TABLE OF CONTENTS

Scheduling - February 15, 1954	B. Grad
Information Process Charting - March 15, 1957	B. Grad R. G. Canning R. H. DeRubbo
Information Process Analysis -November 1959	B. Grad R. G. Canning
A Systems Approach to Integrated Systems Planning - February 1, 1960	B. Grad
The Integrated System Project at General Electric - March 31, 1960	B. Grad
CODASYL (Proposals for a Systems Engineering Language) - April 15 & 25, July 11, 1960	B. Grad
The "Explosion" Operator as used in Production Control - May 15, 1960	B. Grad
Transformation Logic-December 29, 1960	B. Grad
A Model of an Elementary Industrial Processing System - March 1, 1960	B. Grad
Tables Signal Better Communication -GUIDE Presentation - June 1, 1961	B. Grad
Tabular Form in Decision Logic-Datamation Magazine Reprint-July 1961	B. Grad

SCHEDULING

In order to determine an optimum input schedule it is necessary to have some means of testing various schedules and to develop a plan or technique for finding the best one.

The most straight forward approach to the evaluation of a schedule would seem to be the creation of a mathematical model which would simulate the behavior of the factory under study. Then, the next step would be an evaluation based upon the results of the factory model operating on a certain input schedule.

With the factory model and the evaluation equation determined, the only remaining variable is the input schedule itself. In the factory, output is normally specified in terms of customer wants or anticipated wants; this usually establishes a quantity of a specific product desired at a specific date. Therefore, the end result, if within the capacity of the shop, is completely specified and flexibility exists only in the starting date of a lot through the shop and in the priority system used within the factory to determine individual job sequence.

Let us look at each of these three phases of schedule determination separately; for convenience we shall start with the evaluation, then discuss the factory model and finally the variation of the input schedule itself. In general the discussion will be centered on a job control operation, since, with either a batch control or a flow control set-up the problem becomes simpler and easier to analyze.

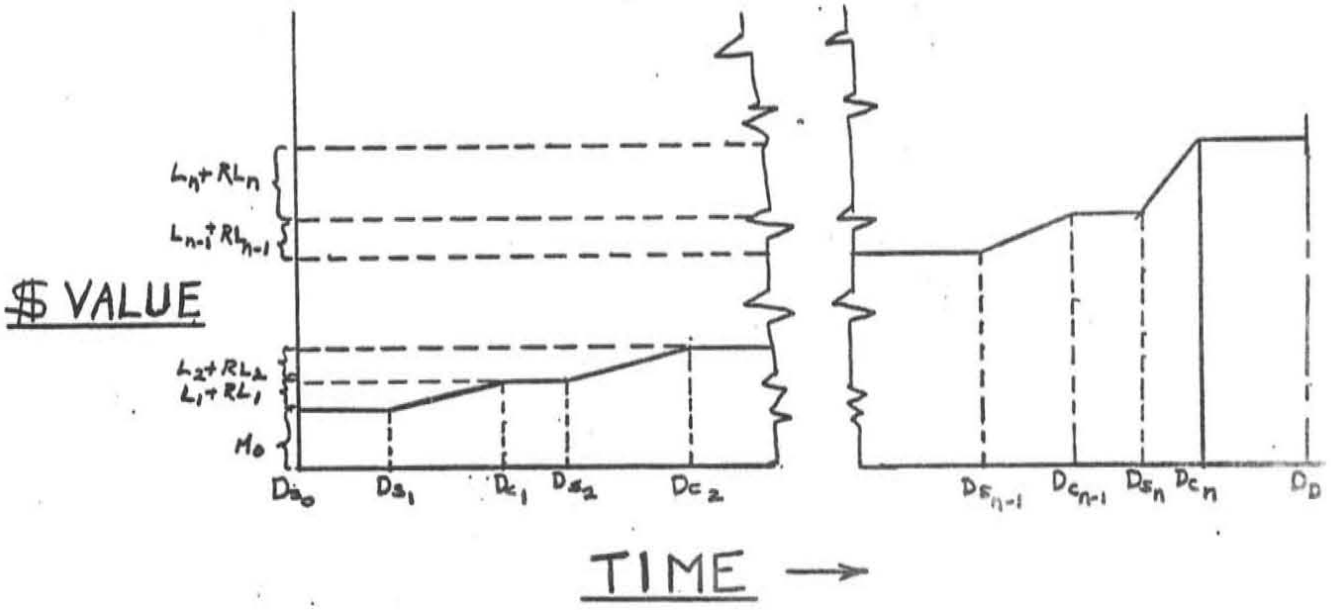
EVALUATION

In evaluating the effects of an input schedule on the factory four measuring sticks seems most significant:

1. Inventory Carrying cost for the lot being manufactured.
2. Equivalent penalty for late completion of the lot.
3. Non-productive man-machine time.
4. Set-up costs.

There is a fifth function - that of the cost of scheduling itself; however, this tends to be inversely dependent upon the lot size which for the sake of simplicity, is being omitted from this discussion. Therefore, it will be assumed that the actual cost of preparing a schedule will be identical in all cases.

The cost of carrying inventory through the entire cycle can be represented in two different ways. The more accurate is as follows:



$$\text{Cost of Carrying INV.} = I\Big\{(D_{c_n} - D_{s_o})\,M_o + (D_{c_1} - D_{s_1})(L_1 + RL_1)\tfrac{1}{2} + (D_{c_n} - D_{c_1})(L_1 + RL_1) + (D_{c_2} - D_{s_2})(L_2 + RL_2)\tfrac{1}{2} + (D_{c_n} - D_{c_2})(L_2 + RL_2) + \cdots + (D_{c_{n-1}} - D_{s_{n-1}})(L_{n-1} + RL_{n-1})\tfrac{1}{2} + (D_{c_n} - D_{c_{n-1}})(L_{n-1} + RL_{n-1}) + (D_{c_n} - D_{s_n})(L_n + RL_n)\Big\} + (\text{if } D_d > D_{c_n}),(D_d - D_{c_n})(M_o + L + RL)$$

Where:

Ds_o = Date material received

D_{s_1} = Date start 1st operation

D_{c_1} = Date completed 1st operation

D_d = Date due complete

M_o = Material Cost

L_1 = Direct Labor 1st operation

R = Ratio of IME to Direct Labor

$$L = \sum_{i=1}^{n} L_i$$

I = Cost of Carrying Inventory ratio per day.

A Good Approximation is:

Cost of Carrying INV. = IAC

A = the Average Inventory over the entire cycle.

C = Entire cycle in days.

Where: $A = M_o + \tfrac{1}{2}(L + RL)$

$$C = D_{c_n} - D_{s_o} + \begin{cases} 0 \text{ if } D_d \leq D_{c_n} \\ D_d - D_{c_n} \text{ if } D_d > D_{c_n} \end{cases}$$

Either of these two equations may be used for inventory evaluation. The second method is generally to be preferred for simplicity and ease of manipulation.

There are three areas of cost which need to be investigated in order to determine the equivalent penalty for late delivery. If a finished product is being shipped directly to a customer, the customer dissatisfaction may be measured through a penalty clause in the contract; however, if there is no such clause then an estimate must be made of the losses which may be suffered through the customer's not obtaining a delivery on the date desired; this might be obtained by estimating the amount which you would be willing to pay to avoid late delivery. Where the product is used as a portion of a final assembly this factor could be a measure of the direct extra cost which would be caused in the assembly department in order to avoid late delivery of the final product. The net cost of customer dissatisfaction can be expressed as $(D_c - D_d)$ FR, where D_c equals the actual date the part is completed, D_d equals the date due complete, F equals the full value of the part, and R equals the customer dissatisfaction ratio per unit product value per day.

The second cost of late delivery is involved in the necessity for carrying the inventory for this lot for a period of time longer than planned. This has been covered in the cost of inventory for the product, by including the time period $(D_{c_n} - D_d)$.

The third area of cost for late delivery is the cost of carrying inventory for other parts which are used together with this part. This is only applicable where the product is used in further assembly. The cost is equal to $(D_c - D_d)$ BI where B equals the inventory value, prior to assembly, of all parts held up pending delivery of this part and I equals the cost of carrying inventory ratio.

The next basic function to be investigated is the non-productive cost when the man-machine combination is idle. The first area here is the payment of funds to operators who have not been able to produce goods because of insufficient part availability. This can be evaluated as $(T_i)(W_m)$. Where T_i equals the idle time in minutes and W_m equals the rate of pay per minute.

The second factor is somewhat more complex in that by the machines not having produced useful goods during the time period T_i a certain loss in output product has been experienced. This loss in output is only significant in that normal profit may have been lost and non-direct expenses not liquidated. This may be expressed as $(VR + O_h W_m) T_i$

Where: V = Replacement value of the machine tool

R = Required return on Investment per minute and

O_h = Overhead to Direct Labor ratio

The fourth and last of the evaluation factors is the needed set-up costs. This can most readily be obtained through a direct comparison of the total set-up dollars expended under one schedule as against the total set-up dollars expended under a different schedule. In some jobs, especially those that use special purpose tooling, this factor may be dropped since there are few opportunities for reduction in set-up costs, except through change in lot size, which has been omitted from this discussion. It should be understood that set-up costs are considered to include not only make-ready charges but also tear-down, clean-up, and put-away costs. Other ways in which this cost may be influenced is by the combining of set-ups on similar jobs and the splitting of jobs for load purposes thereby causing additional unplanned set-ups. This can be expressed as:

$$\sum_{i=1}^{n} \sum_{j=1}^{r} S_{ij}$$

Where S = actual set-up charge experienced on a given job and a given operation.

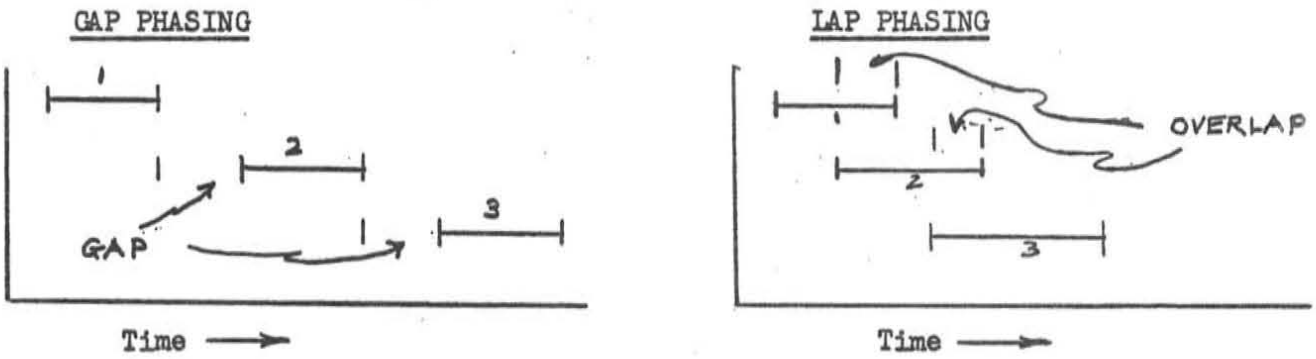
FACTORY MODEL

In preparing a factory model it was decided to imitate as closely as seemed practical actual operation, not the ideal or storybook version. This means that recognition must be made of the effect human errors have on the output of the factory. In preparing the model the operation of a factory seemed to divide itself into three areas. The first is the actual job selection method and is intimately related with the priority system associated with the schedule; this is frequently called dispatching or scheduling. The second activity is the actual transformation of the product through physical, chemical, or electrical means into something different from its original state. This is usually known as machining, assembly, or processing. The third area is the physical movement of the material from one location to another. This is material handling or transportation. Each of these basic areas will now be analyzed.

The dispatching method used in many job shops operates as follows:

A man comes to the dispatcher's window and requests the assignment of a new job; therefore, the man is the forcing function. The dispatcher has a record by operation station of the various jobs which are available for that station at that specific time. In accordance with the existing priority system the dispatcher selects the most urgent job and assigns that to the operator. At some later time the dispatcher will be notified that the job is ready to move to the next station; this may be done by having the operator come to the window for a new job. At this time the dispatcher determines what the next station is and directs a material handler to transport the material to that area. When the material has been delivered to the next station the dispatcher is informed by the material handler and a record is made at that station that the job is available. This basic selection technique will vary somewhat between different factories. However,

the key point is that no job can be moved until it has been operated upon and that no operation can be started unless the material is at that station. This can be modified by using "lap-phasing" instead of the more usual "gap-phasing". They can best be differentiated by looking at a graph of factory progress under both plans:



In this discussion "gap-phasing" will be assumed. The problems of "lap-phasing" are numerous such as determining the number of individual deliveries and therefore is not too frequently used in a job control shop; it is frequently employed in a batch control operation and reaches its ultimate in the flow control shop when the lot size is unity. The variable factors to be considered in the dispatching mechanism include the delay times in receiving information. Dispatching in the batch or flow control shop is of a somewhat simpler nature. There, the starting date may be determined a week or more in advance and it is anticipated that the various consecutive operations will be performed on these jobs maintaining the original sequence. Therefore in these areas a dispatcher only has to control the starting sequence selection. It is even possible to treat the entire group of operations as being performed at a single station; this is especially true where conveyor belts are used.

The station operation model predicts the performance of certain specific tasks during a finite time interval. There are two basic phases to look at.

First is the fixed or pattern data such as

Starting time = T_o

Initial quantity = x

Set-up time = e

Time per unit = p

If the operation were perfect and there were no "lap phasing" or human or machine variations to deal with, it would be a simple matter to predict the completion time (T_c). $T_c = T_o + e + xp$

However we do not face any such ideal situation. The following "noise" factors severely affect the completion time for a lot and their specific impact is on a somewhat random basis.

Operator efficiency ratio = f, $f > 0$

Machine breakdown = b, b = o for indicating machine availability.

$b > o$ indicates time delay until machine is available.

Operator absenteeism = a, a = o for indicating operator availability.

$a > o$ indicates time delay until operator is available.

Spoilage ratio to original quantity = s, $s \geq o$

Re-work ratio to original planned time = r, $r \geq o$

Material, tool, blueprint and paperwork availability = m, m = o for indicating availability of all factors.

$m > o$ indicates time delay until all factors will be available.

Therefore, the actual equation which must be used for predicting the anticipated completion time (T'_c), still omitting lap phasing is:

$$T'_c = T_o + \frac{1 + r}{f} (e + xp) + b + a + m$$

and at T'_c, $x' = (1 - s)x$,

where x' = completed quantity and each factor is determined by an appropriate probability distribution.

