File: General Electric
REF FILE
CAB. TECH

GENERAL ELECTRIC COMPANY

MANUFACTURING SERVICES

MATERIALS SERVICE

570 LEXINGTON AVENUE, NEW YORK 22, NEW YORK . . . TELEPHONE PLAZA 1-1311

February 8, 1961

Mr. Burt Grad
IBM Corporation
P. O. Box 218
Yorktown Research Center
Yorktown Heights, New York

SUBJECT: GECOM Structure Table

Dear Mr. Grad:

The Integrated Systems Team of the General Electric Company is delighted in the interest being generated in the Structure Table concept. Charley Katz has asked that we transmit a copy of the attached paper, "TABSOL - A Fundamental Concept for Systems-Oriented Languages", to each member of the CODASYL Systems Group in case some of you missed the EJCC Meeting. Charley also asked if we could release some of the previous internal General Electric Company reports which describe the development of the Structure Table concept as a systems analysis tool. After review of the internal reports, it was decided that these contain some proprietary material and shouldn't be released at this time. However, the chronological events leading up to the GECOM Structure Table format are well documented in Mr. Kavanagh's EJCC paper.

If you have any questions on the subject matter in the paper, please contact Mr. Katz.

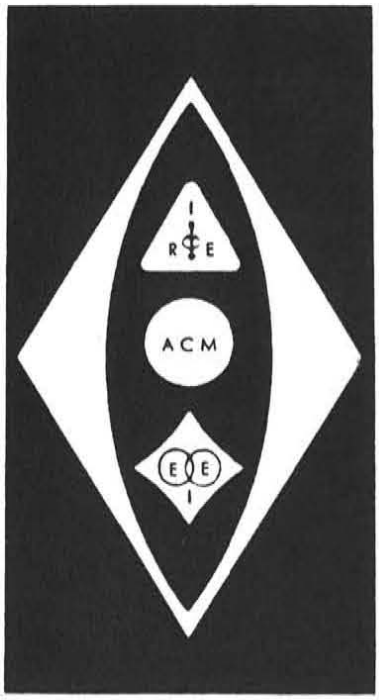
Very truly yours,

Stan Williams

S. B. Williams, Technical Counselor
Integrated Systems Research and Development
PRODUCTION CONTROL SERVICE
Room 2409 - Extension 2549

Attachment

cc: Mr. C. Katz - General Electric Company - Phoenix, Arizona



TABSOL

A FUNDAMENTAL CONCEPT FOR SYSTEMS-ORIENTED LANGUAGES

T. F. Kavanagh
Manufacturing Services
General Electric Company

Presented at the

Eastern Joint Computer Conference
December 14, 1960
Hotel New Yorker
New York, New York

MANUFACTURING SERVICES

GENERAL ELECTRIC

NEW YORK, N. Y.

TABSOL
A FUNDAMENTAL CONCEPT FOR SYSTEMS-ORIENTED LANGUAGES

T. F. Kavanagh

Manufacturing Services
General Electric Company
New York, New York

Summary

Lack of efficient methods for thinking-through and recording the logic of complex information systems has been a major obstacle to the effective use of computers in manufacturing businesses. To supply this need, this paper introduces and describes "decision structure tables," the essential element in TABSOL, a tabular systems-oriented language developed in the General Electric Company. Decision structure tables can be used to describe complicated, multi-variable, multi-result decision systems. Various approaches to the automatic computer solution of structure tables are presented. Some benefits which have been observed in applying this language concept are also discussed. Decision structure tables appear broadly applicable in information systems design.

In addition, they are of interest because they revise many earlier notions on problem formulation and systems analysis technique. Decision structure tables will be an available feature in GECOM, General Electric's new General Compiler, which will be first implemented on the GE 225.

Introduction

Progress in computers can be broadly divided into two categories. First there is the work that essentially accepts computers for what they are, and directs its energies toward further refinement of the original hardware, and operating technique. Research to improve recording density on magnetic tape would certainly fit this description. In the second category are the efforts to advance by developing new areas of application. This latter work is directed toward generalizing the concepts and hardware, so that they apply to an ever-increasing span of problems and situations. Obviously, both groups are vital; but it was this second stimulus -- the desire to expand the area of economic application -- which motivated the research reported in this paper. While the earliest beginnings can be traced as far back as June, 1955, the primary research effort started in November, 1957, under the title of the Integrated Systems Project. Leadership was assigned to Production Control Service, a component in General Electric's Manufacturing Services. The basic purpose of the Project was to probe the potential for automating the flow of information and material in an integrated business system.

Then, as now, computers were making significant contributions in many areas. Unfortunately, one of these areas was not, as some would have it, in the operation and control of manufacturing businesses. Important advances were made in specific applications such as order processing payroll, and inventory record-keeping; but these represented only a small percentage of the total information processing and decision-making in even the smallest manufacturing firm. Still these early successes were very important. They developed confidence in computer performance and reliability; but even more, they encouraged systems engineers and procedures personnel to continue computer applications research. Similarly, management, under growing foreign and domestic competition, rising costs, and a seeming explosion in paperwork requirements, saw intuitively -- or perhaps hopefully -- that computers offered a possible approach to improved productivity, lower costs and sharply reduced cycle times. It was in this environment that the Integrated Systems Project began a comprehensive study of the decision-making and the information and material processing required to transform customer orders into finished products -- a major part of the total business system for a manufacturing firm.

The Decision-Making Problem

Once underway, it was soon apparent

that there was an enormous amount of decision-making required to operate a business. Indeed, the number and complexity of these decisions is perhaps the most widely underestimated and misunderstood characteristic of industrial information systems today. Tens-of-thousands of elementary decisions are made in the typical manufacturing business each working day. All are necessary to guide and control the many functional activities required to design products, purchase raw material, manufacture parts, assemble products, ship and bill orders, and so on. The typical factory is a veritable beehive of decisions and decision-makers; for example:

- "What size fuses shall we use on this order for XYZ Company?"-- a product engineer's decision.
- "What is the time standard for winding this armature coil?"- a manufacturing engineer's decision.
- "What test voltages shall be applied?"-- a quality control planner's decision.
- "What should be the delivery promise on this customer's order?"-- a production control planner's decision.
- "How much will this model cost."-- an accountant's decision.

This list of elementary day-to-day decisions could be expanded to cover all business activities. If this were done, the list would cover hundreds of sheets of paper before each activity listed all the decisions for which it was responsible. Moreover, some of these decisions are repeated many times each day for various sets of conditions. In the end result, one cannot help but be impressed with the multiplicity of these detailed choices and selections. But more importantly, making these decisions costs money, in many cases more money than the direct labor required to make the product. In addition, business performance is greatly affected by the speed and accuracy with which this decision-making is carried out.

