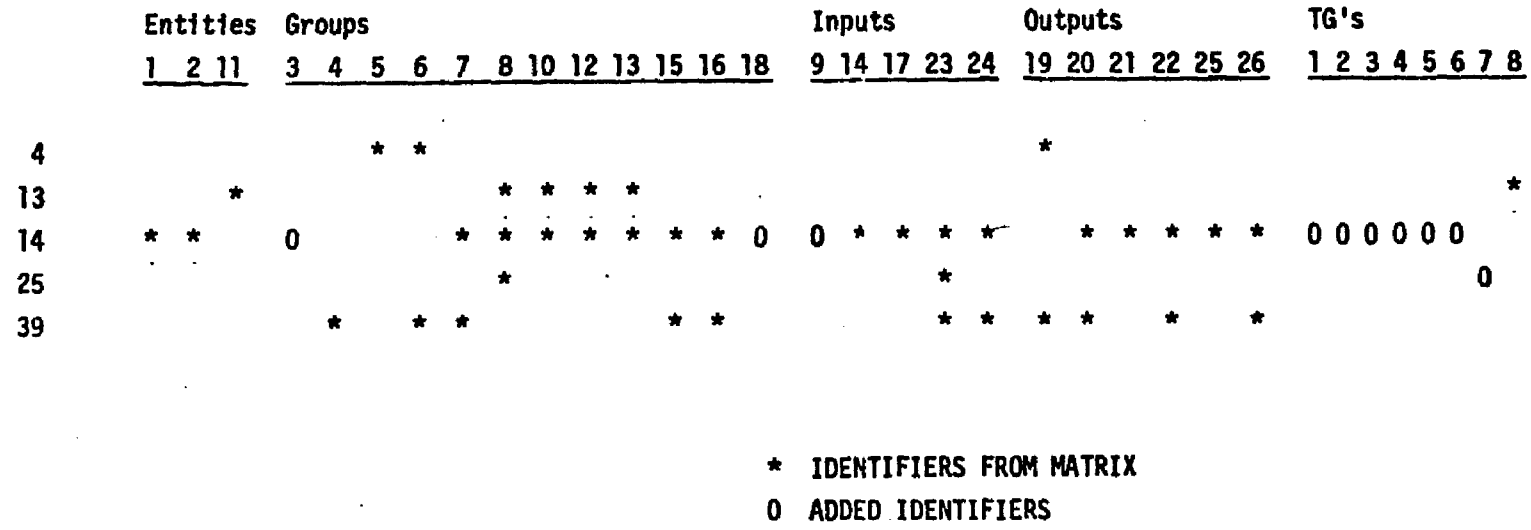


Figure 51. Identifier Incidence Matrix

	Entities			Groups												Inputs					Outputs					
	1	2	11	3	4	5	6	7	8	10	12	13	15	16	18	9	14	17	23	24	19	20	21	22	25	26
1	20	19	0	7	12	5	5	16	1	8	5	5	4	4	4	11	1	2	15	13	5	4	4	4	5	16
2		19	0	7	12	5	5	15	1	8	5	5	4	4	4	11	1	2	15	13	5	4	4	4	5	15
11			4	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0
3				7	7	1	1	7	1	7	1	1	1	1	1	7	1	1	7	7	1	1	1	1	1	7
4					14	4	6	11	1	4	4	4	4	5	4	10	1	1	14	14	5	5	4	5	4	11
5						7	7	4	4	1	4	4	4	4	4	4	1	1	4	4	7	4	4	4	4	4
6							15	5	1	1	5	6	6	5	4	4	1	1	5	5	15	5	6	6	5	5
7								20	3	9	4	4	6	5	4	5	1	2	8	8	5	5	4	6	4	19
8									8	6	2	2	2	1	1	2	1	1	8	6	1	1	2	2	2	2
10										12	2	2	2	1	1	8	1	1	12	12	1	1	2	2	2	8
12											7	7	4	4	4	4	1	1	5	5	5	4	7	4	7	4
13												8	4	4	4	4	1	1	5	5	6	4	8	4	7	4
15													7	5	4	4	1	1	6	6	6	5	4	7	4	6
16														5	4	4	1	1	5	5	5	5	4	5	4	5
18															4	4	1	1	4	4	4	4	4	4	4	4
9																12	1	1	11	11	4	4	4	4	4	10
14																	4	1	1	1	1	1	1	1	1	1
17																		2	1	1	1	1	1	1	1	2
23																			21	19	5	5	5	6	5	12
24																				19	5	5	5	6	5	12
19																					15	5	6	6	6	5
20																						5	4	5	4	5
21																							8	4	7	4
22																								7	4	6
25																									7	4
26																										19

Figure 52. Augmented Consists Comparison Matrix

are subsets of:

(GROUP)	(is a subset of...)
5 CHECK	19 PAY-STATEMENT (OUTPUT)
16 ERROR-LISTING-ENTRY	20 ERROR-LISTING (OUTPUT)
13 H-EMP-REPORT-ENTRY	21 HOURLY-EMPLOYEE-REPORT (OUTPUT)
15 HIRED-REPORT-NETRY	22 HIRED-EMPLOYEE-REPORT (OUTPUT)
8 HOURLY	23 HOURLY-EMPLOYEE-FORM (INPUT)
4 PERSONAL-DATA	23 HOURLY-EMPLOYEE-FORM (INPUT)
4 PERSONAL-DATA	24 SALARIED-EMPLOYMENT-FORM (INPUT)
12 S-EMP-REPORT-ENTRY	25 SALARIED-EMPLOYEE-REPORT (OUTPUT)
10 SALARIED-JOB-DATA	24 SALARIED-EMPLOYMENT-FORM (INPUT)
7 TERM-REPORT-ENTRY	26 TERMINATED-EMPLOYEE-REPORT (OUTPUT)
6 PAY-STUB	19 PAY-STATEMENT (OUTPUT)

The following information is also directly available from the matrix:

(Column)	(is a subset of ...)	(is highly similar to ...)
1	--	2
2	1	
3	1 2 4 7 10 9 23 24 26	
4	--	2 7 23 24 26
5	6 19	1 2
6	19 (equals)	
7	--	1 26
8	23	10 24
10	23 24	
12	13 21 25	1 2 6 7
13	21	6 12 25
15	22	19 23 24 26
16	4 7 15 17 23 19 20 21 26	6 7
18	1 2 4 5 6 12 13 15 16 9 23 24 19 20 21 22 25 26	
9		1 2 23 24 26
14	--	
17	1 2 7 26	
23	--	24
24	23	
19	6 (equals)	
20	4 6 7 15 16 23 24	
21	13	25 21
22	15 21	6 7 26 23 24 19
25	12 13 21	
26	7	1

Step 9. Using the results of Step 8 gather alternative record designs and evaluate them:

A. Consider the data incident to given processes.

B. Determine alternative record combinations.

C. Evaluate the alternatives.

--A. (example) PERSONAL-DATA (Column 4) is incident to various processes.

B. Alternative 1, column equals record.

DATA ITEMS 3, 39, 14 (Identifier, EMPLOYEE-IDENTIFICATION-NUMBER) plus Temporary Groups 1, 2 and 5.

Alternative 2, complete fragmentation with identifier (14) duplicated.

Record 1. TG 1 (= column 3) ADDRESS GROUP plus identifier

Record 2. TG 2 (MISC-EMPLOYEE-DATA) plus identifier

Record 3. TG 5 (= column 18) EMPLOYEE-NAME GROUP plus identifier.

Record 4. DATA ITEM 3 (BIRTHDATE) plus identifier

Record 5. DATA ITEM 39 (SOCIAL-SECURITY-NUMBER) plus identifier.

Alternative 3, a combination.

Record 1. TG 1 plus identifier

Record 2. TG 2 plus DATA ITEMS 3, 39 plus identifier

Record 3. TG 5 plus identifier

The five possible records shown in alternative 2 can be considered five items to be grouped, five, four, three, two and one item at a time, thus generating all possible alternatives. Record size restrictions may, however, reduce the number of alternatives.

C. Evaluating an alternative involves computing the transport volume, storeage costs and the maintenance costs for the given alternatives.

Clustering Methods and Record Design

The problem encountered in determining record design is one of size. The possible record design for a system containing N DATA ITEMS is equal to the number of groups, of any size 1 through N, which can be derived from these DATA ITEMS. This number of possible design alternatives quickly grows. For a system with only five DATA ITEMS the number of alternatives to be considered is over 60; for any IPS with as few as 50 DATA ITEMS, the number of alternatives is too large for enumerative methods.

The clustering methods proposed would begin with either (1) DATA ITEMS as individual items, (2) a partitioning of DATA ITEMS either by identifier or timing requirements per step 7 of the record design hueristic described in the previous section or (3) after step 9 of the hueristic with each item being, itself, a grouping which will be involatile.

The first problem with using a clustering method is defining the proper distance or stress measure. For record design the "wasted" or excess transport volume incurred when a DATA ITEM or GROUP is added to another GROUP is the appropriate measure. Unfortunately this measure

is not completely additive. The following derivation explains the distance measure used.

Consider items to be DATA ITEMS, D_i $i = 1, N$.
A "commonality" of the process incidence must be found. That is the number of processes which the DATA ITEMS are concurrently incident or not incident to. The incidence matrix (Figure 53) shows which items i are input ($I_{ij} = 1$) or output ($I_{ij} = -1$) from a given process j. The commonality matrix (Figure 54) shows how many times two items have a common incidence. C_{ij} is this number. C_{ii} is the row sum for item i in the incidence matrix. The commonality matrix is, of course, symmetrical, $C_{ij} = C_{ji}$. The asterisks indicate a commonality defined as negative infinity--the two items may not be grouped. This situation occurs when an item is input to a process from which the other item is output. The anti-commonality is defined as follows:

$$A_{ij} = \begin{cases} C_{ii} - C_{ij} & i \neq j, \quad C_{ij} \neq * \\ 0 & i = j \\ * \text{ (infinity)} & C_{ij} = * \end{cases}$$

Thus the anti-commonality, A_{ij}, gives the "waste" or added transport required when item i is grouped with item j (Figure 55).

Having developed a concept of commonality and anti-commonality, the distance measure or stress can be derived from the incidence matrix. This measure now determine how frequently item i would be "dead weight" in processes which item j is incident to and vice versa. The measure is then multiplied by the appropriate volumes and record sizes:

Item/Process	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	1	1	0	0	0	0	1	0
2	1	0	1	1	0	1	1	1	1	0	0
3	0	1	0	0	1	1	1	1	0	0	1
4	1	1	1	1	1	0	0	0	0	1	0
5	0	1	0	0	1	1	1	1	0	0	1
6	-1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	-1	0	0	0	0	0	0

Figure 53. Sample Incidence Matrix

Item/Item	1	2	3	4	5	6	7
1	6	3	2	6	2	*	*
2	3	7	3	3	3	*	*
3	2	3	6	2	6	*	*
4	6	3	2	6	2	*	*
5	2	3	6	2	6	*	*
6	*	*	*	*	*	1	0
7	*	*	*	*	*	0	1

Figure 54. Sample Commonality Matrix

Item/Item	1	2	3	4	5	6	7
1	0	4	4	0	4	*	*
2	3	0	4	3	4	*	*
3	4	4	0	4	0	*	*
4	0	4	4	0	4	*	*
5	4	4	0	4	0	*	*
6	*	*	*	*	*	0	1
7	*	*	*	*	*	1	0

Figure 55. Sample Anti-commonality Matrix

Let $C'_{ijk} = \begin{cases} 1 & \text{if } I_{ik} = 0 \quad \underline{\text{and}} \quad I_{jk} = 1 \\ 0 & \text{otherwise} \end{cases}$

Let V_k be the volume of PROCESS k.

Let L_i be the length (size in words) of item i.

Define Stress, $S_{ij} = \underset{k}{\text{SUM}} \; C'_{ijk} V_k L_i + \underset{k}{\text{SUM}} \; C'_{ijk} V_k L_j.$

The problem with this stress measure is that after any initial clustering, the new stress measure must be recomputed. If, for example, items 1 and 2 are combine, the stress between that grouping and item 3, i.e., the "waste" of adding item 3 into that group must be computed--it is not a function of S_{13} or S_{23}. It is necessary to recompute the C', commonality, between the group (items 1 and 2) and item 3. A computer program was developed to perform clustering via these methods. Figure 56 is a flowchart of that program. A listing of the program, together with outputs for the example depicted in figure 31 appears as appendix E. Had the stress measure been additive, a math programming solution to the record design problem would be possible.

The clustering method represents a significant automated approach to record design. Record design is, however, highly dependent on process grouping and on the resultant incidence matrix. The RSM might well include a feedback loop to use results from the record design stage to serve as input for re-evaluating the logical system specifications and proposed changes to them. This falls within the area of sensitivity analysis.

