Discussion Paper No. 310

A METHODOLOGY FOR AUTOMATED SYSTEMS DESIGN

by

Edward A. Stohr*

January 1978

* Northwestern University, Graduate School of Management, Evanston, Illinois 60201

ABSTRACT

Most information systems involve the processing of various types of transactions and their subsequent aggregation according to various classification schemes for the purpose of management reports. A method of systems analysis and information processing system design is discussed in which the logic of such information systems is expressed as a graph and is implemented using only a set of prewritten subroutines. The description involves graph representations of work-flow in the organization and of the multiple classification schemes used for reporting. The logical description of the system can be considered to consist of 'classification' and 'report processing' schemas which coexist with the conventional data base schema defined over the records of the master and transaction files of the data base. Their function is to provide the intelligence necessary to automatically implement the processing required to generate status reports for the organization.

DISCUSSION PAPER NO. 310

A METHODOLOGY FOR AUTOMATED SYSTEM DESIGN

by

Edward A. Stohr

1. Introduction

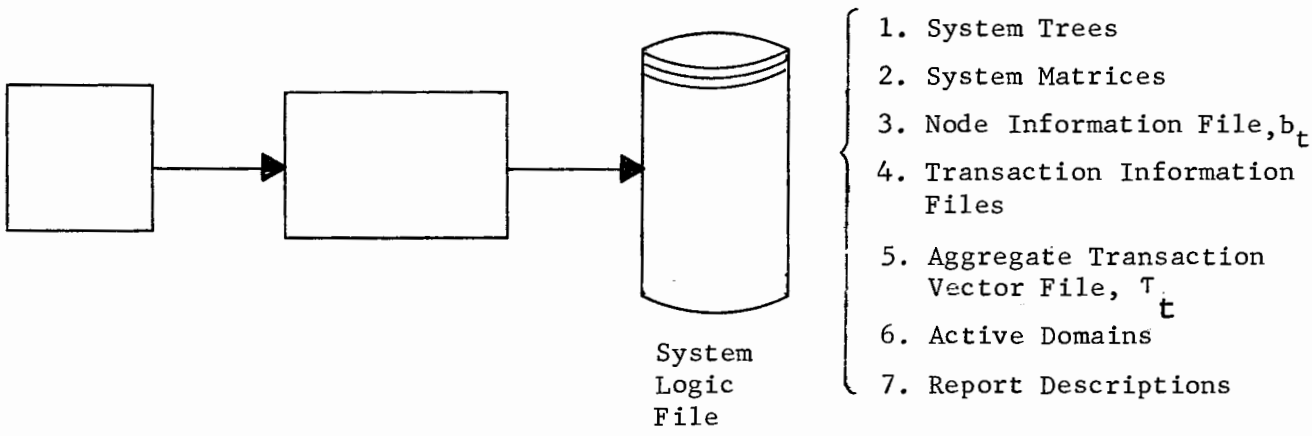
The problem of complexity in systems design is the subject of much present research and a number of different approaches have been proposed. Much progress has been made in improvement of the coding process by techniques such as structured programming (Dijkstra [3]) and 'chief-programmer teams', (Mills [12]). At a more ambitious level, attemps are currently being made to automate the entire systems design process. Techniques for producing and analyzing a machine readable problem statement and data dictionary have been advanced by the ISDOS Project (Teichroew [18], Nunamaker, et.al. [14] and Langefors, [10]). Methods for producing a suitable data base design from the problem statement are also being developed (Nunamaker [13]). Progress relating to the automatic production of running computer programs from the problem statement is being made by Gerritson [7]. This paper is similar in spirit to the latter work, but takes a different approach in that the objective is to eliminate as far as possible the necessity to write specialized code for certain types of application.

The paper describes a software system which is currently being developed where the underlying idea is to express the 'logic' of the information system as data rather than to have it embodied in computer programs. This reduces the necessity for coding and

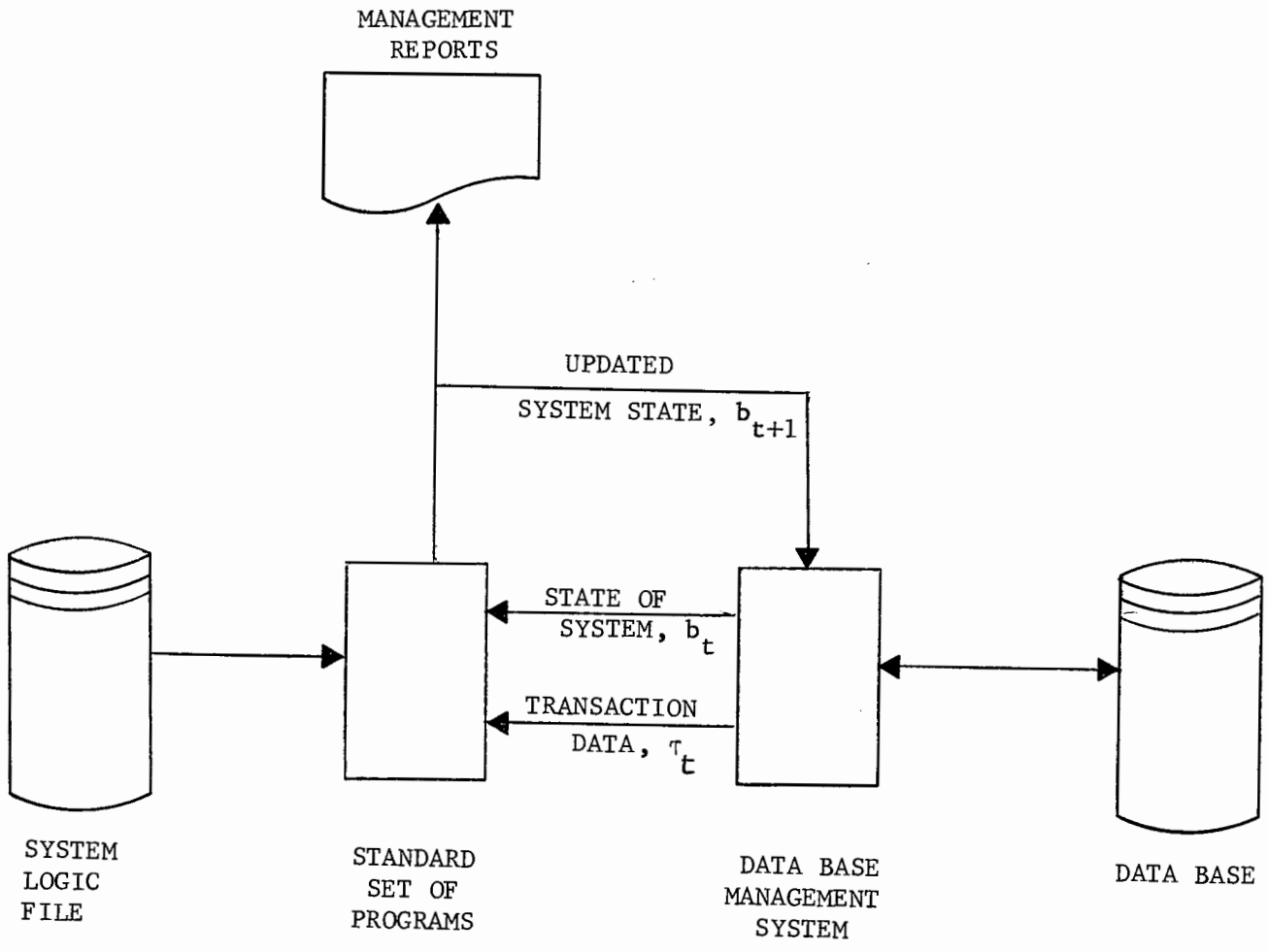
gives the resultant system a high degree of flexibility. As far as possible, the computations necessary to produce management reports are carried out using a 'standard set' of programs which are logically equivalent to matrix and other linear algebraic operations. A key task of the system analysis for such a system involves the production of 'system logic files' which define both managerial requirements for reports of various kinds and the logical relationships existing between the data inputs to the computing process and its outputs. The standard set of programs use the 'system logic files' together with data concerning the current state of the system and current transactions to produce the required managerial reports. The 'system data files' (records describing the state of the system and transaction records) are maintained by traditional methods--possibly by a 'data base management system' (DBMS). The underlying philosophy of the system is shown in Figure 1.

