

Computer-Aided Analysis and Design of Information Systems

J.F. Nunamaker Jr. and Benn R. Konsynski Jr.
University of Arizona
Thomas Ho and Carl Singer
Purdue University

This paper describes the use of computer-aided analysis for the design and development of an integrated financial management system by the Navy Material Command Support Activity (NMCSA). Computer-aided analysis consists of a set of procedures and computer programs specifically designed to aid in the process of applications software design, computer selection and performance evaluation. There are four major components: Problem Statement Language, Problem Statement Analyzer, Generator of Alternative Designs, and Performance Evaluator. The statement of requirements was written in ADS (Accurately Defined Systems) and analyzed by a Problem Statement Analyzer for ADS. The ADS problem definition was supplemented with additional information in order to create a complete problem definition. The analyzed problem statement was translated to the form necessary for use by the SODA (Systems Optimization and Design Algorithm) program for the generation of alternative specifications of program modules and logical database structures.

Key Words and Phrases: computer-aided analysis, information systems, logical system design, problem statement language, problem statement analyzer, physical system design, accurately defined systems, systems optimization and design algorithm

CR Categories: 2.44, 3.50, 4.33, 4.9, 8.1

Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

Authors' addresses: J.F. Nunamaker Jr. and B.R. Konsynski Jr., Computer Science Department and Management Information Systems Program, College of Business Administration, University of Arizona, Tucson, AZ 85721; T. Ho, and C. Singer, Department of Computer Science and Krannert School of Industrial Administration, Purdue University, Lafayette, IN 47907.

1. Introduction

The problems inherent in the increasing use of the computer for information systems applications have provided the motivation for the development of tools for automating the production of applications software. The current methods for building information systems contain numerous deficiencies [1], and so computer-aided analysis of user requirements is proposed as the first step toward automated systems building [2, 3]. This approach follows the work of Langefors [4] and Teichroew [5, 6].

In this paper we describe a number of software systems for computer-aided analysis and how they were used to aid the Navy Material Command Support Activity in the design of a large information system.

The activities performed by the systems for computer-aided analysis consisted of (1) procedures for stating processing requirements, (2) automatic analysis of processing requirements, (3) the design of program structure (i.e. determining how many modules must be generated and the size of each module), (4) the design of logical file structures and a logical database, (5) selection of hardware (central processing unit, core memory size, auxiliary memory, input/output configuration), (6) the allocation of files to storage devices, and (7) the optimal selection of blocking factors for each file.

The software package for computer-aided analysis, design, and construction consists of problem statement, problem analysis, generation, and evaluation of alternative designs for process and data organization, code generation and implementation of the selected design. An overview is shown in Figure 1.

1.1 Computer-Aided Problem Statement Techniques and Analyzers

In 1971, when work began on the design problem discussed later in the paper, no existing problem statement technique was determined to be adequate for the complete expression of user requirements relevant to all aspects of systems design and optimization. Hence, two techniques, Accurately Defined Systems (ADS) and SODA Statement Language (SSL), were used. Features of ADS and SSL were eventually incorporated into Problem Statement Language (PSL) version 3.0 developed by the ISDOS Project [7, 8] at the University of Michigan.

ADS is a product of the National Cash Register Company (NCR) and is described by Lynch [9] and NCR [10]. SODA Statement Language was developed by Nunamaker [1]. Combined, the two techniques

were used to specify the requirements for the Navy Integrated Financial Management System. The time frame for the selection of equipment and design of the information system for the Navy did not allow development of a problem statement analyzer that included the forms orientation of ADS and the time and volume features of SSL. The use of the two languages (ADS and SSL) was an operational convenience to take advantage of existing software.

The combined problem statement consists of information on data volumes and frequency of input and output items. The data description, processing requirements, operational requirements, and time and volume information are expressed in units specified by the problem definer.

ADS is forms-oriented and is used to obtain much of the basic problem definition. SSL is used to express system design parameters and performance requirements, e.g. I/O volumes and frequencies for system design and performance optimization.

1.2 Accurately Defined Systems Methodology

ADS consists of a set of forms and procedures for systematically recording the information that a systems analyst would gather during compilation of the user requirements for the information system to be implemented. The essential elements of an ADS requirements statement include descriptions of (1) inputs to the information system, (2) historical data stored by the information system, (3) outputs produced by the information system, and (4) actions required to produce these outputs and the conditions under which each action is performed. The ADS analyzer used on the Navy project was developed at the University of Michigan [11, 12] and extended at Purdue University.

Computer-aided analysis of an ADS statement performs a number of checks and prepares a series of summaries of the statement of user requirements. The simplest kind of check performed involves the validation of ADS source statements to uncover any violations of the syntax rules of ADS problem statement. Rules relating to naming conventions, numbering conventions, information linking, and the like are specified to guide the user during problem definition.

Summary reports produced by computer-aided analysis include a dictionary of all data element occurrences, indices to all data elements and processes, matrices indicating the data elements required by each process and the precedence relationships among data elements, and graphical displays of the ADS forms submitted for analysis. The description of the relationship between data elements, reports, and variables follows the work of Langefors [4]. The data element dictionary consists of an alphabetical list of the data elements defined in the ADS statement, the places of occurrence of each element, and the information source of each occurrence. The indices assign a unique number to each data element and process to identify row and

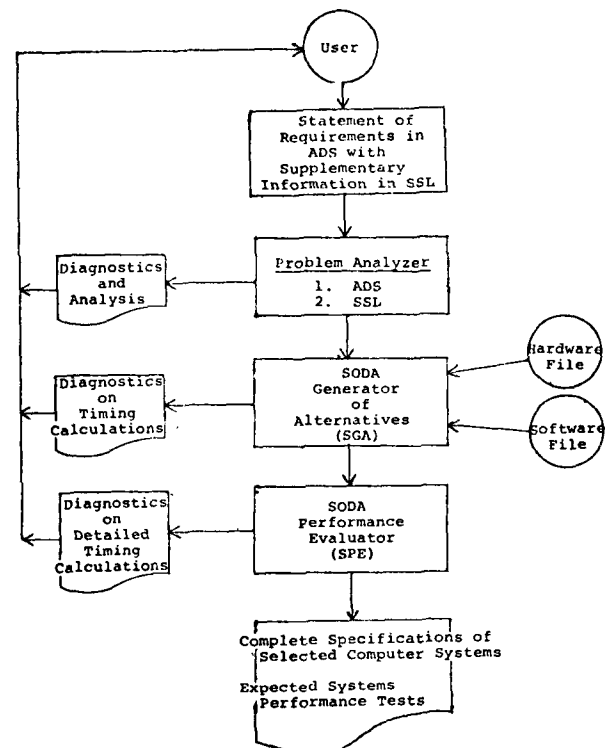
column positions in the matrices indicating incidence and precedence relationships. The incidence matrix uses process numbers as row indices and data element numbers as column indices to identify the data elements used in each computational process. The precedence matrix uses data element numbers as both row and column indices to indicate, for each data element, the data elements that must be computed before the first data element can be calculated. Complex checks of logical consistency and completeness indicate errors in data definition and linking of information sources. Finally, the graphical reports display the five kinds of ADS forms in the tabular manner as they would appear in manual use of ADS.