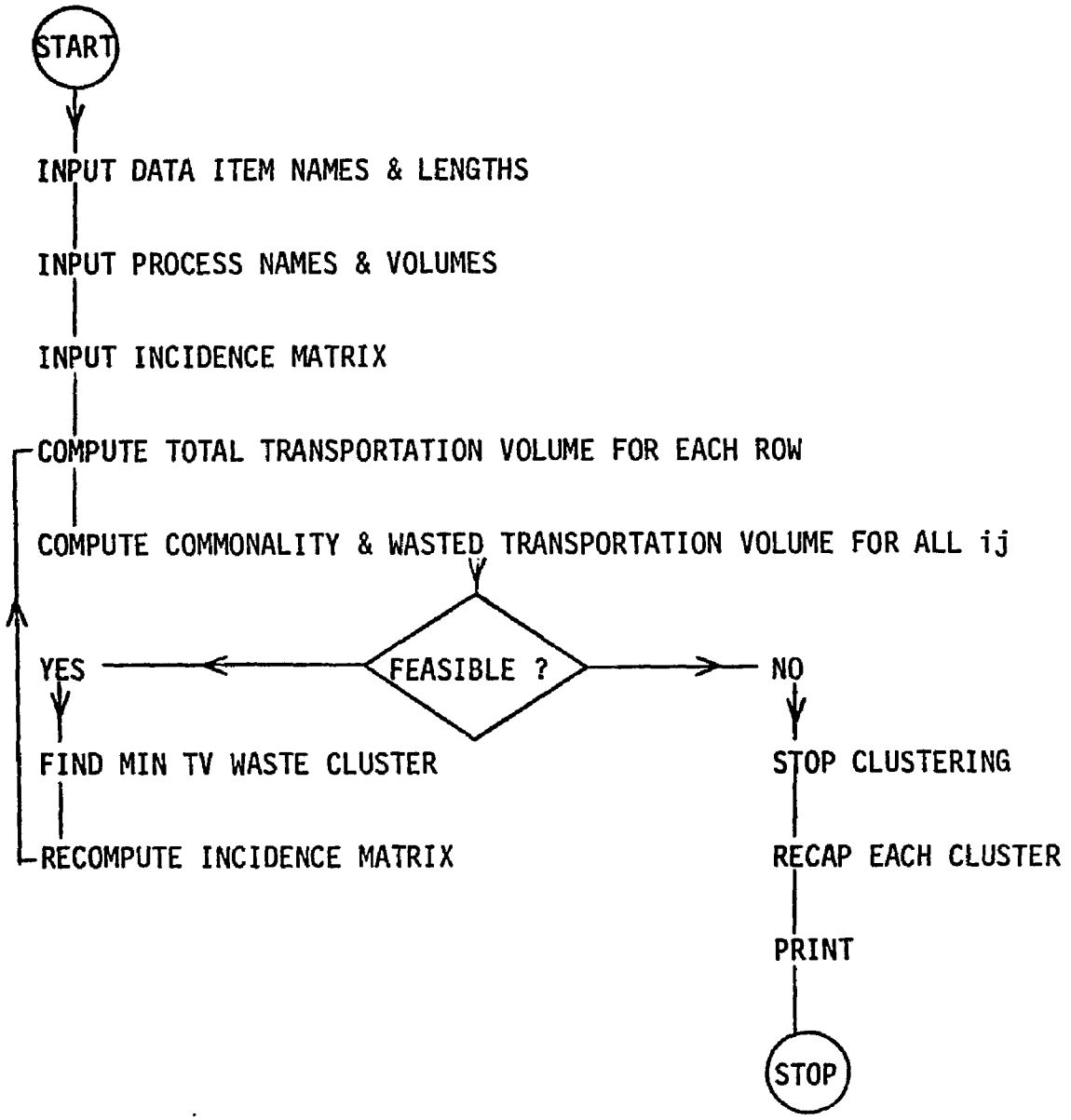


Figure 56. Record Design Program Flowchart

The Clustering Algorithm

The objective function of the clustering algorithm is to minimize the transport volume (TV). Therefore the first approach to grouping items (DATA ITEMS, GROUPS or clusters) is to do so in a manner which minimizes the wasted transport volume (WTV). The WTV, roughly, is a measure of the times data is input to a process but not used by it. Minimizing WTV may not be a straight forward as first appears. Consider the following rows from a data item/process incidence matrix:

PROCESS		1	2	3	4	5	6	7	8	9	
DATA ITEM/ (length)	1.	1	1	1	-1	1	0	0	1	1	(4)
	2.	1	1	0	-1	1	0	0	1	1	(5)
	3.	0	1	1	0	0	0	0	0	0	(4)
	4.	0	0	1	0	0	0	0	0	0	(7)
	5.	1	1	-1	-1	1	0	0	1	1	(3)
process volume		70	20	50	70	90	60	50	40	90	

Figure 57. Data Item/Process Incidence Matrix

The WTV for grouping DATA ITEMS 1 and 2 together is obtained by comparing their rows in the incidence matrix. The waste is caused by transporting DATA ITEM 2 into PROCESS 3 where it is not used:

$$WTV_{12} = L_2 \; V_3 = .5 \times 50 = 250$$

For a more complex example, the stress measure defined in the previous section could be used to derive the same result (there is only one C'_{ijk} term which is non-zero, that is $C'_{2,1,3}$). Similarly, the WTV for grouping items 3 and 4 equals 140. Using minimum WTV to determine which grouping should take place first we choose to form a cluster containing DATA ITEMS 3 and 4. This, even though DATA ITEMS 1 and 2 appear to have more similar process incidence.

This problem is significant because the order of the clustering does matter. In the above example DATA ITEMS 1 and 5 are not compatible because 1 is input to PROCESS 3 and 5 is output. Thus if DATA ITEMS 1 and 2 are clustered, then DATA ITEM 5 is not compatible with that cluster. If on the other hand DATA ITEMS 2 and 5 are clustered, then DATA ITEM 1 is not compatible. Thus the order of the clustering will determine the final configuration. Various alternate expressions of the objective function are possible. Two which have an intuitive appeal are:

$$\text{Minimize} \qquad WTV_{ij} \;/\; (TV_i + TV_j)$$

$$\text{Minimize} \qquad WTV_{ij} \;/\; TV_i$$

Both of these expressions adjust the WTV by a proportionality factor, in effect resulting in a minimum percent WTV. The second expression was also tested by the clustering program.

Figure 58 recaps a portion of the clustering using WTV as an objective function. Figure 59 recaps a similar portion of the clustering using the second expression (above) for making the clustering decision. (NOTE: Figure 58 results from a restatement of Figure E10 and Figure 59 from E12.) As expected the resulting clusters are different. For clarity those clusters with zero incidence (groups of DATA ITEMS with zero incidence) were ignored.

The end condition for the clustering algorithm is to stop when there are no more feasible (WTV finite) clusters remaining. This does not take in account the trade-offs involved with record design. The point at which additional wasted transport volume incurred by clustering is not worthwhile is, of course, hardware dependent.

Cluster Number 1.

1 2 3 4 11 22 32 39 40 44 49 50

1. Combine like Rows (WTV = 0) 1-2-50, 39-40

2. Iteration number 22, combine 22 & 32 (WTV = 400)

1-2-50 22-32 39-40 3 4 11 44 49

3. Iteration number 23, combine 3 & 4 (WTV = 400)

1-2-50 3-4 22-32 39-40 11 44 49

4. Iteration number 34, combine 11 & 39 (WTV = 1500)

1-2-50 3-4 11-39-40 22-32 44 49

5. Iteration number 35, combine 3 & 22 (WTV = 1600)

1-2-50 3-4-22-32 11-39-40 44 49

6. Iteration number 36, combine 44 & 49 (WTV = 2200)

1-2-50 3-4-22-32 11-39-40 44-49

7. Iteration number 42, combine 1 & 11 (WTV = 6000)

1-2-11-39-40-50 3-4-22-32 44-49

8. Iteration number 44, combine 3 & 44 (WTV = 8400)

1-2-11-39-40-50 3-4-22-32-44-49

9. Iteration number 48, combine 1 & 3 (WTV = 34600)

cluster contains all DATA ITEMS

Figure 58. Cluster Recap

Cluster Number 1.

1 2 3 4 5 6 7 10 11 14 16 22 29 32 36 39 40 42 50

1. Combine like Rows 1-2-50 10-36 39-40,
2. Iteration number 27, combine 1 & 32.
3. Iteration number 28, combine 1 & 22.
4. Iteration number 29, combine 1 & 4.
5. Iteration number 30, combine 1 & 3
6. Iteration number 31, combine 1 & 10.

1-2-3-4-10-22-32-36-50 39-40 7 11 14 16 29 42

7. Iteration number 32, combine 1 & 42.
8. Iteration number 33, combine 1 & 16.
9. Iteration number 34, combine 1 & 29.
10. Iteration number 35, combine 1 & 5.
11. Iteration number 36, combine 1 & 14.
12. Iteration number 37, combine 1 & 11.
13. Iteration number 38, combine 1 & 39.

1-2-3-4-5-10-11-14-16-29-32-36-39-40-42-50 7

14. Iteration number 39, combine 1 & 7.

Figure 59. Cluster Recap, Alternate

The above figure shows that a large cluster tends to "attract" additional DATA ITEMS because the large denominator results in a smaller value for the objective function. The percentage WTV also obscures the actual WTV involved with each iteration and makes it more difficult to develop a good stopping criterion.

This chapter has developed a hueristic and a clustering method for the record design problem. Coupled with the closed form solution for set design presented in the previous chapter, this solves the problem of data base design from logical system specifications. The RSM has been employed as the means of both communicating the specification information and (via PSA) for providing tools for the systems designer. The following chapter will outline certain extensions to the RSM which are expected in the future.

CHAPTER VI - EXTENSIONS

In formulating the Requirements Statement Methodology and applying it to data base design, the various problems solved have also opened the door to new questions which remain to be answered. The areas for extension include implementation, expansion and interfacing. The many concepts and features of the RSM need to be implemented, tested and evaluated. The scope of the DID needs to be expanded to include the special considerations such as a customized query language. Finally, the various PSA, RSM and data base design tools need to be interfaced with each other to provide a system which features feedback to the problem definer and sensitivity analysis.

Implementation

First on the list of features to be implemented is, of course, the interactive RSM. A full implementation may feature state of the art graphic display terminals and various other features to insure ease of use. The three forms presented as Figure 8, 9 and 10 are in need of additional software to help in their parsing and use. Similarly the use of generated forms (Figure 11 and 12) could use additional software. The current implementation (over 1000 FORTRAN statements) could be enhanced to the point of a commerical report generator. If various specialized terminals become part of the user specification system, they, too, could aid in this area. Certain of the matrices and

algorithms used for record and set design should be further automated and made easier to use. Monitoring functions such as those which apply to query oriented systems need to be implemented and extensive work is needed in optimizing these types of systems. The last area for possible RSM expansion is in the project control and documentation area. Software to keep track of the system being specified and designed is clearly needed for large projects.

Expansion

The primary areas for possible expansion are those additional features mentioned in the discussion of the DID. The determination of actual timing requirements along the real-time versus batch continiuum would be a major step in the progress of PSL. PSL and PSA need to be expanded to include various additional language constructs. The first construct would be an expansion of the ATTRIBUTES statement so that could be declared, given structure and then required as part of the PSL statement of requirements. For example, the ATTRIBUTE SECURITY-CLASSIFICATION could be declared, given a structure (say READ-ONLY-LEVEL, WRITE-ONLY-LEVEL, OVERRIDE, etc.) and all DATA ITEMS would then be required to have SECURITY-CLASSIFICATION data in their specification. Various features of the RSM should be come integral parts of the PSL and PSA. The data directory is one such feature.

The area of designing for existing systems has been overlooked in most efforts to date. The majority of systems being designed today are, in fact, replacing other systems. Such areas as decompilation from existing software to PSL documentation provide both theoretical and pragmatic challenges. This neatly leads into the area of performance

evaluation, where the RSM can compare the actual (implemented system with the desired system.

Interface

The area of interfacing the various models, languages and techniques may seem like a programming task, but it offers much more opportunity than that. By linking all the parts of the systems design methodology together under the umbrella of the RSM the consistency and completeness the many steps to design could be determined. The possibility of providing feedback to the specification process along with sensitivity analysis opens an untapped door to better, more efficient and cheaper systems. To date there is no perfected method for weighing the implications of any requirements in the logical system specification on the implemented system. By finally putting together all of the pieces of the design process into one compatible model, various system design alternatives could be tested at minimum cost. Any step in this process could be more readily evaluated and such areas as automatic code generation would have an ideal test environment. The ultimate goal for such a system would be to be able to simulate the design, implementation and operation of any given system (for a given PSL statement) and provide sensitivity analysis on various critical logical system specifications.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Kast, Fremont E.; Rosenzweig, Jame E.; ORGANIZATION AND MANAGEMENT, A SYSTEMS APPROACH, McGraw-Hill, New York, 1970.

2. Rosenblatt, S. Bernard; Bonnington, Robert L.; Needles, Belverd E., Jr.; MODERN BUSINESS, A SYSTEMS APPROACH, Houghton Mifflin Co., Boston, 1973.

3. Hitchcock, Robert; Wille, Edgar; THE COMPUTER AND BUSINESS UNITY, American Elsevier Publishing Co. Inc., New York, 1969.

4. Stuart, Humphrey; Yearsley, Ronald (editors); COMPUTERS FOR MANAGEMENT, william Heinemann, Ltd., London, 1969.

5. Blackburn, Thomas W.; White, H. Warren; UNDERSTANDING COMPUTERS, A GUIDE FOR MANAGEMENT, Clarkson N. Potter, Inc., New York, 1969.

6. Birkle, John; Yearsley, Ronald; COMPUTER APPLICATIONS IN MANAGEMENT, Brandon/Systems Press, Princeton, 1970.

7. Brandon, Dick H., MANAGEMENT PLANNING FOR DATA PROCESSING, Brandon/Systems Press, Princeton, 1970.

8. Teichroew, Daniel; Peters, ?; INTRODUCTION TO DATA PROCESSING, Unpublished text, class notes 1965.

9. Steiger, W. H., THE COMMUNICATIONS PROBLEM IN SYSTEMS BUILDING, ISDOS (Information System Design and Optimization System) Working Paper No. 2, Case Western Reserve University, September 1967.

10. Teichroew, Daniel, A SURVEY OF LANGUAGES FOR STATING REQUIREMENTS FOR COMPUTER-BASED INFORMATION SYSTEMS, FJCC, 1972.

11. Young, J. W.; Kent, H., ABSTRACT FORMULATION OF DATA PROCESSING PROBLEMS, Journal of Industrial Engineering, November-December 1958.

12. CODASYL DEVELOPMENT COMMITTEE, AN INFORMATION ALGEBRA PHASE I REPORT, Communications of the ACM, 5 4 April 1962.

13. Langefors, B., SOME APPROACHES TO THE THEORY OF INFORMATION SYSTEMS, BIT 3 1963.

14. Langefors, B., INFORMATION SYSTEMS DESIGN COMPUTATIONS USING GENERALIZED MATRIX ALGEBRA, BIT 5 1965.

15. Lombardi, L., THEORY OF FILES, Proc Easter Joint Computer Conference.

16. Lombardi, L., A GENERALIZED BUSINESS-ORIENTED LANGUAGE BASED ON DECISION EXPRESSIONS, Communications of the ACM Vol 7 No 2, February 1964.