Many of the reports produced by management information systems simply involve the aggregation of data according to multiple classification systems. Examples are financial reports (balance sheet, income statements), managerial accounting reports (standard costing systems, responsibility accounting) and marketing reports (sales classified by product, by region and by salesman). The list is almost endless. The techniques to be described rely on the fact that such classification schemes can be described both graphically and algebraically. The systems analyst will be

SYSTEM DEFINITION PHASE:



REPORT GENERATION PHASE:



MAJOR SYSTEM COMPONENTS

FIGURE 1

able to visualize and define the system primarily in terms of its representation as a graph (flow diagram or tree) while the standard set of programs will implement the algebraic description. The design process will be automated in the sense that the system analyst will communicate with an on-line program and that algorithms will be provided to transform his system definition into the algebraic format of the systems logic file. The systems logic file can be viewed as a 'classification schema' which is superimposed, on the usual hierarchical,network or relational schema defined over the transaction and master files of the data base.

The examples to be described will concern both accounting and non-accounting applications. The system representation to be used was first proposed by Eaves [4] and Butterworth [2] in an accounting context. Related work is contained in Lieberman and Whinston [11], and Shank [17]. Everest and Weber [5] describe a relational schema for accounting systems and suggest an approach similar to that adopted here in a concluding paragraph. The present paper attempts to generalize this work by extending the techniques to various other applications and by designing an automated interface for systems analysis in large-scale data processing systems.

2. Representation of Information Classification Schemes

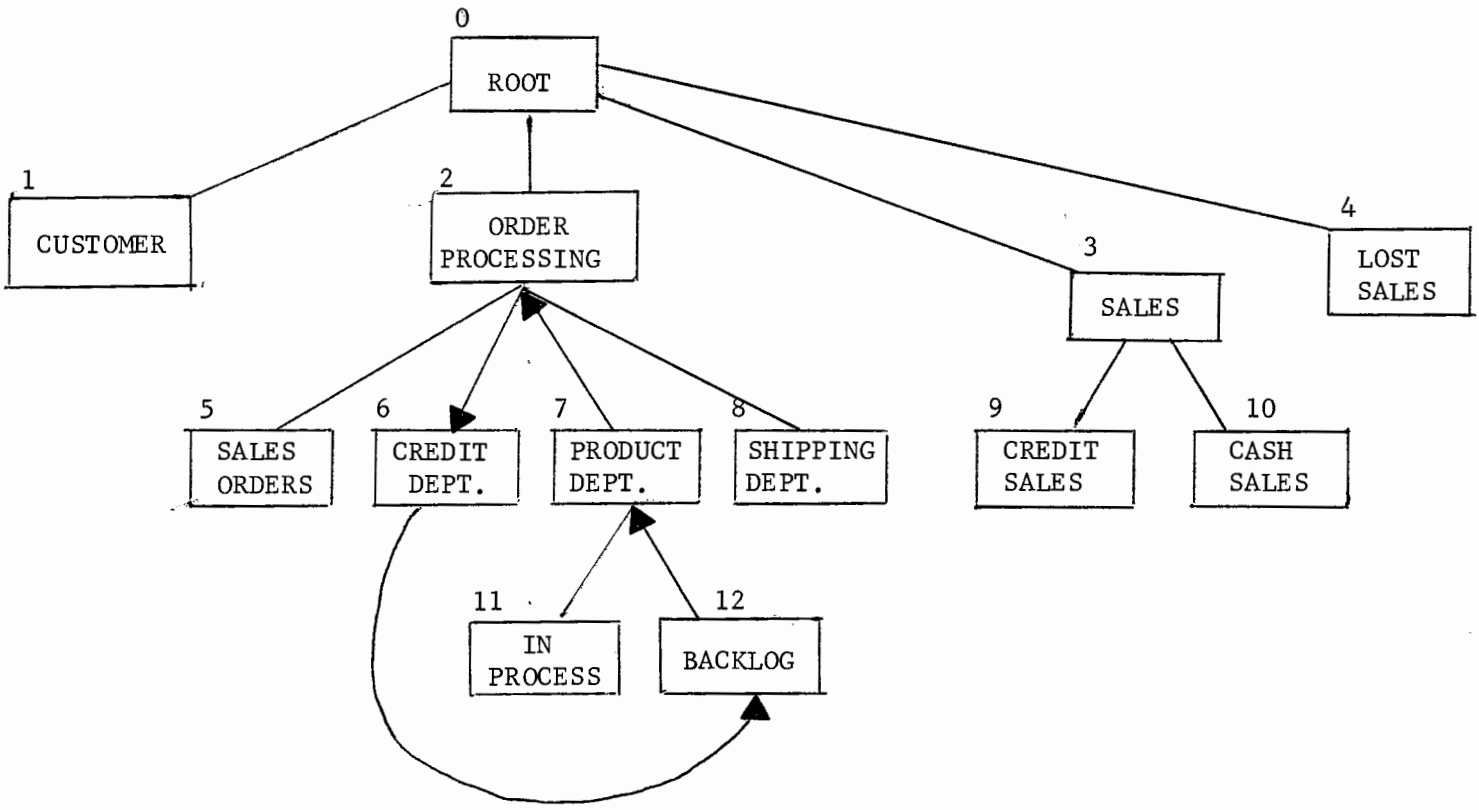
Routine information processing systems usually perform two functions: (1) recording, validating and processing of transactions, (2) aggregation of the values of the transactions in some hierarchical scheme for the production of management reports. In this paper we are primarily concerned with the latter function.