1.3 Systems Optimization and Design Algorithm

The SODA components used on the Navy project were: (1) SODA Statement Language (SSL), (2) SODA Statement Analyzer (SSA), (3) SODA Generator of Alternatives (SGA), and (4) SODA Performance Evaluator (SPE).

SSL consists of a set of forms for systematically gathering data on the volumes and frequencies of system inputs and outputs described in the ADS statement. SSL provides design parameters not available in the ADS description. The essential elements of an SSL statement include requirements data for: (1) inputs to the batch processing subsystem, (2) queries to the teleprocessing subsystem, and (3) reports produced by the information system.

Fig. 1. SODA (Systems Optimization and Design Algorithm).



SSA produces a number of networks which record the interrelationships of processes and data and passes the networks on to the SODA program concerned with the generation of alternative designs.

Each type of input and output is specified in terms of the data involved, the transformation needed to produce output from input, and stored data. Time and volume requirements are also stated. SSA analyzes the statement of the problem to determine whether the required output can be produced from the available inputs. The problem statement stored in machine-readable form is processed by SSA, which: (1) checks for consistency in the problem statement and checks syntax in accordance with SSL, i.e. verifies that the problem statement satisfies SSL rules and is consistent, unambiguous, and complete; (2) prepares summary analyses and error comments to aid the problem definer in correcting, modifying, and extending his problem statement; (3) prepares data to pass the problem statement on to SGA; and (4) prepares a number of matrices that express the interrelationship of processes and data.

SODA Generator of Alternatives (SGA) begins after the requirements have been stated, verified, and analyzed in SSA. SGA accepts, as input, the output of SSA and a statement of the available computing resources, hardware, and utility programs. The hardware and software file consists of data for the computer systems under consideration.

Fundamental to the SODA approach is the automatic generation of designs of program structure and file structure. This is the point at which SODA differs from other techniques such as SCERT [13] and CASE [14]. In order to use either SCERT or CASE, it is necessary that a system already be designed to obtain answers regarding feasibility. With SODA, the user has three options with respect to the generation of alternative system designs: (1) consider only SODA generated designs, (2) consider only designs generated manually, and (3) consider both sources of designs.

SGA takes the information from SSA, analyzes the alternative hardware and software information with respect to a specific design, and generates the specifications for the necessary CPU, core size, program structure, and data structure. SGA essentially computes the expected processing time required for alternative designs for each period of time [1].

SODA Performance Evaluator (SPE) involves examining feasible designs in an attempt to improve system performance. The optimization and performance evaluation phase searches for ways to improve the IPS (Information Processing System) design. SPE may return control to SGA to select another CPU or core size or to select another set of program modules and files.

SPE consists of a number of mathematical programming models and timing routines that are used to (1) optimize the blocking factors for all files, (2)

evaluate alternative designs, i.e. specify the number and type of auxiliary memory devices, (3) assign files to memory devices, and (4) generate an operating schedule for running program modules.

These reports include: (1) a list specifying which of the available computing resources will be used, i.e. which computer system is required to do the job, (2) a list of the program modules specifying the input, output, and computations to be performed in each, (3) a list of files to be maintained specifying their format and manner in which they will be stored, i.e. an assignment of files to memory devices, and (4) a statement of the sequence and manner in which the program modules must be run to accomplish all the requirements.

2. Description of ADS and SODA Software

The NMCSA project was mainly concerned with a macro-level evaluation of alternatives for equipment selection and logical design. The SODA components required for a micro evaluation of alternatives were not used in this project. This section describes the computer-aided information systems design software packages used on the Navy Material Command Support Activity project.

The ADS requirements statement begins with the definition of all system outputs. The definition continues with the identification of information that enters the system in order to describe inputs to the system. The requirements statement is then completed with the definition of historical data retained in the system for a period of time, together with specification of computations and accompanying logic that subsequently use the input and historical data to produce system outputs.

Linking of information elements among the various ADS definitions is accomplished in two ways. First, each data element is assigned a unique name that is always used whenever that element appears in an ADS definition. Second, each use of a data element in a report, history, or computation definition is linked back to its information source elsewhere in the ADS description. All data elements are chained from output to input and each output can ultimately be expressed in terms of inputs to the system. Chaining is accomplished by assigning page and line numbers to all ADS forms, so each use of a data element can be uniquely identified by the form, page, and line on which the element appears. The ADS example describes the requirements of a financial application for the United States Navy Material Command Support Activity.

The financial application produces an output report entitled DIRECT CITATION ORDERS BY ORDER NO. Two data elements appearing on the report are of particular interest: M0002 MO-ENDG-DT (line 2 of Figure 2(d)) and Z053A (line 7 of Figure 2(b)).

Fig. 3. History and Computation Definition Forms.

HISTORY DEFINITION for FEAS Application NAME OF GROUPING DIRECT CITATION
FUND STATUS ORDER # + QUANT G/L
 PREPARED BY _____ DATE _____ PAGE 189 OF 190

NCR

I FIELDS THAT IDENTIFY THIS HISTORY: D1007 ACQYS, A0001 C0008, D0018 REQ1, RE078 D0017
 II WHAT IS THE EXPECTED VOLUME? AV. MAX.
 III DESCRIBE EACH FIELD OF THE GROUP