17. National Cash Register Company, ACCURATELY DEFINED SYSTEMS, 1967.

18. Lynch, H. J., ADS: A TECHNIQUE FOR SYSTEMS DOCUMENTATION, Database Vol 1 No 1, Spring 1969.

19. Myers, D. H., A TIME AUTOMATED TECHNIQUE FOR THE DESIGN OF INFORMATION SYSTEMS, IBM Systems Research Institute, New York, 1962.

20. IBM, THE TIME AUTOMATED GRID SYSTEM (TAG): SALES AND SYSTEMS GUIDE, Publication No Y20-0358-1 (approx 1968).

21. Grindley, C.B.B., SYSTEMATICS - A NON-PROGRAMMING LANGUAGE FOR DESIGNING AND SPECIFYING COMMERCIAL SYSTEMS FOR COMPUTERS, Computer Journal Vol 9 August 1966.

22. Grindley, C.B.B.; Stevens, W.G.R.; PRINCIPLES OF THE IDENTIFICATION OF INFORMATION, File Organization IAG Occasional Publication No 3, Scolts and Zeitlinger N V, Amsterdam, 1969.

23. Grindley, C.B.B., SPECIFICATION AND INTERROGATION OF FORMAL CONTROL SYSTEMS, TIMS XVII Conference, London, July 1970.

24. Grindley, C.B.B., THE USE OF DECISION TABLES WITHIN SYSTEMATICS, Computer Journal Vol 11 No 2, August 1968.

25. Grindley, C.B.B., SYSTEMATICS FIELD TRIALS PROJECT, December 1968.

26. King, P.J.H., SOME COMMENTS ON SYSTEMATICS, Computer Journal Vol 10.

27. Nunamaker, J. F., Jr., ON THE DESIGN AND OPTIMIZATION OF INFORMATION PROCESSING SYSTEMS, PhD Dissertation, Case Western Reserve University, 1969.

28. Nunamaker, J. F., Jr., A METHODOLOGY FOR THE DESIGN AND OPTIMIZATION OF INFORMATION PROCESSING SYSTEMS, Krannert School of Business Paper No 301, Purdue University, March 1971.

29. Ho, Thomas; Nunamaker, J. F., Jr.; A SOFTWARE SYSTEM TO AID STATEMENT OF USER REQUIREMENTS, Krannert School of Business Paper No 401.

30. Ho, Thomas, TOWARD A FORMAL THEORY FOR THE REQUIREMENTS STATEMENT, ANALYSIS, AND DESIGN OF INFORMATION SYSTEMS, PhD Dissertation, Purdue University, December 1974.

31. Teichroew, Daniel, IMPROVEMENTS IN THE SYSTEM LIFE CYCLE, IFID, Stockholm, August 1974.

32. CODASYL Data Base Task Group, CODASYL DATA DESCRIPTION LANGUAGE, ACM, June 1973.

33. Bontempo, Charles J., DATA RESOURCE MANAGEMENT, Data Management, February 1973.

34. Wearing, Michael L., UPGRADE DOCUMENTATION WITH A DATA DICTIONARY, Computer Decisions, August 1973.

35. Uhrowczik, P. P., DATA DICTIONARY/DIRECTORY, IBM Systems Journal No 4 1973.

36. Nunamaker, J. F., Jr.; Ho, Thomas; Konsynski, Benn; Singer, Carl A., ANALYSIS AND DESIGN OF INFORMATION SYSTEMS, (to appear) Journal of the ACM.

37. Swenson, Donald, PERFORMANCE MEASURES FOR A DATA BASE MANAGEMENT SYSTEM, PhD Dissertation, Purdue University, 1974.

38. Stieger, W. H.; Teichroew, Daniel; A PSL - A PROBLEM STATEMENT LANGUAGE, PRELIMINARY USER'S MANUAL, ISDOS W.P. No 8 January 1968.

39. Teichroew, Daniel; Sibley, Edgar H.; Metrick, Lee B.; Sayani, Hasan; PSL - A PROBLEM STATEMENT LANGUAGE FOR INFORMATION PROCESSING SYSTEMS DESIGN, ISDOS Project, University of Michigan, 1969.

40. Koch, Ralph F., REVISIONS AND EXTENSIONS TO THE PROBLEM STATEMENT LANGUAGE, ISDOS W.P. No 30 January 1970.

41. Teichroew, Daniel; Hershey, Ernest A., III; Bastarache, Michel J.; AN INTRODUCTION TO PSL/PSA, ISDOS W.P. No 86 March 1974.

42. Hershey, Ernest A., III; Teichroew, Daniel; Berg, Douglas L. R.; Winters, Edward; Bastarache, Michel J.; PROBLEM STATEMENT LANGUAGE, VERSION 3.0, LANGUAGE REFERENCE MANUAL, ISDOS W.P. No 68 April 1974.

43. ____, Computer listing of examples and usage.

44. Berg, Douglas L. R.; Hershey, Ernest A., III; Bastarache, Michel J.; PROBLEM STATEMENT ANALYZER, COMMAND DESCRITPIONS, ISDOS No 91, 1974.

45. Bastarache, Michel J., PROBLEM STATEMENT ANALYZER, USERS MANUAL (MTS VERSION), ISDOS W.P. No 90 October 1974.

46. Teichroew, Daniel, PSL COMPLETENESS CHECKS, Unpublished notes, 1974.

47. Nunamaker, J.F., Jr.; Konsynski, Benn; COMPANY Z: AN INFORMATION DESIGN PROBLEM, W.P. University of Arizona, Tucson, Arizona January 1975.

48. Warshall, S., A THEORY OF BOOLEAN MATRICES, Journal of the ACM, Vol 9 No 1 January 1962.

APPENDICES

APPENDIX A

D. TEICHROEW: A REQUIREMENTS STATEMENT LANGUAGE

A REQUIREMENTS STATEMENT LANGUAGE[5]

Objectives of a useful requirements statement language

The discussion in the first two sections has established the need for a better way of stating information needs. The analysis in the previous section has shown that, while there have been attempts to develop such languages, they have not been successful in the sense that they are not in wide use today.

The need for such a language exists even more strongly today and therefore research, development, experimentation and evaluation are needed to develop a satisfactory medium for communicating requirements. A set of objectives for a Requirements Statement Language(RSL) is proposed in this section.

The language should accommodate the statement of requirements of the kind that are occurring now as well as those that will occur in the future. It is becoming more and more obvious that the cost of changing from one programming language to another is very high. Unfortunately, the present progression from COBOL, to COBOL with extensions, to Data Base Management Systems results in relatively small incremental improvements. The RSL should provide a quantum jump to a completely new generation of capabilities. The characteristics of the situation to be expected in the future that must be accommodated are:

i. Hardware features will increase in quality and reliability. There will be larger hardware with more parallel capabilities--this implies that unnecessary precedence constraints should be avoided whenever possible.

ii. Interrelationship of varying requirements will increase, e.g., jobs with varying priorities, inquiries to be answered, status data to be monitored, outputs required at predetermined times, data to be gathered and results to be distributed over geographically dispersed points, automatic monitoring and control, etc.

iii. The number and type of users with varying interface requirements will increase, e.g., online

interaction; data entry such as transaction recorder; interrogation, e.g., reservation clerk, users with no programming needed; system builders; analysts and programmers; data administrators; operators; etc.

iv. Systems will become larger and larger and they will become more integrated. This implies: common data bases, any given programmer does not know what else is going on, new functions such as data administrator, etc.

v. Requirements will be more unstructured; immediate response will be required and requirements will be changing rapidly; jobs require more consistency in data and business data function specifications. This implies that the "user" must be able to communicate with the computer system more directly.

vi. The performance of systems will become more important and hence there will be greater emphasis on more explicit recognition and statement of the criteria by which performance is measured and requirements parameters which affect performance.

vii. There will be more need to monitor the system in operation. The systems change over time either in the volume or the capabilities and consequently there must be provision for changing the internal structure of the system without affecting the correct achievement of the requirements.

The language should be suitable for use by humans in the necessary activity of determining and stating requirements.

i. The language or part of it must be usable by the manager or his assistants. This is necessary to eliminate the (computer) systems analyst as intermediary in order to reduce the chance for misunderstanding and to reduce the implementation time. To some, this specification implies that the language must be a subset of English. However, the fact that a subset of English is not English can severely limit the value of a subset of English as a requirements language. One of the objections sometimes raised against anything other than a natural language as a requirements language is that a manager will never take the time to use what to him is an unnatural language. It is unlikely that top managers will ever specify detailed requirements. The situation here will be analogous to the current situation in accounting. When a manager starts out in his career, he

is very familiar with the details of accounting and prepares statements for his immediate superior from the reports furnished by the accounting department. As he rises in the organization, he delegates more and more of this to his assistants but he still understands the accounting language and procedures. The career path of the person using the requirements language will be through the management ranks rather than the computer ranks.

ii. The language must be suitable for the top-down approach for problem definition. Most large systems are defined from the top down. The broad, overall outline is developed first and then successively more details are filled in. The language should permit this process and permit checking the problem statement for consistency and unambiguity at each level before proceeding to the succeeding lower levels. The language should, of course, not prohibit the bottom-up approach where this is appropriate.

iii. The language should be suitable for helping in the determination of requirements. It should augment the capabilities of the analysts or teams of analysts who are carrying out the requirements determination.

iv. The language should facilitate the testing and "exercising" of requirements. It is extremely important that statements of requirements be tested before they are implemented. Tests should be made for consistency and completeness. In addition, the person developing the requirements should be able to state data and test conditions that can be used to verify correctness of the requirements statement.

The language should be suitable for building the system to accomplish the requirements.

i. The language should permit the statement of requirements only and prevent the statement of data processing procedures. This is absolutely necessary in order to make the requirements statement hardware independent and to avoid reconversion costs when the capabilities of the equipment change. It is also necessary to prevent the introduction of restrictions which may limit the efficient use of hardware resources in the later stages of systems building.

ii. The requirements statement must be analyzable by computer programs. The problem statement should not only be readable by a computer program so that the requirements can be stored, but it should also be analyzable so that the problem can be restructured for optimum implementation efficiency without being limited by the sequence used by the problem definers. This is also necessary to permit the automatic construction of the system.

iii. The requirements statement language must permit statement of details necessary for the production of object code. This is necessary if the system is to be constructed automatically. In accordance with the above specifications, however, this detail should not have to be provided all at one time and as much as possible should be available from a library that is built up over time.

iv. The language should permit statements to facilitate the transition process. In most cases, systems already exist with files and programs and it is desirable to be able to move from the present system to the future system in an organized, controlled fashion to reduce inconvenience to the user and reduce cost.

v. The language should be as independent as possible of the particular area of application so that the cost of maintaining separate systems for a number of different applications is eliminated.

APPENDIX B

INSTRUCTIONS FOR "TEMPORARY" FORMS

DATA DICTIONARY (INPUT) FORM

The data dictionary in used to store all the relevant characteristics of a given data item (PSL ELEMENT). The forms serves as a convenient method of communicating much of this information to the data dictionary. Certain items contained in the data dictionary are generated (by a problem statement analyzer) from information which is contained elsewhere in the problem statement. Other items may be input directly via other forms. The layout form, for example, provides for input of the Picture, Validation Rules, etc. The user may input at more than one source, it will be checked for consistency. DATA ITEM NUMBER (DIN) is a unique 4 digit number which allows a convenient reference to a data item in lieu of the data item name.

USER INITIALS to help keep track of the documentation, this translates into RESPONSIBLE-PROBLEM-DEFINER in PSL.

DATA ITEM NAME (or ELEMENT name) is a unique and descriptive name for a data item. It may be up to 70 characters in length. Spaces are not allowed within the name but hyphens may be used to link words.

SYNONYMS must be unique and may be up to 70 characters in length. Synonyms are provided for user convenience and need not be used.

FORTRAN SYNONYM is a fortran name which is used in existing programs to identify a given data item. Like other synonyms, it is optional.

COBOL SYNONYM is similar to the Fortran synonym in use. It must be a "legal" COBOL name.

FORMAT/PICTURE may be specified in either Fortran or COBOL syntax.

TYPE I=integer R=real (decimal) D=Double Precision A=Alphanumeric.

JUSTIFICATION L=left justified R=right justified C=centered (the default is left justified for alphanumeric and right justified for numeric (integer, real or double precision).

VOLUME The total number of occurance of this item. For example, for EMPLOYEE-NAME, volume would be the number of employees. This can be expressed by numbers, or by reffering to another data item (such as NUMBER-OF-EMPLOYEES) which is equal to that number. This reference is made using the DIN. The "times" entry allows for a multiplication factor to be added. For example, for item DEPENDENT-NAME the "depends on" clause can be used to reference the DIN corresponding to NUMBER-OF-EMPLOYEES and "times" can be set to 3.1, meaning there are approximately 3.1 times dependents as employees. Alternatively, NUMBER-OF-DEPENDENTS could have been defined via a process (as equal to 3.1 times NUMBER-OF-EMPLOYEES).

RANGE A minimum and maximum allowable range is entered. 5 digits max.

VOLATILITY is a fraction with a number of time units. The number is entered in the first space, the time unit code (1=year 2=quarter 3=month 4=week 5=weekday 6=day 7=hour 8=minute 9=second) follows. For example, "3.0 6" means 3 days. i.e., the data item lasts for 3 days.

VALIDITY RULES for input an output are entered as appropriate. A check mark indicates that such rules exist.

SECURITY CATEGORY a four digit security category may be assigned to each data item. There is no technique, at present, for establishing and using these categories.

DATA SET INFORMATION refers to the set (data structure) relations among the data. For example, EMPLOYEE-NAME belongs to EMPLOYEE, percent occurance is 100. DEPENDENT belongs to EMPLOYEE, percent occurance is 90 (i.e., 80% of employees have dependents). Conversely, EMPLOYEE contains DEPENDENT, EMPLOYEE-ADDRESS, EMPLOYEE-NAME, etc. The user need only define either the "belongs to" (contained-in) or "contains" (consists-of) clause, the complementary statement is provided by the PSA.

NARRATIVE may be used as desired to clarify the description.