One important feature of this probability arrangement is that no single run-through of a schedule will be sufficient for evaluation. It will be necessary to use a statistically determined number of tries or samples in order to predict with, say 90% accuracy, the mean evaluation for a given schedule.

However, it should be noted that each of these "noise" factors can, with the data which is available, be statistically determined and need not be guessed or estimated. The accuracy with which the probability distributions for these noise factors is established will, to a great degree, determine the usefulness of the final results.

The third function in the factory, that of material handling, can also be analyzed statistically, if necessary. In a batch control or flow control operation where automatic or semi-automatic movement exists, this transportation mechanism is exceedingly simple in that it is an essentially predictable function. However, in the job control shop, indications are that this can be a random relationship not fixed by the source station and delivery station or by their distance, time of day, or any other determinable factor. All this means is that the transportation time may have to be derived and used in the same manner as the noise factors.

VARIATION OF INPUT SCHEDULING

The basic objective of all this evaluation and factory simulation is the improvement of the input schedule itself. Two basic approaches suggest themselves: one is the possibility of random variation of the sequence of items in the input schedule, thereby providing a set of different priority systems. Each new sequence

that is attempted should result in some change in the net evaluation. However, the difficulty here arises in the tremendous magnitude of the available permutations. For instance, for just 100 jobs the total number of arrangements for a single operation is 10^{158}. If there were, say 10 operations to be performed on each job, then this number would have to be raised to the tenth power to cover all of the various possibilities. Even with the fastest computer on the market today such a number of trials is not economically feasible especially since the time consumed would be such to make the data old before an answer was obtained. A second possible fallacy in this random arrangement approach is the absolute magnitude of the noise factors compared to the average cost variances. It should be possible statistically to compute at what magnitude of the noise factors the cost variances are insignificant.

However, this above technique might be applied through random arrangement of stratified data. For instance, if the various jobs to be manufactured were arranged in sequence based upon their due complete date and the amount of work left to be performed then it would seem rational to establish rules that no job may be moved more than n positions down this stratified table. This approach can be even further simplified by dividing the jobs into a set of groups of n items each. Then, within the group random arrangements might be tested, but no job could be shifted to a different group. However, even this approach leads to voluminous trials since if there were 100 jobs and 10 jobs to each sub-group there would be $(3.6 \times 10^6)^{10}$ trials. Another possibility is testing of each group independently and then fixing permanently the results of this group prior to the testing of the next group. However, this would have to be proven as statistically valid. All of these approaches require the weekly (or other short period) re-analysis in order to determine the comparative priority numbers.

The second basic way of approaching the variation of the input schedule is by using an Operations Research type analysis. This involves a study of the effect that various input factors have on the output factors and by appropriate correlation establishing the anticipated variation in output from a specific change in input. Certain examples may be cited which explain more clearly this approach. As stated previously, an input schedule specified certain factors. First, it establishes the quantity desired at a certain date. Next, it provides for each item a starting date supposedly selected so as to meet the due complete date. Third, associated with the schedule, is a basic priority system which gives each job in the factory a comparative preference. Since, in normal operation the quantity and finish date are fixed the only variation that can be made is the starting date and the nature of the priority system itself. Now if an adequate factory model exists it should be possible to intelligently vary the basic manufacturing cycles (hence the starting dates) for the various items. The examination of these results should give excellent clues as to what the optimum cycle should be for each item. Another series of tests might be conducted having the factory model choose the jobs in accordance with different priority systems such as dispatching by due complete date, by starting date, by starting date for each operation, by selected numbers, or by a combination of due date and amount of work to be done. Upon completion of these studies it would seem to be statistically possible to determine for a specific plant the best priority system to use. These studies need not be done every week, but could be performed at semi-annual intervals or as the key factors changed.

There is a further advantage to this approach in that once an effective factory model exists it would be relatively easy to pre-compute the effect changing product mix or changing output would have on the factory.

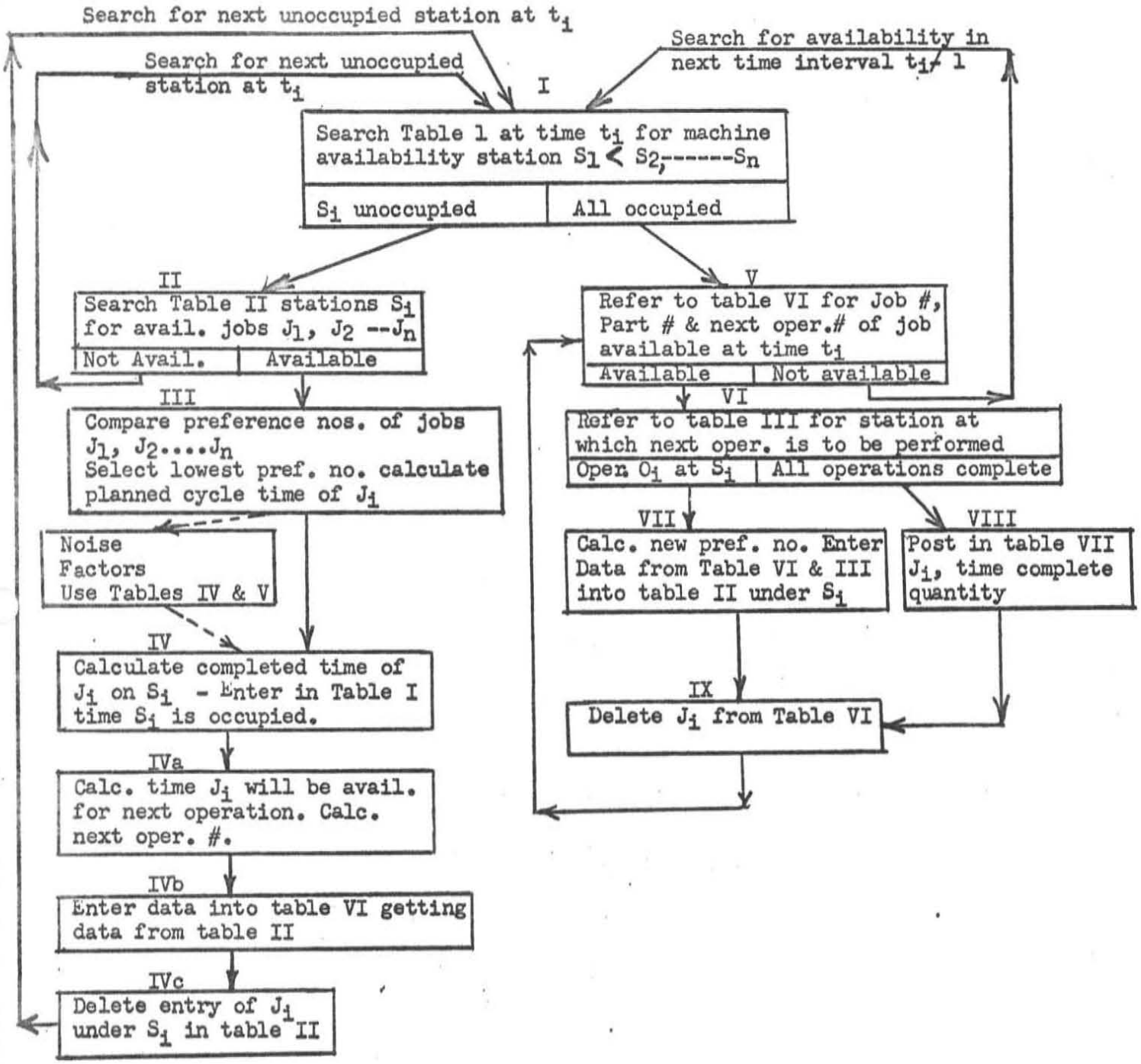
ANALYSIS

From the foregoing paragraphs it seems reasonable to conclude that excellent savings might be realized from the creation of an effective factory model associated with a good evaluation plan. This entire plan must be converted to a detailed computer program in order to obtain a realistic test. It is essential that the time per run be very brief so that multiple reviews will be physically and economically feasible.

The potential gain seems great and further investigation using a computer appears to be extremely desirable.

In order to provide experience on the application of computers to this basic scheduling problem it was decided to try to imitate manually the computer operations on a simple set of data. A concept of the overall flow of information is on the following page:

SCHEDULING INSTRUCTIONS



The tables required are included as exhibits and numbered as follows:

I Machine utilization and status by station number

II Waiting operation file by station number

III Planning and routing data by part number

IV Random number generation function accumulator (not included as an exhibit)

V Noise factor statistical selection (not included as an exhibit)

VI Parts in process by time available

VII Evaluation data

VIII Input schedule

The operations required are described briefly with the results obtained in the specific problem studied. These results are posted in the tables concerned with asterisks to indicate the added data. For the sake of simplicity the noise factors are assumed to have 100% probability for the listed value:

$f = 1.0$	$s = 0$
$b = 0$	$r = 0$
$a = 0$	$n = 0$

Step 1.

Examine time .1 in table I. for example machine availability:

All machines occupied.

Step 2.

Search table VI. for parts available at time .1:

None available

Step 3.

Repeat for time .2:

All machines occupied, no parts available.

Step 4.

Search table I. for machine available at time .3:

Station 03 is available.

Step 5.

Search table II. for station 03 for parts available:

Jobs M and 0 are available.

Step 6.

Compare preference number jobs available and select that job with lowest preference number:

Job 0 has a preference number of $\not$ 47.0; therefore job M with a lower preference number will be placed on station 03.

Step 7.

Calculate length of time station 03 will be occupied in manufacturing job M:

$T_o = e \not xp = 2.5$

Step 8.

Indicate on table I. machine hours utilized at station 03:

Machine will be occupied from time 0.3 through time 2.7; $T_c = (T_s - .1) \not T_o$.

Step 9.

Calculate time that part will be avilable for machine operation:

Job M complete at station 03 at time 2.7. Transportation time as obtained from random number table = .5; therefore time available for next operation = 3.2.

Step 10.

Obtain next operation number:

Add operation number in table II. to the quantity 1; therefore the next operation number = 2.

Step 11.

Enter data in table VI. and results of calculations:

Time available, job number, part number, quantity, next operation number and preference number.

Step 12.

Delete entry in table II. under station 03 for job M:

Step 13.

Continue searching table I. for available machine capacity during time .3:

All machines occupied.

Step 14.

Search table VI. for parts available at time .3:

Job F. available part 006 for operation 6.

Step 15.

Refer to table III. for 006, operation 6:

Operation 6 says deliver to destination - part is complete; part is complete at time .3.

Step 16.

Post to table VII:

Part 006 completed.

Step 17.

Delete job F from table VI:

Step 18.

Continue searching table VI for parts available at time .3:

No parts available.

Step 19.

Search table I. and table VI for time .4:

All machines occupied, no parts available.

Step 20.

Search table I. and table VI. for time .5:

All machines occupied, no parts available.

Step 21.

Search table I. at time .6:

Station 01 available.

Step 22.

Search table II. for station 01 for parts available:

Jobs A., C., J., L., and N. are available.

Step 23.

Compare preference numbers and select job with the lowest preference number:

Job A. / 32.8

Job C. / 71.5

Job J. - 15.6

Job L. / 5.0

Job N. - 23.9

Therefore job N. is selected for assignment.

Step 24.

Calculate length of time station 01 will be occupied in manufacturing job N:

$T_o = 3.6$

Step 25.

Indicate on table I. machine hours utilized at station 01:

$T_c = 4.4$

Step 26.

Calculate time job M will be available for next operation:

$4.4 / .1 = 4.5$

Step 27.

Calculate next operation number:

$2 \neq 1 = 3$

Step 28.

Enter data in table VI:

Step 29.

Delete entry in table II. for station 01 under job N:

Step 30.

Continue to search table I for unoccupied machine at time .6:

All machines occupied.

Step 31.

Search table VI. for parts available at time .6:

No parts available.

Step 32.

Search tables I and IV for time .7:

All machines occupied, no parts available.

Step 33.

Search tables I and IV for time .8:

All machines occupied, no parts available.

Step 34.

Search tables I and IV for time .9:

All machines occupied, no parts available.

Step 35.

Search table I. for machines available at time 1.0:

All machines occupied.

Step 36.

Search table VI for parts available at time 1.0:

Job D., part 006 available for operation #3.

Step 37.

Refer to table III for part 006:

Operation 3 to be performed at station 02.

Step 38.

Calculate new preference number:

$$\pi' = \pi + SF(e_2 + xp_2) = -3.3$$

Step 39.

Enter data under station 02 in table II:

Job number, part number, preference number, operation number, set up, time per unit, and quantity; data obtained from table II and table IV.

Step 40.

Delete Job D. from table IV:

This process is continued until the week's schedule has been planned.

For this problem the priority system is based on the use of a preference number calculated from the due complete date of the job and the amount of work remaining to be performed. The formula used for calculating the preference number is:

$$\pi = D_d - SF(e + xp)$$

Where:

D_d = date due complete

π = preference number

$$SF = \frac{1}{\text{Mfg. cycle efficiency}}$$

e = set-up time

x = quantity

p = per unit time

M = Mfg. Cycle Efficiency

$$M = \frac{\sum_{j=1}^{r} M_j}{r}$$ where there are "r" jobs possible to manufacture.

$$M_j = \frac{T_j}{R_j}$$

$$T_j = \sum_{i=1}^{n} e_i + x p_i$$

$$R_j = D_{c_n} - D_{s_o}$$ where there are n operations per part.

A good approximation is:

$$M \cong \frac{\left[\sum_{i=1}^{n} e_i + X p_i\right]_{J=1}^{J=r}}{\frac{\sum_{j=1}^{r} C_j}{r}} = \frac{\left[\sum_{i=1}^{n} e_i + X p_i\right]_{j=1}^{j=r}}{\sum_{j=1}^{r} C_j}$$

Where C = planned cycle.

This in effect compares the total planned on machine time to the actual time that the job was in the process of manufacture.

This simple approach to the problem of scheduling production through the use of an electronic computer should provide a good beginning for computer programing. Additional studies will be performed in the near future in an effort to determine the computer time required for performing the necessary calculations as well as an evaluation of the potential savings to be realized.