Composing a detailed list of these elementary business decisions is more than an academic exercise. For one thing, such an analysis of an actual operating business will demonstrate conclusively that these elementary decisions are handled quite rationally (which is somewhat contrary to popular opinion.) One must be careful not to be misled by quick, superficial explanations which gloss over fundamental reasoning. In our present-day manual systems which emphasize files of quick answers, the logic behind the decision is often left unrecorded. As a result it is easy to lose contact with their rational nature, and frequently we tend to feel these decisions are substantially more intuitive than is actually the case. At times, some persistent as well as penetrating analysis (often through extensive interviewing of the operating personnel presently on the job) is required to uncover the true parameters and relationships on which operating decisions are really based. This arduous work is more than justified, for it establishes a sound conceptual foundation for automation, and hence the practical application of the concepts and techniques developed in this paper. Thus, once it is established that these operating decisions are rational, it should follow that they can be structured in a consistent logical framework. Such a structure is presented in this paper.

Operating vs. Planning Decisions

At this point let us define terminology a little more precisely, and stress that we are speaking about the detailed, elementary decisions required to "operate" a business as opposed to "planning" one. First, a decision in its simplest form consists of selecting one unique alternative from an allowed set of possible actions. Operating decisions are defined in the context of this paper as selecting the appropriate course of action in accordance with given problem conditions to operate the business successfully. Operating decisions may be assumed to be made under "conditions of certainty." The solution for a specific set of problem conditions will always be the same. Under these premises, the action or outcome decided on can always be predicted. In a pragmatic sense, the decision-making process may be classed as "causal"; that is, B may be said to follow from A. For example, an engineer's decision to install fuses might follow from a customer's requirement for independent circuit protection.

The relevant factors or parameters affecting the decision can also be determined. The relationship values are known. For example, in most homes, the current carrying capacity of the house wiring is the only parameter value one needs to know to select an appropriate fuse. In an industrial application, however, the values of at least three additional parameters are usually required: voltage, time and type of fuse mounting. The strategy and the alternate outcomes are known; that is, the per-

missible fuses are known. To continue the illustration, the fuse selection may be limited to those carried in the stockroom; otherwise the bounds of the operating decision system are exceeded and the decision-maker would appeal to a higher authority.

To approach the analysis of operating decisions from another viewpoint, it might be compared to a linear programming problem, and as will become evident, a linear programming solution might be considered as somewhat of a mathematical bound for the class of decision-making systems under discussion.

These operating decisions are quite apart from the planning decisions of a business. The "planning", "administrative", or "policy" decisions in a business are basically those prior commitments which permitted all the assumptions about operating decision systems in the preceding paragraphs (i.e. certainty, causality, known relationships, etc.) Some examples of planning decisions are:

- "Shall fuses, circuit breakers, or both be used on the product line?"-- a product engineer's planning decision.
- "Should this group of parts be made on the screw machine or from die castings?"-- a manufacturing engineer's planning decision.
- "Should this component be inspected before or after the milling operation?"-- a quality control planning decision.
- "What rule shall be used to determine the correct order quantity?"-- a production control planner's decision.
- "What is an appropriate cost-of-money?"-- an accountant's planning decision.

These are typical planning decisions made in designing an operating decision system. To make the distinction clear, consider the design engineer who is motivated by cost considerations to put fuses on the economy part of the product line, while specifying circuit breakers on more deluxe models. Or consider the production control planner who selects one of the common square root formulas for determining all order quantities. Once he puts this decision rule in the operating system, order quantities for every part will be determined using this square root formula with specific values for cost, lead time, usage shelf life, etc., appropriate to the specific item being ordered. Assuming the operating decision system is automatic, and this is the intention, the production control planner need not make any order quantity determinations himself. Rather he will be watching the measures of operating system performance (inventory level, number of shortages, ordering costs, etc.) to see how well his decision rule is working. Incidentally, it's worth noting that the production control systems designer will be using a "cost-of-money" figure supplied by accountants and an annual requirements figure projected by salesmen. Of course, the objective of this fundamental decision analysis is to suggest a conceptual scheme which will permit automating all the routine operating decision-making required to direct a business, thus permitting the engineers, planners, and other technical advisors, to concentrate on doing a better job in design.

Specifying Decision Systems

But great difficulties still remain. As already pointed out, operating decision systems are invariably large and complex, containing multi-variable, multi-result decision problems with sequence of solution difficulties thrown in on the side. One serious problem which arises quickly is the actual development of the decision logic itself. Numerous techniques have been proposed ranging from precise, legalistic verbal statements to complex mathematical equations. Among these however, it appears that matrix-type displays and flow charts are the most common. The matrix-type displays appear under a variety of names: collation charts, tabulated drawings, standard time data sheets, etc. For example, engineers have frequently used collation charts to show direct relationships between end-product catalog numbers and component identification numbers. Typically, however, collation charts are a tabulation of past decisions rather than a description of the logic used to derive them. Matrix-type displays often suffer from redundancy and frequently become large and unwieldy as operating tools. Similarly, they make no allowance to sequential decision-making.

Flow charts handle this sequence problem very nicely. This graphic method describes a decision system by the extensive use of symbols for "mapping" the various operations. A variety of flow chart techniques are used in factory methods and office procedures work.

They are particularly effective in relatively straightforward, sequential decision chains but run into difficulty when describing multi-variable, multi-result decision processes. As an illustration, flow charts have been used extensively to document the detailed logic of computer programs; but some harried computer programming supervisors still maintain that the best way to transfer program knowledge is to reprogram the job. The difficulty of interpreting someone else's flow charts is certainly one of the major trials in today's computer technology.

In addition to these more popular tools numerous other diagramming or charting techniques have been useful in limited problem areas. However, the basic problem remained: there was really no effective, uniform method for thinking about and specifying decision systems as complex as those required to operate a business. To help solve this problem, the Integrated Systems Project developed a new technique which combines key characteristics of earlier methods and adds some new features of its own. This new technique is called the decision structure table. The balance of this paper will describe what decision structure tables are, how they work, and the results of their use in General Electric.

Structure Table Fundamentals

Structure tables provide a standard method for unambiguously describing complex, multi-variable, multi-result decision systems. Thus, each structure table becomes a precise statement of both the logical and quantitative relationships supporting that particular elementary decision. It is written by the functional specialist in terms of the criteria or parameters affecting the decision and the various outcomes which may result.

A structure table consists of a rectangular array of terms, or blocks, which is further subdivided into four quadrants, as shown in Figure 1. The vertical double line separates the decision logic on the left from the result functions or actions which appear on the right. The horizontal double line separates the structure table column headings or parameters above from the table values recorded in the horizontal rows below. Thus, the upper left quadrant becomes decision logic column headings, and is used to record, on a one per column basis, the names of the parameters (P_{0j}) effecting the decisions. The lower left quadrant records test values (p_{ij}) on a one per row basis, which the decision parameter identified in the column heading may have in a given problem situation. The upper right hand quadrant records the names of result functions or actions to be performed (R_{0j}) as a result of making the decision, once again on a one per column basis. Similarly the lower right quadrant shows the specific result values (r_{ij}) which pertain, directly opposite the appropriate set of decision parameter values. Thus, one horizontal row completely and independently describes all the values for one decision situation.