INPUT/OUTPUT (LAYOUT) FORM

This is the primary form for identifying data when it is either input to or output by the system. The form will be used in conjunction with a graphic layout form which will allow the "picture" of the input or output to be drawn. Conceptually, a different form will be used for graphic layout, display of a card, a report, a crt display, etc. The data items contained on each of these media is identified by location (a 3-vector coordinate) on the graphic form. All other data requested on the layout form is optional. The general idea is to allow the user to fill out whatever bits of information are available at the time. If picture or format is readily available at the time of filling out the layout form then the user has the opportunity to enter that information; if, on the other hand, a validation rule is not yet established or not readily apparent, it can be filled out at some other time and referenced via the data dictionary.

The form type, medium and frequency constitute the heading. Detailed instruction on how these are to be filled out can appear on the form itself or on separate documentation. The form will have room for 40 (approximate) lines of data. Date is a group of three 2-digit number--day, month, year.

FORM NUMBER is 3 digits precede by an "L". The page number allows for continuation of information unto other forms. Page number is two digits.

TITLE is 41 characters maximum and is for convenience in identify the form.

FORM TYPE input, output, both

MEDIUM card, tape, disc, printer, crt (video display)

FORM FREQUENCY five methods are available to communicate form frequency:

1. X times per Y, where X is a 4 digit number and Y is a 1 digit code corresponding to given time intervals.

2. X times per Y, where Y represents the day(s) of the week (1=sunday, 2=monday, etc.)

3. X times per Y, where Y represents day(s) of the month.

example: "1 times per 01 10 20" means something happens on the first, tenth and twentieth day of the month.

4. If a DIN has been defined appropriately, form frequency can refer to this data item times a constant (default = 1).

5. If desired, the process definition form may be used to express a logical relation which defines form frequency.

Every item on the graphic layout form is identified by a number which corresponds to the line number. It's location is identified by the three vector of three digit number. Example: 1-15-45 means page=1, line=15 and column=45. The medium determines what the vector stands for. The above example was for printer; for card it might mean card deck number = 1, card number = 15 and card column = 45.

DIN assigned to the data item is entered (without the D prefix).

THE FOLLOWING ITEMS ARE ALL OPTIONAL

PICTURE or FORMAT as on the Data Dictionary (Input) Form.

VALIDATION RULE enter "V" then:

if number: P if number must be positive

N if number must be negative

X if number must not equal zero

"blank" if only constraint is that it be a valid number.

if alphanumeric: "blank" may not exceed size expressed by format.

= must equal size in format

RANGE enter "R" then two 5 digit number, min and max

PROCESS enter "P" then number of process definition form which defined valid data item.

PERCENT OCCURANCE if data does not occur on all forms of this type, enter percent occurance (default is 100%).

PROCESS DEFINITION FORM

The process definition form is a combination process description form and decision table form (suitable for both complex logic and validation rules). The form is rather free format, but the analyzer printout of this form will be in more conventional decision table form (when used as such). The tabs (computations and conditions) will be in a rather free (Fortran-like) format.

EXAMPLES OF VALID TABS:

A=B+C+ (4*D*LOG C) a computation-type statement which defines a process resulting in A.

(A + B) ≠ (C*D) a conditional tab which is part of an "if" section for a decision table

Column 9 identifes the type of tab involved:

I "if" a conditional tab

T "then" a computation-type statement which defines a process dependent on a condition.

A "always" a computation-type statement which defines a process which always occurs (i.e. is not tied to a conditional tab)

C "continuation" continue the line above

N "narrative" or comment

F "footnote" narrative for bottom of page

V "valid" to identify "THEN VALID" condition

X "invalid" to identify "THEN INVALID" condition

TABS are entered in columns 10-60

GRID Columns 61-80 have the decision table-like grids

T or Y for true (with conditional tabs)

F or N for false

* or X to tie computation-type tabs to conditional tabs.

APPENDIX C

PSL SYNTAX AND COMPLETENESS CHECKS

PSL SYNTAX

SECTION A. REAL-WORLD-ENTITY (RWE)

(Structure Statements)

1. PART(OF) RWE (one and only one)
2. SUBPARTS(ARE) RWE's

A tree-structure is formed.

(Document Flow Statements)

3. GENERATES INPUTS (to the IPS)
4. RECEIVES OUTPUTS (from the IPS)

(Data Structure Statements)

5. RESPONSIBLE(FOR) SETS

Gives sets which are part of this REAL-WORLD-ENTITY.

DISCUSSION: RWE's are the parts of an organization. They receive and generate documents and are responsible for given groupings (SETS) of data.

SECTION B. OUTPUT

(Structure Statements)

1. PART(OF) OUTPUT (one and only one)
2. SUBPARTS(ARE) OUTPUTS

(Document Flow Statements)

3. RECEIVED(BY) RWE's

Data Structure Statements)

4. CONSISTS(OF) GROUPS and/or ELEMENTS

5. CONTAINED(IN) SETS

(PROCESS/DATA LINKAGE)

6. DERIVED(BY) PROCESS
   USING SETS, INPUTS, ENTITIES, GROUPS and/or ELEMENTS

7. GENERATED(BY) PROCESS (only one)

Derived is used for process which derive values for the output; Generated is for a single process which generates the output.

(Timing and Conditional Statements)

8. HAPPENS (system-parameter)
   TIME-PER INTERVAL name.

SECTION C. INPUT

(Structure Statements)

1. PART(OF) INPUT (one and only one)

2. SUBPARTS(ARE) INPUT

(Document Flow Statements)

3. GENERATED(BY) RWE

(Data Structure Statements)

4. CONSISTS(OF) GROUPS and/or ELEMENTS

5. CONTAINED(IN) SETS

(PROCESS/DATA LINKAGE)

6. RECEIVED(BY) PROCESS (one and only one)

(Timing and Conditional Statements)

7. HAPPENS (system-parameter)
   TIMES-PER INTERVAL name

SECTION D. ELEMENT

(Data Structure Statements)

1.	CONTAINED(IN)	GROUPS, ENTITIES, INPUTS and/or OUTPUTS

(PROCESS/DATA LINKAGE)

2.	DERIVED(BY) USING	PROCESS SETS, INPUTS, ENTITIES, GROUPS and/or ELEMENTS
3.	UPDATED(BY) USING	PROCESSES INPUTS, SETS, ENTITIES, GROUPS or ELEMENTS
4.	USED(BY) (TO)DERIVE	PROCESS SETS, ENTITIES, GROUPS or ELEMENTS

(OTHER Statements)

5.	SUBSETTING-CRITERION(FOR)	SET
6.	ASSOCIATED(WITH)	RELATION name

(Descriptive Statements)

7.	VALUES(ARE)	(min) THRU (max)
8.	IDENTIFIES	ENTITIES

SECTION E. GROUP

(Data Structure Statements)

1.	CONSISTS(OF)	GROUPS and/or ELEMENTS
2.	CONTAINED(IN)	ENTITIES, INPUTS, OUTPUTS or GROUPS

(PROCESS/DATA LINKAGE)

3.	DERIVED(BY) USING	PROCESSES SETS, INPUTS, ENTITIES, GROUPS or ELEMENTS
4.	UPDATED(BY) USING	PROCESSES INPUTS, SETS, ENTITIES, GROUPS or ELEMENTS
5.	USED(BY) (TO)DERIVE/ UPDATE	PROCESS SETS, ENTITIES, GROUPS or ELEMENTS

(OTHER Statements)

6.	SUBSETTING-CRITERION(FOR)	SET
7.	ASSOCIATED(WITH)	RELATION name

(Descriptive Statements)

8.	IDENTIFIES	ENTITIES

SECTION F. SET

(Structure Statements)

1.	SUBSET(OF)	SETS
2.	SUBSETS(ARE)	SETS

(Data Structure Statements)

3.	CONSISTS(OF)	INPUTS, OUTPUTS and ENTITIES
4.	RESPONSIBLE-REAL-WORLD-ENTITY(IS)	RWE

(PROCESS/DATA LINKAGE)

5.	DERIVED(BY) USING	PROCESSES SETS, INPUTS, ENTITIES, GROUPS or ELEMENTS
6.	UPDATED(BY) USING	PROCESSES SETS, INPUTS, ENTITIES, GROUPS or ELEMENTS
7.	USED(BY) (TO)DERIVE/UPDATE	PROCESSES SETS, ENTITIES, GROUPS or ELEMENTS

(OTHER Statements)

8.	SUBSETTING-CRITERIA(ARE)	SUBSETTING-CRITERION, ELEMENT or GROUP

(Timing and Conditional Statements)

9.	VOLATILITY-MEMBER	(comment-entry)
10.	VOLATILITY-SET	(comment-entry)

(Descriptive Statements)

11.	DERIVATION	(comment-entry)
12.	CARDINALITY(IS)	(system-parameter)

SECTION G. ENTITY

(Data Structure Statements)

1.	CONSISTS(OF)	GROUPS and/or ELEMENTS
2.	CONTAINED(IN)	

(PROCESS/DATA LINKAGE)

3.	DERIVED(BY) USING	PROCESSES SETS, INPUTS, ENTITIES, GROUPS or ELEMENTS
4.	UPDATED(BY) USING	PROCESSES SETS, INPUTS, ENTITIES, GROUPS or ELEMENTS
5.	USED(BY) (TO)DERIVE/ UPDATE	PROCESSES SETS, ENTITIES, GROUPS or ELEMENTS

(OTHER Statements)

6.	RELATED(TO) VIA	ENTITY RELATION name

(Timing and Conditional Statements)

7.	VOLATILITY	(comment entry)

(Descriptive Statements)

8.	CARDINALITY(IS)	(system-parameter)
9.	IDENTIFIED(BY)	GROUP or ELEMENT

SECTION H. PROCESS

(Structure Statements)

1.	PART(OF)	PROCESS (one and only one)
2.	SUBPARTS(ARE)	PROCESSES

(PROCESS/DATA LINKAGE)

3.	DERIVES	SETS, OUTPUTS, ELEMENTS, ENTITIES or GROUPs
	USING	ELEMENT, GROUP, INPUT, ENTITY or SETS
4.	GENERATES	OUTPUTS
5.	RECEIVES	INPUTS
6.	UPDATES	ENTITIES, SETS, GROUPS or ELEMENTS
	USING	INPUTS, SETS, ENTITIES, GROUPS or ELEMENTS
7.	USES	SETS, GROUPS, ELEMENTS, INPUTS or ENTITIES
	(TO)DERIVE/ UPDATE	SETS, ENTITIES, GROUPS or ELEMENTS
8.	UTILIZED(BY)	PROCESSES
9.	UTILIZES	PROCESSES

(OTHER Statements)

10.	MAINTAINS	RELATION names or SUBSETTING-CRITERIA

(Timing and Conditional Statements)

11.	HAPPENS TIMES-PER	(system-parameter) INTERVAL name
12.	INCEPTION-CAUSES	EVENTS
13.	TERMINATION-CAUSES	EVENTS
14.	TRIGGERED(BY)	EVENTS

(Descriptive Statements)

15.	PROCEDURE	(comment-entry)

SECTION I. CONDITION

(TRUE/FALSE) WHILE (comment-entry) BECOMING (TRUE/FALSE) IS

CALLED EVENT name.

This allows a descriptive statement to define a condition which is defined as an EVENT.

SECTION J. EVENT

(Timing and Conditional Statements)

1.	HAPPENS TIMES-PER	(system-parameter INTERVAL name
2.	(ON)INCEPTION(OF)	PROCESSES
3.	(ON)TERMINATION (OF)	PROCESSES
4.	TRIGGERS	PROCESSES
5.	WHEN BECOMES	CONDITION name (TRUE/FALSE)

SECTION K. INTERVAL

INTERVAL (name) CONSISTS(OF) (system-parameter)
INTERVAL name

SECTION L. RELATION

1.	ASSOCIATED- DATA(IS)	ELEMENT or GROUP
2.	BETWEEN and	ENTITY ENTITY
3.	CARDINALITY(IS)	(system-parameter)
4.	CONNECTIVITY(IS)	(system-parameter) TO (system-parameter. i.e. many to one, etc.
5.	DERIVATION(IS)	(comment-entry)
6.	MAINTAINED(BY)	PROCESSES

SECTION M. PROBLEM-DEFINER

1.	MAILBOX(IS)	MAILBOX name
2.	RESPONSIBLE(FOR)	any name

SECTION N. DEFINE

1.	APPLIES(TO)	KEYWORD, SECURITY, SOURCE or MAILBOX
2.	MAINTAINED(BY)	PROCESS
3.	SUBSETTING-CRITERION(FOR)	SETS
4.	VALUES(ARE) (min) THRU (max) etc.	

The Defines statement allows information to be added outside the section in which it would normally appear.

SECTION O. DESIGNATE

DESIGNATE (name) AS A SYNONYM FOR (name)

SECTION P. MEMO

1.	APPLIES(TO)	any name except another MEMO

In addition to the PSL sections, there are certain statements which apply to nearly all sections:

1. ATTRIBUTES(ARE). Record additional characteristics of the data such as type, length, frequency, etc. The attribute is flexible in that it allows a list of additional characteristics to be supplied. It is restrictive in that it is limited to list structure. Future implementation of PSL and PSA may allow more complex attributes statements; or may "hard-wire" (i.e. establish statements) certain characteristics into the PSL.
2. DESCRIPTION (comment entry). This allows narrative to be entered to state information which cannot be stated easily within the syntax of the given PSL section.

3. KEYWORDS(ARE). Keywords are established to link the areas of interest. Thus all sections dealing with payroll can be assigned the keyword payroll. The PSA allows retrieval on the keyword (see discussion of PSA below).
4. RESPONSIBLE-PROBLEM-DEFINER assigns the user (problem-definer) to the section being defined.
5. SECURITY(IS) is the current implementation of security keys. This is security for the PSL (to keep problem-definers from entering PSL sections which are not their own) as opposed to security for the target system.
6. SOURCE(IS) links information with its source, say a part of the previous documentation for the system or the person interviewed.
7. SYNONYMS(ARE) allows the user to define additional synonyms.