Line No.	NAME	FORM	TYPE	Page	Line	SOURCE
1	D0019 DOC-TY		A	3		
2	D0017 DOC-NR		A	13		
3	P0065 PROC-ACT-CD		A	1		
4	RE001 RE-DOC-NR		A	13		
5						
6	D1028 DRCT-D-AMT-ER-OTH-SAVCS		N	13		
7	AC045 ACCT-CL-CHRG		A	80		
8	A0001 ADMING-ORCC		A	2		
9	C0008 CNTRY-WORLD		A	2		
10	D1007 DRCT-CIT-EDG-LN-INDR		A	1		
11						
12						
13	RE078 RRRG-FIN-MGR-C		N	3		
14						
15						
16						
17						
18						
19						
20	D1031 DRCT-CIT-AUTHN-RCVD		N	13		
21	D3211 DRCT-CIT-UNCMD-UNOBLD		N	13		
22	D7810 DRCT-CIT-OTSTN-INITS		N	13		
23	D3231 DRCT-CIT-OTSTN-CMPT		N	13		
24	D3231 DRCT-CIT-OTSTN-OBLN		N	13		
25	D3130 DRCT-CIT-CONTR-NBK		N	13		
26	D1060 DRCT-CIT-ED-DSDB		N	13		
27	D7811 DRCT-CIT-INITS-LIQD		N	13		
28	D7812 DRCT-CIT-INITS-CONTR		N	13		
29	NAD12 NAVILCO-RAN-CON-NR		A	13		
30	RE002 GENR-CODE		A	2		
31						

(a)

FEAS FUND 2053A **NCR**
 GROSS-OBLIGATION- D3231 DRCT-CIT-OTSTN-OBLN
 NET-OBLIGATION- D3231 DRCT-CIT-OTSTN-OBLN
 PERIOD: 1 2 3 4 5 6 7 8 9 10 11 12

Line	Form	Type	Page	Line	Source
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

(b)

ADS possesses obvious advantages over the traditional narrative requirements statement technique. Narrative statements are ambiguous and often incomplete, while ADS provides a standardized and systematic approach to system definition. ADS is both exact and precise while remaining hardware independent. ADS promotes effective communication among systems personnel by imposing a discipline that enables the efficient use of human and machine resources. The ADS technique enables checking for accuracy, consistency, and completeness of the requirements statement.

To determine the computer resource demands of the information system for which the design process was performed, additional data supplementary to the ADS statement was necessary. This need required the following data for each input and report described in the ADS statement: (1) ADS page number, (2) frequency of occurrence, (3) volume, and (4) a brief description.

This information was represented on SSL forms, along with query profile information which included frequency, size, source, and file reference information.

2.1 ADS Analyzer

The first module of the Problem Statement Analyzer for ADS (PSA/ADS) performs source deck validation, lists the input cards, creates a file containing all valid card images, and constructs a dictionary table to be used by other PSA/ADS modules. Source deck validation checks compliance with ADS syntax rules and detects errors that include (1) specification of an illegal form type, i.e. not Report, Input, History, Computation, or Logic, (2) improper form format, (3) an illegal data element name, and (4) invalid page or line numbering.

For each valid ADS entry, the dictionary table records:

1. The place of occurrence such as form type, page number, and line number.
2. The data element name.
3. The information source that specifies form type, page number, and line number.

The dictionary is sorted, in ascending order, according to the data element name and place of occurrence.

The second module of PSA/ADS prints the data element dictionary and constructs a symbol table containing data element names. From the sorted dictionary table, the second module lists the data elements in alphabetical order and provides the place(s) of occurrence and information source(s) for each data element.

An example of a dictionary entry for the data element D3231 DRCT-CIT-OTSTN-OBLN appears in Figure 4(a). The two occurrences of D3231 noted in the previous ADS example are indicated. During dictionary printing, the second module performs logical checks to detect the following errors and warnings: (1) no source of information, (2) no update for history data, (3) data element not used, (4) redundant inputs, (5) redundant histories, (6) a data element defined as both an input and the result of a computation, and (7) invalid back reference.

Also, the second module assigns a unique number to each data element and prints an alphabetical list of the data elements used in the ADS statement. Then the sorted dictionary table is again sorted, in ascending order, according to form type (report, input, computation, logic, history), page number, line number, and entry type.

The third module of PSA/ADS creates a file containing records of the computational processes defined in the ADS statement, prints a list of the computational processes, and generates matrices displaying the incidence and precedence relationships among the data elements and processes defined in the ADS statement. The third module reads entries from the twice-sorted dictionary table and, for each computation entry, the module writes one or more (depending on the number of operands in the computation) records on the file

of computational processes. Each record has the symbol table pointer of the data element that appears as the result of the computation entry and the symbol table pointer of the data element that appears as an operand of the computation for which the first pointer identifies the result.

The third module generates the incidence matrix indicating the data elements that serve as results and as operands for each process. These relationships are easily derived from the result-operand pairs in the sorted process file. An example of the incidence relation for data element Z053A and the relevant subset of the incidence matrix appears in Figure 4(b). Also, an alphabetical list of the processes is generated, with the operands of each process listed alphabetically.

Finally, the twice-sorted process file is used to generate the precedence matrix indicating the direct precedents of each process. Data element I is said to be a precedent of data element J if I must be computed before J can be computed. A *direct* precedent of J is a precedent of J that is not also a precedent of any other precedents of J.

The fourth and final module reads the sorted card image file and prints the input in a tabular format similar to that of the ADS forms developed by NCR. Examples of the ADS forms from the previous ADS example appear in Figure 5.

2.2 Process Generation and Program Module Specifications from ADS Definition

The ADS problem statement contains the basic information required to generate program module specifications from processes that may be grouped into program modules to eliminate unnecessary transport of data from history files to program modules. For example, if it is determined that two processes require the same inputs and occur in the same processing cycle, e.g. daily, then the two processes become candidates for grouping into a single program module.

SODA Generator of Alternatives (SGA) performs process generation by compiling four comprehensive summaries for each ADS-described report: (1) an input summary, (2) a history input summary, (3) a computation summary, and (4) a history output summary.

Fig. 4. PSA/ADS dictionary and incidence relation entries.

(a)

DICTIONARY		ADS / III				RELEASE 1				PAGE 79	
VARIABLE NAME	OCCURS ON:				WITH BACK REFERENCE:				NOTE		
	FORM	PAGE	LINE	USE	FORM	PAGE	LINE	ASGN			
D3231 DRCT-CIT-OTSTN-OBLN	C	925	1	LT.		0	0		(N/A)		
	C	925	2	LT.		0	0		(N/A)		
	C	925	3	LT.		0	0		(N/A)		
	C	925	4	LT.		0	0		(N/A)		
	C	925	5	LT.		0	0		(N/A)		
	C	53	1	RT.	H	169	24	PS	+		
	C	122	90	RT.	H	169	24	PS	+		
	C	122	106	RT.	H	169	24	PS	+		
	C	126	2	RT.	H	169	24	PS	+		
	C	126	6	RT.	H	169	24	PS	+		
	C	925	1	RT.	H	169	24	PS	+		
	C	925	2	RT.	H	169	24	PS	+		
	C	925	3	RT.	H	169	24	PS	+		
	C	925	4	RT.	H	169	24	PS	+		
	C	925	5	RT.	H	169	24	PS	+		
	H	169	24		C	925	1	PS	+		

(b)

INCIDENCE RELATIONS		ADS / III				RELEASE 1				PAGE 378	
PROC	VAR	RESULTANT VARIABLE	VAR	INPUT VARIABLE(S)							
488	632	Z053A	218	D1060	DRCT-CIT-PD-DSBD						
			219	D2130	DRCT-CIT-CONTR-HBK						
			222	D3231	DRCT-CIT-OTSTN-OBLN						

INCIDENCE MATRIX		ADS / III				RELEASE 1							
P / VARIABLE NUMBERS		205	210	215	220	225	230	235	240	245	250	255	260
R	487
O	488
C	489
	490
	491
	492
	493
	494
	495

Fig. 5(a). PSA/ADS Report Definition Form.