There is, of course, no limit to the number of columns (decision parameters and result functions) in any given structure table. Even the degenerate case where the number of decision parameters goes to zero is permissible. Also there is no limit on the number of decision situations (rows). Thus, the dimensions (columns by rows) of any specific structure table are completely flexible, and are a natural outgrowth of the specific decision being described. A series of these structure tables taken in combination is said to describe a decision system.

Rather than become further involved in abstract notation, let's consider some actual illustrations to develop an insight into the nature of structure tables. For example, the oversimplified illustrative structure table in Figure 2 states that an elementary decision on transportation from New York to Boston in the afternoon is (according to the person who developed the decision logic) a function of three decision parameters: Weather, Plane Space, and Hotel Room. Weather has only two value states, Fair or Foul; Place Space is either OK or Sorry; and Hotel Room can be Open or Filled. In terms of results, Plane or Train are the only permissible means of Transportation. Following the illustrative problem, we see by inspection that the solution appears in the second row. Therefore, Train is the correct value for Transportation, Other Instructions are Cancel Plane, and this is the End of the decision problem.

The intent of this simple structure table is to provide a general solution to this particular decision situation, and if the problem of afternoon trips to Boston ever arises (and one assumes that it frequently does), then an operating decision can quickly be made by supplying the current value of Weather, Plane Space, and Hotel Room, and, of course, solving the structure table. Solving a structure table consists of examining the specific values assigned the decision parameters in the problem statement and comparing or "testing" these values against

the sets of decision parameter values recorded in the structure table rows. Testing proceeds column by column from the first decision parameter to the last (left to right) and thence row by row (top to bottom). If all tests in a row are satisfied, then the solution is said to be in that row and the correct result values appear in the same horizontal row directly opposite to the right of the double line. When a test is not satisfied, the next condition row is examined.

When a particular structure table has been solved, it is often necessary to make more decisions. To specify what decision is to be made next, the last result column of the structure table may be assigned as a director to provide a link to the next structure table. Notice the last row in the illustrative structure table which specifies that for any value of Weather, with no Plane Space, and no Hotel Room, the decision-maker is directed to solve the next structure table, Transportation, New York-Boston, a.m. -- which is another structure table describing how to select a means of transportation in the morning.

In a similar fashion, the systems designer would use a whole system of structure tables to describe a more realistic operating decision problem. He completely controls the contents of each table, as well as its position in the sequence of total problem solution. He may decide to skip tables, or, if desired, he may resolve tables to achieve the effect of iteration. In any event, the entire system of tables, just as each individual structure table, will be solved using specific decision parameter values appearing in the problem statement. In other words, solving a set of structure tables consists essentially in re-applying the systems designer's operating decision logic.

Having completed this quick and very simplified introduction to structure tables, let us now return to consider each structure table element in greater detail. This will provide a deeper insight into the power of the structure table technique, as well as a better understanding of how they are used to describe operating decision systems. The illustrations are drawn from actual operating decision problems.

Structure Table Tests

Comparisons or tests between problem parameter values (pv) and decision parameter test values (tv) need not be simple identities, such as those used in the previous illustration. Actually the problem parameter values may be compared to the decision test values in any one of the following ways in any structure table block:

EQ	pv = tv	problem value is equal to test value.
GR	pv > tv	problem value is greater than test value.
LS	pv < tv	problem value is less than test value.
NEQ	pv ≠ tv	problem value is not equal to test value.
GREQ	pv ≥ tv	problem value is greater than or equal to test value.
LSEQ	pv ≤ tv	problem value is less than or equal to test value.

This broad selection of test types (or relational operators as they are known technically) greatly increases the power of individual structure tables and sharply reduces size. It permits testing limits or ranges of values rather than only discrete numbers. In Figure 3, TABLE 1000 uses several difference test types to bracket continuous and discontinuous intervals. Also note in Figure 3, that the relational operator may be placed in the test block immediately preceding the test value, or in the column heading immediately following the decision parameter name. When this latter notation is used, the relational operator in the column heading applies to all test values appearing immediately below.

Test values are not limited to specific numbers on alphanumeric constants (indicated by quotation marks); a test block may also refer to the contents of any name. In this case of course, the current contents of that named field are compared to the problem parameter value in accordance with the test type. For example, TABLE 1005 in Figure 3 tests the current value of INSUL~TEMP against MAX~TEMP to make certain that insulation temperature ratings are satisfactory.

In addition to these simple comparisons it is also possible to formulate compound structure table blocks involving two decision parameters or test values using a relational or logical operator.

The following logical operators may be used:

OR	tv_1 OR tv_2	first test value or the second test value.

AND	pv_1 AND pv_2	first problem value and second problem value.
NOT	tv_1 NOT tv_2	first test value and not second test value.

Also the truth or falseness of a compound decision parameter or test value statement can be tested with the symbols:

T	true
F	false

Lastly, any arithmetic expression may be used in place of a parameter name, and complicated blocks involving several names and operators are also permitted. Although in this latter case, it is worth noting that the language capability far surpasses any requirements experienced to date in formulating operating decision systems.

In writing structure tables, the situation often arises where, except for one or two special situations, one course of action is adequate for all input values. The concept of an "all other" row was introduced to avoid enumerating all possible logical combinations of the decision parameter values. The "all other" concept can be verbalized as follows: "If no solution has been found in the table thus far, the solution is in this last row regardless of the problem values." While this greatly reduces table size, it also implies that the problem was stated correctly and does indeed lie within the boundaries of the decision system. The related concept of "all" which appears in the Transportation: New York-Boston, p.m. can be similarly verbalized: "regardless of the problem value proceed to the next column." It was introduced so that a given table need not contain all permissible states of any given decision parameter and also to handle the case where a test in a given column had no significance. In all the above situations the appropriate structure table blocks are left blank signifying no test.

Structure Table Results

Similarly structure table results are not limited to assigning alphabetic constants or numeric values to the result functions or actions named in column headings to the right of the double line. Actually there are four result functions:

"ASSIGN" - which is implied when a named field appears as a result function. This indicates that the result value appearing in (or named by) the solution row is to be assigned or placed in the field named in the column heading.

"CALCULATE" - which is implied by the use of an equal sign after a name appearing as a result value. This indicates that the results of the formula evaluation named in the structure table block should be assigned to the field named as the result function in the column heading. Actually this is not the only way to perform calculations as any arithmetic expression may be used as a result value.

PERFORM - which performs the data processing or arithmetic operations referred to in the label appearing in the result value block. When this is completed, the next result function is executed.

GO - links the structure table to the label appearing in the result value block. There is no implied return in a GO function.