Burton Grad
Production Control Services Section
2/15/54

BG:D

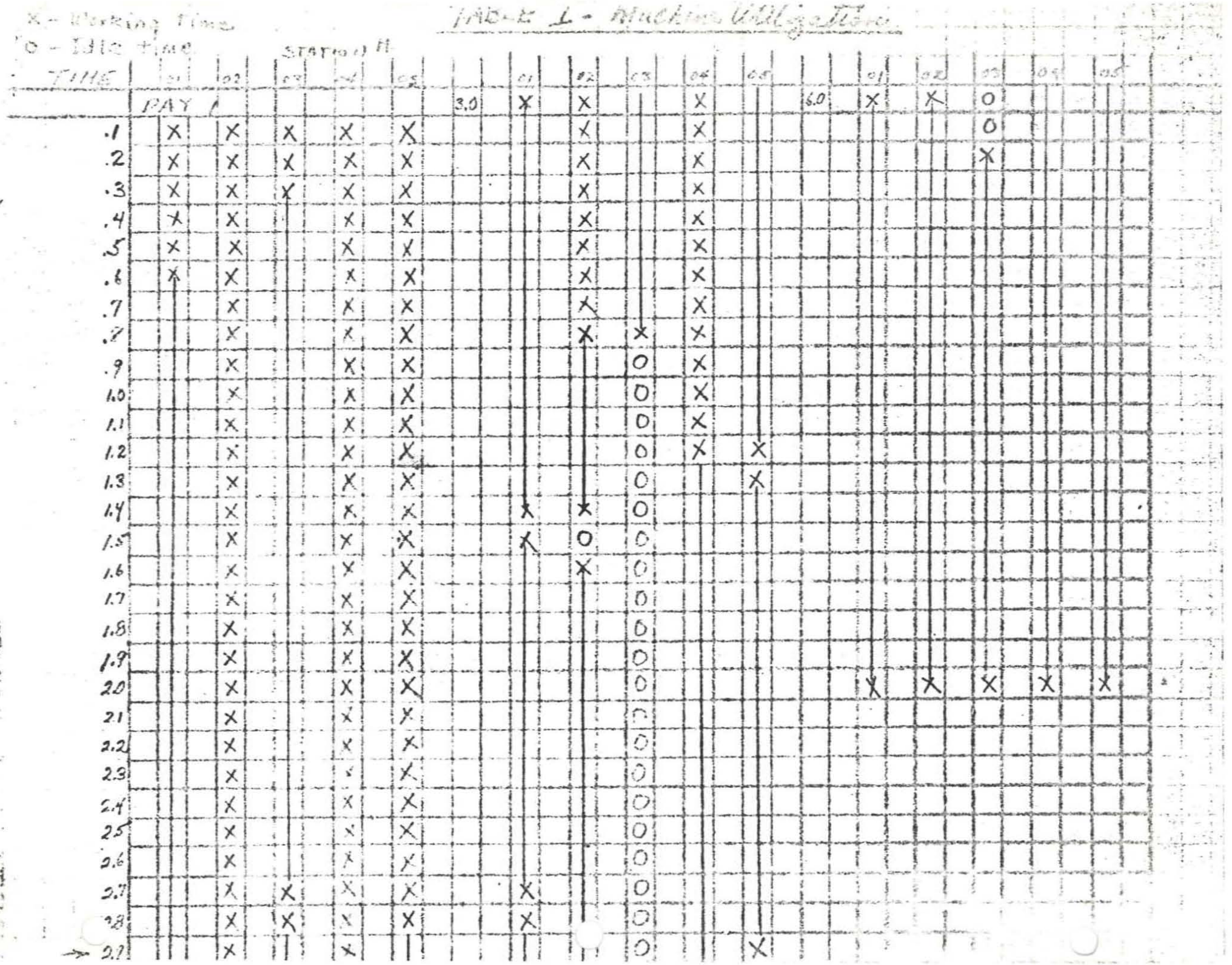
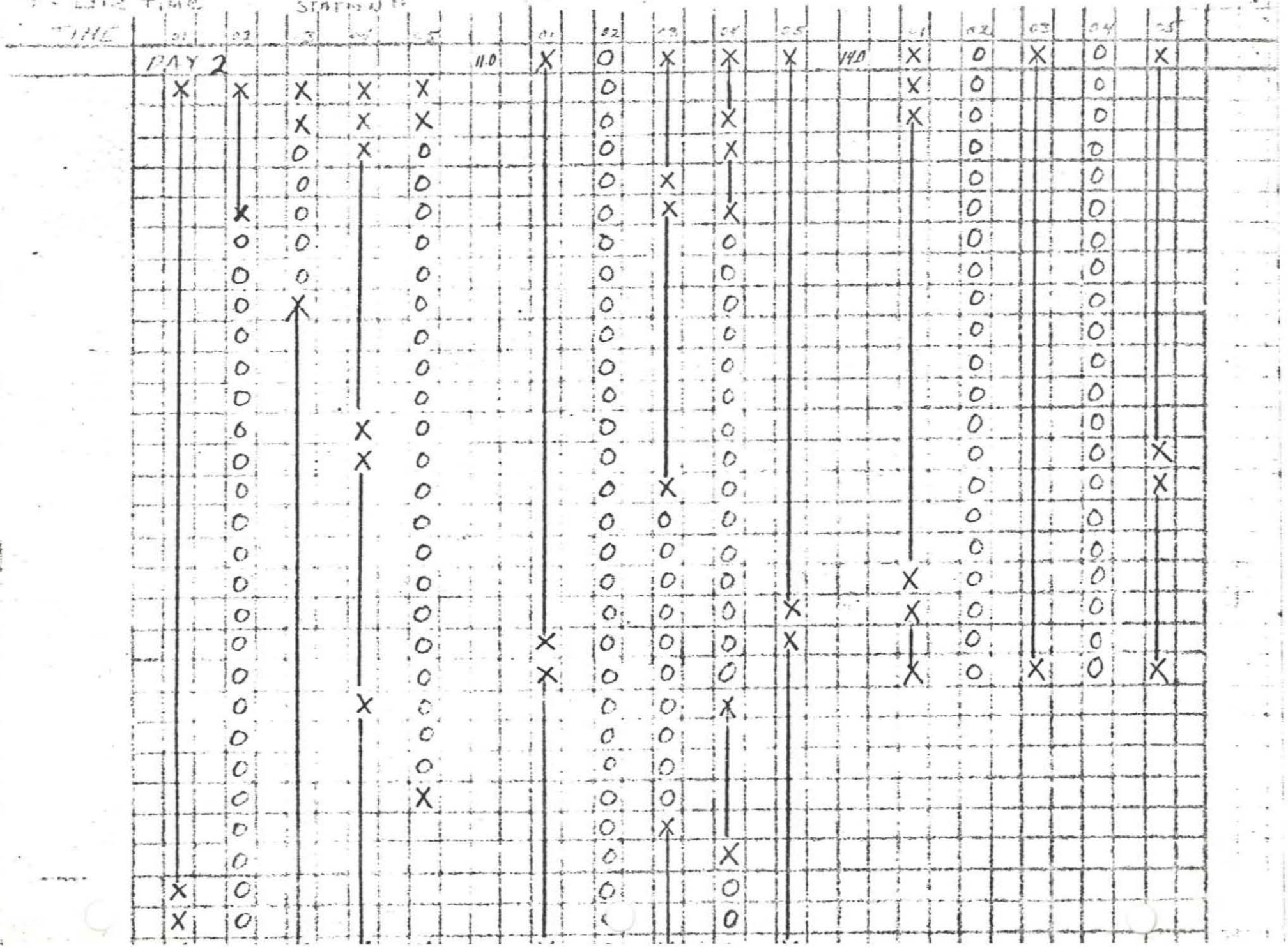

X - Working Time
O - Idle Time
TABLE 1 - Machine Utilization
STATION II
TIME
DAY 1
01
02
03
04
05
3.0
6.0
.1
.2
.3
.4
.5
.6
.7
.8
.9
1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2.0
2.1
2.2
2.3
2.4
2.5
2.6
2.7
2.8
2.9

TABLE I - Machine Utilization

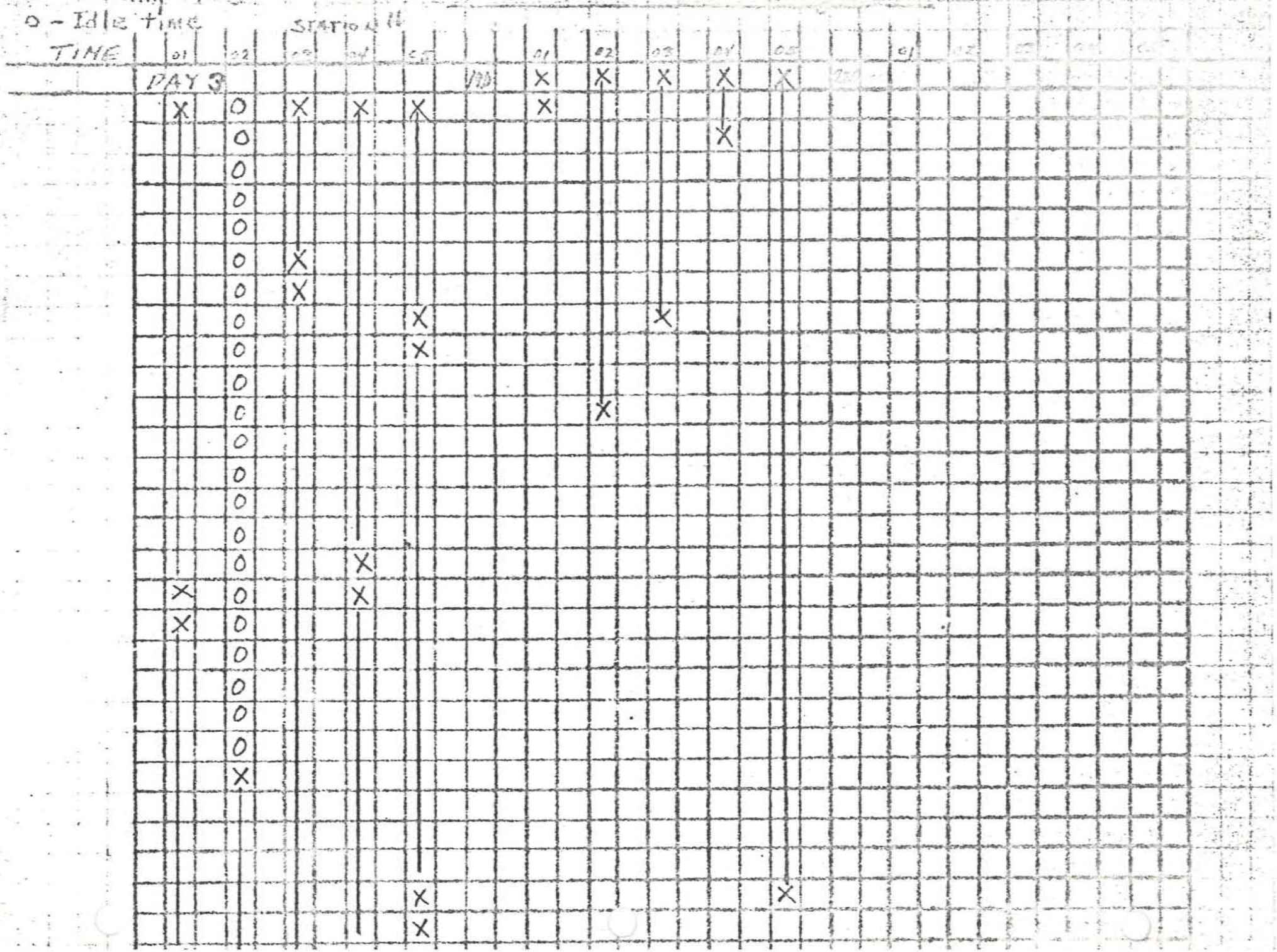
X - working time
0 - Idle time



X - Working time

O - Idle time

TABLE 4 - Machine Utilization



x - Working Time
o - Idle time

TABLE 1 - machine utilization

STATION H

TIME	00	01	02	03	04	05		01	02	03	04	05		01	02	03	04	05
	DAY 4						1170						360					

TABLE II

WAITING OPERATION FILE

Job Number	Part Number	Pref. Number	Oper. #	S.U.	Time/Unit	Quan.
Station #01						
A	003	+32.8	1	.2	.7	2
C	004	+71.5	4	.3	.7	1
J	007	-15.6	1	.5	.2	4
L	002	+ 5.0	3	.5	.9	5
N	001	-23.9	2	.3	.9	4
M*	004*	+42.0*	2*	.2*	.6*	3*
H*	005*	+ 9.0*	4*	.2*	.4*	5*
O*	003*	+51.7*	4*	.1*	.4*	1*
E*	002*	+18.9*	5*	.4*	.2*	4*
B*	010*	+50.7*	5*	.5*	.9*	5*
L*	002*	+34.1*	5*	.4*	.2*	5*
R*	001*	+52.4*	2*	.3*	.9*	3*
Station #02						
D*	006*	- 3.3*	3*	.1*	.2*	3*
N*	001*	- 7.3*	3*	.4*	.9*	4*
M*	004*	+51.5*	3*	.1*	.6*	3*
Station #03						
M	004	+31.3	1	.1	.8	3
O	003	+47.0	3	.2	.9	1
J*	007*	-10.0*	2*	.5*	.4*	4*
N*	001*	+ 9.8*	4*	.3*	.6*	4*
D*	006*	+ 4.0*	5*	.4*	.2*	3*
H*	003*	+18.4*	5*	.2*	.6*	5*
Q*	002*	+62.3*	1*	.5*	.6*	2*
S*	010*	+34.1*	1*	.3*	.5*	4*
Station #04						
I	007	+51.7	3	.1	.2	5
B*	010*	+33.2*	4*	.1*	.8*	5*
D*	006*	- .3*	4*	.1*	.3*	3*
K*	008*	+54.7*	5*	.1*	.1*	2*
J*	007*	- 1.0*	3*	.1*	.2*	4*
J*	007*	+13.5*	5*	.2*	.1*	4*
P*	001*	+60.1*	1*	.1*	.5*	2*
R*	001*	+45.6*	1*	.1*	.5*	3*
Station #05						
E	002	+12.5	4	.3	.3	4
K	008	+44.9	4	.5	.9	2
G*	009*	+40.9*	5*	.2*	.3*	5*
J*	007*	+ 2.8*	4*	.5*	.5*	4*
L*	002*	+26.4*	4*	.3*	.3*	5*
I*	007*	+56.4*	4*	.5*	.5*	5*
N*	001*	+21.3*	5*	.5*	.5*	4*
A*	003*	+38.6*	2*	.1*	.9*	2*

Table III

Part Number	Operation	Station Number	S.U.	Oper. Time/pc.
Part # 001				
	1	04	.1	.5
	2	01	.3	.9
	3	02	.4	.9
	4	03	.3	.6
	5	05	.5	.5
			1.6	3.4
Part # 002				
	1	03	.5	.6
	2	02	.4	.9
	3	01	.5	.9
	4	05	.3	.3
	5	01	.4	.2
			2.1	2.9
Part # 003				
	1	01	.2	.7
	2	05	.1	.9
	3	03	.2	.9
	4	01	.1	.4
	5	02	.1	.4
			.7	3.3
Part # 004				
	1	03	.1	.8
	2	01	.2	.6
	3	02	.1	.6
	4	01	.3	.7
	5	05	.2	.8
			.9	3.5
Part # 005				
	1	01	.3	.6
	2	03	.5	.2
	3	02	.4	.8
	4	01	.2	.4
	5	03	.2	.6
			1.6	2.6
Part # 006				
	1	03	.3	.7
	2	01	.1	.4
	3	02	.1	.2
	4	04	.1	.3
	5	03	.4	.2
			1.0	1.8

TABLE III (Cont.)