However, the methods adopted imply a highly structured approach to the design of the processing function as well.

A representation of an order-entry and sales processing system is shown in Figure 2. The leaves of the tree represent the various (possible) stages in the processing of an order. The intermediate nodes represent the departmental/functional/accounting classification scheme. Finally, the root of the tree is an artificial node which binds the various sub-trees together.

A transaction changes the state of the system and can be uniquely represented by a directed arc joining two leaves in the tree. The direction of the arc corresponds to the direction of the flow of paperwork, money, material, etc. associated with the transaction. Thus the transaction defined by the arc (6,12) represents the backlogging of an order after it has been approved by the credit department. The addition of an arc defines a unique loop in the tree. Except for the node nearest the root, all the nodes on this circuit have their states changed by the transaction. For the transaction just mentioned, the loop passes through nodes 6, 12, 7 and 2, in the direction indicated. The number of orders, their value and physical quantity of product involved in the credit department (node 6) are decreased while the corresponding quantities in the production department (node 7) and in particular, in the backlog queue (node 12) are increased.

Thus a preliminary description of the processing system can be given by a hierarchical classification scheme and a list of transactions as shown in Figure 2. Note that various forms such



TRANSACTIONS

No.	Defining Arc	Description
1	(1,5)	CUSTOMER ORDER TAKEN BY SALESMAN
2	(5,6)	ORDER SENT TO CREDIT REVIEW
3	(6,11)	CUSTOMER CREDIT APPROVED; ORDER IN PROCESS
4	(6,12)	CUSTOMER CREDIT APPROVED; ORDER BACKLOGGED
5	(6,1)	CUSTOMER CREDIT DISAPPROVED; ORDER RETURNED
6	(12,11)	BACKLOGGED ORDER PLACED ON PRODUCTION SCHEDULE
7	(11,8)	PRODUCTION COMPLETED ON ORDER
8	(8,9)	SHIPMENT TO CUSTOMER ON CREDIT SALE
9	(8,10)	SHIPMENT TO CUSTOMER ON CASH SALE
10	(9,4)	CUSTOMER RETURNS UNITS RECEIVED AND REJECTED (CREDIT SALES)
11	(10,4)	CUSTOMER RETURNS UNITS RECEIVED AND REJECTED (CASH SALES)

CLASSIFICATION AND AGGREGATION
- ORDER PROCESSING AND SALES -

FIGURE 2

as purchase orders, shipping documents, invoices, etc. would be associated with the leaf nodes, and transaction files with the arcs. To complete the system description in the sense of a form driven system analysis technique such as SOP [9] or a computed-aided technique such as the Problem Statement Language (PSL) [17] more detail can be added, however these descriptive techniques will not be elaborated upon here. Instead, we will describe an algebraic analog of the information in Figure 2 and show how it can be used to generate the required managerial reports.

Flows toward the root node in Figure 2 will be considered positive and those away from the root node negative. If there are m nodes (not counting the root node), and n transactions, the information in Figure 2 can be represented by an mxn 'systems matrix', S, as shown in Figure 3. Here each row corresponds to a node and each column to a transaction (loop) with the 1's and -1's indicating the direction of flow according to the above convention. Note that the node nearest the root for any given loop has a zero entry in the matrix (the flow at that node is both towards and away from the root). A column of S describes the effect of 1 unit of the associated transaction type on each node, while a row shows how its associated node is affected by each transaction. For simplicity we will consider the case where the n transactions are measured on a single dimension (for example, dollar value). Let τ be an n-dimensional vector with element τ_j representing the

	Transaction										
Node	(1,5)	(5,6)	(6,11)	(6,12)	(6,1)	(12,11)	(11,8)	(8,9)	(8,10)	(9,4)	(10,4)
1	-1	0	0	0	1	0	0	0	0	0	0
2	0	0	0	0	-1	0	0	-1	-1	0	0
3	0	0	0	0	0	0	0	1	1	-1	-1
4	0	0	0	0	0	0	0	0	0	1	1
5	1	-1	0	0	0	0	0	0	0	0	0
6	1	1	-1	-1	-1	0	0	0	0	0	0
7	0	0	1	1	0	0	-1	0	0	0	0
8	0	0	0	0	0	0	1	-1	-1	0	0
9	0	0	0	0	0	0	0	1	0	-1	0
10	0	0	0	0	0	0	0	0	1	0	-1
11	0	0	1	0	0	1	-1	0	0	0	0
12	0	0	0	1	0	-1	0	0	0	0	0

SYSTEMS MATRIX
- ORDER PROCESSING AND SALES -

FIGURE 3

cumulative flow of transaction type j for the time period.[1] In the example, τ_4 is the total dollar value of orders backlogged during the time period. Then the product $S\tau$ gives the net change in the dollar value of transactions at each node in the tree. Let the subscript t represent time and define $b_t \in R^m$ such that b_{it} = value of transactions at node i at time t. Then the net change in state of the processing system is described by the difference equation

(1) $\quad b_0 = 0$

(2) $\quad b_{t+1} = b_t + S\tau_t \qquad\qquad t = 0,1,2,\ldots$

Let $S^+(S^-)$ be the matrix obtained from S be replacing all 1's (-1's) by zeroes then the cumulative inflows (outflows) from each node are given by (3) and (4):