An important complement to the PSL is the PSA. The full use of PSA is explained in the appropriate manuals. The following discussion briefly reviews the types of reports (and analysis) available from PSA:

1. CONSISTS-COMPARISON. This report compares the contents of SETS, INPUTS, OUTPUTS, ENTITIES and/or GROUPS. Each of these is broken down into its smallest components (ELEMENTS or GROUPS) by tracing the CONSISTS statements in the PSL. A BASIC CONTENTS MATRIX is drawn with the SETS, INPUTS, OUTPUTS, ENTITIES and GROUPS specified as rows and the columns corresponding to ELEMENTS or GROUPS. An asterisk indicates that the ELEMENT or GROUP in a given column

appears in the "object" corresponding to a given row. The I-J'th entry indicates the number of items (ELEMENTS or GROUPS) objects I and J have in common. A report, the CONTENTS SIMILARITY SUMMARY, shows which "objects" are subsets of, or equivalent to, other "objects."

2. CONSISTS-MATRIX. This report also uses the CONSISTS and CONTAINS statements in the PSL to show which objects contain given ELEMENTS, GROUPS, INPUTS, OUTPUTS or ENTITIES. This is then shown via a list and with the aid of a matrix. The number of objects which contain the given ELEMENT (etc.) is also given.

3. CONTENTS. This report uses the CONSISTS statement to show the CONTENTS of SETS, INPUTS, OUTPUTS, ENTITIES and GROUPS. The structure is presented to show the heirarchy of the items contained. For example, A contains B which contains C and D, is represented as:

```
CONTENTS REPORT for A
      1.   B
         2   C
         2   D
```

4. DATA-PROCESS. This report provides information about PROCESSES and data objects (SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS and ELEMENTS). The DATA PROCESS INTERACTION MATRIX shows which data objects are INPUT, OUTPUT or UPDATED by a given process. The PROCESS INTERACTION MATRIX shows which processes interact with each other (i.e. which precede or succeed each other via data flow).

5. DICTIONARY. The dictionary report lists all names used in the PSL and such information as DESCRIPTION, SYNONYM, KEY-WORDS, etc.
6. ENTITY-IDENTIFIER. The IDENTIFIER INFORMATION REPORT is a matrix showing ELEMENTS versus the ENTITIES which are IDENTIFIED(BY) these ELEMENTS.
7. FORMATTED-PROBLEM-STATEMENT. This report takes all the information presented in the PSL and presents it in a clear manner. Information which is presented via a complementary statement is added to the report, etc. Thus regardless of the method or location of entry of given information, it is captured and reported with the appropriate name.
8. FREQUENCY. The HAPPENS statement is traced to all the INPUTS, OUTPUTS, EVENTS and PROCESSES which have HAPPENS statements.
9. KWIC. The KWIC INDEX permutes names about the dashes and presents a listing of all these names and their permutations in the PSL. This is most useful when problems with redundant names, etc. occur. The KWIC INDEX may also serve to retrieve information which concerns a given area of interest (similar to using KEYWORD).
10. NAME-GEN. The NAME-GEN provides a list of names retrieved using some selection criteria. These criteria include KEYWORDS, name TYPE (i.e. all PROCESSES), all SUBPARTS of a given name, all sections defined by a given PROBLEM-DEFINER,

etc. The NAME-GEN is most useful in limiting the contents of other reports. Thus, for example, a CONSISTS-MATRIX for those sections defined by a given PROBLEM-DEFINER or dealing with given KEYWORDS can be extracted.

11. NAME-LIST. The NAME-LIST produces a listing of all names in the PSL data base. Ordering may, optionally, be by TYPE, thus all SETS are listed together, etc.

12. PICTURE. The PICTURE presents data in a graphical format. The PICTURE may be generated for SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS, ELEMENTS, REAL-WORLD-ENTITIES and PROCESSES. Options include "NOFLOW" to omit information relating PROCESSES with their INPUTS and OUTPUTS; "NODATA" to exclude data relating PROCESSES with SETS, ENTITIES, GROUPS and ELEMENTS; "NOSTRUCTURE" to omit structure information (from SUBPARTS, CONSISTS and SUBSETS statements.

13. PRINT-ATTRIBUTE-VALUES. The ATTRIBUTE REPORT lists each ATTRIBUTE name for every name which has this ATTRIBUTE, lists the VALUE.

14. PROCESS-INPUT-OUTPUT. This report provides data flow information by linking PROCESSES to INPUTS and OUTPUTS via USES, RECEIVES, GENERATES, DERIVES and UPDATES statement.

15. PUNCH-COMMENT-ENTRY. This report retrieves comment-entries used in DESCRIPTION, DERIVATION, PROCEDURE, VOLATILITY, etc., statements.

16. STRUCTURE. This report presents the structures which result from use of the SUBPARTS statement. It can be called for REAL-WORLD-ENTITIES, INPUTS, OUTPUTS and PROCESSES.

COMPLETENESS CHECKS[46]

COMPLETENESS CHECKS FOR SPECIFCATION OF INPUTS

System Flow

1. Every INPUT must be GENERATED by some RWE.

Structure

1. All INPUT structures having SUBPARTS must terminate in INPUTS which have a media ATTRIBUTE (whose value can be "TO BE DETERMINED", TBA) and which contain data values.
2. An INPUT cannot have both a SUBPART statement and a CONTAINS statement. Only the lowest level INPUT can CONTAIN ELEMENTS.

Data Contents

1. All INPUTS at the lowest level, i.e. those that have the media ATTRIBUTE must consist of GROUPS and ELEMENTS. Any groups must be reducible to ELEMENTS.

Processing

1. All INPUTS must be RECEIVED by a PROCESS or it must have SUBPARTS all of which are RECEIVED by PROCESSES.
2. Every ELEMENT contained in an INPUT must be USED by one of the PROCESSES which RECEIVED the INPUT.

Size and Volume

1. Every INPUT must have a HAPPENS/TIMES statement.

COMPLETENESS CHECKS FOR SPECIFICATION OF OUTPUTS

System Flow

1. Every OUTPUT must be RECEIVED by some RWE.

Structure

1. All OUTPUT structures having SUBPARTS must terminate in OUTPUTS which have a media ATTRIBUTE (whose value can be "TO BE DETERMINED", TBA) and which contain data values.
2. An OUTPUT cannot have both a SUBPART statement and a CONTAINS statement. Only the lowest level OUTPUT can CONTAIN ELEMENTS.

Data Contents

1. All OUTPUTS at the lowest level, i.e. those that have the media ATTRIBUTE must consist of GROUPS and ELEMENTS. Any groups must be reducible to ELEMENTS.

Processing

1. All OUTPUTS must be GENERATED by a PROCESS or it must have SUBPARTS all of which are GENERATED by PROCESSES.
2. Every ELEMENT CONTAINED in an OUTPUT must be DERIVED by one of the PROCESSES which GENERATED the OUTPUT.

Size and Volume

1. Every OUTPUT must have a HAPPENS/TIMES statement or the PROCESS which GENERATES it must have a HAPPENS/TIMES statement.

COMPLETENESS CHECKS FOR SPECIFICATION OF SETS

Structure

1. All SETS must "eventually" consist of INPUTS, OUTPUTS or ENTITIES.

Processing

1. Every SET must be USED or UPDATED by some PROCESS.

Size and Volume

1. Every SET must have a CARDINALITY.
2. Every SET must have a VOLATILITY-MEMBER and VOLATILITY-SET statement.

COMPLETENESS CHECKS IN SPECIFICATION OF ENTITIES

1. Every ENTITY must be CONTAINED in at least one SET.

Data Contents

1. All GROUPS in an ENTITY must CONSIST of ELEMENTS.

Processing

1. Every ENTITY must be UPDATED by some PROCESS.
2. Every ELEMENT in an ENTITY must serve at least one purpose:
   - IDENTIFIER of the ENTITY
   - USED by some PROCESS, or
   - UPDATED by some PROCESS.

Size and Volume

1. Every ENTITY must have a VOLATILITY statement.
2. Every ENTITY must have CARDINALITY.

COMPLETENESS CHECKS FOR SPECIFICATION OF GROUPS

Data Structure

1. Every GROUP must "eventually" CONSIST of ELEMENTS.

Processing

1. Processing statements in which GROUPS appear must apply to all ELEMENTS in the GROUP.

COMPLETENESS CHECKS FOR SPECIFICATIONS OF ELEMENTS

Data Structure

1. Every ELEMENT must be CONTAINED in at least one INPUT, OUTPUT or ENTITY.
2. An ELEMENT cannot be CONTAINED in a GROUP, ENTITY, SET, INPUT or OUTPUT more than once.

Processing

1. Every ELEMENT CONTAINED in an INPUT must be USED in some way.
2. Every ELEMENT CONTAINED in an ENTITY must serve a purpose.
3. Every ELEMENT CONTAINED in an OUTPUT must be DERIVED by some PROCESS.

COMPLETENESS CHECKS IN SYSTEM DYNAMICS

1. Every PROCESS must be triggered by some EVENT.
2. Every PROCESS must have a HAPPENS/TIMES statement.
3. Every CONDITION must be named in at least one EVENT.
4. Every EVENT must TRIGGER at least one PROCESS.
5. Every CONDITION must have a TRUE WHILE or a FALSE WHILE statement.
6. Every EVENT must be caused by one of the following:

   (i) a CONDITION,
   (ii) the INCEPTION of an EVENT, or
   (iii) the TERMINATION of an EVENT.

COMPLETENESS CHECK FOR SPECIFICATIONS OF EVENTS

Size and Volume

1. Every EVENT must have a HAPPENS/TIMES statement or the equivalent information must be derivable from EVENT and CONDITIONS statements.

System Dynamics

1. Every EVENT must TRIGGER at least one PROCESS.
2. Every EVENT must be caused by one of the following:
   (i) a CONDITION,
   (ii) the INCEPTION of an EVENT, or
   (iii) the TERMINATION of an EVENT.

COMPLETENESS CHECK FOR SPECIFICATIONS OF CONDITIONS

1. Every CONDITION must have a TRUE WHILE or a FALSE WHILE statement.
2. Every CONDITION must be named in at least one EVENT.

COMPLETENESS CHECKS IN SYSTEM FLOW

1. Every RWE must either GENERATE some INPUT or RECEIVE some OUTPUT or be RESPONSIBLE for some SET.
2. Every INPUT must be GENERATED by some RWE and RECEIVED by some PROCESS.
3. Every OUTPUT must be GENERATED by some PROCESS and RECEIVED by some RWE.
4. Every SET must be USED or UPDATED by some PROCESS.

COMPLETENESS CHECKS FOR SPECIFICATIONS OF PROCESS

Structure

1. A PROCESS which does not have any SUBPARTS, must have a PROCEDURE statement.

Processing

1. Every PROCESS must acquire some data either by USING or UPDATING.
2. Every PROCESS must produce data by DERIVING data or by UPDATING it.

Size and Volume

1. Every PROCESS must have a HAPPENS/TIMES statement or the equivalent information must be derivable from EVENT and CONDITION statements.

System Dynamics

1. Every PROCESS must be triggered by some EVENT.

OTHER COMPLETENESS CHECKS

SYSTEM PARAMETER

1. Should have a value.
2. Every CONSISTS OF statement in INPUTS, OUTPUTS, ELEMENTS, GROUPS, ENTITY, SETS which does not have a SYSTEM PARAMETER will be listed.

ATTRIBUTES

1. Should have an attribute value

ATTRIBUTE-VALUE

1. Should not apply to more than 1 ATTRIBUTE.

UNDEFINED NAMES

1. Should not exist

COMPLETENESS AND CONSISTENCY CHECKS IN SYSTEM STRUCTURE

1. All the completeness statements in system flow apply to each subpart as it is defined.

2. At each subdivision, the totality of statements about the subparts must be consistent with the statement about the objects to which the parts belong.

COMPLETENESS CHECKS ON SYSTEM SIZE AND VOLUME STATEMENTS

1. Every INPUT must have a HAPPENS/TIMES statement.

2. Every OUTPUT must have a HAPPENS/TIMES statement or the Process which GENERATES it must have a HAPPENS/TIMES statement.

3. Every EVENT and PROCESS must have a HAPPENS/TIMES statement or the equivalent information must be desirable from EVENT and CONDITION statements.

4. Every SET, ENTITY and RELATION must have a CARDINALITY.

5. Every CONSISTS OF statement in INPUTS, OUTPUTS, ELEMENTS, GROUPS, ENTITY, SETS which does not have a SYSTEM PARAMETER will be listed.

6. Every SET must have a VOLATILITY-MEMBER and VOLATILITY-SET statement.

7. Every ENTITY must have a VOLATILITY statement.

CONSISTENCY CHECKS ON SYSTEM SIZE AND VOLUME STATEMENTS

1. If the completeness check may be satisfied in more than one way, as in checks 2 and 3, and if both are stated, they must lead to the same result.

APPENDIX D

COMPANY Z, EXCERPTS[47]

COMPANY Z, EXCERPTS

Figure D1 through D5 give a glimpse of the Company Z system design problem. The user is given an introduction to the problem (figure D1); a description of the organization (figure D2) and detailed specifications for the information processing required for Company Z.

Company Z

A. Introduction

Company Z is a medium size manufacturer of an electrical widgit and ranks second in an industry of four major manufacturers. Last year's sales totaled $12,000,000. All manufacturing, distributing, and administration functions are performed in one plant, located on the outskirts of a large West Coast city.

The structure of the firm with respect to data processing is as follows:

a) All data processing is carried out in the DPC. It is responsible for preparing documents and reports for each department and for management.

b) Each department has the responsibility for carrying out its individual functions, as described in more detail below. Their only responsibilities in the area of data processing are to ensure that documents going to DPC are complete and correct with respect to format.

c) Control is exercised by having each department compile certain activity and control totals. These totals are sent to the Internal Audit Department, and in some cases to DPC.

Figure D1. Company Z, Introduction

B. The Departments

The functions of the departments are outlined below:

a) Sales. The function of the Sales Department is to obtain as large a dollar volume of sales as possible. It affects the sales levels through its relations with customers. Customers' orders are documented and the information passed on to DPC. In return, the Sales Department needs reports showing sales.

b) Accounting. This department has the task of managing accounts receivables and accounts payables and of generating payrolls. It strives to minimize credit losses and at the same time attempting to offer as liberal credit terms as possible. In managing accounts payable, it strives to take advantage of all trade credit offered from vendors. Invoices to customers and payaments from customers, invoices from vendors and payments to vendors, and payroll go through this department. It needs frequent reports on status of customer accounts, summary reports showing the extent to which available discounts are used by customers, customer invoices, payment authorizations, and other reports and documents to fulfill its objectives.

c) Shipping. Ths Shipping Department is responsible for ensuring efficient delivery at minimum costs. It needs reports showing weights and volumes shipped, breakage and other expenses by type of transportation.