Most of these result functions are illustrated in Figure 3 and Figure 4. In Figure 4, for example, TABLE 2000 assigns the alphabetic constant "FLAT-STRIP" to ASSEMBLE. In the first and third result columns, arithmetic expressions appear as result values. In TABLE 2005 the implied CALCULATE is used for formula evaluation. TABLE 2005 also uses the PERFORM function to solve TABLE 2008 or carry out some other data processing operations depending on the particular solution row. TABLE 2005 is linked by the GO operation to TABLE 2010, 2015, 2020.

TABLE 1005 in Figure 3 shows an interesting use of the GO function. After the winding has been specified in TABLE 1000, assumedly on a lowest cost basis, the product engineer evidently wants to check the insulation temperature rating with the maximum expected operating temperature. If the insulation temperature rating should turn out to be greater everything is fine and the decision-maker proceeds to TABLE 1007. If not, first TYPE-N and then TYPE-T insulation are specified to

supercede TYPE-F, thus getting progressively higher insulation temperature ratings by redirecting the structure table to solve itself.

Frequently, a result function or action will not have a value for all rows. This is common when several result functions are determined by the same structure table. In this situation the phrase "not exist" has been used in verbalizing and the structure table block is left blank.

The use of formulas as structure table results can greatly reduce the size of the table. As an illustration, suppose that a given result function has twenty-six values (10, 12, 14 16, ...60). Ostensibly, the structure table to select the appropriate result value would have twenty-six rows. This decision could be reduced to one row by calculating the result value as some function of the decision parameter as shown in Figure 6. Obviously, all result relationships are not so conveniently proportional but a surprising number of result functions can be described with simple linear and exponential expressions. The curve fitting problem can be greatly simplified by using structure table rows to break the curve into convenient intervals that can be represented by such simple mathematical expressions.

Preambles and Postscripts

Each structure table is preceded by a heading which identifies the table by number and indicates its dimensions in terms of decision parameter columns, result function or action columns, and value rows. Tables may be numbered from TABLE 1 to TABLE 9999999 and allowance is made for up to 999 decision parameter or result functions. Provision is also made for 999 condition rows.

Following the heading is a NOTE which may contain any combination of alphabetic or numeric characters. The NOTE may be used to give the structure table an English name and provide a verbal description of the decision being made. Subsequent to this any labels naming expressions or arithmetic calculations referred to by "CALCULATE" or PERFORM operators in the body of the structure table may be defined. For example, note the definition of TIME ~ 1 and TIME ~ 2 in TABLE 2005 of Figure 4. The structure table proper follows BEGIN.

If no solution row is found in the structure table proper, or if the structure table has executed all results or taken all actions without reaching a GO function then control is passed to the area directly below the structure table. Here are recorded any special instructions pertaining to that particular decision. Of particular note is the situation where no solution row has been found. Such a failure is regarded as an "error." In certain types of decision systems, this may be exactly what the systems designer intended. However, error conditions most often indicate a failure of the decision logic to cope with a certain combination of input values. The systems designer should set up to notify himself whenever such an error occurs by designing an error routine which will provide him with a source language printout identifying the table that failed and the problem being solved at the time. With this problem printout and the source language structure tables, the systems designer has all the data he needs to trouble shoot the system in his own terminology. Thus, each structure table should be followed by the statement: IF NOT SOLVED GO_____. In this way any structure table failures will always be uncovered. Frequently, the situation arises, as mentioned earlier, that regardless of the solution row, the next structure table solved is the same. In this case the statement: GO_____, may be written after or below the preceding error statement, to serve as a universal link to the next structure table.

The areas immediately preceding and succeeding the structure table proper may also be used for input-output, data movement, and other data processing operations.

The Dictionary

The precise name and definition of each decision parameter and result function are recorded in a "dictionary." This dictionary becomes an important planning document in the systems engineer's work for it provides the basic vocabulary for communicating throughout the entire decision system. The dictionary should note a parameter's minimum and maximum values, as well as describe how it behaves. If the parameter is non-numeric in nature, the dictionary should record and define its permissible states. Significantly, the systems engineer formulates both the structure table and the dictionary using his own professional terminology.

The dictionary will also prove useful in compiling and editing structure tables for computer solution. It also follows that problems presented to the resulting operating decision system must also be stated in precisely the same terms as the structure tables. To those as yet uninitiated to the perversity of computers, this may seem a simple matter; unfortunately,

it is not. Interestingly however, one of the more promising application areas for structure tables appears to be in stating the logic for compilers and edit programs.

Summary

The foregoing description of decision structure tables is not meant to be a fully definitive language specification. The intention is to introduce the reader to the decision structure table concept and to discuss their characteristics in sufficient detail to provide the reader with enough understanding to evaluate their inherent flexibility and application potential. Many additional features are available which aid in formulating concise, complete decision structure table systems and also to facilitate input-output operations, but the reader will find that the fundamentals already described are adequate for structuring most operating decision logic.

Automatic Solution of Structure Table Systems

Decision structure tables have proven to be an excellent method for analyzing or formulating the logic of complex industrial information systems, but after taking such great care to precisely record each elementary decision in this highly structured format, it is only natural to speculate on the possibility of solving structure tables automatically with an electronic computer. Before plunging into the computer world, however, it is worth noting that some systems engineers have had very favorable experience using structure tables on a manual basis -- especially as a problem analysis technique, and also in limited applications in manual clerical systems.

Numerous methods for solving structure tables automatically suggest themselves. First, the tables could be coded by hand. Such an approach would use structure tables as a direct substitute for flow charts. Actually this really isn't as bad as it initially sounds. Many benefits would accrue from making this precise readable format the standard method for stating decision logic. It also offers the possibility that a series of macro-instructions could be developed, thereby permitting untrained personnel to code tables without detailed knowledge of computers or programming. However, this approach suffers some distinct disadvantages in comparison with the other alternatives outlined below.

Second, a generalized interpretive program could be written to solve any structure table. This offers the possibility of using a translator to work directly from keypunched structure tables without any manual detail coding. This approach makes economical use of memory since the basic programming to solve any table appears only once and the structure table itself offers a compact statement of decision logic. This reduces the amount of reading time required to bring the problem logic into the computer. File maintenance via recompiling structure table tapes also appears quick and simple. However, interpretive programs usually run more slowly; and this implies some penalty in total machine running time.

A third approach would be to create a structure table program generator in which an object computer program would be generated from the source structure tables. This approach would provide faster computer running times for maximum efficiency. A generator program would probably require more complicated coding than an interpretive translator. In addition, the generated object program would not be as concise as the structure tables themselves. However, where computer running time is of paramount concern, this approach has considerable appeal.