(3) $\quad b^+_{t+1} = b^+_t + S^+\tau_t$

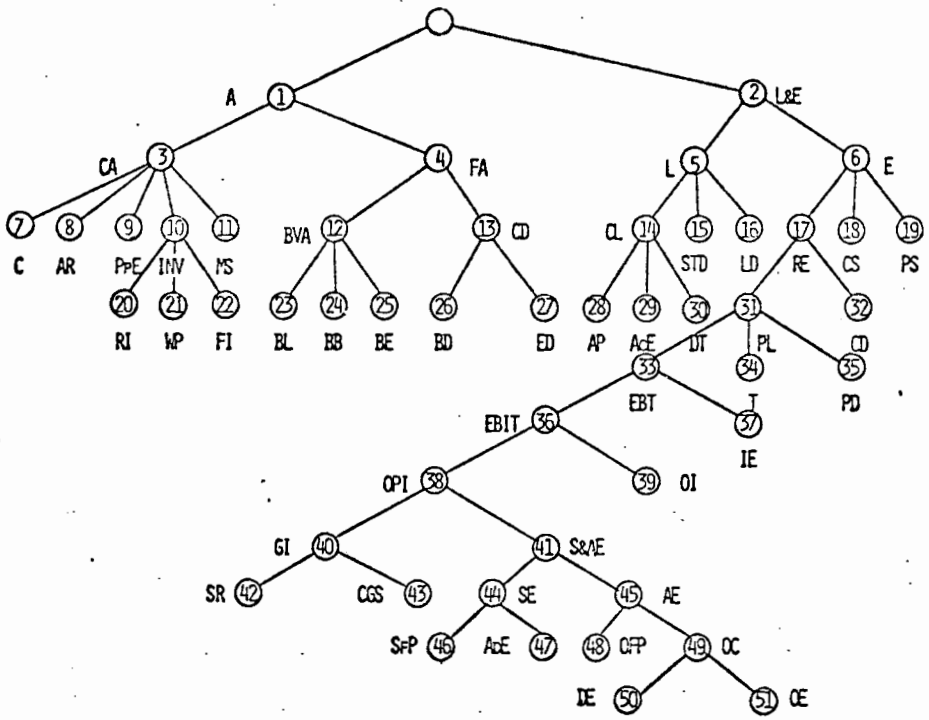
(4) $\quad b^-_{t+1} = b^-_t + S^-\tau_t$.

In addition to its use in the present context, the algebraic representation (1) through (4) is useful in simulation and forecasting applications, [1]. It must be emphasized that the algebraic representation represents the logical description of the process only. The physical implementation involves quite different data structures and processes for reasons of efficiency and economy of storage.

---

[1] The case where the transaction volume is to be recorded in terms of more than one dimension (for example, dollar value and physical quantity) is accommodated simply by considering τ to be an nxk matrix where k is the number of dimensions considered.

A representation of the traditional balance sheet and income statement accounts for a small business is shown in Figure 4. Each node in Figure 4 corresponds to an 'account.' The tree has two sub-trees representing the asset accounts and the liability and equity accounts. The income statement accounts appear as a sub-tree under a 'Profit and Loss' account, which serves to connect the income and balance sheets. A standard cost accounting system can be added as a sub-tree of the inventory account as discussed later. All of the other leaves of the tree might have (one or more) sub-trees attached to them if and when further detail is required. Alternatively, the tree in Figure 4 might represent the accounts for only one division ofa multi-division company in which case it might be regarded simply as a sub-classification of the corporate accounting system. It often happens that more than one classification will be associated with a given node of the tree. This occurs for example at the inventory node, if management requires both a departmental cost break-down ('responsibility accounting') and product cost break-down (e.g. 'standard cost accounting'). In this case the hierarchical 'classification schema' is no longer a tree but a 'confluent hierarchy' with more than one hierarchy having common leaf nodes. Finally, the approach also provides flexibility with regard to accounting methods. For example, to add a capability for producing 'replacement cost' accounting reports as well as 'historic cost' reports it is only necessary to generate the required transactions and to add 9 or 10 rows and columns to the systems matrix (Frigo [6]).



Acct. (Node)	Symbol	Description
1	A	Assets
2	L&E	Liab. & Equities
3	CA	Cur. Assets
4	FA	Fixed Assets
5	L	Liabilities
6	E	Equities
7	C	Cash
8	AR	Accounts Rec.
9	PpE	Prepaid Exp.
10	INV	Inventory
11	MS	Marketable Sec.
12	BVA	Bk. Val. Assets
13	CD	Cum. Deprec.
14	CL	Cur. Liab.
15	STD	Short-term Debt
16	LD	Long-term Debt
17	RE	Retained Earnings
18	CS	Common Stock
19	PS	Preferred Stk.
20	RI	Raw-Mat. Inv.
21	WP	Work-in-Process
22	FI	Finished Gd. Inv.
23	BL	Bk-Value of Land
24	BB	Bk-Value of Bldg.
25	BE	Bk-Value of Equip.
26	BD	Bldg. Depreciation
27	ED	Equp. Depreciation
28	AP	Acct. Payable
29	AcE	Accrued Expenses
30	DT	Deferred Taxes
31	P/L	Profit/Loss
32	CD	Common Dividend
33	EBT	Earn. Before Taxes
34	T	Taxes
35	PD	Pref. Dividends
36	EBIT	Earn. Bef. Int & Txs
37	IE	Interest Exp.
38	OPI	Oper. Income
39	OI	Other Income
40	GI	Gross Income
41	S&AE	Sell. & Adm. Expense
42	SR	Sales Revenue
43	CGS	Cost of Gds. Sold
44	SE	Selling Expense
45	AE	Adm. Expenses
46	SfP	Sales force Pay Exp
47	AdE	Advertising Exp.
48	OfP	Office Payroll Exp.
49	OC	Other Costs
50	DE	Depreciation Exp.
51	OE	Other Expense