Figure D2. Company Z, Departments

d) Warehouse. The Warehouse attempts to maximize the use of space while minimizing the cost of handling goods, parts, and raw materials. It receives picking tickets in the order in which the items are stored in the warehouse. It needs reports on the frequency with which items are received from vendors and are ordered by the production department and by customers.

e) Receiving. This department performs primarily an inspection function. It receives a copy of purchase orders on which the quantity ordered has not been entered. After inspection, the copy is marked with quantity accepted, quantity rejected and additional costs, if any, and then returned to DPC.

f) Purchasing. This department is responsible for obtaining as favorable terms as possible from vendors and for locating sources of supply. It needs reports showing the performance of each vendor. All purchase orders to vendors go through the Purchasing Department. It is responsible for keeping DPC informed about any new information on vendors.

g) Production. The Production Department is responsible for producing an economical product, in accordance with quality standards and in time to meet scheduled needs. The department produces components of the finished product from raw materials and assembles these together with purchased parts into the finished product. It needs reports on cost and volume performance.

Figure D2, cont.

Internally Initiated Reports

An internally initiated report is due strictly to some need within the company for information. With one exception, all reports in this category come from the DPC. Time cards are the only input to the DPC other than the inputs which were described under externally initiated reports. The following documents are internally initiated:

FROM DPC:

1. Paychecks
2. Tax Report-Employee
3. Tax Report-Company
4. Accounts Receivable Report
5. Accounts Payable Report
6. Inventory Status Report
7. Profit Analysis Report
8. Back-Order Report
9. Customer Report
10. Credit Report
11. Warehouse Report
12. Sales Report

TO DPC:

1. Time Cards

Each of these reports are specified below:

Figure D3. Company Z, Internally Initiated Reports

1. PROBLEM: Paycheck

Each department holds the responsibility for presenting the DPC with the labor information for its employees. This report is created once a week. It should include the following information: social security number, employee name, department number, hourly or salary employee, regular hours worked, overtime hours worked.

INPUT:

As the DPC receives the time cards from the department it prepares the information for processing. The format will be as follows:

FIELD SIZE	NATURE OF DATA
9	social-security-number
20	employee-name
1	department-number
1	wage-code
8	regular-hours
8	overtime-hours

PROCESSING:

The following steps will be performed to process a time-card:

1. Using the social-security-number as a key, the employee's record is accessed. This record gives the employee's wage rate or salary and holds cumulative totals on wages, taxes, etc. Once the record is accessed, further processing can continue.
2. Calculate Pay.
   a) if wage-code = 0 (salaried)
      then go to b.
      else regular-wage = regular-hours * wage-rate,
           overtime-wage = overtime-hours * wage-rate * 1.5,
           total-wage = regular-wage + overtime-wage, go to 3.
   b) total-wage = salary-wage

Figure D4. Company Z, Paycheck

3. Calculate Deductions -- federal and state taxes, social security, etc.

4. Print Check.

OUTPUT:

1. Paychecks.

The paychecks will be printed on pre-printed forms. The form will be in two parts -- one part above the other. The upper part being the check and the lower a record of all earnings and deductions to that time. The check will contain the following items: employee name, employee address, date, amount. The lower part will provide space for the following information: date, amount of check, period for which payment is made, total earnings, social security tax, federal income tax, state income tax, total deductions, net pay, total earnings to date, total social security tax withheld to date, total federal income tax withheld to date, total state income tax withheld to date. The two parts of the form will be separated by a perforated edge.

Figure D4, cont.

2. PROBLEM: Tax Report-Employee

The Tax Report is generated by the DPC at the request of the Accounting Department. Its purpose is to report the total amount of FICA, Federal and State Tax withheld from each employee.

INPUT:

There is no input requirement from the accounting department. The report can be generated from the totals maintained in the history items for each employee.

PROCESSING:

Because all the information needed to generate the report can be found in each employee's record, the only processing required to generate the report will be to transport the data from the employee record to the output medium. No calculations will be required.

OUTPUT:

For each employee, the report will contain the following information:

LINE NO.	FIELD	DATA-ITEM
1	1-20	employee-name
	21-23	blank
	24-34	social-security-number
	35-37	blank
	38-45	total-social-security-tax
	46-48	blank
	49-56	total-federal-tax
	57-59	blank
	60-67	total-state-tax

Figure D5. Company Z, Tax Report-Employee

APPENDIX E

THE CLUSTERING PROGRAM

THE CLUSTERING PROGRAM

The clustering algorithm developed in chapter five is presented in this appendix. Figure E1 is a listing of the FORTRAN program which was written per the flowchart presented in that chapter. The input data follows. Figure E2 is the printout of the DATA ITEM names and lengths (in words). Similarly Figure E3 is the printout of the PROCESS names and the volumes associated with each process. The DATA ITEM/PROCESS Incidence Matrix appears as Figure E4. (A "1" indicates that the DATA ITEM is input to the PROCESS and a "2" is used to indicate that the DATA ITEM is output. This change in notation allows a neater representation of large matrices.) Although the matrix was input to the algorithm manually, it could be generated by the RSM in future implimentations.

Figure E5 is the first output of the program. The transport volume is computed for each DATA ITEM. Note that certain DATA ITEMS have a zero transport volume. This indicates a gap in the input data which is typical of user specified data. Certain DATA ITEMS were not given any incidence and this is the result. The first step of the algorithm is to group rows which have like process incidence. Here, again, the zero incidence DATA ITEMS cloud the picture as they are all grouped together. The groupings which are of interest are 1-2-50, 7-8, 10-36 and 39-40. Figure E7 shows the output after the first

iteration of the clustering algorithm itself. DATA ITEMS 22 and 32 have been grouped with a resulting wasted transport volume (WTV) of 400. The clusters formed to this point are listed serially along with their volumes. Figure E8 lists those DATA ITEMS which are not yet in any cluster or grouping. Figure E9 shows the results of the last feasible iteration using this objective function. DATA ITEMS 6 and 14 have been combined with a WTV of 104300. The combination of DATA ITEMS which are already in a cluster serves to combine those clusters. Notationally each cluster is represented by the lowest numbered DATA ITEM in it. Figure E10 is a recap of the clustering showing the chronological order of the clustering. It is the basis for the last section of chapter five. Figure E11 shows the last iteration of the algorithm as modified to use a different clustering objective function and Figure E12 is the corresponding recap of the clustering. This also is discussed in the last section of chapter five.

```
      PROGRAM SINGER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C    THIS PROGRAM WILL CLUSTER THE DATA ITEMS BASED ON ACTUAL TV SAVINGS
C
C  DIMENSIONS
      INTEGER  PROCS(30,5), OUT,HIST(400,2)
      DIMENSION IM(60,30),NAME(60,5), DILEN(60),NCL(60)
      DIMENSION PVOL(30),IND(60),TR(60),HTV(400)
      ITR=0
      OUT=6
      IN=5
      CTOT=0.
C
C  READ DATA ITEM NAMES AND LENGTHS
      WRITE(OUT,902)
  902 FORMAT(1H1,10X,"DATA ITEM NAMES AND LENGTHS"//)
      I=0
  101 I=I+1
      READ(IN,904) DILEN(I),(NAME(I,J),J=1,5) ,K
  904 FORMAT(F10.0, 5A6,I3)
      IF(K.EQ.999) GO TO 102
      WRITE(OUT,905) I,(NAME(I,J),J=1,5) , DILEN(I)
      GO TO 101
  102 NROW=I-1
C
C    READ IN PROCESS NAMES AND VOLUMES
      I=0
      WRITE(OUT,901)
  901 FORMAT(1H1 10X, "PROCESS NAMES AND VOLUMES" //)
  103 I=I+1
      READ( IN,904)        PVOL(I),(PROCS(I,J),J=1,5), K
      IF(K.EQ.999) GO  TO 104
      WRITE(OUT,905) I,(PROCS(I,J),J=1,5),  PVOL(I)
  905 FORMAT(I3,".  " 5A6,F12.0)
      GO TO 103
  104 NCOL=I-1
C
C    INPUT THE DATA ITEM/PROCESSS INCIDENCE MATRIX
      WRITE(OUT,908)
  908 FORMAT(1H1,10X,"DATA ITEM/PROCESS INCIDENCE MATRIX"//)
      DO 110 I=1,NROW
      READ(IN,906) (IM(I,J),J=1,NCOL)
  906 FORMAT(50I1)
  907 FORMAT(I5,2X,10(2X,5I1))
  110 WRITE(OUT,907) I,(IM(I,J),J=1,NCOL)
```

Figure E1. Program Listing

```
C   COMPUTE TOTAL TRANSPORT  FOR EACH ROW    (TO BE MULTIPLIED BY LENGTH LATER)
      WRITE(OUT,988)
  988 FORMAT(1H1,"    DATA ITEM    TRANSPORT VOLUME    *  LENGTH"//)
      DO 128 I=1,NROW
      TR(I)=0.
      DO 123 K=1,NCOL
      IF(IM(I,K).EQ.0) GO TO 123
      TR(I)=TR(I)+PVOL(K)
  123 CONTINUE
      X=TR(I)*DILEN(I)
  128 WRITE(OUT,981) I,(NAME(I,J) ,J=1,5), TR(I),X
  981 FORMAT(I5,".    " 5A6,2F12.0)


      WRITE(OUT,986)
  986 FORMAT(//)
C
C  GROUP IDENTICAL ROWS--DATA ITEMS WITH SAME INCIDENCE
      DO 130 I=1,NROW
  130 IND(I)=0
C  IND IS INDICATOR    =0  NO GROUP  = -1  FIRST ROW OF GROUP  =+ MEMBER OF GROUP
      NROW1=NROW-1
C    DO NESTED LOOP  COMPARE EACH ROW WITH EVERY OTHER ROW
      DO 131 I=1,NROW1
      IF(IND(I).NE.0) GO TO 131
      L=I+1
           DO 132 J=L,NROW
      IF(IND(J).NE.0) GO TO 132
        DO 133 K=1,NCOL
      IF(IM(I,K).NE.IM(J,K)) GO TO 132
  133 CONTINUE
      IND(J)=I
      IND(I)=-1
C  ADJUST LENGTH TO REFLECT THIS COMBINATION
      DILEN(I)=DILEN(I)+DILEN(J)

      WRITE(OUT,910) I,J,X
  910 FORMAT(" XXXXXXXXXX       ROWS"2I5," ARE ALIKE  ")
      ITR=ITR+1
      HIST(ITR,1)=I
      HIST(ITR,2)=J
      HTV(ITR)=0.
  132 CONTINUE
  131 CONTINUE
```

Figure E1, cont.

```
C    COMPUTE COMMONALITY AMONG THE CLUSTERS
  150 IC=0
      DO 140 I=1,NROW1
      IF(IND(I).GT.0) GO TO 140
      L=I+1
      DO 141 J=L,NROW
      IF(IND(J).GT.0) GO TO 141
      C=0.
      DO 142 K=1,NCOL
      IF((IM(I,K)+IM(J,K)).EQ.3) GO TO 141
C   C IS WASTED TRANSPORT VOLUME
      IF(IM(I,K)-IM(J,K)) 191,142,192
  191 C=C+PVOL(K)*DILEN(I)
      GO TO 142
  192 C=C+PVOL(K)*DILEN(J)
  142 CONTINUE
C     STORE BEST PAIR
      IF(IC.EQ.0) GO TO 146
      IF(C.GT.CMIN) GO TO 141
  146 IC=1
  147 CMIN=C
      CTOT=CTOT+C
      ISAV=I
      JSAV=J
  141 CONTINUE
  140 CONTINUE
C     NOW GROUP THESE TWO AND REPEAT THE COMPUTATIONS  (ITERATE)
      IF(IC.EQ.0) GO TO 500
      IF(IND(JSAV).NE.-1) GO TO 180
      DO 181 I=1,NROW
      IF(IND(I).EQ.JSAV) IND(I)=ISAV
  181 CONTINUE
  180 IND(JSAV)=ISAV
      IND(ISAV)=-1
      DILEN(ISAV)=DILEN(ISAV)+DILEN(JSAV)
C   PRINT      THIS ITERATION
      ITR=ITR+1
      WRITE(OUT,920) ISAV,JSAV,CMIN,CTOT,ITR
  920 FORMAT(1H1,10X,"THIS ITERATION HAS COMBINED"2I5,2F10.0,I5)
      HIST(ITR,1)=ISAV
      HIST(ITR,2)=JSAV
      HTV(ITR)=CMIN
```

Figure E1, cont.

```
C    ADJUST THE NEW INCIDENCE MATRIX FOR THE COMBINED CLUSTERS
     DO 200 K=1,NCOL
     IF(IM(JSAV,K).GT.0) IM(ISAV,K)=IM(JSAV,K)
 200 CONTINUE
 501 NCLUST=0
C    FIRST SEARCH FOR CLUSTERS  THEN FOR UNGROUPED DATA ITEMS
     DO 170 I=1,NROW1
     IF(IND(I).NE.-1) GO TO 170
     NCLUST=NCLUST+1
      WRITE(OUT,924) NCLUST
 924 FORMAT(//,20X,"CLUSTER NUMBER"I5,/)
     WRITE(OUT,925)  I,(NAME(I,K),K=1,5),TR(I)
 925 FORMAT(I5,2X, 5A6,10X,4F12.0)
     L=I-1
     DO 171 J=L,NROW
     IF(IND(J).NE.I) GO TO 171
     NCL(J)=NCLUST
     WRITE(OUT,925) J,(NAME(J,K),K=1,5),TR(J)
 171 CONTINUE
 170 CONTINUE
     WRITE(OUT,927)
 927 FORMAT(///," SINGLE DATA ITEMS" ///)
     DO 175 I=1,NROW
     IF(IND(I).NE.0) GO TO 175
     NCLUST=NCLUST+1
     WRITE(OUT,926) I, (NAME(I,J),J=1,5), NCLUST
 175 CONTINUE
 926 FORMAT(I5,2X,10A6,10X," IS CLUSTER NUMBER"I5)
     GO TO 150
 500 WRITE(OUT,922)
 922 FORMAT(1H1  10X,"NO FEASIBLE CLUSTERING, ALGORITHM END")
     WRITE(OUT,960)
 960 FORMAT(//, 10X,"RECAP OF EACH CLUSTER"/)
     DO 300 K=1,NCLUST
     WRITE(OUT,961) K
 961 FORMAT(//,10X,"CLUSTER NUMBER"I3,//)
     DO 301 L=1,ITR
     J=HIST(L,2)
     IF(NCL(J).NE.K) GO TO 301
     WRITE(OUT,962) L,HIST(L,1),J,HTV(L)
 962 FORMAT(" ITERATION NUMBER"I5,2I8,10X,F10.0)
 301 CONTINUE
 300 CONTINUE
     STOP
     END
```

Figure E1, cont.

```
  192 C=C+PVOL(K)*DILEN(J)
  142 CONTINUE
C     STORE BEST PAIR
      VOLTOT(I)=0.
      DO 441 K=1,NCOL
      IF(IM(I,K).EQ.0) GO TO 441
      VOLTOT(I)=VOLTOT(I)+PVOL(K)*DILEN(I)
  441 CONTINUE
      IF(VOLTOT(I).LT.1.) VOLTOT(I)= 1.
      C=C/VOLTOT(I)
      IF(IC.EQ.0) GO TO 146
```

Figure E1, cont.