Part Number	Operation	Station Number	S.U.	Oper. Time/pc.
Part # 007	1	01	.5	.2
	2	03	.5	.4
	3	04	.1	.2
	4	05	.5	.5
	5	04	.2	.1
			1.8	1.4
Part # 008				
	1	01	.5	.7
	2	04	.1	.3
	3	01	.5	.3
	4	05	.5	.9
	5	04	.1	.1
			1.7	2.3
Part # 009				
	1	03	.2	.2
	2	02	.1	.6
	3	05	.3	.2
	4	04	.1	.9
	5	05	.2	.3
			.9	2.2
Part # 010				
	1	03	.3	.5
	2	04	.1	.5
	3	05	.3	.6
	4	04	.1	.8
	5	01	.5	.9
			1.3	3.3

Oper. 6 - across the board
deliveries to destination
part complete

TABLE VI

PARTS IN PROCESS

	Time Available	Job #	Part #	Quantity	Next Oper. #	Pref. #
	.3	F	006	1	6	+61.4
	1.0	D	006	3	3	- 8.9
	2.9	B	010	5	4	+19.1
	3.8	H	005	5	4	- 9.8
	4.2	G	009	5	5	+21.3
*	3.2	M	004	3	2	+31.3
*	4.5	N	001	4	3	-23.9
*	3.9	O	003	4	4	+47.0
*	4.3	E	002	4	5	+12.5
*	4.5	D	006	3	4	- 3.3
*	8.3	B	010	5	5	+33.2
*	6.0	G	009	5	6	+40.9
*	6.1	J	007	4	2	-15.6
*	8.7	N	001	4	4	- 7.3
*	10.9	L	002	5	4	+ 5.0
*	8.3	K	008	2	5	44.9
*	8.3	J	007	4	3	-10.0
*	9.3	D	006	3	5	- .3
*	11.9	N	001	4	5	+ 9.8
*	10.3	J	007	4	4	- 1.0
*	11.3	I	007	5	4	+51.7
*	13.0	J	007	4	5	+ 2.8
*	13.4	H	005	5	5	+ 9.0
*	11.6	K	008	2	6	+54.7
*	12.6	D	006	3	6	+ 4.0
*	15.6	N	001	4	6	+21.3
*	14.5	E	002	4	6	+18.9
*	13.8	J	007	4	6	+13.5
*	17.1	H	005	5	6	+18.4
*	16.1	A	003	2	2	+32.8
*	17.3	L	002	5	5	+26.4
*	18.2	M	004	3	3	+42.0
*	18.0	R	001	3	2	+45.6
*	20.1	H	005	5	6	+18.4
*	19.0	A	003	2	3	+39.6
*	19.3	R	001	3	2	+45.6
*	19.5	L	002	5	6	+34.1
*	20.4	M	004	3	4	+51.5
*	21.8	I	007	5	5	+56.4

EVALUATION DATA TABLE

TABLE VII

Job #	Part #	Wkg. Days Date Due	A	$D_d - D_s$	B	time opt.	Qty. opt.
A	003	8	$ 42.42	10	$2596		
B	010	9	102.55	10	1703		
C	004	10	48.90	10	8674		
D	006	1	155.40	10	1857	12.5	3
E	002	3	62.82	10	7585	14.5	4
F	006	8	53.30	10	8482	.3	1
G	009	6	236.78	10	5099	6.0	5
H	005	4	92.85	10	3499	17.1	5
I	007	9	104.80	10	1030		
J	007	2	84.65	10	9235	13.8	4
K	008	7	34.18	10	6738	11.6	2
L	002	5	77.35	10	9458		
M	004	10	142.65	10	6990		
N	001	4	118.20	10	7767	15.6	4
O	003	7	22.00	10	3962		
P	001	12		10			
Q	002	12		10			
R	001	12		10			
S	010	12		10			

TABLE VIII

INITIAL SCHEDULE

	JOB #	PART #	QUANTITY	DATE DUE	AT OPER#	STARTED AT	AVAIL. STAT#
	A	003	2	8	1		01
	B	010	5	9	3	- .6	05
	C	004	1	10	4		01
	D	006	3	1	2	- .8	01
	E	002	4	3	4		05
	F	006	1	8	5	- .4	03
	G	009	5	6	4	- .5	04
	H	005	5	4	3	- .7	02
	I	007	5	9	3		04
	J	007	4	2	1		01
	K	008	2	7	4		05
	L	002	5	5	3		01
	M	004	3	10	1		03
	N	001	4	4	2		01
	O	003	1	7	3		03
Addit.	P	001	2	12	1		04
Jobs	Q	002	2	12	1		03
	R	001	3	12	1		04
	S	010	4	12	1		03
	T	009	5	13	1		
	U	005	3	13	1		
	V	002	5	14	1		
	W	008	1	14	1		
	X	003	5	15	1		
	Y	008	2	15	1		
	Z	009	5	16	1		
	AA	007	4	16	1		
	AB	010	4	17	1		
	AC	009	4	17	1		
	AD	009	4	18	1		
	AE	003	3	18	1		
	AF	001	5	19	1		

INFORMATION PROCESS CHARTING

Burton Grad
Manufacturing Services
General Electric Company

Richard G. Canning
Consultant
Vista, California

Ralph H. DeRubbo
Specialty Motor Department
General Electric Company

March 15, 1957

Manufacturing Services
General Electric Company
570 Lexington Avenue
New York 22, New York

INTRODUCTION

The information process charting which is discussed in this write-up is aimed at helping to analyze the so-called "office automation" aspect of the project. It is a technique to be used during the data gathering phase of the project, when we are trying to find out as much about the present business as time will allow. It should be emphasized, however, that this process charting technique is only part of the overall data gathering phase. In addition to this charting, we must also obtain a considerable amount of statistical data about such things as the flow of information and the size of the files. Similar types of studies must be made of the factory operations, too.

The sequence of presentation in this Information Process Charting write-up will be:

I Description of the charting symbols used, together with a brief discussion of some of the less obvious points.

II Description of the charting form used.

III Discussion of the overall charting technique.

IV Definitions of terms, to aid in understanding the discussion.

I. PROCESS CHARTING - GENERAL

We are interested in learning what the mechanized system must be able to handle. For the "office automation" aspect of the Project, then, we are interested in the following types of information:

1. The logically necessary alternative procedures which are needed in the business, for handling the main flow as well as the so-called "exception" cases.

2. The management control reports that are developed by the organization, and method of use.

3. The reports and other pieces of information that must be transmitted outside of the organization, for legal or other reasons.

4. At first, a preliminary pin-pointing of all complex computation operations and decision areas. Later, we will need a more detailed understanding of these operations.

Conversely, there are certain types of information in which we are not interested:

1. We are not interested in the manual procedures that are used, per se--e.g. how many clerks and typists work on a form during its preparation, or points along its route where the form is stored temporarily.

2. We are not interested in the fact that several people are involved in an operation, each doing part of the overall job.

3. We are not interested in the layout of the clerical work area or the types of office equipment used.

4. We are not interested in the "exceptions" which arise due to internal clerical errors (however, we are interested in the types of errors in information transmitted into the organization from outside, over which the organization has no control).

The more conventional charting techniques have not proved themselves adequate for uncovering the types of information in which we are interested. Such conventional techniques have generally been aimed at one of two objectives: (a) trying to find out where human clerical time is being spent, so as to shorten the overall process time, or reduce the man-hours to do a job, or (b) lay out a prescribed sequence of operations in a reasonably complex job, for the guidance of the clerks and operators--e.g. flow charts of punched card operations. We are not yet interested in either of these objectives, in this study, so it is not surprising that their associated charting techniques turn out to be relatively ineffective for our present purposes.

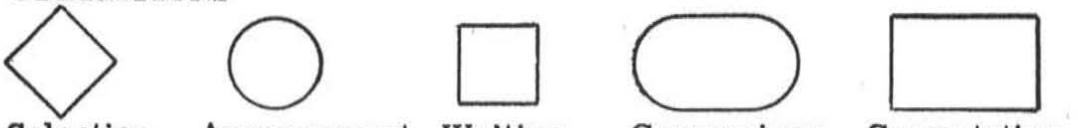
Rather, to obtain the understanding we require, we must learn the flow of information throughout the business, for directing manufacturing and the other activities, and for feedback and control. All uses of the information must be accounted for, so we soon find the scope of the project spreading throughout the entire organization.

To accomplish this objective, we have designed and tested a new technique which we call Information Process Charting. This technique is based on (a) reasonably precise definitions of basic data processing operations, represented in the form of symbols, and (b) a charting procedure to make sure that the necessary descriptive information accompanies each symbol.

PROCESS CHARTING - SYMBOLS

Fundamentally, there are only seven symbols in this technique, five of them representing "operations" (where someone does something), and two representing conjunctive, or connective devices. These basic symbols are shown below:

OPERATIONS



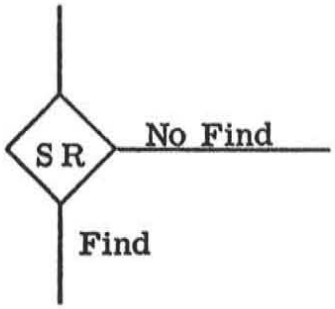
Selection — Arrangement — Writing, Erasing — Comparison, Branching — Computation, Decision

CONNECTIONS

Entry — Exit

As will be seen shortly, there are different variations for some of these symbols, so that in practice 11 different symbols are used. Each of these 11 symbols is defined on the following pages.

SELECT, SEARCH (SR)



1. Search always means to look for a specific record in a file, and not to look for a field of information on a record.

2. In searching, the file is always assumed to be present and available.

3. Indicate which file is being searched, the field(s) of information by which the search is made (the key), and which record the search key was obtained from.

4. Search implies that the file is sequenced by the key being used; otherwise, a Separate operation should be used.

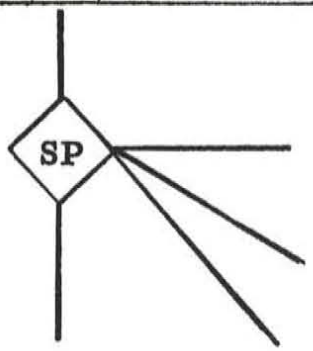
5. Always indicate both the Find and the No Find cases; even though the latter may be impossible, unlikely, or trivial.

6. We are interested in analyzing those cases of No Find that are logically important - e.g. there has not yet been time for the record to reach the file, the record has not been created, etc.

7. Both the input records and the selected file record are available at the output.

8. If the record is a separate unit record (e.g. a separate form), it is assumed to be removed from the file by the search operation, and must be merged back with the file if that is its disposition. If the record is simply an entry on a line, such as an entry in a log book, then it stays in the file but is, of course, available at the output also.

SELECT, SEPARATE (SP)



1. Separate means to select one or more records from a group or file according to some "key"--e.g. separating copies of a form according to the distribution field, or separating a batch of incoming mail by type of document.

2. Name what is being separated - e.g. multiple copies of a specified record, records of a specified file, etc.

3. Indicate the key by which the separation is being made. It is generally sufficient just to name the type of document, without giving details about the key.

4. A separate operation may have numerous outputs, one of which usually is the "main line" being charted. Use exits for all the other outputs, and use a separate line on the chart for each such exit, so as to be able to describe it properly.

5. If some outputs go to a simple filing operation, use the "file, then stop" exit described under Exits.

6. If multiple copies of a form are being separated, the disposition of all copies must be shown.

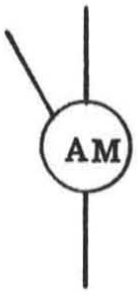
ARRANGE,SEQUENCE (AS)

1. Sequence means to arrange a group of records into ascending or descending order. It does not mean "sort into different groups", which is the Separate operation.

2. Name the records being sequenced, and indicate the key on which the sort is being made. If more than one field, indicate their order.

3. There is only one input and only one output to a sequencing operation.

ARRANGE, MERGE (AM)



1. Merge means to combine two or more sequences of records into one sequence.

2. Merging assumes that all records and/or files are already sequenced by the key.

3. Merge also covers the operation of placing a record into a file in proper sequence. In this operation, the file is assumed to be present and available.

4. Name the records and/or files being merged and fields by which the merge is made.

5. There are always at least two inputs to a merge operation, and always just one output.

6. In a "file, then stop" operation (described under Exits) use a merge and a dead-end exit, both in the exit column.

MODIFY, INSERT (+)

1. Modify insert means

   (a) To create a new record, e.g. a new Purchase order;

   (b) To add one or more fields of information to an existing record; e.g. signing a freight bill.

2. Indicate the record from which data is taken, the record into which it is inserted and the field(s) being inserted.

3. If a new document is created, show the disposition of both old and new documents.

4. It is usually not necessary to chart out the insertion of each field in a report (as discussed under Compute). Use one Insert for the whole report, or at least that amount which is done as one job.

5. As discussed under Search, a record should not be Searched to obtain a field of information prior to an insert. It is assumed that the entire record is read "at a glance" and all fields in that record are available. If there is a logical liklihood that a certain field will blank, use a Compart and Branch, comparing the contents of that field with "blank".