No.	Transaction	Node No. (From), (To)	Acct. Entries Db/Ct	Trans. No.
1.	Cash Sales	42,7	C/SR	100
2.	Sales on Account	42,8	AR/SR	101
3.	Sales Force Sal. Exp.	7,46	SfP/C	102
4.	Advertising Expense	7,47	AdE/C	103
5.	Cost of Goods Sold	22,43	CGS/FI	104
6.	Coll. of Acct. Rec.	8,7	C/AR	105
7.	Office Salaries Exp.	7,48	OfP/C	200
8.	Lab. Costs Pd. in Cash	7,21	WP/C	201
9.	Mat. Used This Period.	20,21	WP/RI	202
10.	Overhead Pd. in Cash	7,21	WP/CD	203
11.	Misc. Overhead Exp. in the Cur. Prd, but Pd for in Prior Prds.	9,53	OE/PE	206
12.	Dep. Exp. in this Period	26,50	DE/BD	208
13.	Inventory Increase	21,22	FI/WP	209
14.	Cash Pur. of Inventory	7,20	RI/C	300
15.	Pur. of Inv. on Acct.	28,20	RI/AP	301
16.	Payment of Acct. Pay.	7,28	AP/C	302
17.	Equip. Bought for Cash	7,25	BE/C	303
18.	Pur. of Equip. on Acct.	28,25	BE/AP	304
19.	Sale of Equip. for Cash (at Cost)	28,7	C/BE	305
20.	Gain/Loss from Sale of Equipment	39,7	C/OI	306
21.	Dep. of Sold Equip. Cleared from Bks.	7,27	ED/C	307
22.	Rev. from Marketable Sec.	39,7	C/OI	400
23.	Interest Expense	7,37	IE/C	401
24.	Taxes Paid in Cash	7,34	T/C	402
25.	Adj. for Tax Difference	30,34	T/DT	403
26.	Deferred Taxes Pd. in Cash	7,30	DT/C	404
27.	Pref. Div. Pd. in Cash	7,35	PD/C	405
28.	Com. Div. Pd. in Cash	7,32	CD/C	406
29.	Com. Div. Pd. in Stock	18,17	RE/CS	407
30.	Repayment of Lgterm Debt	7,16	LD/C	408
31.	Repayment of Shtterm Debt	7,15	STD/C	409
32.	Proc. from Shtterm Loan	15,7	C/STD	410
33.	Pur. of Marketable Sec.	7,11	MS/C	411
34.	Proc. from Sale of Mktable Securities	11,7	C/MS	412
35.	Proc. from Issuance of Common Stock (at Par)	18,7	C/CS	413
36.	Proc. from Issuance of Com. Stock (in Excess of Par)	17,7	C/RE	414
37.	Proc. from Iss of Pref Stk	19,7	C/PS	415
38.	Issue of Longterm Debt	16,7	C/LD	416

FIGURE 4

Classification and Transaction Schemas for an Accounting System

The systems matrix for the accounting system has a 'plus' sign associated with an accounting 'debit' and a 'minus' sign with a 'credit.' At time t+1 the conventional accounting balance, b_{t+1}, in all accounts (including the standard cost, responsibility accounting, etc. accounts if present), is obtained very simply by using equation (2).[1] Reports corresponding to the traditional balance sheet, income,and sources and uses of funds statements can be specified simply in terms of the appropriate subsets of b_t.

When a report is to be generated the system to be described consults the classification schema ('system logic file'), automatically determines the required subset of τ_t and generates the submatrix of S required to produce the report. Again it should be noted that the elements of b_t and τ_t correspond in the physical data base to file records and will normally contain many data items. Thus in the accounting application there will normally be fields in the b_t records containing descriptive information, the budgeted amount, the balance in previous time periods, etc. The aggregate transaction vector τ_t summarizes the total transaction activity over a time period. The elements (records) of τ_t can be maintained permanently by the system or can be generated as required by an aggregation process performed on the underlying transaction files.

We end this section by stating some simple properties of this representation [2]:

(1) A row in S (or S^+ of S^-) corresponding to a summary account is the sum of the rows for all the accounts immediately subsidiary to that account.

---

[1] Care must be taken to zero-out the income statement accounts in b_t before (2) is applied.

(2) The sum of the balances of the leaf node accounts is zero and as a consequence the system is always 'in balance.'

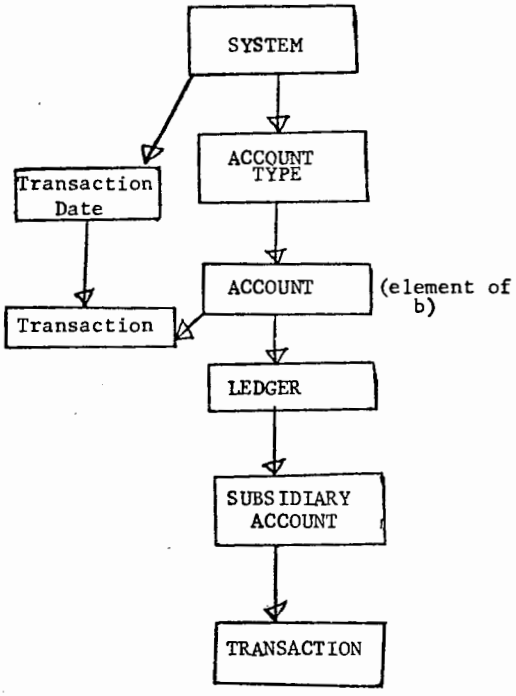
3. Logical System Design

This section outlines the design of a system which uses the raw data carried by transactions, together with the previous state, b_t, of the system, to produce an updated system state, b_{t+1}, and to perform the computations for a wide range of management reports. Some of the reports may be 'preplanned' others 'ad hoc.' We are not concerned with the format of the reports -- only that the required information is made available to the report writing subroutines. The distinguishing features of the proposed system are that it uses only a few 'standard routines', that it can be used for a wide variety of applications -- both accounting and non-accounting, and that the logic of the system is expressed by data structures rather than by a programming language -- consequently, the system possesses great modularity and flexibility.

The logic of the system is described in terms of Bachman diagrams in Figure 5. The objects in full outline are static data structures explicitly stored by the system; the dashed outlines indicate structures which are generated as required by requests for reports. Assuming that the organization's data base is maintained by a data base management system (DBMS) there will exist a schema such as that depicted in Figure 5(a). The DBMS can be used in the usual way for the input, update and maintenance of the

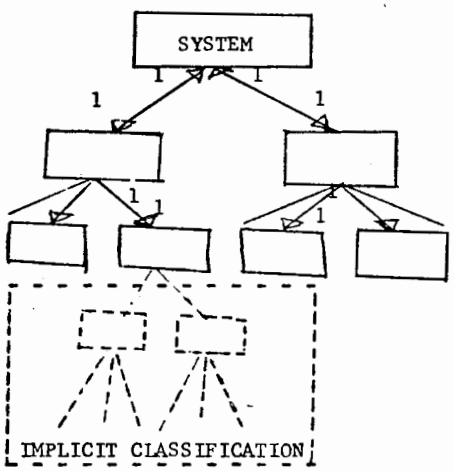
(a) TYPICAL CONVENTIONAL SCHEMA

(for an accounting system, see [ ])

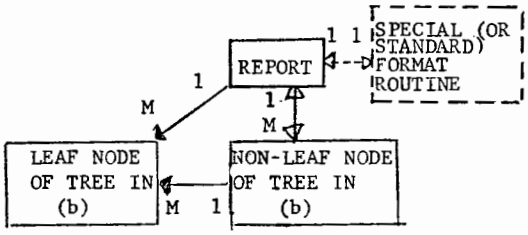


(b) CLASSIFICATION SCHEMA

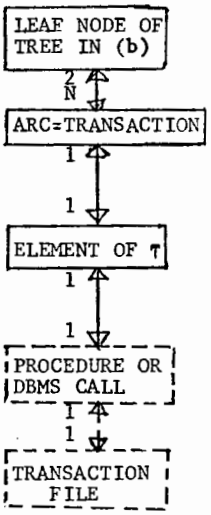
(relationships between elements of b).