DATA ITEM NAMES AND LENGTHS

1.	TIME-IN	6.
2.	TIME-OUT	6.
3.	NORMAL-PAY-RATE-CODE	2.
4.	OVERTIME-PAY-RATE-CODE	2.
5.	SOC-SEC-YEAR-RO-DATE-EMP	6.
6.	HOURS-WORKED	3.
7.	SOC-SEC-FED-PERCENT	3.
8.	SOC-SEC-CEILING	7.
9.	LOCALITY-NAME	25.
10.	STATE-NAME	25.
11.	STATE-TAX-YEAR-TO-DATE-EMPLOYE	7.
12.	STATE-CODE	2.
13.	STATE-TAX-AMOUNT	6.
14.	LOCAL-TAX-YEAR-TO-DATE	7.
15.	LOCAL-TAX-AMOUNT	6.
16.	BONUS	6.
17.	LOCALITY-CODE	4.
18.	EMPLOYEE-SSAN	9.
19.	EMPLOYEE-STREET-NUMBER	25.
20.	EMPLOYEE-CITY-STATE-ZIP	35.
21.	NUMBER-OF-DEPENDENTS	2.
22.	NUMBER-OF-ALLOTMENTS	2.
23.	ALLOTMENT-AMOUNT	5.
24.	OVERTIME-PAYRATE	4.
25.	OVERTIME-PAY	6.
26.	OVERTIME-HOURS	3.
27.	REGULAR-PAYRATE	4.
28.	REGULAR-HOURS	3.
29.	REGULAR-PAY	6.
30.	UNION-DUES	5.
31.	SUPERVISORS-SSAN	9.
32.	RETIREMENT-PLAN-CODE	2.
33.	RETIREMENT-DEDUCTION-AMOUNT	5.
34.	DEPARTMENT-NUMBER	4.
35.	DIVISION-NUMBER	3.
36.	HEALTH-BENEFIT-PLAN-CODE	2.
37.	HEALTH-BENEFIT-PLAN-AMOUNT	5.
38.	JOB-SKILL-CODE-EMP	3.
39.	SICK-LEAVE-REMAINING	5.
40.	SICK-LEAVE-USED-THIS-PERIOD	3.
41.	GROSS-PAY	6.
42.	NET-PAY	6.
43.	DEDUCTIONS	6.
44.	FED-TAX	6.
45.	FED-TAX-YEAR-TO-DATE	6.
46.	EMP-PROMOTION-CLOCK-DATE	5.
47.	EMP-COMP-START-DATE	6.
48.	FED-TAX-RATE	3.
49.	SOC-SEC-DEDUCTION	5.
50.	NORMAL-HOURS	3.
51.	TOTAL-ALLOCATIONS	3.
52.	STATE-TAX-RATE	3.
53.	LOCAL-TAX-RATE	3.

Figure E2. Data Item Names and Lengths

PROCESS NAMES AND VOLUMES

1.	WP-HOURLY	0.
2.	WP-HOURLY-PAYCHECK	0.
3.	WP-HOURLY-SOC-SEC	0.
4.	WP-HOURLY-TAX	0.
5.	WP-HOURLY-DEDUCTIONS	0.
6.	WP-ALLOTMENTS	0.
7.	WP-DISBURSEMENTS-NON-TAX	0.
8.	REG-HOURS	100.
9.	REG-PAY-RATE	100.
10.	OVT-PAY-RATE	100.
11.	OVT-HOURS	100.
12.	REG-PAY	100.
13.	OVT-PAY	100.
14.	GROSS-PAY	100.
15.	LOCAL-TAX	100.
16.	STATE-TAX	100.
17.	FED-TAX	100.
18.	SOC-SEC	100.
19.	ALLOTMENTS	100.
20.	HEALTH-BEN	100.
21.	RETIREMENT	100.
22.	LOCAL-TAX-YTD	100.
23.	STATE-TAX-YTD	100.
24.	FED-TAX-YTD	100.
25.	SOC-SEC-YTD	100.
26.	DEDUCTIONS	100.
27.	NET-PAY	100.
28.	ALLOTMENT-TOTAL	100.
29.	SICK-LEAVE-YTD	100.

Figure E3. Process Names and Volumes

DATA ITEM/PROCESS INCIDENCE MATRIX

1	00000	00100	10000	00000	00000	0000
2	00000	00100	10000	00000	00000	0000
3	00000	00010	00000	00000	00000	0000
4	00000	00001	00000	00000	00000	0000
5	00000	00000	00000	00000	00000	2000
6	00000	00000	00000	00000	00000	0000
7	00000	00000	00000	00100	00000	0000
8	00000	00000	00000	00100	00000	0000
9	00000	00000	00000	00000	00000	0000
10	00000	00000	00000	00001	00000	0000
11	00000	00000	00000	00000	00200	0000
12	00000	00000	00000	00000	00000	0000
13	00000	00000	00000	00000	00100	1000
14	00000	00000	00000	00000	01000	0000
15	00000	00000	00002	00000	01000	1000
16	00000	00000	00010	00000	00000	0000
17	00000	00000	00000	00000	00000	0000
18	00000	00000	00000	00000	00000	0000
19	00000	00000	00000	00000	00000	0000
20	00000	00000	00000	00000	00000	0000
21	00000	00000	00000	00000	00000	0000
22	00000	00000	00000	00010	00000	0000
23	00000	00000	00000	00020	00000	1000
24	00000	00002	00100	00000	00000	0000
25	00000	00000	00000	00000	00000	0000
26	00000	00000	20100	00000	00000	0000
27	00000	00020	01000	00000	00000	0000
28	00000	00200	11000	00000	00000	0000
29	00000	00000	02010	00000	00000	0000
30	00000	00000	00000	00000	00000	0000
31	00000	00000	00000	00000	00000	0000
32	00000	00000	00000	00000	10000	0000
33	00000	00000	00000	00000	20000	1000
34	00000	00000	00000	00000	00000	0000
35	00000	00000	00000	00000	00000	0000
36	00000	00000	00000	00001	00000	0000
37	00000	00000	00000	00002	00000	1000
38	00000	00000	00000	00000	00000	0000
39	00000	00000	00000	00000	00000	0001
40	00000	00000	00000	00000	00000	0001
41	00000	00000	00021	11100	00000	0100
42	00000	00000	00000	00000	00000	0200
43	00000	00000	00000	00000	00000	2100
44	00000	00000	00000	02000	00010	1000
45	00000	00000	00000	00000	00020	0000
46	00000	00000	00000	00000	00000	0000
47	00000	00000	00000	00000	00000	0000
48	00000	00000	00000	00000	00000	0000
49	00000	00000	00000	00200	00001	1000
50	00000	00100	10000	00000	00000	0000
51	00000	00000	00000	00000	00000	0020
52	00000	00000	00000	10000	00000	0000
53	00000	00000	00001	00000	00000	0000

Figure E4. Data Item/Process Incidence Matrix

	DATA ITEM	TRANSPORT VOLUME	* LENGTH	
1.	TIME-IN	200.		1200.
2.	TIME-OUT	200.		1200.
3.	NORMAL-PAY-RATE-CODE	100.		200.
4.	OVERTIME-PAY-RATE-CODE	100.		200.
5.	SOC-SEC-YEAR-RO-DATE-EMP	100.		600.
6.	HOURS-WORKED	0.		0.
7.	SOC-SEC-FED-PERCENT	100.		300.
8.	SOC-SEC-CEILING	100.		700.
9.	LOCALITY-NAME	0.		0.
10.	STATE-NAME	100.		200.
11.	STATE-TAX-YEAR-TO-DATE-EMPLOYE	100.		700.
12.	STATE-CODE	0.		0.
13.	STATE-TAX-AMOUNT	200.		1200.
14.	LOCAL-TAX-YEAR-TO-DATE	100.		700.
15.	LOCAL-TAX-AMOUNT	300.		1800.
16.	BONUS	100.		600.
17.	LOCALITY-CODE	0.		0.
18.	EMPLOYEE-SSAN	0.		0.
19.	EMPLOYEE-STREET-NUMBER	0.		0.
20.	EMPLOYEE-CITY-STATE-ZIP	0.		0.
21.	NUMBER-OF-DEPENDENTS	0.		0.
22.	NUMBER-OF-ALLOTMENTS	100.		200.
23.	ALLOTMENT-AMOUNT	200.		1000.
24.	OVERTIME-PAYRATE	200.		800.
25.	OVERTIME-PAY	0.		0.
26.	OVERTIME-HOURS	200.		600.
27.	REGULAR-PAYRATE	200.		800.
28.	REGULAR-HOURS	300.		900.
29.	REGULAR-PAY	200.		1200.
30.	UNION-DUES	0.		0.
31.	SUPERVISORS-SSAN	0.		0.
32.	RETIREMENT-PLAN-CODE	100.		200.
33.	RETIREMENT-DEDUCTION-AMOUNT	200.		1000.
34.	DEPARTMENT-NUMBER	0.		0.
35.	DIVISION-NUMBER	0.		0.
36.	HEALTH-BENEFIT-PLAN-CODE	100.		200.
37.	HEALTH-BENEFIT-PLAN-AMOUNT	200.		1000.
38.	JOB-SKILL-CODE-EMP	0.		0.
39.	SICK-LEAVE-REMAINING	100.		500.
40.	SICK-LEAVE-USED-THIS-PERIOD	100.		300.
41.	GROSS-PAY	600.		3600.
42.	NET-PAY	100.		600.
43.	DEDUCTIONS	200.		1200.
44.	FED-TAX	300.		1800.
45.	FED-TAX-YEAR-TO-DATE	100.		600.
46.	EMP-PROMOTION-CLOCK-DATE	0.		0.
47.	EMP-COMP-START-DATE	0.		0.
48.	FED-TAX-RATE	0.		0.
49.	SOC-SEC-DEDUCTION	300.		1500.
50.	NORMAL-HOURS	200.		600.
51.	TOTAL-ALLOCATIONS	100.		500.
52.	STATE-TAX-RATE	100.		300.
53.	LOCAL-TAX-RATE	100.		300.

Figure E5. Data Item Transport Volume and Length

```
ROWS     1     2   ARE ALIKE
ROWS     1    50   ARE ALIKE
ROWS     6     9   ARE ALIKE
ROWS     6    12   ARE ALIKE
ROWS     6    17   ARE ALIKE
ROWS     6    18   ARE ALIKE
ROWS     6    19   ARE ALIKE
ROWS     6    20   ARE ALIKE
ROWS     6    21   ARE ALIKE
ROWS     6    25   ARE ALIKE
ROWS     6    30   ARE ALIKE
ROWS     6    31   ARE ALIKE
ROWS     6    34   ARE ALIKE
ROWS     6    35   ARE ALIKE
ROWS     6    38   ARE ALIKE
ROWS     6    46   ARE ALIKE
ROWS     6    47   ARE ALIKE
ROWS     6    48   ARE ALIKE
ROWS     7     8   ARE ALIKE
ROWS    10    36   ARE ALIKE
ROWS    39    40   ARE ALIKE
```

Figure E6. Listing of Like Data Items

```
          THIS ITERATION HAS COMBINED    22    32        400.

                        CLUSTER NUMBER     1      10000.    22

  1   TIME-IN                                                  200.
  2   TIME-OUT                                                 200.
 50   NORMAL-HOURS                                             200.

                        CLUSTER NUMBER     2

  6   HOURS-WORKED                                               0.
  9   LOCALITY-NAME                                              0.
 12   STATE-CODE                                                 0.
 17   LOCALITY-CODE                                              0.
 18   EMPLOYEE-SSAN                                              0.
 19   EMPLOYEE-STREET-NUMBER                                     0.
 20   EMPLOYEE-CITY-STATE-ZIP                                    0.
 21   NUMBER-OF-DEPENDENTS                                       0.
 25   OVERTIME-PAY                                               0.
 30   UNION-DUES                                                 0.
 31   SUPERVISORS-SSAN                                           0.
 34   DEPARTMENT-NUMBER                                          0.
 35   DIVISION-NUMBER                                            0.
 38   JOB-SKILL-CODE-EMP                                         0.
 46   EMP-PROMOTION-CLOCK-DATE                                   0.
 47   EMP-COMP-START-DATE                                        0.
 48   FED-TAX-RATE                                               0.

                        CLUSTER NUMBER     3

  7   SOC-SEC-FED-PERCENT                                      100.
  8   SOC-SEC-CEILING                                          100.

                        CLUSTER NUMBER     4

 10   STATE-NAME                                               100.
 36   HEALTH-BENEFIT-PLAN-CODE                                 100.

                        CLUSTER NUMBER     5

 22   NUMBER-OF-ALLOTMENTS                                     100.
 32   RETIREMENT-PLAN-CODE                                     100.

                        CLUSTER NUMBER     6

 39   SICK-LEAVE-REMAINING                                     100.
 40   SICK-LEAVE-USED-THIS-PERIOD                              100.
```