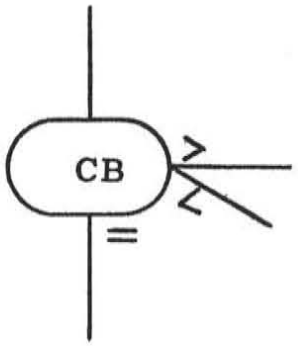
MODIFY, DELETE (-)

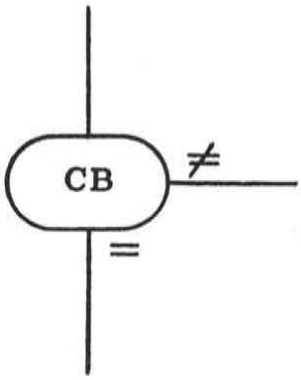
1. Modify delete means to delete one or more fields from an existing record.

2. Indicate record and field deleted.

3. Complete records are destroyed by means of a dead end exit, and not by means of a Delete operation.

COMPARE AND BRANCH (CB)





1. Compare and branch is a basic choice operation involving a defined or prescribed decision; e.g. if x is greater than y, take one course of action, etc.

2. Compare and branch is one of the operations to be used for showing how alternative processing paths are selected - for locating the "exceptions". (Other operations which do similar but not identical functions in this regard are Separate, Search, and Decision-Making).

3. If at all possible, the two things that are being compared, x and y, should be expressed as numbers. For instance, "compare the number of copies with 1" to determine if there are zero, one, or more copies. Or "compare the contents of a certain field with blank" to determine if anything is written in that field.

4. Always indicate the record (s) and fields (s) involved.

5. Y should always be the standard, if there is one.

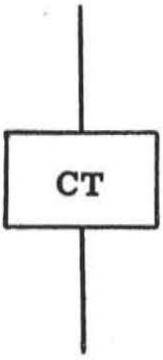
6. There are two varieties of outputs from this symbol: (a) = and ≠, to indicate "equal" and "not equal" when this expresses the situation adequately; (b) >, <, and =, when each of these three alternatives must be taken into account. ">" means "greater than", and "<" means "less than".

7. If one of the outputs is impossible, so mark it.

8. If the same action results from two outputs, the two signs may be shown together on one output line.

9. <u>All outputs must always be accounted for.</u>

COMPUTE (CT)



1. Compute refers to arithmetic operations; e.g. computing new shop load, delivery schedule, etc. using addition, subtraction, multiplication or division.

2. If the formula is easily expressed, note it down in the Remarks column of the charting form, or on a supplementary note sheet.

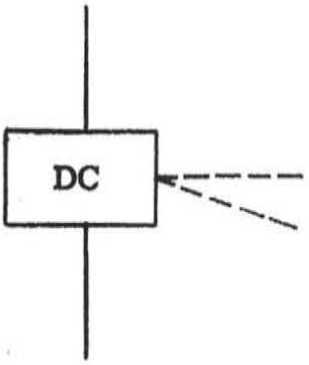
3. If the method of computation is at all complicated, do not spend the time to analyze it in detail, but simply group the whole computation into one compute operation. There is one exception to this rule: where significant branching occurs between portions of the computation, each of these portions should be shown as a compute operation, and the branching indicated by Compare and Branch operations. In either case, a detailed analysis of the computation will be made later.

4. Likewise, it is not necessary to indicate how each field in a statistical report is computed; just show one computation for the whole report, followed by an insert, for recording it.

5. If not too difficult, it would be desirable to have a two or three sentence write-up of the computation, indicating what is being done, for the more complex computations. This write-up can be on a supplementary note sheet.

6. Indicate the records involved and the fields being computed.

DECISION-MAKING (DC)



1. Decision making generally involves judgment; e.g. estimating what will be the volume of customer orders during the next three months.

2. These operations are usually quite difficult to analyze, (because of the large number of factors involved), so do not spend the time to analyze them. They will be analyzed at a later date.

3. However, it is desirable to chart the processing of information up to the point where the decision is made, and also after the decision is made. For instance, information is usually pulled together from a number of sources and often presented in report form before the decision-making operation takes place.

4. After the decision is made, indicate that the results are recorded, by an Insert operation.

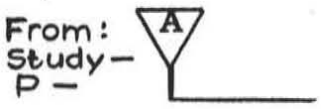
5. In some cases, alternative processing steps will be taken, depending upon alternative results of the decision. In this case, several exits may be shown from one decision symbol.

6. List the major factors involved in making the decision, in the Remarks column or on a supplemental note sheet if necessary.

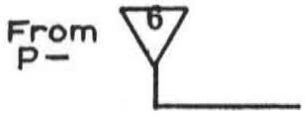
7. If there are well-defined intermediate steps or computations during portions of the decision process, these should be charted in the normal manner.

8. The decision symbol may also be used to represent non-routine activities of professional and semi-professional personnel which are difficult to chart. Examples might include data gathering and analysis prior to a decision, where methods of data gathering and analysis vary with the situation.

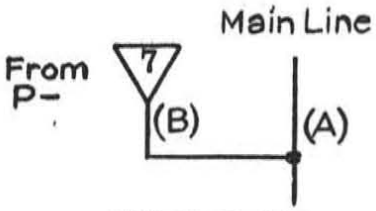
ENTRY (EN)



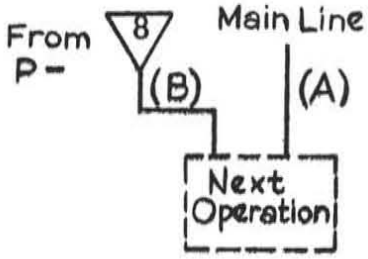
Inter Activity Entry



Intra Activity Entry



"And" Entry



"Or" Entry

1. An entry indicates the beginning of a series of related operations (a routine). An entry may have inputs from more than one exit.

2. There are four types of entries:

   a. Inter activity entry, (inputs from outside the system, or from another activity study within the system). Letter such entries beginning with "A".

   b. Intra-activity entry (entries from exits from within this same study). Number these entries, beginning with "1".

   c. "And" entry. If records A and B must go through a series of steps together, and record A has been charted as the main line, bring record B in with an "and" entry. This condition is represented by connecting the entry to the main line, as indicated.

   d. "Or" entry. If record A or record B (but not both) can go through the same series of steps, and A has been charted as main entry. The entry symbol is connected to the next operation symbol, and not to the main line.

3. On lettered entries, indicate study, page, and exit supplying the record, if possible. On numbered entries, specify the page number of all exits which are inputs to the entry.

4. The analyst will have to decide whether the first entry is to be numbered or lettered. If the activity originates at point where the chart begins, use number "1". If the activity is supplied by forms or information from some other activity, or from outside the system, use letter "A".

5. Indicate all entering records (but not fields).

6. An entry may be initiated at any point on the chart, but must be made to the main line flow. A direct entry may not be made to a secondary flow.

7. An entry does not necessarily have to have an entering document or form. Also, an entering record may be in the form of direct measurements or observations.

EXIT (EX)

To Study--
P--
Entry--
Inter Activity Exit

To P--
Intra Activity Exit

Dead End Exit

"File, then stop" Exit

1. There are four types of exits:

   a. Going out to some other study; letter sequentially, beginning with "A", indicate study and page number, if possible. All exits going to the same outside entry should use the same letter. It is desirable to name the exit on the chart in terms of the activity--e.g. copy to Accounts Payable

   b. Going to an entry within this same study (a "normal" exit); number the same as the entry to which it is going, and indicate page number of that entry.

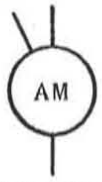
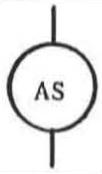
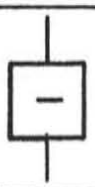
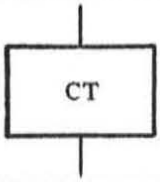
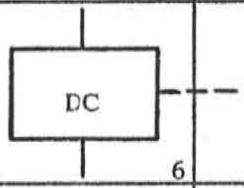
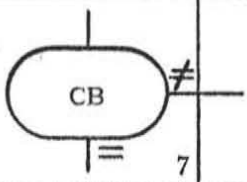
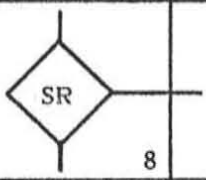
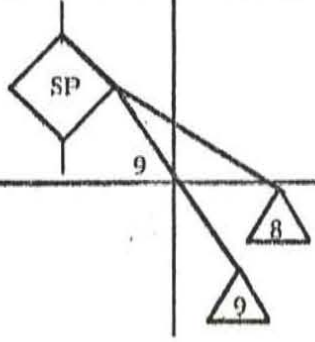
   c. A "dead end" exit, either destruction of a record or ceasing to process a record, indicate by an X in the exit symbol.

   d. A "file , then stop" exit; use a "merge" operation, then a dead end exit, both in the exit column Name the file and sequence "Key"

2. An exit should go to only one entry.

3. Do not use an exit at the end of a page just so as to refer to the next page; just stop the line at the bottom of the page and pick it up at top of next page. Be sure page numbers are correct to prevent confusion.

4. As discussed under the Separate operation, use exits for all the secondary outputs of Separate. Spread these exits vertically, not horizontally (to provide space for proper description) but take up as little space on the page as possible.

Entries	Main Line Flow	Exits	Secondary Flow		Records		Record or Field		Fields	Remark
	 1				Name of Record or File	By	Key	with	Name of File	
	 2				Name of Records Being Sequenced	By	Key			
	 3			on	Record onto which data is being inserted		Field(s) being inserted	from	Record from which data is obtained	
	 4			from	Record from which data is deleted	Delete	Field(s) being deleted			
	 5				Field(s) being computed	from	Field(s) supplying information	on	Records from which field(s) were obtained	Formula, if easily obtained
	 6				Brief name of decision					List of major factors
	 7			using	Record involved	compare	Field	with	Field	
					Record	and	Record	by	Field(s)	
	 8				Name of file being searched	by	Key	from	Record from which key was obtained	Do not search a record fo a field
	 9				Name Records or file being Separated	by	Key			
					(Description of Exit 8)					
					(Description of Exit 9)					

PROCESS CHARTING - DISCUSSION

There are a few points of special interest that should be discussed briefly, about the use of the above symbols.

First, the operation of "reading" is implied in all of the above symbols, and is not called out as an independent step. It is assumed that the person doing the processing must first read the information from the documents.

Next, it should be re-emphasized that whenever Compute and Decision-Making operations are encountered, the analyst should not spend much time in detailing these operations. The process charting simply pin points the computations and decisions, and gives some understanding of what is involved. After the data gathering phase, many of these operations will be analyzed in detail.

Occasionally an analyst will come across a "loop" operation, where a series of steps is performed on a number of items, before going on with the processing. An example: computing the standard cost for each operation in making a part, before the standard cost of the finished part can be determined. Indicate such loops as simply as possible; e.g. "steps___through___ repeated for each item on the record."

Also, in some possible loop operations (where there may be one or more items on the list), a totalling or summary is performed afterward if there was more than one item on the list. Do not try to chart this out, using conditional transfers. Simply indicate the totalling operation right after the last operation of the loop, and indicate in the Remarks column, that it is used if necessary.

Occasionally a matching operation will be encountered, such as a group of time cards being matched against a payroll file. It is desired to know if there are any time cards for which there are no payroll records, and if there are any payroll records for which there are no time cards, as well as matching payroll records and time cards for computing gross pay. Such situations as this may be charted by two Search operations in sequence. In the case above, search the payroll file by the time cards employee numbers indicating Find and No Find. Then search the time card "file" by the payroll record employee numbers indicating Find and No Find.

The "and" and "or" concept should be mentioned briefly. Sometimes two records must be brought together, so as to go through a series of processing steps: record A and record B. If record A has been charted as the main line, then record B can be brought in by using an "and" entry, as shown on page 12, by connecting the entry to the main line.

Note that this "and" entry is somewhat different from a Merge or a Search operation. If record A is being charted on the main line and B is located in a file, then B is obtained by a Search operation.

Also, if two files are in a common sequence, they can be combined into one file of the same sequence by a Merge operation. Similarly, a record can be placed into its proper place (in sequence) in a file by the Merge operation.

The "And" entry implies no searching or no sequencing. There is no file; only records are involved. Two or more records are brought together, so as to be able to form a larger record.

For the "or" situation, record A or record B (but not both together) can go through the same sequence of steps. There are two permissable ways to chart this. (1) Use an "or" entry for record B; draw the line from the entry into the next operation box. Then, at the end of the sequence of steps, use a Separate operation to split the two types of documents apart. (A Compare and Branch operation is more correct logically, but the Separate operation is easier to use in the present case.) (2) Sometimes, however, it would be confusing to use the method described above, since records A and B are so dissimilar. When charting the processing of record B, then, the analyst (when he comes to the same sequence as he used in charting record A) can leave a blank line on his chart and insert the note "Use steps___through___from page___."

II CHART FORM

Below is a sample of the chart form used, showing the column headings:

Entries	Main line Flow	Exits	Second-ary flow		Rec-ords		Record or field		Fields	Re-marks

The meaning of the headings is reasonably self-explanatory. All entries will be shown in the first column, properly numbered or lettered and page numbers of pertinent exits shown. The next column is where the bulk of the charting will be done--the Main Line, or the sequence of steps being described. The exit column is similar to the entry column; in the case of a "file, then stop" exit, both the merge symbol and the dead end exit may be shown in the exit column.

The secondary flow column has been provided to reduce the need for exiting and re-entry to handle minor variations in procedures. The use of this secondary flow column should be quite restricted. For one thing, no separate entries should be made to the secondary flow column; rather, it is limited to branches out of the main line via Compare and Branch, Search, Separate, or Decision operations. In addition, if the secondary flow involves over 6 operation symbols, an exit should be made to another page of the chart, and a re-entry provided.

There should be only one operation symbol per line--either an entry, main line, secondary flow, or exit symbol--so that the proper notation can be made in the other columns of the chart.

The three narrow columns which have no headings are provided for inserting prepositions, conjunctions or verbs. For example, "Insert on Record A Field X from Record B."

Recommended procedures for filling out the remaining columns are shown on page 10. The last column is for remarks; if the remarks exceed the space provided, refer to a supplementary note sheet, as "See Note 1."

Following is a sample of the identification block at the bottom of the chart form:

Subject Charted Procurement (Direct Mtls)	
Chart Begins Initiation of Purch. Ord.	Last Exit # 9
Chart Ends Preparation of Purch. Ord.	Last Entry # 4
Charted By A. Close	File Code
Page 1 of 9	Date 12/3/56

It is very desirable to have these blocks filled out completely, indicating the activity being studied, who is doing the charting, the date, and so on. The blanks for "Chart begins" and "Chart ends" are most helpful when searching through a group of charts to find a particular sequence. In these blanks, indicate roughly what part of the whole activity is being covered by this one chart.