(c) REPORT SCHEMA



(d) TRANSACTION & PROCESSING SCHEMA



(e) UPDATING PROCESS

$$b_{t+1} = b_t + S_{T_t}$$

where schemas (b) and (d) imply S

FIGURE 5
Logical Structure of Report Management System

master and transaction files and for the routine or ad-hoc queries which require the record relationships maintained by the DBMS. Normally it would also be used to produce the required management reports (such as the balance sheet, income statement, etc. in an accounting system) which are based on hierarchical classification schemes. In the proposed system the classification schema will be logically and physically distinct from the schema used to maintain the data base. The logical separation helps to remove 'the confusion between the real entity and a classification scheme applying to that entity...' (Everest and Weber [5]). The physical separation is imposed for reasons of efficiency in the search and retrieval operations to be described later. This also allows the report to be generated by standard sub-routines' rather than special purpose data manipulation language (DML) code which would have to 'navigate' through the higher levels of the hierarchy.

The logical relationships shown in Figures 5(b), (c) and (d) and the 'standard subroutines' which operate on this information to produce management reports and update the state of the system will be referred to collectively as the Report Management System (RMS). The RMS invokes the DBMS as shown in Figure 5(c) whenever it is necessary to perform an aggregation operation on the underlying transaction files.

The classification schema (Figure 5(b)) has been described previously,the only new item being the 'implicit classification' sub-tree. To illustrate the meaning of this consider the order processing and sales system introduced earlier and assume that there are two products (P_1,P_2), two customer classes (C_1,C_2),

and two regions (R_1,R_2). Management will require performance information classified according to these dimensions as well as according to those indicated by the classification scheme in Figure 2. We look at an order as consisting logically of the data fields:

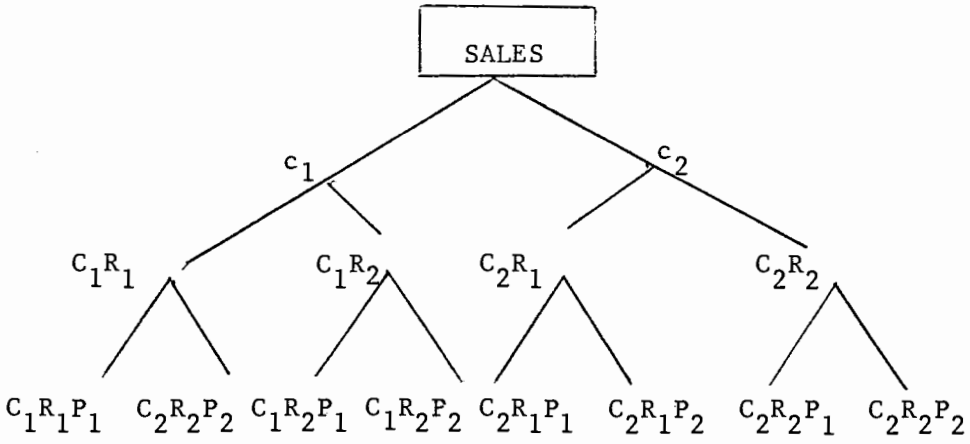
ORDER NO	DATE	CUSTOMER CODE	REGION CODE	QUANTITY	PRICE

Suppose management requires a report on the current month's sales classified by product within region within customer class, (Figure 6(a)). To be consistent with the representation developed in Section 2 we might add the sub-tree shown in Figure 6(b) to the SALES node in Figure 2, thus augmenting S by an 'aggregation' matrix. In Figure 6(b), node C_1 indicates that total sales are to be accumulated for customer 1, node C_1R_1 indicates that total sales are to be 'accumulated for customer 1 in region 1, etc. The transaction vector τ is augmented by: $(C_1R_1P_1, C_1R_1P_2,\ldots,C_2R_2P_2)$, where $C_iP_jR_k$ is the total value of transactions for customer class i product j and region k for the time period. Note that we now have two distinct classification schemes for sales -- cash versus credit sales and total sales by product, region and customer class. This is indicated by the Bachman diagrams in Figure 6(c). Obviously this method rapidly becomes unwieldy when there are more than one or two products, regions and customer classes -- especially since sub-trees similar to that in Figure 6 might be attached to any of the nodes in Figure 3. These sub-trees are not therefore stored explicitly. Instead, they are generated by the standard routines when and if required; or, more precisely, the standard

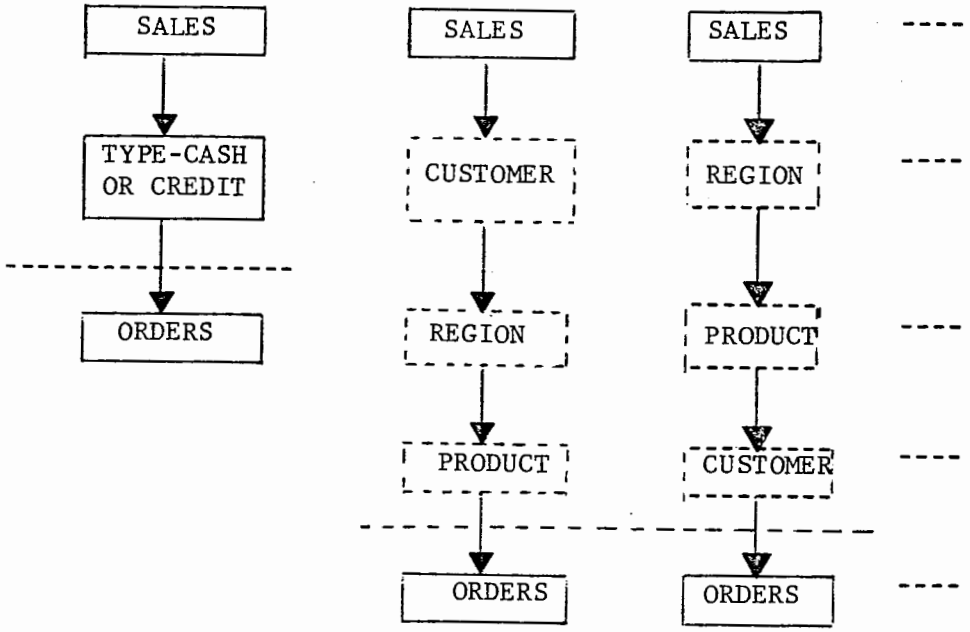
(a) Report Request:

REPORT SALES CLASSIFIED BY PRODUCT WITHIN REGION WITHIN CUSTOMER

(b)



(c)



AGGREGATION SCHEMES FOR SALES REPORTS

FIGURE 6

routines carry out steps which are logically equivalent to using (2) with the appropriately augmented system matrix.

The 'Report Schema' (Figure 5(c)) and the 'Transaction Processing Schema' (Figure 5(d)) will be explained more fully in subsequent sections. In summary, the 'systems logic file' of Figure 1 contains:

(1) Systems Trees: physical representations of the hierarchical relationships in the classification schema. The systems trees are defined during the system definition phase and are used to generate the systems matrices. They are also used during the report generation phase to determine the transactions which are required to produce a given report. Separate trees may be stored for each major system or subsystem and each mode of classification within a system.

(2) System Matrices (in sparse matrix form): the algebraic representation of the system of aggregation. They can be used directly as input to the report generation routines. They are also used during the report generation phase to determine all the nodes affected by a given transaction. One system matrix may be associated with each of the trees in (1). Systems matrices are stored only for much used routine reports since the RMS is capable of generating the data structure necessary to perform the aggregations for ad-hoc reports.

(3) Node Information File: this file contains the code number and name of each node in the systems trees together with other pertinent information. It also contains pointers to the

nodes' positions in the systems trees and matrices and to the state file, b_t.

(4) Transaction Information File: this file contains the code number and name for each defined transaction together with pointers to the elements of τ (if they are stored explicitly) and to the DBMS calls required to retrieve and aggregate the information in the underlying transaction files.

(5) Active Domains: list of valid code values for elements in a classification scheme (see next section).

(6) Report Descriptions: Header information for the report together with information defining the order in which information is to be presented to the report writing routines or alternatively the names of any special purpose report routines.

4. System Definition Phase

Figure 7 demonstrates the general form of the proposed interface for the data base administrator. The underlined words are prompts from the system definition programs or keywords in a DBMS query language such as that described in Haseman and Whinston [8]. Some details are omitted from the figure for reasons of brevity. In general code numbers as well as names would be input to identify the nodes and transactions, however only names are used in the illustration.

The name of the classification schema is input first and the root declared to be 'SYSTEM' since this is the highest level of the hierarchy. If multiple subclassification schemes are present the root may be a leaf or non-leaf node in an already

SYSTEM NAME:

SALES ORDER PROCESSING

CLASSIFICATION NAME:

SALES ORDER PROCESSING

ROOT:

SYSTEM

NODES:

CUSTOMER < ROOT

ORDER PROCESSING < ROOT

SALES < ROOT

SALES ORDERS, CREDIT DEPT, PRODUCT DEPT, SHIPPING DEPT <
ORDER PROCESSING

IN PROCESS, BACKLOGGED < PRODUCTION DEPT

:

IMPLICIT SUBCLASSIFICATION SCHEMA:

COMBINATIONS (CUSTOMER, PRODUCT, REGION) < BACKLOG, IN PROCESS, SALES

DOMAINS:

CUSTOMER - WHOLESALE, RETAIL

PRODUCT - PRODUCT 1 THRU PRODUCT 2

REGION - EAST, WEST

ELIMINATE:

(, PRODUCT 1, WEST)

TRANSACTION FLOWS:

CUSTOMER ORDER TAKEN BY SALESMAN: OUTSIDE, SALES

ORDER SENT TO CREDIT REVIEW: SALES, CREDIT

CREDIT APPROVED, ORDER IN-PROCESS: CREDIT, IN-PROCESS

CREDIT APPROVED, ORDER BACKLOGGED: CREDIT, BACKLOG

:

TRANSACTION SEARCH INFORMATION:

CUSTOMER ORDER TAKEN BY SALESMAN:

FILE ORDER, AGGREGATE (PRICE X QUANTITY) WHERE STATUS = 1;

ORDER SENT TO CREDIT REVIEW:

FILE ORDER, AGGREGATE (PRICE X QUANTITY) WHERE STATUS = 2;

:

SYSTEM DEFINITION PHASE
FIGURE 7

existing schema. The hierarchical structure and node names are input next with the symbol '<' meaning 'is the child of.'

The "Implicit Sub-Classification Schema" allows a further breakdown and aggregation of the information coded on the transactions themselves. Thus, the 'Combinations' line declares that summary transaction information is to be retrieved by the system for every combination of customer, region and product. Logically the multiple subclassification schemes indicated in Figure 6(c) are declared to exist under the Backlog, In Process and Sales nodes. However, the associated classification schema will not be generated and stored explicitly in the systems file. The 'domains' for customer, product and region have to be declared and maintained so that the requisite valid combinations can be generated for validating and reporting purposes. The list following the 'eliminate' prompt indicates infeasible combinations of domain values (thus, for example, product 1 is not sold in the western region).

The transactions are described next with the name (and/or code number) of the transaction followed by the names of the two leaf nodes which define the transaction's effect on the nodes in the tree. The order in which the nodes are listed defines the direction of flow (from, to). For each defined transaction the 'transaction search' information describes how the associated element of τ is updated. In this case this is done by invoking a DBMS query language command which opens the specified file and aggregates over an implied or actual field based on a condition statement. In general more than one transaction file may be

involved. However in the illustration it is assumed that the transaction flows indicated in Figure 3 are all recorded in the 'ORDER' file and that they are distinguished by a status code field appended to the record defined earlier. Thus orders in the (logical) transaction stream defined by the arc (1,5) have STATUS = 1, etc.