Because of the available time and money, all the early efforts of the Integrated Systems Project toward automatic structure table solution were essentially interpretive. It is interesting that a simple, yet adequate, tabular systems-oriented language could be provided in this way for somewhat less than a man year's effort. Similarly work to date in the area of formula calculations indicates that a comprehensive system of mathematical notation like that required for scientific work is probably not necessary in many operating business decision systems. Initial efforts on the IBM 702 were followed with experimental TABSOL languages for the IBM 305, IBM 650 and the IBM 704. These applications to different computers represented more than simple extrapolations to different pieces of hardware. In each an effort was made to expand capabilities of the language. In addition, the peculiarities of the equipment were explored, since one great concern was to free the user from a programming system usable on one and only one computer. As you might suspect, this wasn't always completely possible on the smaller computers, lacking tape or core memories. Nevertheless, the most recent Manufacturing Service effort on the IBM 650 produced a language with named fields, indexing, a two-address arithmetic, completely generalized structure table formats, and considering the alphabetic restrictions of the

equipment, remarkably flexible output formats.

Although these experimental languages proved quite adequate, one could not help but look toward the tremendous power of one of the more conventional languages. For one thing, the prospects for structure table application in other problem areas brightened, and it seemed reasonable that this power would be desirable in future work. Further our own tabular systems language development had brought us to the point of direct competition with the major language efforts already underway. Here General Electric's Computer Department entered on the scene. The Computer Department was developing a new concept in compiler building for use with General Electric computers. The first version of this new General Compiler, called GECOM, will be available to GE 225 users in May, 1961. It is designed primarily around COBOL, with some of the basic elements of ALGOL, and is now to contain all of TABSOL. To state the results of joining TABSOL with GECOM simply, it places the power of a full-fledged language at the command of every structure table block. Within General Electric, we obviously have a very high regard for the contribution of decision structure tables in information systems design. Significantly, the same committees who developed COBOL are now actively investigating tabular systems-oriented languages as the language of the future. By drawing on the CODASYL work and utilizing the extensive research and development experience already available within General Electric, the Computer Department expects that GECOM will provide users with a system compatible with both the present-day common business language, COBOL, and also the tabular systems-oriented language, TABSOL. Incidentally, the decision structure tables appearing in Figures 3, 4 and 5 are written in conformance with GECOM specifications.

Applications of Structure Tables

As somewhat implied in the illustrations a substantial amount of experience has been gained in applying structure tables to a wide variety of operating decision-making problems over the past three years. But perhaps the most interesting experience, at least from the researcher's point of view, was the very research work which spawned decision structure tables themselves. Earlier, it was mentioned that the Integrated Systems Project undertook a careful study of the essential information and material processing required to directly transform customer orders into finished products. For example, the product must be engineered prior to shipment, but the payroll, though revered by all of us can well be done at some other time, out of the main flow of events. Using this rough rule of thumb, the following activities were studied (Figure 7): order editing, product engineering, drafting, manufacturing methods, and time standards, quality control, cost accounting, and production control. These activities account for a fairly substantial portion of the business system. Normally, they would include 100% of the direct labor and 100% of the direct material as well as about 50% of the overhead. All the production inventory investment lies within the scope of this system and obviously most of the plant and equipment investment. Fortunately, the inputs and outputs to this system are simple and well-defined: the customer order comes in and the finished product goes out. With this in mind, it was possible to treat all activities within these bounds as one integrated, goal-oriented operating decision system and develop decision structure tables accordingly. Working with a small product section in one of the Company's Operating Components, a significant portion of the functional decision logic was successfully structured. Further the resulting structure tables were directly incorporated into a computer-automated operating decision system which transformed customer orders for a wide variety of finished products directly into factory operator instructions and punched paper tape to instruct a numerically programmed machine tool. This prototype system was demonstrated to General Electric management in November, 1958. Starting at the beginning, (Figure 8) the computer system edited the customer order and using the product engineer's design structure tables, developed the product's component characteristics and dimensional details. These in turn were used in the manufacturing engineer's operation structure tables to develop manufacturing methods and determine time standards. And so the flow of information cascaded down through the various business functions computing the quality control procedures, the product costs and the manufacturing schedules; eventually issuing shop paperwork and machine program tapes.

Since the completion of this work further research and development of the structure table concept was conducted in a variety of functional areas for different kinds of businesses in General Electric: defense, industrial apparatus, and consumer-type products. In addition, structure tables have been used in entirely different applications such as compilers. They also appear to hold great promise in complex computer simulation programs.

Benefits of Structure Tables

As a result of these efforts, we have come to believe that the decision structure table is a fundamental language concept which is broadly applicable to many classes of information processing and decision-making problems. They offer many benefits in learning, analyzing, formulating and recording the decision logic:

1. Structure tables force a logical, step-by-step analysis of the decision. First the parameters affecting the decision must be specified; then suitable results must be formulated. The nature of the structure table array is such that it forces consideration of all logical alternatives, even though all need not appear in the final table. Similarly, the precise structure table format highlights illogical statements. This simplifies manual checking of decision logic. The decision logic emphasizes causal relationships and constantly directs attention to the reasons why results are different. Personal design preferences can be resolved and intelligent standardization can be fostered.

   This is no mean capability. Indeed, it was very instructive to witness the development of methods and time standards logic in parallel with the development of the engineering logic during the initial Integrated Systems Project study. Through analysis of the decision structure tables written by the various functional specialists, everyone was able to achieve an insight into the product and the business rarely obtained in so short a period of time. The facts of life in product design, factory methods, and standardization were brought into the open very rapidly.

2. Structure tables are easily understood by humans regardless of their functional background. This does not imply that anyone can design or create new structure tables to describe a particular decision-making activity; but it does mean that the average person, with the aid of a dictionary, can readily understand someone else's structure tables. Thus, structure tables form an excellent basis for communication between functional specialists and systems engineers. Structure tables also go a long way toward solving the difficult systems documentation problem.

3. Structure table format is so simple and straightforward that engineers, planners, and other functional specialists can write structure tables for their own decision-making problems with very little training and practically no knowledge of computers or programming. Given a few ground rules, regarding formats and dictionaries, the structure tables written by these functional people can be keypunched and used directly in operating decision systems without ever being seen by a computer programmer. This cuts computer application costs as well as cycle times.

4. Structure table errors are reported at the source language level, thus permitting the functional specialist to debug without a knowledge of computer coding.

5. Structure tables solved automatically in an electronic computer offer levels of accuracy unequalled in manual systems. Note, however, that any such mechanistic systems lose that tremendous ability of humans to compensate for errors or discrepancies.

6. Structure tables are easy to maintain. Instead of changing all the precalculated answers in all the files, it is often only necessary to change a single value in a single table. For example, when changing the material specified for a component part under current file reference systems, it would be necessary to extract, modify and refile all drawings and parts lists calling for any variation of the component part. Using structure tables, it would only be necessary to alter those structure tables which specified the component material.