In some cases, the entry "Miscellaneous Exceptions" is as close as the analyst will be able to come to describing the subject covered by the particular chart.

III CHARTING TECHNIQUE

1. General Procedures

   a. Draw all charts (even preliminary ones) on the 8 1/2 x 22 inch forms provided, using No. 2 (soft) pencil. Do not use any pencil as hard as No. 4, as this hardness will not reproduce well. When using a soft pencil, it will be necessary to be careful not to smudge the lines.

   b. It is suggested that both the senior analyst and the analyst draw charts of initial interviews. These charts should then be checked against each other for consistency of approach.

   c. The senior analyst should review all new process charts on a frequent basis.

   d. It is anticipated that project review meetings will be held weekly during the data gathering phase, to discuss the progress of the studies. Discussion of charting technique will be held at those meetings.

2. Beginning a chart

In beginning the charting of an activity, do not lump all possible types of transactions together artificially. If all types of transactions come in from a common source, such as the mail room, then this fact should be shown, followed by a Separate operation to select out one type of transaction for analysis. But if the transactions come from different sources, just select the most common one and start charting it.

If the information originates at the beginning of the chart, the first entry should be numbered "1". If the activity is started by the receipt of a document from another part of the system, or from outside the system, then the first entry should be lettered with"A". It is hard to define this point too precisely, and it is somewhat arbitrary. For instance, the receipt of mail from outside the business can be considered as external to the system, so that the first entry should be lettered, not numbered.

The objective of the lettering is to quickly point out those entries (and exits) that tie in with other charts in the system. If the analyst knows that an entry is fed from some other chart, he should use a letter.

3. Ending a Chart

When the analyst comes to the bottom of a chart, and is simply going to continue charting on the next page, he should not use an exit and a re-entry on the next page. Simply draw the line to the bottom of the chart, and pick it up at the top of the next chart.

4. List of Open Exits

After charting has been completed, the analyst should prepare a list of the exits that refer to other studies. Each exit should be described briefly, so that it can be understood from just the list (without having to refer to the charts), and the other study to which it refers should be indicated.

5. Working from Other Charting Forms

It is sometimes possible to do much of the charting from flow charts and procedures charts prepared for other purposes. The big difficulty here is that these other charts usually concentrate on the main line flow only, and do not specify the condiditions under which alternative paths of processing are selected. This branching will have to be uncovered by interviews.

Another example of charts that can be used is an IBM procedures chart. These charts have usually been developed for instructing machine operators as to the sequence of machines to be used on a job. As such, they very much reflect the particular limitations of the machines. If a process chart were to be drawn from these charts directly, there would be a considerable waste of effort. For instance, a half of a page of charting would be needed to show the insertion of an invoice number in a deck of billing cards, due to the way that this operation must be done on the machines. Actually, all that would be desired would be the one operation - "insert". Therefore, the person making process charts from IBM flow charts will have to do some abstracting before drawing the process charts.

6. Control Totals

The use of control totals is very common, both in manual and punched card operations. The purpose and technique of these control totals are quite well understood, therefore, the analyst should not go to any pains to chart out the method of preparing control totals, of segregating different types of control totals, or of specifying what is done when an error is detected. Simply indicate a Compute operation, an Insert operation and later a Compare and Branch operation. For the "Not Equal" output of this last, simply add the note, "Correct Clerical Error".

IV DEFINITIONS

File A collection of data, generally applying to one aspect of a business. A file consists of a group of "records" (to be defined), usually each of which contains the same type of information. The records are arranged (in most cases) in a specific logical sequence. Example: personnel file, inventory file, model list file, etc. (Note that the word "file" is used as a noun, not a verb.)

Record One member of a file; a record usually contains all pertinent information on a single item. Examples: a personnel record for one employee, an inventory record for a raw material type, or an entry in a log book.

Field A sub-division of a record; one or more related characters of data referring to a single aspect of an item. Examples: date of birth of an employee, employee's name, etc.

Character A character of data may be a numerical digit (0 to 9), an alphabetic character (a to z) or a special character ($, blank, @, etc.)

Data Known facts; generally, recorded facts, as used herein. The term "data" is often used synonymously with "information" although data generally includes redundancy and errors as well as information.

Information Communicated knowledge (more precise engineering definitions are available but will not be presented here). It is important to know that information can be measured in terms of "binary digits" (bits), where each bit is equivalent to one yes-no decision.

Form A printed form used in manual data processing; generally is covered by the definition of "records".

Transactions A record of an event, an action. Examples: a new customer order, a shipment to a customer, an employee clocking in at work, etc.

Key One or more fields of data within a record used for controlling the handling of the record -- sequencing, merging, searching, and separating. Example: Employee name in sequencing personnel records in the personnel file.

Main Line The predominant flow of data in the specific area being charted. A sequence of operations which is a secondary branch and exit on one page of a chart will generally be the main line on another page where it is charted.

INFORMATION PROCESS ANALYSIS

BY
BURTON GRAD AND RICHARD G. CANNING

Reprinted from
THE JOURNAL OF INDUSTRIAL ENGINEERING
Vol. X, No. 6, November-December, 1959

Information Process Analysis

by **BURTON GRAD**

General Electric Company, New York

and **RICHARD G. CANNING**

Consultant, Vista, California

INFORMATION Process Analysis is a new charting technique developed by the General Electric Company to aid the introduction of electronic data processing systems in the office. Actually, it has applications considerably beyond this original purpose. However, the present discussion will deal primarily with its application to electronic data processing.

The introduction of electronic data processing equipment is forcing a greater awareness of the *systems* aspect of information processing than has, say the introduction of punched card techniques. Because of their speed, cost, and capabilities, these EDP systems are causing management to cut across existing functional lines in the development of new procedures. This is true not only of the different areas within the office but it is becoming equally true in the integration of office and factory.

How is this systems concept different from what has been done in the past? In a relatively few cases, procedural work has had a true systems approach. However, in the vast majority of cases, the emphasis has been on cost improvement projects designed to do a certain part of the business in a more efficient manner: for example, better methods for drilling a hole or for preparing factory paperwork: substituting a less expensive part which will perform the same function as the previous part, and so on. Most of these projects do not consider (or consider only briefly) the *inter-relationships* between the various activities of the business. The "bricks" are analyzed in great detail but the composition of the "mortar" is too often ignored.

There always seem to be numerous valid reasons for *not* studying these interrelationships. First, they are more difficult to analyze than are the activities themselves, and require a broader knowledge of the total business. The system interrelationships cut across existing functional and sub-functional lines so that no one is quite sure to whom they belong. Finally, their study takes longer and does not have the glory of an immediate reward.

Even in the face of these arguments, however, experience has shown that the "bits and pieces" approach by itself cannot produce the gains which can be realized from a study of the business as a whole—a systems study. Also, the advantages of electronic data processing lie, to a great extent, in tying together logically related activities. Maximum system speed and accuracy result from integration along lines of information flow, rather than within individual functions.

In a systems study, care must be taken to make the analysis much more comprehensive than the usual "procedures" analysis. Improving a particular process or activity is not the primary goal, but rather examining the necessity for having the process at all. This approach is not hardware-oriented; it is an effort to find out *why* things are done at all and then, after constructing a logical pattern for operating the process, to determine the real equipment needs.

The Information Process Analysis technique to be described was designed to meet the needs of such a systems study.

THE SYSTEMS APPROACH

In beginning an EDP systems study, we are more interested in *what* is going on in the business rather than in *how* it is being performed or *who* is performing it. For example, there are certain types of information which, though of interest in a conventional procedures study, are *not* of interest in a systems study:

1. We are not interested in the manual procedures that are used per se—e.g., how many clerks and typists work on a form during its preparation, or points along its route where the form is stored temporarily.
2. We are not interested in the fact that several people are involved in an operation, each doing part of the over-all job.
3. We are not interested in the layout of the clerical work area or the types of office equipment used.
4. We are not interested in the "exceptions" which arise due to internal clerical errors; however, we are interested in the types of information errors transmitted into the organization from outside over which the organization has no control.

Instead, since we are interested in learning *what* the mechanized system must be able to handle, we *are* interested in the following types of information:

1. The logically necessary alternative procedures which are needed in the business for handling main line flows as well as so-called "exception" cases.
2. The management control reports that are developed by the organization, and the purposes they serve.
3. The reports and other pieces of information that must be

transmitted outside of the organization, for legal or other reasons.

4. At first, a preliminary pin-pointing of all complex computation operations and decision areas. Later, we will need a more detailed understanding of these operations.

Conventional charting techniques usually involve symbols representing such clerical activities as transport, store, delay, inspect, and a general symbol representing all operations. These conventional techniques have been aimed at, and these symbols are useful for, one of two main objectives: a. to find out where human clerical time is being spent, so as to shorten the over-all process time, or reduce the man-hours to do a job, or b. to lay out a prescribed sequence of operations for a reasonably complex job, for the guidance of the clerks and operators—e.g., flow charts of punched card operations. Since our primary interest in a systems study is not in either of these objectives, it is not surprising that their associated charting techniques prove relatively ineffective in a systems study.

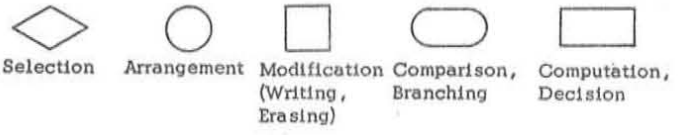
Instead, we are interested in learning of the flow of information throughout the business, for directing manufacturing and other activities, and for feedback and control. *All* uses of the information must be accounted for, so we soon find the scope of the systems study spreading throughout the entire organization. The Information Process Analysis technique designed to meet this objective is based on: a. reasonably precise definitions of basic data processing operations, represented in the form of symbols, and b. a charting procedure to make sure that the necessary descriptive information accompanies each symbol.

In other words, conventional charting techniques treat information systems as though they were material handling systems, where the material is paper. Since this new technique deals with the information itself, not the paper, we are able to concentrate on the different data processing operations.

INFORMATION PROCESS CHARTING TECHNIQUE

Fundamentally, there are only seven symbols used, five of them representing operations (where someone does something), and two representing conjunctive, or connective devices. These basic symbols are shown below:

OPERATIONS

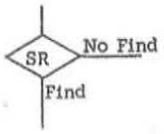


CONNECTIONS

In practice, there are variations for some of these symbols, so that a total of 11 different terms are used. Each of these 11 terms and symbols is described briefly.

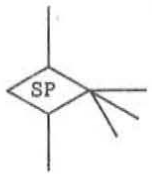
The Charting Symbols

1. Select, Search (SR)



Search means to extract a particular record from a file of similar records which are sequenced by the field on which the search is being conducted. Illustration: Search the Planning File for the Planning Card covering part number 374255-1.

2. Select, Separate (SP)



Separate means to select one or more records from a group or file, according to a key field. Sequencing by the key is not necessary. Illustration: Separate the copies of the Purchase Order by destination.

3. Arrange, Sequence (AS)

Sequence means to arrange (sort) a group of records into ascending or descending order according to a key field. Illustration: Sequence time cards by employee pay number.

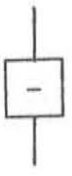
4. Arrange, Merge (AM)

Merge means to combine two or more groups of records which are already in sequence by one or more key fields into a single sequence on the same keys, or to place a record in a file. Illustrations: Merge new Parts Lists with the Parts Lists File by Parts List number; merge the employee time cards with the employee job cards by employee pay number.

5. Modify, Insert (+)

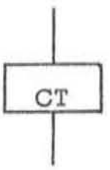
Insert means to create a new record or to add one or more fields of information to an existing record. Illustrations: Prepare a new Purchase Order; sign a Freight Bill.

6. Modify, Delete (−)



Delete means to remove one or more fields of information from an existing record. Illustration: Delete a terminated employee's pay number from the active employee ledger.

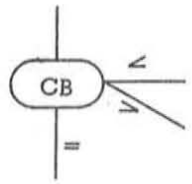
7. Compute (CT)



Compute refers to an arithmetic formula incorporating basic arithmetic operations: Add, subtract, multiply, divide, exponentiation, trigonometric functions, etc. It does not contain any comparison or choice operations. If the result of a computation is used in a comparison, this should be indicated separately. Illustrations: Total week-

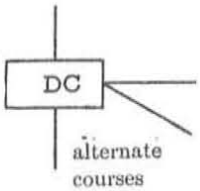
8. Compare and Branch (CB)

ly pay equals hourly rate times number of hours worked; stock on hand at end of period equals initial stock plus receipts minus disbursements.



Compare and Branch is the basic choice operation which involves a defined or fully prescribed decision. Illustrations: If the product model number is incomplete, then pass the order to Engineering; if an employee's accumulated salary year-to-date is greater than $4200, do not deduct social security.

9. Decision-Making (DC)



Decision-Making is a higher level than Compare and Branch. It is used when a choice is not based on a clear cut set of rules; in other words, judgment is involved in a decision-making process. It is possible in a decision-making operation to indicate what factors are considered and often even the relative importance of these factors. To the extent that exact "weights" can be determined and all alternate paths noted the operation reduces to a series of Compare and Branch operations. Illustrations: Determine the quantity of Model XYZ that will be sold within the next 12 month period; decide whether a job applicant is suitable for a particular task.

10. Connection, Entry (EN)

An *Entry* serves to start a routine or to bring additional information into it. It may come from another part of the same chart or from a different activity entirely. Illustrations: The customer order entering the order service routine; a pay voucher coming to Cost Accounting from Payroll.

11. Connection, Exit (EX)

An *Exit* is the means by which an activity is terminated. It may go to another part of the same chart, it may go to another activity, or it may be the end of the routine. Illustrations: All requisitions requiring special engineering review go to Engineering; a pay check is given to an employee for his previous week's work.

The foregoing descriptions are necessarily brief; more complete definitions for each operation symbol may be obtained from (1). A summary of their use on the charting form is shown in Figure 1.

EXAMPLES OF PROCESS CHARTS

To give an idea of the technique in action, we have included two examples, one trivial and one from an actual systems study. Figure 2 indicates some of the "data processing" operations which might be followed by a young man in search of a date.