Finally, the required production reports are listed (not shown). Logically all that is required here is the name of the report and the list of nodes in the required order. If all of the nodes in a sub-tree are to be included in the report it is only necessary to state the root node of the sub-tree -- the system uses the classification schema to automatically produce the report. The hierarchical level of the nodes in the schema is used to help format the report.

This completes the logical description of the information classification and reporting schema. Subsequent modifications and additions to the schema can be made in a similar manner without affecting the logic of the system processing and report generating routines. File space for any resulting additions to the vectors b_t and τ would have to be allocated. However, it is important to note that the order in which the files b_t and τ are stored is immaterial and that these files should never have to be sorted into any order other than that declared by (or implicitly derived from) the system definition.

5. Report Generation Phase

We first describe the process by which regular production reports are generated. Assuming that the relevant state (b_t) and transaction (τ_t) vectors are available, it is only necessary to execute calls to two 'standard' report generating routines -- one to perform the aggregation via (2), (3) or (4), and another to print the resultant information. (A specially written report formating routine can be substituted for the latter routine if desired.) Information concerning any special formating requirements is stored in the 'Report Schema' (Figure 5(c)). These calls could be executed from a host language program whose only other function would be: (1) to open and close the files containing the required report-definition and the associated vectors b_t and τ_t, and (possibly) (2) to update the vector b_t.

The system design outlined in Section 3 can also be used to provide the intelligence necessary for the generation of ad hoc reports. One example of a report request is given in Figure 6(a). As another example suppose that this period and year-to-date sales are to be printed classified on a cash or credit basis only and that this is not a regular (predefined) production report. The request would be stated:

REPORT SALES

The steps involved are as follows:

(a) The relevant system tree (Figure 2) is accessed and the sales node located. The leaf nodes in the sales sub-tree are then located. Only transactions beginning or ending at these nodes need be identified.

(b) The relevant leaf nodes are Credit Sales and Cash Sales. Reference to the associated rows in the Systems Matrix (Figure 3) identifies the four transactions which are involved -- namely, those associated with columns 8 through 11 of the Systems Matrix (and elements 8 through 11 of the transaction vector).

(c) Generating the sub-matrix defined by the rows and columns mentioned in (b) and retrieving the relevant elements of b and τ the report can quickly be obtained using the logic of equation (2).

6. Conclusion

This paper has described the logical design of a system for managing the processing of information for management reports. Some considerations relevant to the physical implementation are outlined below; others will be discussed in a later paper.

The first questions concern the storage of the records associated with the b_t and τ_t vectors. These must be readily accessible to RMS. In many existing systems the elements of b_t and τ_t are stored together with the associated application file. Thus this month's aggregate transactions (debits and credits) to Accounts Receivable might be stored in the Accounts Receivable master file. For RMS to function efficiently it will be helpful to consolidate the state and aggregate transaction records for a given classification schema into two separate files. In addition the data-base administrator must decide which elements of τ_t are to be stored, explicitly and which are to be generated

as required by aggregation processes defined on the underlying transaction files, as illustrated by the DBMS calls in Figure 7. Extensive information of this kind is stored on most existing systems - indeed past values of the state and transaction variables are usually maintained also because of their usefulness in management reports as a yardstick for gauging current results.

The second set of questions concern the updating of the elements of b_t and τ_t. In an on-line system it is possible to update the appropriate elements of these records as soon as each transaction received by the system has been validated. This will enable up-to-date ad hoc management reports to be obtained at any time.

Finally, it is possible to define the logical structures necessary to implement RMS via the data definition language of the DBMS itself. In the system currently being implemented this alternative was not adopted because it was desired to optimize the retrieval capabilities of RMS (storage of b_t and τ_t) for a given probability distribution of management requirements for reports. These aspects are discussed more fully in Ormancioglu, [16].

REFERENCES

1. Blin, J.M., E.A. Stohr and M. Tanniru, "Development of a Corporate Information System," Proc. COMPSAC 77, Chicago, November 1977, pp.

2. Butterworth, John, "The Accounting System as an Information Function," Journal of Accounting Research (Spring 1972), pp.1-27.

3. Dijkstra, E.A., "A Constructive Approach to the Problem of Program Correctness," BIT 8 (1968), 174-186.

4. Eaves, B. Curtis, "Operational Axiomatic Accounting Mechanics," The Accounting Review, Vol. XLI, No. 3, July 1966.

5. Everest, Gordon C. and Ron Weber, "A Relational Approach to Accounting Models," The Accounting Review, Vol. LII, No. 2, April 1977.

6. Frigo, Mark L., "Utilization of Replacement Cost Information Within an Accounting System" (unpublished paper).

7. Gerritson, Rob, "Understanding Data Structures," Working Paper 75-08-01, Decision Sciences Department, The Wharton School, University of Pennsylvania.

8. Haseman, William D. and Andrew B. Whinston, Introduction to Data Management, Richard D. Irwin, Homewood, Illinois, 1977.

9. International Business Machines Corporation, "Study Organization Plan Documentation Techniques" (Form No. C20-8075), White Plains, New York 01961.

10. Langefors, Borje, Theoretical Analysis of Information Systems, Auerbach Publishers, Inc., Philadelphia, 1973.

11. Lieberman, Arthur Z. and Andrew B. Whinston, "A Structuring of an Events-Accounting Information System," The Accounting Review, April 1975.

12. Mills, H.D., "Chief Programmer Teams Principles and Procedures," IBM Corp., Gaithersburg, Md., FSC 71-5108, 1971.

13. National Cash Register Company, "A Study Guide for Accurately Defined Systems," Dayton, Ohio, 1968.

14. Nunamaker, J.F., Jr., "A Methodology for the Design and Optimization of Information Processing Systems," Proceedings, Spring Joint Computer Conference, AFIPS Press, Montvale, New Jersey, 1971, pp.283-294.

15. Nunamaker, J.F., Jr., Benn R. Konsynski, Jr., Thomas Ho, and Carl Singer, "Computer-Aided Analysis and Design of Information Systems," Comm. ACM, 19, December 1976.

16. Ormancioglu, Levent, "An Automated Report Generation System-Design and Optimization," unpublished Ph.D. Dissertation, Northwestern University.

17. Shank, John K., Matrix Methods in Accounting, Addison-Wesley Publishing Company, 1972.

18. Teichroew, D., "Problem Statement Analysis: Requirements for the Problem Statement Analyzer (PSA)," ISDOS Working Paper No. 43, Departmentof Industrial Engineering, University of Michigan, Ann Arbor, 1971.