Summary

In closing, we recommend that the reader demonstrate the effectiveness of decision

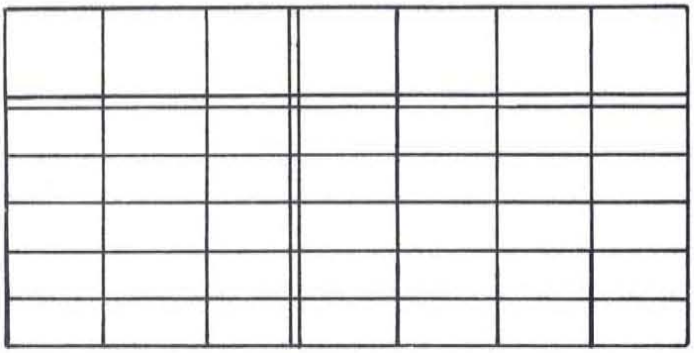
structure tables to himself by "structuring" a few simple decisions. For example, write a structure table which will enable your wife to decide how to pack your suitcase of any business trip. Perhaps a simple business decision such as those mentioned earlier would provide a more instructive example. The first structure tables are usually difficult to write, because most of us do not, as a general rule, probe deeply into the logic supporting our decisions. However, once this mental obstacle is overcome, "structuring" facility develops rapidly. If the reader will take the time to "structure" a few decisions and actually experience the deeper insight and clarity which this technique provides, then decision structure tables need no apologist, they will speak for themselves.

Acknowledgement

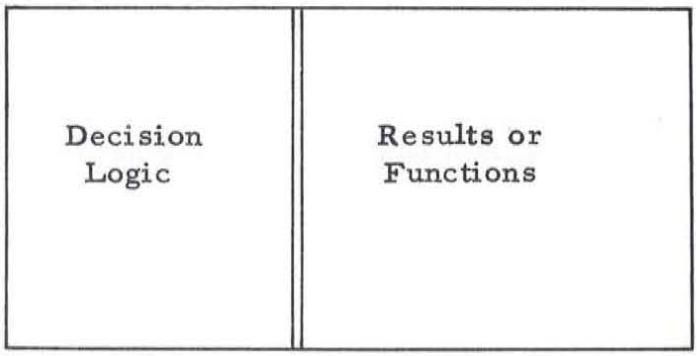
In contrast to most technical papers which essentially document only the work of the author, this discussion reports on the efforts of over seventy-five General Electric men and women. In particular, credit is due Mr. Burton Grad, who though no longer with General Electric, was a principal originator of the decision structure table concept. Mr. Malcolm C. Boggs, Mr. Daniel F. Langenwalter, Mr. Herbert W. Nidenberg, and Mr. Theodore E. Schultz representing Service Components and personnel from some fifteen different Operating Components within General Electric have contributed toward bringing these ideas to their present state of development and application. Acknowledgement is also due Mr. Charles Katz of General Electric's Computer Department who was instrumental in joining TABSOL and GECOM.

Decision Structure Table

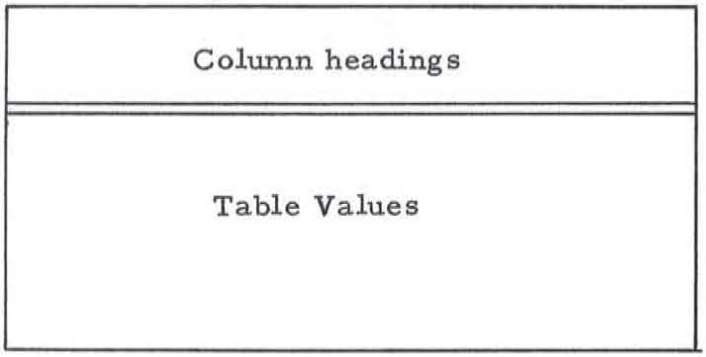
...a rectangular array of terms, or blocks...



...vertical double line...



...horizontal double line...



...structure table values...

P_{01}	P_{02}	P_{03}	R_{01}	R_{02}	R_{03}	R_{04}
p_{11}	p_{12}	p_{13}	r_{11}	r_{12}	r_{13}	r_{14}
p_{21}	p_{22}	p_{23}	r_{21}	r_{22}	r_{23}	r_{24}
p_{31}	p_{32}	p_{33}	r_{31}	r_{32}	r_{33}	r_{34}
p_{41}	p_{42}	p_{43}	r_{41}	r_{42}	r_{43}	r_{44}

Figure 1

Problem Statement: Select Transportation, New York - Boston, p.m.

Weather: Foul

Plane Space: OK

Hotel Room: Open

Decision Structure Table: Transportation, New York - Boston, p.m.

Weather	Plane Space	Hotel Room	Trans-portation	Other In-structions	Next Decision
Fair	OK	Open	Plane		End
Foul	OK	Open	Train	Cancel Plane	End
	Sorry	Open	Train		End
	OK	Filled		Cancel Plane	NY-Bost. a.m.
	Sorry	Filled			NY-Bost. a.m.

Solution:

If the value of Weather is Foul, and

the value of Plane Space is OK, and

the value of Hotel Room is Open,

Then

the value of Transportation is Train, and

the value of Other Instructions is Cancel Plane, and

the value of Next Decision is End.

Figure 2

TABLE 1000. DIMENSION C4 A5 R10.
NOTE TABLE FOR DETERMINING DETAIL VARIABLE PART CHARACTERISTICS FOR A LINE OF SENSING COILS IN ACCORDANCE WITH CUSTOMER END PRODUCT SPECIFICATIONS.
BEGIN.

SERVICE EQ	UNITS EQ	VALUE	VALUE	TURNS	DIA	RESIST	INSUL	INSUL ~TEMP
"DC"	"MAMP"	GR 180	LS 450	0.3/I	.001	2.6*TURNS	"TYPE-F"	150
:	:	:	:	:	:	:	:	:
"DC"	"MVLT"	GREQ 45	LSEQ 150	26	.008	1.84	"TYPE-F"	150
"DC"	"MVLT"	GR 150	LSEQ 330	13	.002	0.46	"TYPE-F"	150
"DC"	"VOLT"	GREQ 0.9	LSEQ 300	60	.002	39.0	"TYPE-F"	150
"DC"	"VOLT"	GR 300	LSEQ 1100	120	.002	137.0	"TYPE-F"	150
:	:	:	:	:	:	:	:	:
"AC"	"WATT"			230	.002	150.0	"TYPE-N"	200

IF NOT SOLVED GO ERROR~COIL.
MOVE "COPPER" TO MATERIAL.
GO TABLE 1005.
END TABLE 1000.

TABLE 1005. DIMENSION C2 A3 R3.
NOTE TABLE TO MAKE CERTAIN THAT INSULATION TEMPERATURE RATING EXCEEDS MAXIMUM OPERATING TEMPERATURE.
BEGIN.

MAX~TEMP	INSUL	INSUL	INSUL~TEMP	GO
LSEQ INSUL~TEMP				TABLE 1007
GR INSUL~TEMP	"TYPE-F"	"TYPE-N"	200	TABLE 1005
GR INSUL~TEMP	"TYPE-N"	"TYPE-T"	250	TABLE 1005

IF NOT SOLVED GO ERROR~COIL.
END TABLE 1005.