The steps on the chart are relatively self-explanatory. Alphabetic entries and exits provide connections to and from other charts while numeric entries and exits refer to different parts of the same chart. Line 3 shows a routine decision step, involving a Compare and Branch operation, while Line 15 shows a difficult decision involving judgment. Line 8 shows the procedure for entering a new record into a file (merging). In general, it will be seen that the technique brings out clearly the various alternative circumstances that can arise. Also, space is provided for a brief explanation of each step in the process. Thus, charts drawn by one person may be easily read by others.

Figure 3 is a reproduction of one part of a chart on tooling activities in a manufacturing organization. The particular operation being charted is the receipt of raw material at the tool crib, where the material is destined to be made into a tool. Line 4 shows a searching for a copy of the purchase order after the material is received; the charter may or may not chart the "no find" situation depending upon how significant it is. Line 6 indicates that the tool crib attendant checks to see who the material is for; the "not equal" branch indicates that it is for someone else, and Exit 3 connects to the charting of that condition. Line 7 here refers to the fact that "someone else" to whom the material is to go wishes it to be stored temporarily in the tool crib; after it has been suitably marked in the other operation, the information again enters the main line being charted. The remaining operations

INFORMATION PROCESS CHART

Entries	Main Line Flow	Exits	Secondary Flow		Records		Record or Field		Fields	Remarks
	AM 1				Name of Record or File	By	Key	with	Name of File	
	AS 2				Name of Records Being Sequenced	By	Key			
	+ 3			on	Record onto which data is being inserted		Field(s) being inserted	from	Record from which data is obtained	
	- 4			from	Record from which data is deleted	Delete	Field(s) being deleted			
	CT 5				Field(s) being computed	from	Field(s) supplying information	on	Records from which field(s) were obtained	Formula, if easily obtained
	DC 6				Brief name of decision					List of major factors
	CB < = > 7			using	Record involved	compare	Field	with	Field	
					Record	and	Record	by	Field(s)	
	SR 8				Name of file being searched	by	Key	from	Record from which key was obtained	Do not search a record for a field
	SP 9				Name Records or file being Separated	by	Key			
	8					(Description of Exit 8)				
	9					(Description of Exit 9)				

FIG. 1

shown here indicate the steps taken to determine where to store the material in the tool crib. The original chart continues beyond line 10 for this one function, and in fact, eleven of these 18-line charts were used in charting the tooling activities.

EVALUATION AND SPECIAL PROBLEMS

The speed with which Information Process Charts can be prepared seems to bear a close relation to the speed of programming a problem for a large scale digital computer. Interviewing, charting, rechecking and summarizing result in about 2 steps per hour, although this improves with experience. Remember though that these steps are more powerful than the normal computer operation codes since each symbol may represent an entire subroutine like "Sequence," "Merge," etc.

As in any new technique there is a significant learning curve effect. As experience is developed, speed and accuracy improve considerably. It is also apparent that different types of problems require somewhat different viewpoints and charting "tricks."

In first applying the technique the procedure was described with brief examples to a group of trainees and first line supervisors and specialists. The initial charting accuracy was substantially less than expected, probably because the training techniques were at fault. It is our conclusion that the best teaching method would be a practical example (like receiving of purchased parts) performed individually by each member of the team and then discussed and analyzed as a group.

In choosing the particular format and charting arrangement much consideration was given to the location of the symbols. In contrast to the usual computer charting which uses a "large" sheet of paper and writes in each block a description of the operation, we felt that the in-line page type arrangement was more easily traced and understood by the non-charter, yet this produces its own problems in excessive numbers of sheets and a lack of "Gestalt" or total grasp.

Since grasp and insight are among the main reasons for choosing process charting in the first place, it would have been most unfortunate if they had been lost because of difficulties of paper representation. Two approaches were used to help solve this problem. First, the original process charting forms were modified to allow the parallel indication of a secondary flow beside the main line. This permits a visual continuity and apparently saves many exits and entries. The second solution was the introduction of summary charts which served to review the over-all pattern of a particular business activity. These gave a sort of index to the detail charts and helped significantly in grasping and absorbing the major implications of the activity.

It is also evident that this summary process is vital to the desired insight into the ramification of the whole business. In other words, there needs to be a hierarchy of summarization eventually leading to a "Master Diagram" of the key processes in the business.

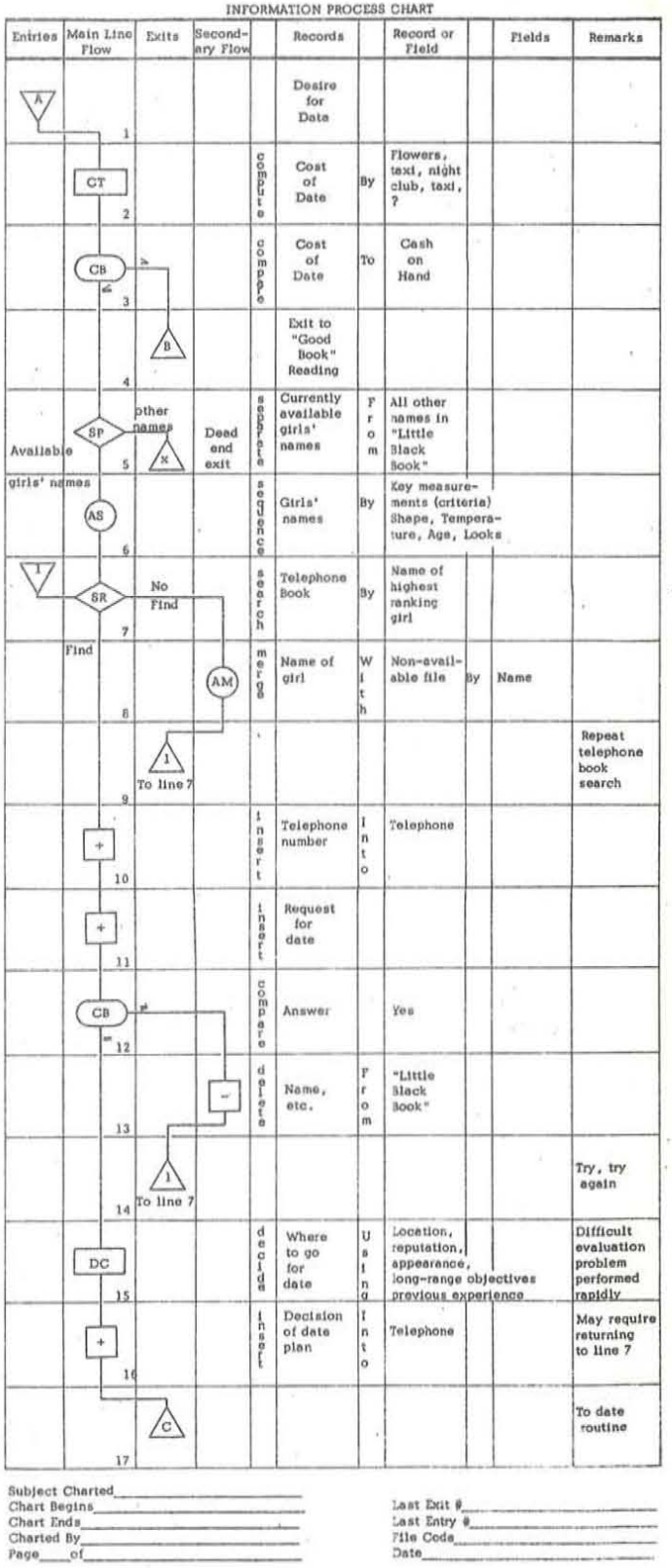


FIG. 2

To elaborate on the charting, reference is made to the column headings shown in Figures 1, 2 and 3. The mean-

INFORMATION PROCESS CHART

Entries	Main Line Flow	Exits	Second-ary Flow		Records		Record or Field		Fields	Remarks
2	1				Tool crib attendant receives copy of purchase order for any tool crib routed item being ordered					
	AM 2			merge	Copy of purchase order	in	Purchase order book	by	Purchase order number	
B	3				Material with 1 copy of receiving report arrives at tool crib					
	SR Find 4 / No Find 6			search	Purchase order book	by	Purchase order number	from	Receiving report	
	+ 5			insert	Quantity received, date received	on	Purchase order			
	CB = 6 / ≠ 3			compare	Destination on purchase order	with	Tool crib number			
5	7				From chart 2--material destined for a planner, engineer or foreman					
	CB = 8 / ≠ 4			compare	Material	with	Special purpose tool, gauge or fixture which has an assigned tool number			
	SR Find 9 / No Find 7			search	Tool number file	by	Tool number	for	Location	
	+ 10			create	3 copies of tool card AF-280FW	from	Purchase order, dimension card		And insert all fields	

FIG. 3

ings are reasonably self explanatory. The following definitions of terms are quoted from (1):

File A collection of data, generally applying to one aspect of a business. A file consists of a group of "records" (to be defined), usually each of which contains the same type of information. The records are arranged (in most cases) in a specific logical sequence. Example: personnel file, inventory file, model list file, etc. (Note that the word "file" is used as a noun, not a verb.)

Record One member of a file; a record usually contains all pertinent information on a single item. Examples: a personnel record for one employee, an inventory record for a raw material type, or an entry in a log book.

Field A sub-division of a record; one or more related characters of data referring to a single aspect of an item. Examples: date of birth of an employee, employee's name, etc.

Character A character of data may be a numerical digit (0 to 9), an alphabetic character (a to z) or a special character ($, blank, @, etc.)

Data Known facts; generally, recorded facts, as used herein. The term "data" is often used synonymously with "information" although data generally include redundancy and errors as well as information.

Information Communicated knowledge (more precise engineering definitions are available but will not be presented here). It is important to know that information can be measured in terms of "binary digits" (bits), where each bit is equivalent to one yes-no decision.

Form A printed form used in manual data processing; generally is covered by the definition of "records."

Transactions A record of an event, an action. Examples: a new customer order, a shipment to a customer, an employee clocking in at work, etc.

Key One or more fields of data within a record used for controlling the handling of the record—sequencing, merging, searching, and separating. Example: Employee name in sequencing personnel records in the personnel file.

Main Line The predominant flow of data in the specific area being charted. A sequence of operations which is a secondary branch and exit on one page of a chart will generally be the main line on another page where it is charted.

All entries will be shown in the first column, properly numbered or lettered and page numbers for pertinent reference exits shown. The next column is where the bulk of the charting will be done—the main line, or the sequence of steps being described. The exit column is similar to the entry column.

The secondary flow column has been provided to reduce the need for exiting and re-entry to handle minor variations in procedure. We believe that the use of this secondary flow column should be quite restricted. For one thing, no separate entries should be made to the secondary flow column; rather, it is limited to branches out of the main line via Compare and Branch, Search, Separate, or Decision operations. In addition, if the secondary flow involves over 6 operation symbols, an exit should be made to another page of the chart, where it is charted as a main line.

There should be only one operation symbol per line—either an entry, main line, secondary flow, or exit symbol—so that the proper notation can be made in the other columns of the chart.

The three narrow columns which have no headings are provided for inserting prepositions, conjunctions or verbs. For example, "Insert *on* Record A Field X *from* Record B."

PROCESS CHARTING—DISCUSSION

There are a few points of special interest that should be discussed briefly, about the use of the foregoing symbols.

First, the operation of "reading" is implied in all of the above symbols, and is not called out as an independent step. It is assumed that the person or machine doing the processing must first read the information from the documents.

Next, it should be re-emphasized that whenever Compute and Decision-Making operations are encountered, the analyst should not initially spend much time in detailing the routines. The process charting simply pin points these computations and decisions, and gives some understanding of what is involved. After the initial data gathering phase, many of these operations will need to be analyzed in greater detail.

Occasionally an analyst will come across a "loop" operation, where the same series of steps must be performed on a number of items, before continuing with the processing. An example of a loop might be computing the

standard cost for each operation in making a part, before the standard cost of the finished part can be determined. Indicate such loops as simply as possible; e.g., "steps ——— through ——— repeated for each item on the record."

Also, in some possible loop operations (where there may be one or more items on the list), a totaling or summary is performed afterward if there were more than one item on the list. Do not try to chart this out, using conditional transfers. Simply indicate the totaling operation right after the last operation of the loop, and indicate in the Remarks column, that it is used if necessary.

Occasionally a matching operation will be encountered, such as a group of time cards being matched against a payroll file. It may be of interest to know if there are any time cards for which there are no payroll records, and if there are any payroll records for which there are no time cards, as well as matching the payroll records and time cards for computing gross pay. Such situations as this may be charted by two Search operations in sequence. In the case above, search the payroll file by the time cards' employee numbers indicating "Find" and "No Find." Then search the time card "file" by the payroll records' employee numbers indicating "Find" and "No Find."

The "and" and "or" concepts should be mentioned briefly. Sometimes two records must be brought together, so as to go through a series of processing steps; record A and record B. If record A has been charted as the main line, then record B can be brought in by connecting an "and" entry to the main line with a solid circle. See Figure 4.

Note that the "and" entry implies no merging, searching or sequencing. There is no file; only individual records are involved. Two or more records may be brought together, so as to form a larger record. If record A is being chartered as the main line and B is located in a file, then B should first be obtained by a Search operation.

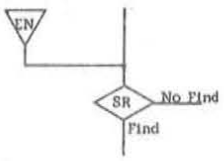


FIG. 4

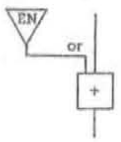


FIG. 5

For the "or" situation, record A *or* record B (but not both together) can go through the same sequence of steps. If the two types of records are similar or are logically related, it is often desirable to show this by an "or" entry and a second input line to an operation box, such as Figure 5. Then, at the end of the common sequence of steps, a Separate operation may be used to split the two types of documents apart. (A Compare and Branch operation is more correct logically, but the Separate operation is often easier to use.)

Sometimes, however, records A and B are so dissimilar that it would be confusing to use this "or" method. When charting the processing of record B, the analyst (when he comes to the same sequence of steps as he used in charting record A) can leave a blank line on his chart and insert the note, "Use steps ——— through ——— from page ———."

FUTURE USAGE POSSIBILITIES

We have tried to look ahead and deduce the logical implications of the process analysis technique and have found that many ideas can be suggested.

One idea is concerned with the use of punched cards for analysis of individual operations and the preparation of summaries. For example, by defining the records and fields carefully we could use one punched card for each line on the process chart. This is similar to work reported through the American Management Association by two Lehigh professors (4). This might be a convenient procedure for reducing the clerical content required to draw the flow charts.