REPORT FORMS		ADS / III	RELEASE 1	PAGE 415			
REPORT	DEFINITION FOR	NAME OF REPORT					
IPMS FUND STATUS REPORTING		DIRECT CITATION ORDERS BY ORDER NO					
	APPLICATION	PREPARED	PAGE 53 OF 192				
		DATE 12/4/72					
III CROSS REFERENCE TO FIELD NAMES (WHEN FIELD HEADING INCLUDES MULTIPLE CODES, LIST ALL CODES ON TABLE BELOW).							
			SOURCE				
	L		H	H	P	L	
	I	N	E	C	A	I	
	B	A	O	X	I	G	
	E	M	O	O	E	E	
	NAME						
B →	1	AD001 ADNING-OPC-C	0		H	169	8
	2	RO002 HO-INDG-DT	0		I	201	2
	3	CO017 DOC-NR	0		H	169	2
	4	D1031 DRCT-CIT-AUTHN-RCVD	0		H	169	20
	5	D9810 DRCT-CIT-OTSTM-INIT5	0		H	169	22
A →	6	D3221 DRCT-CIT-OTSTM-CRTRT	0		H	169	23
	7	Z053A	0		I	53	1
	8	PA005 PG-NR	0		I	201	1
X CHECK COLUMN TO INDICATE THAT A FIELD DOES NOT PRINT EXCEPT WHEN THE VALUE OF THE FIELD CHANGES, OR AT THE TOP OF A NEW PAGE							
IV MAJOR LEVEL 1 AD001 ADNING-OPC-C MINOR LEVEL 2 D0017 DOC-NR							

Fig. 5(c). PSA/ADS History Definition Form.

HISTORY FORMS		ADS / III	RELEASE 1	PAGE 853					
HISTORY	DEFINITION FOR	NAME OF HISTORY							
IPMS FUND STATUS		DIRECT CITATION ORDER # & QUASI C/L							
	APPLICATION	PREPARED BY	PAGE 169 OF 250						
		DATE 2/16/73							
SORT KEY(S)									
	MAJOR LEVEL 1	D0007 DRCT-CIT-PDG-LN-INDR							
	LEVEL 2	AC045 ACCT-CL-CHARGS							
	LEVEL 3	AD001 ADNING-OPC-C							
	LEVEL 4	CO008 CTRY-WORLD							
	LEVEL 5	D0018 DOC-TY							
	LEVEL 6	RE001 RE-DOC-NR							
	LEVEL 7	RE078 RORG-FIN-RCR-C							
	MINOR LEVEL 8	D0017 DOC-NR							
			SOURCE						
	L		H	H	P	L			
	I	N	E	C	A	I			
	B	A	O	X	I	G			
	E	M	O	O	E	E			
	NAME								
			HOW LONG IS DATA RETAINED ?						
	1	D0018 DOC-TY	101	0	A	3	I	310	1
	2	D0017 DOC-NR	101	0	A	15	I	310	2
	3	PR065 PRCC-ACT-CD	0	0	A	1	C	157	20
	4	RE001 RE-DOC-NR	101	0	A	13	C	157	21
	6	D1031 DRCT-CIT-AUTHN-RCVD	0	0	A	13	C	157	21
	7	AC045 ACCT-CL-CHARGS	0	0	A	60	I	310	5
	8	AD001 ADNING-OPC-C	0	0	A	2	I	310	9
	9	CO008 CTRY-WORLD	0	0	A	2	I	310	32
	10	D1031 DRCT-CIT-AUTHN-RCVD	0	0	A	1	C	157	16
	13	RE078 RORG-FIN-RCR-C	0	0	A	3	I	310	29
	20	D1031 DRCT-CIT-AUTHN-RCVD	0	0	A	13	C	900	1
	21	D1211 DRCT-CIT-OTSTM-INIT5	0	0	A	13	C	915	1
	22	D9810 DRCT-CIT-OTSTM-INIT5	0	0	A	13	C	930	1
	23	D3221 DRCT-CIT-OTSTM-CRTRT	0	0	A	13	C	920	1
C →	24	D1231 DRCT-CIT-OTSTM-CBLN	0	0	A	13	C	925	1
	25	D1130 DRCT-CIT-OTSTM-NBR	0	0	A	13	C	910	1
	26	D1060 DRCT-CIT-PD-DSBD	0	0	A	13	C	905	1
	27	D9811 DRCT-CIT-INIT5-LIQD	0	0	A	13	C	935	1
	28	D9812 DRCT-CIT-INIT5-COSTRA	0	0	A	13	C	940	1
	29	NA012 NAVILCO-RGN-COF-NR	0	0	A	13	C	157	19
	30	CE002 CENR-CODE	0	0	A	3	C	157	18
NOTE 1 - X IF FIELD MAINTENANCE HAS BEEN PROVIDED									

Since the source of each report item is specified in the ADS statement, all sources that are either input items or history items are included in the input and history input summaries, respectively. For report items whose sources are computation items, the input and history input items that are used as operand factors in the computations are placed into the input and history input summaries, since the sources of all computation operand factors are specified. Also, the computations required to produce the report items are placed into the computation summary. Finally, the history output summary is compiled by listing all history items whose sources are items listed in either the input, history input or computation summaries. Therefore, the history output summary indicates those history items that might be updated by the elementary module being specified.

After generating a process for each ADS-specified report, SGA searches for candidates for program module grouping in two ways. First, if some process requires history inputs (or inputs) either identical to or forming a subset of the history inputs (or inputs) required by another process, the two processes are identified as candidates for grouping. Second, if a predominant (approximately 90 percent) subset of the history inputs (or inputs) of some process is identical to a predominant subset of some other process, the two processes are identified as candidates for grouping. Finally, if the two candidates for grouping occur in the same processing cycle, grouping into a single program module is recommended by SGA.