Figure 3

TABLE 2000. DIMENSION C3 A3 R4.
NOTE TABLE TO SPECIFY VARIABLE FACTORY OPERATION CHARACTERISTICS FOR THE INITIAL SENSING COIL WINDING FROM PART CHARACTERISTICS.
BEGIN.

SUPPORT~TYPE EQ	MATERIAL EQ	TURNS	START~W	ASSEMBLE	FINISH~W
"TABED-HOLE"	"COPPER"		TURNS		
"FLAT-STRIP"	"COPPER"	LS 100	2	"FLAT-STRIP"	TURNS-2
"FLAT-STRIP"	"COPPER"	GREQ 100	TURNS/2	"FLAT-STRIP"	TURNS/2
"FLAT-STRIP"	"ALUMNM"		TURNS	"2 FLT-STRP"	

IF NOT SOLVED GO ERROR~COIL. GO TABLE 2005.
END TABLE 2000.

TABLE 2005. DIMENSION C2 A3 R3.
NOTE TABLE TO CALCULATE TIME STANDARD FOR PREVIOUS OPERATION.
TIME~1 = 125*DIA*TURNS.
TIME~2 = 1000*DIA/SQRT (TURNS).
BEGIN

TURNS	TURNS	TIME	PERFORM	GO
LS 15		TURNS + 0.88	SETUP	TABLE 2010
GREQ 15	LS 100	TIME~1 =	SETUP	TABLE 2015
GREQ 100		TIME~2 =	TABLE 2008	TABLE 2020

IF NOT SOLVED GO ERROR~COIL.
GO TABLE 2005.

Figure 4

```
TABLE 1010.  DIMENSION C2 A1 R3.
NOTE  COIL QUANTITY DETERMINATION.
BEGIN.
```

SERVICE EQ	UNITS NEQ "WATTS"	COIL~QUAN
"AC"		0
"DC" OR "AC"	T	QUAN
"DC"	F	2*QUAN

```
IF NOT SOLVED GO ERROR~COIL.  GO TABLE 1100.
END TABLE 1010.

TABLE 1500.  DIMENSION C4 A3 R10.
NOTE  COIL LOAD DATA AND CYCLE TIMES.
BEGIN.
```

SERVICE EQ	UNIT EQ	ACY EQ	INSP EQ	NORM~CYCLE	MIN~CYCLE	COIL~LOAD
"AC"	"AMPS" OR "MAMP"	1	"COML"	15	11	QUAN
"AC"	"WATT"	1	"COML"	15	11	2.2* QUAN
.	.	.	.	.	.	.
"DC"	"AMPS" OR "MAMP"	2	"COML"	15	9	0.9* QUAN
"DC"	"VOLT" OR "MVLT"	2	"COML"	15	9	0.9* QUAN
"DC"	"AMPS" OR "MAMP"	1	"GOVT"	20	16	1.4* QUAN

```
IF NOT SOLVED GO ERROR~COIL.
MIN~DATE = TODAY + MIN~CYCLE.
NORM~DATE = TODAY + NORM~CYCLE.
GO TABLE 1510.
END TABLE 1500.
```

Figure 5

TABLE 1510. DIMENSION C2 A2 R3.
NOTE COIL PROMISE DATE DETERMINATION.
BEGIN.

COIL~LOAD LSEQ	CUST DATE	PROMISE	GO
CUM~CAP (NORM~DATE)	GREQ NORM~DATE	CUST~DATE	NORM~LOAD
CUM~CAP (MIN~DATE)	GREQ MIN~DATE	CUST~DATE	RUSH~LOAD
CUM~CAP (CUST~DATE)		CUST~DATE	EMER~LOAD

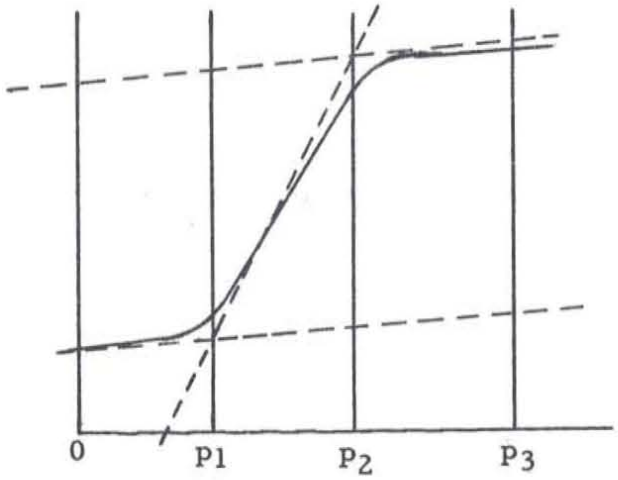
IF NOT SOLVED GO OVERLOAD.
END TABLE 1510.

Figure 5a

P	R
0	10
1	12
2	14
3	16
.	..
.	..
.	..
.	..
25	60

P	P	R
0	25	(2*p) + 10

.... The use of formulas as structure table results can greatly reduce structure table size, as shown by the simple straight line expression above. Structure tables may also be used to partition complicated curves into convenient segments as shown below. ....