Figure E7. First Iteration

SINGLE DATA ITEMS

```
  3   NORMAL-PAY-RATE-CODE
  4   OVERTIME-PAY-RATE-CODE
  5   SOC-SEC-YEAR-RO-DATE-EMP
 11   STATE-TAX-YEAR-TO-DATE-EMPLOYE
 13   STATE-TAX-AMOUNT
 14   LOCAL-TAX-YEAR-TO-DATE
 15   LOCAL-TAX-AMOUNT
 16   BONUS
 23   ALLOTMENT-AMOUNT
 24   OVERTIME-PAYRATE
 26   OVERTIME-HOURS
 27   REGULAR-PAYRATE
 28   REGULAR-HOURS
 29   REGULAR-PAY
 33   RETIREMENT-DEDUCTION-AMOUNT
 37   HEALTH-BENEFIT-PLAN-AMOUNT
 41   GROSS-PAY
 42   NET-PAY
 43   DEDUCTIONS
 44   FED-TAX
 45   FED-TAX-YEAR-TO-DATE
 49   SOC-SEC-DEDUCTION
 51   TOTAL-ALLOCATIONS
 52   STATE-TAX-RATE
 53   LOCAL-TAX-RATE
```

Figure E8. Single Data Items

```
      THIS ITERATION HAS COMBINED     6    14    104300.

                    CLUSTER NUMBER    1        1502200.    49

 1   TIME-IN                                              200.
 2   TIME-OUT                                             200.
 3   NORMAL-PAY-RATE-CODE                                 100.
 4   OVERTIME-PAY-RATE-CODE                               100.
11   STATE-TAX-YEAR-TO-DATE-EMPLOYE                       100.
22   NUMBER-OF-ALLOTMENTS                                 100.
32   RETIREMENT-PLAN-CODE                                 100.
39   SICK-LEAVE-REMAINING                                 100.
40   SICK-LEAVE-USED-THIS-PERIOD                          100.
44   FED-TAX                                              300.
49   SOC-SEC-DEDUCTION                                    300.
50   NORMAL-HOURS                                         200.
                    CLUSTER NUMBER    2

 5   SOC-SEC-YEAR-RO-DATE-EMP                             100.
 7   SOC-SEC-FED-PERCENT                                  100.
 8   SOC-SEC-CEILING                                      100.
10   STATE-NAME                                           100.
24   OVERTIME-PAYRATE                                     200.
26   OVERTIME-HOURS                                       200.
36   HEALTH-BENEFIT-PLAN-CODE                             100.
41   GROSS-PAY                                            600.
43   DEDUCTIONS                                           200.
51   TOTAL-ALLOCATIONS                                    100.
52   STATE-TAX-RATE                                       100.
53   LOCAL-TAX-RATE                                       100.
                    CLUSTER NUMBER    3

 6   HOURS-WORKED                                           0.
 9   LOCALITY-NAME                                          0.
12   STATE-CODE                                             0.
14   LOCAL-TAX-YEAR-TO-DATE                               100.
15   LOCAL-TAX-AMOUNT                                     300.
17   LOCALITY-CODE                                          0.
18   EMPLOYEE-SSAN                                          0.
19   EMPLOYEE-STREET-NUMBER                                 0.
20   EMPLOYEE-CITY-STATE-ZIP                                0.
21   NUMBER-OF-DEPENDENTS                                   0.
25   OVERTIME-PAY                                           0.
27   REGULAR-PAYRATE                                      200.
28   REGULAR-HOURS                                        300.
30   UNION-DUES                                             0.
31   SUPERVISORS-SSAN                                       0.
34   DEPARTMENT-NUMBER                                      0.
35   DIVISION-NUMBER                                        0.
38   JOB-SKILL-CODE-EMP                                     0.
46   EMP-PROMOTION-CLOCK-DATE                               0.
47   EMP-COMP-START-DATE                                    0.
48   FED-TAX-RATE                                           0.
                    CLUSTER NUMBER    4

13   STATE-TAX-AMOUNT                                     200.
16   BONUS                                                100.
23   ALLOTMENT-AMOUNT                                     200.
29   REGULAR-PAY                                          200.
33   RETIREMENT-DEDUCTION-AMOUNT                          200.
37   HEALTH-BENEFIT-PLAN-AMOUNT                           200.
42   NET-PAY                                              100.
45   FED-TAX-YEAR-TO-DATE                                 100.
```

Figure E9. Last Feasible Iteration

CLUSTER NUMBER 1

ITERATION NUMBER	1	1	2	0.
ITERATION NUMBER	2	1	50	0.
ITERATION NUMBER	21	39	40	0.
ITERATION NUMBER	22	22	32	400.
ITERATION NUMBER	23	3	4	400.
ITERATION NUMBER	34	11	39	1500.
ITERATION NUMBER	35	3	22	1600.
ITERATION NUMBER	36	44	49	2200.
ITERATION NUMBER	42	1	11	6000.
ITERATION NUMBER	44	3	44	8400.
ITERATION NUMBER	48	1	3	34600.

CLUSTER NUMBER 2

ITERATION NUMBER	19	7	8	0.
ITERATION NUMBER	20	10	36	0.
ITERATION NUMBER	24	52	53	600.
ITERATION NUMBER	26	5	43	600.
ITERATION NUMBER	27	24	26	700.
ITERATION NUMBER	28	10	51	900.
ITERATION NUMBER	37	41	52	2400.
ITERATION NUMBER	38	7	10	2900.
ITERATION NUMBER	41	5	24	5000.
ITERATION NUMBER	45	7	41	11900.
ITERATION NUMBER	47	5	7	25700.

CLUSTER NUMBER 3

ITERATION NUMBER	3	6	9	0.
ITERATION NUMBER	4	6	12	0.
ITERATION NUMBER	5	6	17	0.
ITERATION NUMBER	6	6	18	0.
ITERATION NUMBER	7	6	19	0.
ITERATION NUMBER	8	6	20	0.
ITERATION NUMBER	9	6	21	0.
ITERATION NUMBER	10	6	25	0.
ITERATION NUMBER	11	6	30	0.
ITERATION NUMBER	12	6	31	0.
ITERATION NUMBER	13	6	34	0.
ITERATION NUMBER	14	6	35	0.
ITERATION NUMBER	15	6	38	0.
ITERATION NUMBER	16	6	46	0.
ITERATION NUMBER	17	6	47	0.
ITERATION NUMBER	18	6	48	0.
ITERATION NUMBER	30	27	28	1100.
ITERATION NUMBER	33	14	15	1400.
ITERATION NUMBER	43	14	27	7300.
ITERATION NUMBER	49	6	14	104300.

CLUSTER NUMBER 4

ITERATION NUMBER	25	16	29	600.
ITERATION NUMBER	29	33	37	1000.
ITERATION NUMBER	31	13	23	1100.
ITERATION NUMBER	32	42	45	1200.
ITERATION NUMBER	39	13	33	4200.
ITERATION NUMBER	40	16	42	4800.
ITERATION NUMBER	46	13	16	20400.

Figure E10. Recap of Clustering

```
THIS ITERATION HAS COMBINED    13    24    2.66667

                                          98.59124    49
                CLUSTER NUMBER    1

 1  TIME-IN                                              200.
 2  TIME-OUT                                             200.
 3  NORMAL-PAY-RATE-CODE                                 100.
 4  OVERTIME-PAY-RATE-CODE                               100.
 5  SOC-SEC-YEAR-RO-DATE-EMP                             100.
 6  HOURS-WORKED                                           0.
 7  SOC-SEC-FED-PERCENT                                  100.
 8  SOC-SEC-CEILING                                      100.
 9  LOCALITY-NAME                                          0.
10  STATE-NAME                                           100.
11  STATE-TAX-YEAR-TO-DATE-EMPLOYE                       100.
12  STATE-CODE                                             0.
14  LOCAL-TAX-YEAR-TO-DATE                               100.
16  BONUS                                                100.
17  LOCALITY-CODE                                          0.
18  EMPLOYEE-SSAN                                          0.
19  EMPLOYEE-STREET-NUMBER                                 0.
20  EMPLOYEE-CITY-STATE-ZIP                                0.
21  NUMBER-OF-DEPENDENTS                                   0.
22  NUMBER-OF-ALLOTMENTS                                 100.
25  OVERTIME-PAY                                           0.
29  REGULAR-PAY                                          200.
30  UNION-DUES                                             0.
31  SUPERVISORS-SSAN                                       0.
32  RETIREMENT-PLAN-CODE                                 100.
34  DEPARTMENT-NUMBER                                      0.
35  DIVISION-NUMBER                                        0.
36  HEALTH-BENEFIT-PLAN-CODE                             100.
38  JOB-SKILL-CODE-EMP                                     0.
39  SICK-LEAVE-REMAINING                                 100.
40  SICK-LEAVE-USED-THIS-PERIOD                          100.
42  NET-PAY                                              100.
46  EMP-PROMOTION-CLOCK-DATE                               0.
47  EMP-COMP-START-DATE                                    0.
48  FED-TAX-RATE                                           0.
50  NORMAL-HOURS                                         200.

                CLUSTER NUMBER    2

13  STATE-TAX-AMOUNT                                     200.
24  OVERTIME-PAYRATE                                     200.
26  OVERTIME-HOURS                                       200.

                CLUSTER NUMBER    3

15  LOCAL-TAX-AMOUNT                                     300.
23  ALLOTMENT-AMOUNT                                     200.
27  REGULAR-PAYRATE                                      200.
28  REGULAR-HOURS                                        300.
33  RETIREMENT-DEDUCTION-AMOUNT                          200.
37  HEALTH-BENEFIT-PLAN-AMOUNT                           200.
44  FED-TAX                                              300.
49  SOC-SEC-DEDUCTION                                    300.

                CLUSTER NUMBER    4

41  GROSS-PAY                                            600.
43  DEDUCTIONS                                           200.
45  FED-TAX-YEAR-TO-DATE                                 100.
51  TOTAL-ALLOCATIONS                                    100.
52  STATE-TAX-RATE                                       100.
53  LOCAL-TAX-RATE                                       100.
```

Figure E11. First Iteration, Alternate Objective Function

CLUSTER NUMBER 1

ITERATION NUMBER	1	1	2	0.00000
ITERATION NUMBER	2	1	50	0.00000
ITERATION NUMBER	3	6	9	0.00000
ITERATION NUMBER	4	6	12	0.00000
ITERATION NUMBER	5	6	17	0.00000
ITERATION NUMBER	6	6	18	0.00000
ITERATION NUMBER	7	6	19	0.00000
ITERATION NUMBER	8	6	20	0.00000
ITERATION NUMBER	9	6	21	0.00000
ITERATION NUMBER	10	6	25	0.00000
ITERATION NUMBER	11	6	30	0.00000
ITERATION NUMBER	12	6	31	0.00000
ITERATION NUMBER	13	6	34	0.00000
ITERATION NUMBER	14	6	35	0.00000
ITERATION NUMBER	15	6	38	0.00000
ITERATION NUMBER	16	6	46	0.00000
ITERATION NUMBER	17	6	47	0.00000
ITERATION NUMBER	18	6	48	0.00000
ITERATION NUMBER	19	7	8	0.00000
ITERATION NUMBER	20	10	36	0.00000
ITERATION NUMBER	21	39	40	0.00000
ITERATION NUMBER	27	1	32	.63333
ITERATION NUMBER	28	1	22	.45098
ITERATION NUMBER	29	1	4	.35526
ITERATION NUMBER	30	1	3	.29524
ITERATION NUMBER	31	1	10	.34058
ITERATION NUMBER	32	1	42	.36508
ITERATION NUMBER	33	1	16	.30682
ITERATION NUMBER	34	1	29	.24786
ITERATION NUMBER	35	1	5	.23333
ITERATION NUMBER	36	1	14	.22816
ITERATION NUMBER	37	1	11	.20402
ITERATION NUMBER	38	1	39	.20000
ITERATION NUMBER	39	1	7	.20841
ITERATION NUMBER	48	1	6	1.79518

CLUSTER NUMBER 2

ITERATION NUMBER	40	24	26	.87500
ITERATION NUMBER	49	13	24	2.66667

CLUSTER NUMBER 3

ITERATION NUMBER	41	15	37	.88889
ITERATION NUMBER	42	15	33	.59091
ITERATION NUMBER	43	15	23	.45000
ITERATION NUMBER	44	15	27	.52381
ITERATION NUMBER	45	15	28	.35500
ITERATION NUMBER	46	15	49	.36071
ITERATION NUMBER	47	15	44	.33333

CLUSTER NUMBER 4

ITERATION NUMBER	22	41	53	.41667
ITERATION NUMBER	23	41	52	.27778
ITERATION NUMBER	24	41	51	.58333
ITERATION NUMBER	25	41	43	.44538
ITERATION NUMBER	26	41	45	.38587

Figure E12. Recap of Clustering with Alternate Objective Function

VITA

Carl Allen Singer was born on the third of May, 1946, on a westbound freight train heading towards Stettin, Poland. He came to the United States of America in 1949 and grew up in Cleveland, Ohio. After graduating from Charles F. Brush High School in Lyndhurst, Ohio, he attended Case Institute of Technology, receiving a B.S. in Organizational Sciences in 1968. He then was a graduate student in the Operations Research Department of Case Western Reserve University. After working as a systems analyst in industry, he attended the University of Michigan, receiving an M.S. in Industrial Engineering (Management Information Systems) in 1970. He then joined the U.S. Army serving as a military analyst in the Office of the Chief of Staff. He received a direct commission and is currently a Captain in the U.S. Army Reserve, Ordnance Corps. Upon leaving active duty, Mr. Singer worked as an Operations Research Analyst and Systems Analyst for the Defense Department. In 1973 he enrolled at Purdue University to complete his doctorate. Mr. Singer is currently a Management Scientist with Chase Econometrics in Bala Cynwyd, Pennsylvania.