SGA generates one complete feasible set of program module specifications. Phase 1 of SGA is a deterministic algorithm and traces the sources of data items that appear on reports in order to generate elementary processes. Phase 2 of SGA is a heuristic algorithm and searches for elementary processes to be grouped into program modules according to the specified decision rules.

2.3 SODA Macro Simulation

In most cases, it is not appropriate to perform a micro-level performance evaluation, when many system design factors have not been specified. Therefore, a macro simulation can be useful as an aid in the specification of the complete systems design.

The SODA macro simulation model was used to evaluate the performance of the alternative computer systems under various simulated workload conditions.

The macro simulation reports serve as an aid in design and performance evaluation of alternative design factors. The macro simulation was used to test the sensitivity of the performance considerations on various hardware and software design parameters. Among the system factors simulated at the macro level were (1) the number and capabilities of various devices, (2) specification of system software organization, (3) distribution of teleprocessing arrivals during

various periods in the day, (4) query profiles, (5) scheduling of I/O devices to channels, (6) resource queue characteristics and (7) batch scheduling and job profiles.

The macro simulation was used to isolate potential problem areas and determine bottlenecks. This analysis was used to evaluate resource utilization, system throughput, batch turnaround, query response time, overall system behavior, etc.

The SODA Macro Simulator is an event oriented simulator not unlike that presented by MacDougall [15]. The major structures maintained by the model during simulation are a job status table, a list of events, various queues, resource status tables, and job and resource utilization statistics tables (Figure 6). Examples of the SODA Macro Simulator output are presented in Figure 7.

The manner in which the model simulates job interaction with the database was of particular interest, owing to the sensitivity of the data management system with respect to particular applications. The procedure differs from that used for ordinary I/O requests to and from disk.

Database accesses were classified by function and complexity. The primary functions were updating and retrieval. In addition, each function was further subdivided into various categories of access complexity. Complexity was characterized by such factors as the number of keys specified in a retrieval request and the number of indices that must be searched in order to find the desired database record. Overall, database performance was determined by the size of the database and by the type of physical storage structures employed to represent logical data structures.

The characteristics, e.g. size and physical storage structure, of the database were specified initially in the simulation model. Each type of database request, e.g. update or retrieval, was characterized by the various levels of complexity permissible. Based on this data, the model generated estimates of the processing time of each type and of the complexity of database request made by the jobs in the simulated mix.

3. Experience With a Large "Real World" Application

The work specifically reported in this paper was done for the United States Navy Material Command Support Activity (NMCSA). The statement of requirements for a financial management system was expressed in ADS by a large accounting firm. The ADS analyzer was used to check the ADS statement of requirements for completeness, consistency, and logical accuracy.

The ADS analyzer produced information and reports that were used by the SODA Statement Analyzer. SODA was then used to generate preliminary designs of program structure and logical database structure

for the batch application part of the system, and to recommend a computer system for the entire financial management system.

3.1 Organizational Environment

NMCSA financial management personnel are currently using ADS to state the requirements of a large information system. This Integrated Financial Management System (IFMS) is a large-scale design and implementation effort for more effective financial management, particularly procurement accounting, within the agency. The systems design effort commenced in May 1971, and is expected to continue for four to five years at a cost of twelve million dollars.

The purpose of the system is to centralize information flow and satisfy the information requirements of a variety of decision makers. The principal communicators in the system are the requiring manager or financial manager, participating manager, and procurement manager. The decision makers observe a hierarchy of authority delegation and reverse order for accountability.

Initial authority is received by the Comptroller of the Navy from the office of the Secretary of the Navy. In the case of a fund allocation, the Comptroller issues a program fund allocation with program values at the line item budget level. The funding then is routed through the responsible office to the administering office. Here, funding proceeds to the requiring/financial manager level where responsibility lies for execution, modification, and delivery of the system within cost and schedule objectives. At this level, monthly approval is considered for interim actions. The requiring/financial manager issues project directives which delegate specific authority to the participating manager level operatives. Figure 8 describes the organizational hierarchy (left) and the corresponding budget level (right).

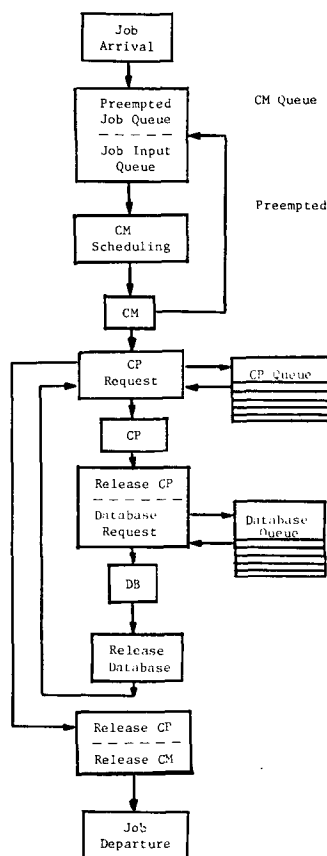
Fund status inquiries are used to maintain fund control at the project directive line item level. Contract status inquiries are used to control delivery dates and funds for contract items.

Fund status inquiries are characterized by funded project directives and planning project directives. The planning project directive allows for procurement of long lead-time items. The funded project directive serves a shorter cycle time and therefore has a high activity level of inquiry and update.

Contract status inquiries are primarily identified with the procurement manager. Once a funded project directive is established, the information updates revolve around completion status of tasks in line with procurement. The procurement manager updates the contract status, which is observed by the participating manager and the requiring/financial manager.

The basic objective is to significantly improve the timeliness of accounting/financial management information reported by the agency's accounting system

Fig. 6. Job processing steps in SODA Macro Simulator.



to reduce input, processing, and reporting time. This facilitates elimination of most memorandum record systems.

The first module of the system to be implemented is the Procurement Accounting and Reporting Subsystem (PARS). This subsystem is intended to normalize information flow concerning procurement accounting. This subsystem includes: (1) funds accounting, (2) planning documents, (3) contract accounting, (4) fiscal and program status, and (5) other items related to appropriations.

3.2 Behavioral Experience With ADS

The first objective of the introduction of ADS into any environment is gaining user acceptance. ADS represents deviation from the established practices and initial resistance to change often occurs. As a result, many questions are raised regarding ADS and its impact upon the organization.