P	P	R
0	P_1	lx + a
P_1	P_2	mx + b
P_2	P_3	nx + c

Figure 6

PRESENT MAIN LINE SYSTEM

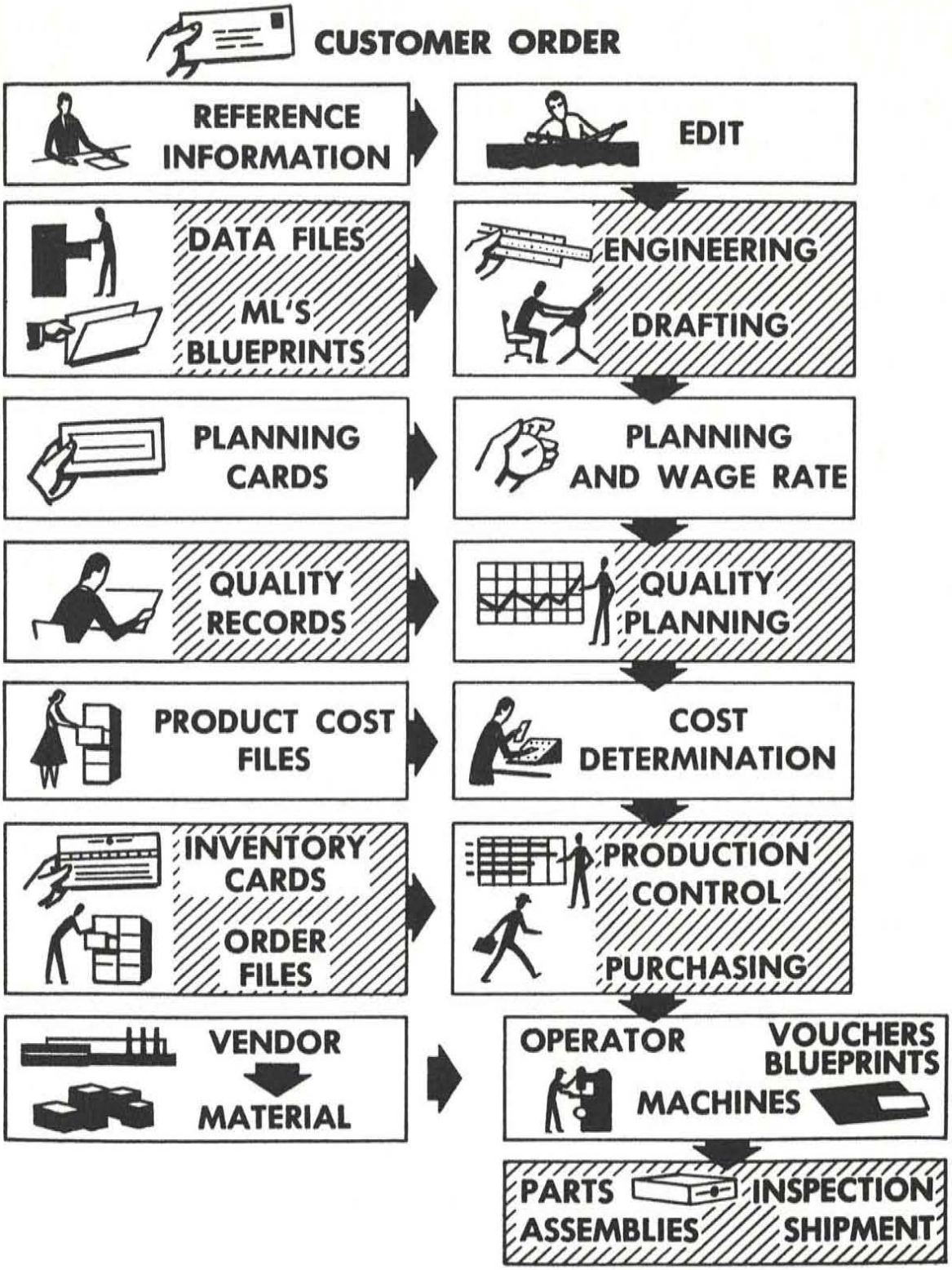


Figure 7

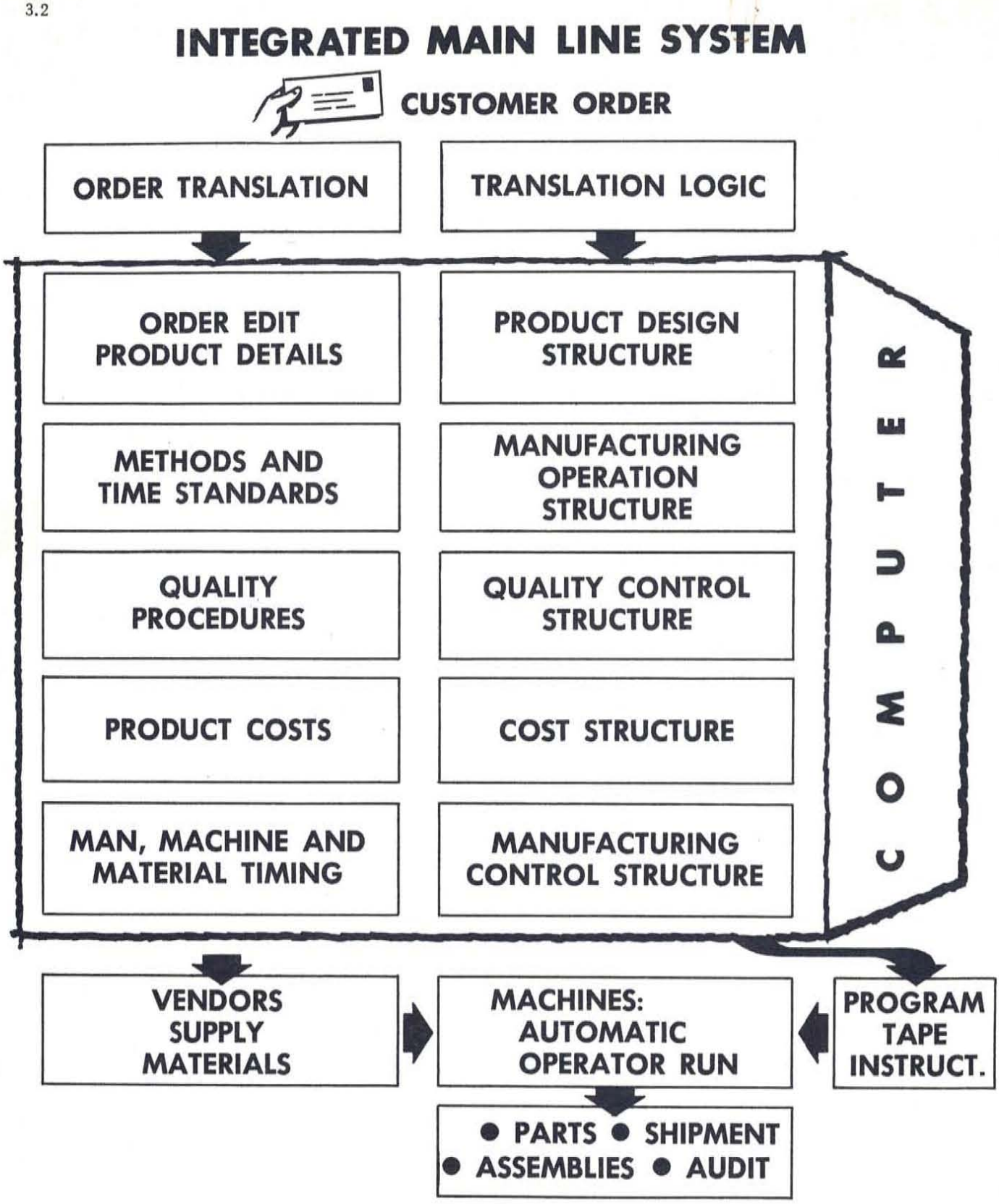

INTEGRATED MAIN LINE SYSTEM
CUSTOMER ORDER
ORDER TRANSLATION
TRANSLATION LOGIC
ORDER EDIT PRODUCT DETAILS
PRODUCT DESIGN STRUCTURE
METHODS AND TIME STANDARDS
MANUFACTURING OPERATION STRUCTURE
QUALITY PROCEDURES
QUALITY CONTROL STRUCTURE
PRODUCT COSTS
COST STRUCTURE
MAN, MACHINE AND MATERIAL TIMING
MANUFACTURING CONTROL STRUCTURE
COMPUTER
VENDORS SUPPLY MATERIALS
MACHINES: AUTOMATIC OPERATOR RUN
PROGRAM TAPE INSTRUCT.
● PARTS ● SHIPMENT
● ASSEMBLIES ● AUDIT

Figure 8