In response to this initial user reaction, an ADS training program is advisable. However, ADS is so simple and straightforward that less than one day of intensive training is all that is necessary to adequately prepare individuals in its use. Then further training is required only to deal with the specific restrictions imposed upon the use of ADS by the ADS Analyzer

Fig. 7. Examples of SODA Macro Simulator Output.

SUMMARY FOR HOUR (14-15)

RESOURCE USAGE/QUEUEING STATISTICS

RESOURCE	USE-TIME	USE-CT	WAIT-TIME	WAIT-MAX	Q-LENGTH	QL-MAX	TIME-QLN	NO. WAITS	AVG Q-LEN
CM	0.000	3	0.000	0.000	0	0	0.000	0	0.00
CTLBLK	9218.977	3	0.000	0.000	0	0	0.000	0	0.00
CPU	2106.012	8573	1263.608	1.107	0	3	1263.608	4343	.35
CHAN1	1888.370	3126	908.948	2.139	0	2	908.948	980	.25
CHAN2	1887.526	3182	872.237	2.245	1	2	871.829	993	.24
MPLXR	288.842	1132	.932	.932	0	1	.932	1	.00

HOURLY SUMMARY

95216 LINES PRINTED
 TP JOBS HANDLED 1130 AVG RESPONSE TIME 5.02 SEC
 JOBS STARTED 1132, JOBS COMPLETED 1132
 JOB COMPLETION RATE 1132.000 JOBS/HR.

SYSTEM LOG

TIME	MESSAGE
15. 0. 0	JOB 1 COMPLETED OUTPUT 191 CHRS TIME IN SYSTEM 4.06 SEC.
15. 0. 2	TP JOB REQUEST. JOB PROFILE FUND STATUS QUERIES REQUEST FROM PARM ASSIGNED JOB NR 1 CM OK.
15. 0. 5	JOB 3 COMPLETED OUTPUT 169 CHRS TIME IN SYSTEM 1719.10 SEC.
15. 0. 5	B-JOB 3 COMPLETED WITH 4000 LINES PRINTED. TIME IN SYSTEM 1719.10 SEC.
15. 0. 6	JOB 2 COMPLETED OUTPUT 96 CHRS TIME IN SYSTEM. 7.69 SEC.
15. 0. 7	JOB 1 COMPLETED OUTPUT 137 CHRS TIME IN SYSTEM 5.16 SEC.
15. 0. 7	TP JOB REQUEST. JOB PROFILE FUND CERTIFICATION REQUEST FROM PARM ASSIGNED JOB NR 1 CM 26K.
15. 0.12	JOB 1 COMPLETED OUTPUT 152 CHRS TIME IN SYSTEM 4.57 SEC.
15. 0.12	TP JOB REQUEST. JOB PROFILE FUND CERTIFICATION REQUEST FROM PROM ASSIGNED JOB NR 1 CM 24K.
15. 0.17	JOB 1 COMPLETED OUTPUT 81 CHRS TIME IN SYSTEM 4.13 SEC.
15. 0.17	TP JOB REQUEST. JOB PROFILE FUND CERTIFICATION REQUEST FROM PROM ASSIGNED JOB NR 1 CM 36K.
15. 0.21	JOB 1 COMPLETED OUTPUT 119 CHRS TIME IN SYSTEM 4.49 SEC.
15. 0.22	TP JOB REQUEST.
15. 0.30	BATCH JOB SUBMITTED.

RESOURCE USAGE/QUEUEING STATISTICS

RESOURCE	USE-TIME	USE-CT	WAIT-TIME	WAIT-MAX	Q-LENGTH	QL-MAX	TIME-QLN	NO. WAITS	AVG-Q-LEN
CM	28535.314	1	4240.243	25.071	0	5	4240.243	706	.15
CTLBLK	77779.241	1	4784.039	14.280	0	6	4784.039	1148	.17
CPU	19629.986	54978	28152.518	3.391	0	4	26152.518	28960	.91
CHAN1	12286.307	20312	2850.203	2.383	0	2	2850.203	3475	.10
CHAN2	12218.062	20344	2770.283	2.255	0	2	2770.283	3453	.10
MPLXR	.1837.264	7161	34.618	18.335	0	2	34.618	9	.00

SIMULATION SUMMARY

TIME 28800.045 SECONDS
 754637 LINES PRINTED
 TP JOBS HANDLED 6996 AVG RESPONSE TIME 8.89 SEC
 BATCH JOBS COMPLETED 11 AVG TURNAROUND 1180.73 SEC
 CPU BATCH 810.40 TP 12305.12
 JOBS STARTED 7163, JOBS COMPLETED 7161
 JOB COMPLETION RATE 895.124 JOBS/HR.

WORK LOAD SUMMARY

	NO. JOBS STARTED	NO. JOBS COMPLETED	JOB COMPLETION RATE	CPU UTILIZATION (SEC)	AVG CM/JOB (K-BYTES)	AVG (SEC) RESPONSE (TURNAROUND)	PERCENT CHANNEL UTILIZAT.	PERCENT CPU UTILIZATION	PERCENT TIME IDLE
TP	6996	6996	874.50	12305.116	25.4	8.89	71.34		
SCN	12	11	1.38	810.400	32.7	1180.73	20.47		
INTERFACE	0	0	0.00	0.000	0.0	0.00	0.00		
DEVELOP	154	154	19.25	6468.000	89.9	68.80	8.19		
SYS MAINT	0	0	0.00	0.000	0.0	0.00	0.00		
SHIFT 1	7163	7161	895.12	19629.986	26.8	11.98	100.00	68.16	92

END OF RUN

software. For example, the Analyzer restricts the length of data element names to forty characters.

The use of a form-oriented procedure such as ADS still requires a significant investment of time and effort to realize the return of a complete and consistent logical systems design. Still, a number of users with ADS experience agree that ADS has saved them considerable time during the specification of logical system design, because of the capability of the ADS Analyzer to provide feedback information to the user. The user should be able to do a better job of specifying his requirements, because he receives feedback much sooner in the system design cycle utilizing computer analysis of ADS. Ordinarily, in a completely manual narrative system, ambiguities and omissions in the logical system description are not discovered until physical design or even coding is well under way. By then, many aspects of the system design have been specified, so that resolution of difficulties may be impossible.

Physical system design is not the responsibility of the ADS user. Completion of the ADS logical description is followed by the physical system design process that provides the specifications for programming.

3.3 Performance of ADS

One physical year, consisting of 12 man-years of effort by problem definers, was required to generate a consistent statement of requirements. Though no actual logs were kept, approximately five iterations of each form were required before the final statement was found to be complete and consistent.

However, even though the problem statement was complete and consistent, this did not guarantee that the problem was defined correctly, i.e. the wrong requirements may have been stated properly.

Experience has demonstrated that ADS is adequate for specification of the logical system. However, an ADS description does not provide sufficient information for generation of physical system design. Data on system performance requirements was collected to supplement the ADS description in SODA Statement Language. Relevant data include specification of the frequency of occurrence of each ADS-described input and report and of the volume of each input, report, and history.

Other needed enhancements to computer-aided ADS include facilities for describing data structures and lookup tables and for decision tables expressing processing logic and input validation rules. Finally, additional software for generating report layouts and program test data would add significantly to computer aided ADS capabilities. Many of these enhancements are included in the SODA Statement Analyzer.

3.4 Program Module Grouping for the Navy Example

A large number of alternative designs were evaluated in determination of a hardware software configuration. The hardware options were manually reduced

Fig. 8. Organizational hierarchy and corresponding budget level.

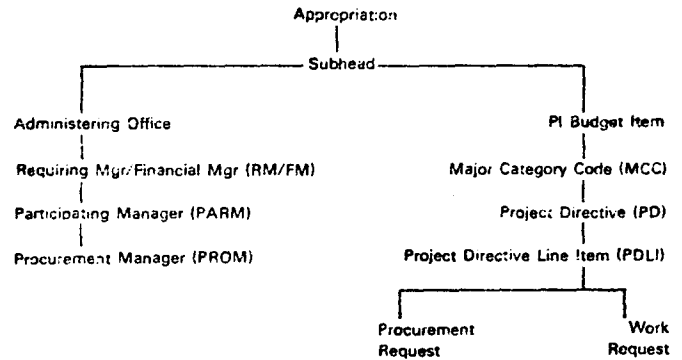
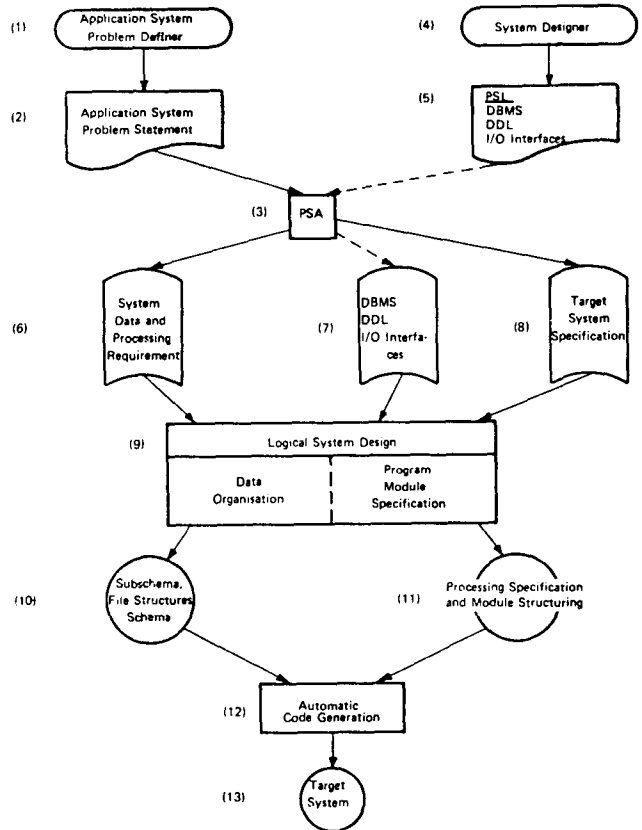


Fig. 9. Overview of the information systems design and construction process.



to three CPU classes. A feasibility study revealed the importance of maintaining a high degree of compatibility with another Navy 360/50 system, and due to interface problems the file design was constrained to IMS files. The alternatives to be evaluated were reduced to an IBM 360/50, 65 and 370/155 with a wide range of memory choices. This wide range of memory options led to the evaluation of 50 hardware configurations within the framework of the SODA Macro Simulator. The program designs were generated as a result of partitioning the application areas. This was done in an iterative fashion consistent with the generation of

the more than 50 alternative configurations. Software designs in general were generated.

For the information system under consideration SGA generated 62 program modules to produce the 79 ADS-specified reports. The PARS Module of IFMS for the Navy consisted of 647 processes and 791 unique data items. The size of the incidence matrix was 647×791 and the data element precedence matrix was 791×791 .

For each program module, the module size and the number of arithmetic operations are derived from the quantity and complexity (e.g. alternative logic paths) of computations in the summary produced by SGA. Volume and size of history records input are derived from the history input summary produced by SGA. SGA performs summary analyses on all ADS-specified inputs required to produce each history item. User-provided data on input requirements was then used to derive the volume of the history item under scrutiny. The size of the history item is provided in the ADS description. Finally, twenty record groups were generated with each group containing history items that are used together in a fashion that implies logical connectivity. Each group of records forms the basis for defining history file structures. An overview of the program module specifications for fiscal reporting tasks is presented in Table I.

Note that process grouping into modules and history record grouping into files were performed in a manner that spreads the workload equally among the modules to the greatest extent possible. Workload sharing is made possible by minimizing the variance in the number of computations in each module and by minimizing the variance in the number of records in each file group-

ing. The computer-aided designs "looked like" manually generated designs with respect to program structure.

The most prominent difference between manual and computer-aided generated designs is that the automated designs allow program structures to cut across functional boundaries unless constrained. This creates organizational problems with respect to maintenance and updating.

4. Current Developments

The work described in this paper used techniques and procedures developed prior to 1969. An overview of computer-aided procedures for information systems design and construction being developed at the University of Arizona [8] is illustrated in Figure 9. The application system problem definer (1) states the system requirements in a Problem Statement Language (PSL[5, 7] (2)). The problem statement is analyzed by the Problem Statement Analyzer (PSA[6, 7] (3)) which determines the consistency and completeness of the problem statement and gives the problem definer feedback for modification of the problem statement. The PSL/PSA used in this effort was developed under the ISDOS Project at the University of Michigan. The language supports a nonprocedural description of system requirements and facilitates a top-down approach to requirement statement.

The process of defining the system requirements is handled in an iterative fashion using PSA. PSA serves as a tool in the problem statement process by continually giving feedback to the problem definer and maintaining a database containing the problem statement. PSA

Table I. Batch Program Module Workload Summary.

Application/program ID	Task type	Freq./month	Memory required (K bytes)	Language	File ID	Medium code	Avg. record length (char.)	Record volume	Avg. output length (lines)
Fiscal reporting									
1. Program budget status	Print	1	150	Cobol	H1 H4 H5 H11	Disk	861	2200	340
2. Appn. status by FY and acct.	Print	1	50	Cobol	H1 H5 H11	Disk	243	2200	23000
3. Report on reimbursables	Print	1	35	Cobol	H1 H4	Disk	232	225	24000
4. Report on obligations	Print	1	100	Cobol	H1 H4 H5	Disk	437	2000	1000
5. Analysis of appropriations and fund balances	Print	1/year June	50	Cobol	H1 H4 H5 H11	Disk	476	2200	800
6. Line item report	Print	1	35	Cobol	H1 H4	Disk	232	225	24000
7. Summary line item report	Print	1	50	Cobol	H1 H5 H11	Disk	263	2200	4700
8. Procurement program progress report	Print	1	35	Cobol	H1 H5 H11	Disk	268	2200	4000
9. Worksheet	Print	2/year June, Dec.	25	Cobol	H1 H5	Disk	141	2000	4000

produces approximately 26 reports which describe process and data relationships and provide documentation as well as analysis of the problem statement. Once a complete and consistent problem statement is obtained, the PSA database contains sufficient information to proceed with the Logical System Design phase (3): Data Organization and Program Module Specification.

It was decided that for purposes of standardization of procedure and portability, a subset of the data structuring techniques of the Data Base Task Group (DBTG) Report [16] would be used as the basis for data organizations generated by the system. This allowed a formal basis for file structuring while permitting considerable flexibility concerning the particulars of file design. Thus, the Data Base Management System, Data Definition Language Analyzer, and I/O interface requirements were stated in the Problem Statement Language (5), and a requirements database was produced (7). The dotted line indicates that this process was a one-time procedure, since the requirements database will be the same for each system generation.

Program modules are derived according to target system specifications (hardware and system software) and operational (manual intervention) factors. Process characteristics and attributes of interprocess relationships are analyzed, and graph theory and multidimensional analysis techniques are used to derive program module specifications.

The process of Logical Data Organization (9) can broadly be described as the definition of logical storage structures and their associated data relationships. It is the purpose of this phase to determine effective subschema for the various processing modules, the number and contents of the system databases, and an effective schema for each database.

The automatic code generation and physical design (12) phase includes representation of data manipulation specifications in the form of functional primitives, determination of data structures required for processing of control sequences within the framework of the program module, generation of "bookkeeping" operations for program module invocation, termination and communication, and generation of target system code (13) for implementation.

The design system maintains a database of information on system requirements. As design decisions are made the database is adjusted accordingly. The design models interface with the database through a central control processor which handles design questions through the use of a query processor.

The design models include simulators, optimizers, and statistical and data summary analyzers. These models form the basis of computer-aided physical system design. Output analysis programs initiated by the control mechanism display information for the user or update the database as necessary. In addition

to the logical design and code generation models, some of the physical system design models in present use are database load and edit programs, direct access storage device time generator, file organization processor, file assignment, channel subsystem, CPU-channel network queueing, line topology organizer, concentrator location, and blocking factor determination.

Acknowledgments. We would like to thank Robert Taylor and William O'Brian of the Navy Material Command Support Activity for their cooperation and assistance in the project. In addition we wish to thank an anonymous referee for his constructive comments.

Received October 1973; revised January 1975

References

1. Nunamaker, J. A methodology for the design and optimization of information processing systems. Proc. 1971 AFIPS SJCC, Vol. 38, AFIPS Press, Montvale, N.J., pp. 283-294; reprinted in *System Analysis Techniques*, J.D. Cougar and R. Knapp, Eds., Wiley, New York, 1974, pp. 359-376.
2. Nunamaker, J., and Whinston, A. A macro approach to the planning and cost allocation of computer services. *Management Informatics* 2, 4 (Aug. 1973), 177-189.
3. Nunamaker, J., Ho, T., Konsynski, B., and Singer, C. SODA: Systems optimization and design algorithm—an aid in the selection of computer systems and for the structure of computer program modules and data base. In *Information Systems and Organizational Structure*, E. Grochla and N. Szyperki, Eds., Walter de Gruyter Pub., Berlin and New York, 1975, pp. 127-150.
4. Langefors, B. Some Approaches to the Theory of Information Systems. *BIT* 3, (1963), 229-254; reprinted in *System Analysis Techniques*, J.D. Cougar and R. Knapp, Eds., Wiley, New York, 1974, pp. 292-309.
5. Teichroew, D. Problem statement languages in MIS. Proc. Int. Symp. of BIFOA (Management Information Systems—A Challenge to Scientific Research), Cologne, July 1970, pp. 253-270; reprinted in *System Analysis Techniques*, J.D. Cougar and R. Knapp, Eds., Wiley, New York, 1974, pp. 310-327.
6. Teichroew, D. Problem statement analysis: Requirements for the problem statement analysis (PSA). In *Systems Analysis Techniques*, J.D. Cougar and R. Knapp, Eds., Wiley, New York, 1974, pp. 336-358.
7. Teichroew, D., Hershey, E., Bastarache, M. An introduction to PSL/PSA. ISDOS Working Paper No. 86, Dep. Indus. and Operations Eng., U. of Michigan, Ann Arbor, Mich., March 1974.
8. Nunamaker, J., and Konsynski, B. From problem statement to automatic code generation. *Systemeering* 75, Studentlitteratur, Lund, Sweden, 1975, pp. 215-240.
9. Lynch, H. ADS: A technique in system documentation. *Database*, 1, 1 (Spring 1969), 6-18.
10. A Study Guide for Accurately Defined Systems. National Cash Register Co., Dayton, Ohio, 1968.
11. Merten, A. and Teichroew, D. The impact of problem statement languages in software evaluation. Proc. 1972 AFIPS FJCC, Vol. 41, Pt. II, AFIPS Press, Montvale, N.J., pp. 849-858.
12. Thall, R. A manual for PSA/ADS: A machine-aided approach to analysis of ADS. ISDOS Working Paper No. 35, Dep. Indus. and Operations Eng., U. of Michigan, Ann Arbor, Mich.; Oct. 1970.
13. Herman, D.J., and Ihrer, F.C. The use of a computer to evaluate computers. Proc. 1964 AFIPS SJCC Vol. 25, AFIPS Press, Montvale, N.J., pp. 383-390.
14. The Case for CASE: Computer Aided System Evaluation. Tesdata Systems Corp., Chevy Chase, Md., 1971.
15. MacDougall, M.H. Computer system simulation: An introduction. *Computing Surveys*, 2, 3, (Sept. 1970), 191-210.
16. CODASYL Data Base Task Group Report. ACM, New York, April 1971.