Another possibility would be the evolution and development of higher level symbols to represent recurrent data processing elements. Examples of this might include edit, translate, and verify. These should be particularly meaningful for summary charting and would also indicate computer sub-routines which should prove useful.

While much of our present charting seems record oriented, this is merely a space reduction convenience. The record stands for or represents the fields it contains. Since all operations are performed on the fields themselves, it might be possible, with appropriate identification and coding techniques, to define all functions in terms of the fields instead of the records. This would be advantageous in leading toward nonredundant systems. Reference (3) is an extremely thought provoking paper on one aspect of this subject.

Since Information Process Analysis can be used to describe any data processing operation, it might be interesting to investigate the application of this language to computer programming. Because these charts are at a somewhat higher level than the charts now used, a significant saving in time and effort could result.

Another unexpected area which was uncovered was the strong similarity between physical processing and data processing systems. This is described in (2). As an analogy, we can consider that the part corresponds to record, and each hole, groove or surface is a field inserted in the record. Parts may be associated together in a "file" (stock room) which can be searched for a particular part. Compare and Branch can be used to represent inspection operations, and merge would imply parts accumulation to precede assembly. While this simile can be overdrawn, there nevertheless appears to be a sound foundation for further study with the implication that physical processing systems are directly analagous to data processing systems.

Since the process charting technique is organized around a generalized set of rigorously defined symbols, it may help to solve another common problem: the present inability to communicate solutions to various business systems problems. The Department of Defense has initiated a commendable program for the development of a Common Business Language—a computer programming system that uses English sentences which can be compiled into running programs for most machine types. But this Common Business Language will most likely be at a more detailed level than the language described in this paper, and may not be as satisfactory for communications at a systems level. It seems to us that the progress of data processing as a science requires establishing such common problem-oriented languages so that we can more successfully communicate with our fellow systems designers.

REFERENCES

(1) GRAD, BURTON, CANNING, RICHARD G., AND DERUBBO, RALPH H., "Process Charting," April, 1957, available through Mr. R. G. Canning, 614 South Santa Fe Avenue, Vista, California.

(2) GRAD, BURTON, AND O'NEAL, W. C., "Making Operations," April, 1957, available through Mr. Canning.

(3) LIEBERMAN, IRVING J., "A Mathematical Model for Integrated Business Systems," *Management Science*, July 1956, pp. 327-336.

(4) RICHARDSON, WALLACE J., AND HEILAND, ROBERT E., "Integrated Procedures Control," *Engineering for Paperwork Control*, American Management Association, Office Management Series No. 143.

I. B. M. CONFIDENTIAL

March 31, 1960

Subject: The Integrated System Project at General Electric

The enclosed material represents a reporter-like description of work that I was responsible for while employed at General Electric.

All the information contained has already been released, much of it at a System Development Corporation meeting in July, 1959. While it is realized that many details are lacking, this is necessarily so in order to avoid any implication of disclosing General Electric confidential information.

However, it is my feeling that the general area covered and certain of the techniques described are of value to designers of industrial processing systems.

Burton Grad

Burton Grad
Programming Systems
IBM Applied Programming

BG:gms

THE INTEGRATED SYSTEMS PROJECT

AT GENERAL ELECTRIC

A summary of non-confidential information about the accomplishments and philosophy of a research activity directed at the design of an automatically operated business.

Burton Grad
Programming Systems
March 28, 1960

THE INTEGRATED SYSTEMS PROJECT AT GENERAL ELECTRIC

General Electric's Services' organizations have attempted to design an advanced automatic system, one that would be able to respond more efficiently and more economically to incoming customers' orders.

Certain general objectives were established in order to accomplish this task:

1. The new system was to be economically practical and technically feasible; it should be broadly applicable to many departments of General Electric.

2. The system should be multi-functionally integrated and provide a close linkage between the office and factory.

3. It was to be designed with bold innovation in order to break the historically accepted business systems patterns.

They hoped through research to develop new concepts and tools for use in designing such new systems. They hoped to develop new criteria for technically and economically sound approaches to automation that would help determine which particular new techniques should be used in specific businesses. They wanted to provide a foundation for future progress through research and development.

To pick an initial area for this exploration, they analyzed some of the current weaknesses of industrial systems.

1. Typically, delivery cycles are quite long when compared to the product's cost. This is particularly true of manufacturing cycles in relation to the actual processing time. Work-in-process inventories are correspondingly excessive.

2. Indirect labor costs are increasing steadily. Many factories even joke about the fact that they can't make a shipment until the paper weighs as much as the product.

3. A third area is the high redundancy of information used in factory paperwork. For example, on a line of shafts used by a successful motor manufacturing department, it was found that some three hundred different drawings had been prepared

over the course of two years to take care of each minor variation. On each of these drawings, there was some sixty to seventy fields of information. Of these fields, better than 80% were completely fixed. For every shaft only 20% were truly variable.

With these and other significant problems in mind, they sought the areas of an industrial business system that would have the greatest impact in these areas of opportunity. This heart of the business process was called the Main Line System. This Main Line System included requisition editing, product engineering and drafting, manufacturing and quality planning, cost determination, production control and purchasing plus the actual shop operations.

Substantial amounts of money are involved in the Main Line System. Normally, 100% of the direct labor and 100% of the direct material is tied up in the Main Line System. At least 40% of the indirect manufacturing expenses are also in this area. All of the productive raw and in-process inventories are in this category as well as approximately 80% of the plant and equipment investment. In total, this area probably accounts for 75 to 80% of product costs and a similar percentage of investment.

To perform a research and development job on these multi-functional problems, a multi-functional team was organized representing the various business functions: Engineering, Accounting and Manufacturing. In this particular study, Marketing and Employee Relations were not included because the particular system defined did not require their extensive contribution.

In a decentralized company like General Electric, planning such a program is not uncomplicated. There are two types of problems that arise:

1. The integration of staff planning people into a closely knit team is complicated by the fact that there is no component in the organization responsible for multi-functional systems work. Therefore, effective work requires mutual participation of the functional services who have no common manager short of the Chairman of the Board, who is the Chief Executive Officer. Basic problems like leadership, budgets, relative functional roles, decision making, reporting, etc. were major problems which had to be overcome.

2. A second problem in a large decentralized company is developing concepts in a framework that will be both understandable and meaningful to the many operating components. Because they have such a variety of products, processes and markets, that generality is elusive.

They felt that an ivory tower approach would not provide an effective atmosphere for integrating systems design work nor would it particularly aid in selling any new concepts which were developed. What was needed was a real business -- a "living laboratory". This selected operation had to be representative of the breadth of businesses in which General Electric engages. Also in picking a business, they wished to select one where the existing information was in sufficiently clear form to be readily usable since they felt that "you can't automate a mess". They sought a well run business where they could concentrate on advanced development rather than having to devote time to cleaning up existing problems.

They also felt that by carrying on their research in a particular business, the systems team would have its attention focused on specific, clearly defined problems rather than the more vague, imaginary difficulties. In this way, the creative contributions were concentrated on the areas most needing improvement.

A business system has five elements:

1. It has information resources including the various decision criteria which are currently in the form of reference files.

2. It must have decision makers capable of taking the transaction inputs and matching them with the information resources to determine a course of action.

3. It must have communications channels enabling it to transmit its decisions and in turn to receive feedback information concerning operating performance.

4. It must have a physical processor which actually transforms material through the use of men, machines, and energy in accordance with the instructions given it.

5. The physical processor must have access to the physical resources of men, machines, materials and energy.

After the multi-functional organization was completed, a clear and specific design program was followed. The first stage was that of data gathering. This involved getting all the facts concerning present inputs and outputs, volume of devices, design variations, manufacturing facilities, historical performances, etc. This phase began in November 1958 and took approximately six months; it led directly into the second phase: problem analysis. During problem analysis, all the information gathered during the first phase was digested, reviewed and an effort made to determine clear cause and effect relationship between changing external conditions and changing internal performance.

The third phase of the program was that of preliminary systems design. This achieved a first specification of what might be called the basic system. This lasted approximately one month and brought into play the design efforts of not just the general systems designers but all the specialists in the various areas.

The fourth stage was that of detailed systems design. This refined the specifications in great detail. It clearly indicated those phases which needed to have their technological feasibility proven and those that had already been clearly demonstrated in previous work.

The fifth phase was that of construction of a prototype to demonstrate application of the new ideas. It was a bread board model, and not yet an actual operating model.

The sixth and final phase of the program was that of testing, training and evaluating. The bread board model has been tested against a variety of circumstances and found to be very satisfactory. The training objective was carried out during 1959 along with initial evaluations of potential savings.

While these are quite conventional steps, the important new concept was the application of the systems approach to business systems problems. With this systems approach, they treated the entire Main Line System as though it were a big black box with only one transaction input, the customer's order, and only a single basic output, the finished product. All that went on in between was subject to analysis and redesign. The systems approach was intended to design a new Main Line System and provide an opportunity to ignore present techniques and ignore all of the conventional organizational or functional divisions of work and to really concentrate, without inhibitions, on reconceiving the solution.

A review of the steps included in this Main Line System will give a clearer understanding of the particular scope of this project.

The present Main Line System starts with a customer's order. This specifies what the customer wants in functional terms, such as size, color, rating and other product requirements. A typical order then goes through certain conventional steps.

1. It is edited to eliminate ambiguities and to put the order into the proper, most usable, internal form.

2. Then this order is engineered and drafting prepares documents needed, namely blueprints, bills of material, etc.

3. Based upon this design information, the manufacturing engineers then perform the operational planning on how to make the product and what the time allowance should be for the various labor and machine operations.

4. In a similar manner, the quality control planning procedures are determined, establishing standards, methods and frequency of quality analysis.

5. And then, using the existing records and files, cost information is accumulated, compiled and analyzed.

6. Production control then takes over to determine when the parts are needed as well as how many are to be purchased and made. Typically, this includes the functions of customer promising, scheduling and inventory control.

7. Finally, instructions in the form of vouchers, purchase requests, etc., along with blueprints and other necessary papers are transmitted to the factory to direct the manufacture of internally made components or to purchasing for outside material procurement.

In each of these steps, information is taken from the previous function, typically in the form of written documents, and used to produce the next document or output with the aid of information reference files: material lists, blueprints, planning cards, quality records, cost cards, etc. In short, the Main Line System converts the customer's order into

a finished product. Present systems are usually based on human-to-human communication with extensive file reference. The use of mechanical aids is generally still limited. The shop area is often characterized by job-shop type facilities, high buffer stocks between operations and long manufacturing cycle times.

The results of the work have indicated that a customer's order can now be automatically converted into parts of a specially designed product, performing all of the Main Line System's steps inside the computer. This automatically provides all of the factory's action documents: purchase orders, operator instructions, quality instructions, punched paper tapes to run numerically controlled machines, customer promises, bills of material, stock order recommendations, withdrawal notices, shipping papers, etc.

As a result of this Project, many new techniques were developed to help the various General Electric departments design integrated, automatic systems. For example, new techniques have been developed for decision analysis. New techniques have been conceived for part and product representation and identification. New ideas have been formulated for computer programming. All of these concepts taken together have changed the economic feasibility of installing integrated, automated business systems.

They feel that there are many benefits from these concepts. In order to clarify them, the nature of each function in the computer and some of the resultant benefits will be covered.

The determination of "What to Build" is the key role of engineering; the requisition engineering activities can now be computerized. The computer can translate a customer's wants into the specific details of the materials, parts and assemblies needed to satisfy those wants. In addition, this computerized process can avoid the necessity of having to create many of the documents and records with which we have become too familiar. Outstanding savings can be realized in the preparation of model lists, bills of material, blueprints, etc.

Included in the benefits from this engineering advance should be substantially reduced engineering time and cost through the elimination of many of the routine steps which humans now take. There should be less drafting expense through eliminating many of the tasks which drafting has historically performed. A clear, logical statement of the engineering scope of a product line should make it easier to obtain an optimal level of standardization. A properly designed computerized engineering system should be easier to change and be more flexible.

With knowledge of the product design details, manufacturing engineering is then in a position to determine the best routing, work methods and time standards. Much of this work on "How to Build" the product can also be completely taken over by the computer. The possibility of automatically preparing accurate operational descriptions coupled with correct time standards for every job certainly has considerable appeal.

Another intriguing area is in the communication of the computer with numerically controlled machine tools. Three new features should have wide application:

1. A single program tape controls an automatic machine for the entire day.

2. Machined parts are automatically identified as an integral part of the program.

3. Computers are used to automatically generate machine tool programs.

"Tape-for-a-Day" Machine Tool Control. Typically, users of numerically programmed machines have achieved repetitiveness in operations by cycling a loop of punched paper tape. Thus, if ten pieces are required, the operator glues the back-end of the tape to the front and allows the looped tape to run around ten times. In this system, the same objective is accomplished by providing ten machine tool control programs in a single length of tape. Further, the same length of tape also includes a program for all other pieces to be manufactured by the machine that day. Thus, one length of paper tape provides an integrated, sequenced control program for a numerically controlled machine tool for the entire day.

Machined Parts Automatically Identified. Parts processed on numerically controlled equipment are sometimes identified in a secondary manual operation. This can be avoided by introducing an identification step in the machine tool program. For example, parts can be identified with shop identification numbers by spotting a shallow blind hold in a code matrix stamped on the part itself:

identification hole →

1	1A	2	6	51	62	64	75
●							

A more suitable, generalized version of this code matrix idea would be:

	0	1	2	3	4	5	6	7	8	9
(hundreds)										
(tens)										
(units)										

Computer Generated Machine Tool Programs. The generation of numerical machine tool programs was done on an electronic computer. This, of course, facilitates developing the "tape-for-a-day". While electronic computers are not essential, mechanizing the production of punched paper tapes (or cards) to run automatic machines improves accuracy and reduces cost.

Among the other benefits is reduced planning time since the computer takes over a former manual job. There would also be reduced planning costs since computers can do this job for less money than humans and probably, most important, the best method, more accurate planning and consistent time standards should result because of the computer's ability to follow the exact instructions that it has been given.

Quality Control: at what point to inspect or test, the quality evaluation method, appropriate time standards, frequency of evaluation, and criteria for acceptance or rejection. Here again it was found that a computer program can be prepared which will perform all of these tasks automatically. This would, in effect, determine how to evaluate the product and its components. Included in the benefits are fewer quality corrections through having the proper balance between quality failure and quality appraisal costs. There should be fewer complaints through a careful analysis of customers' needs and product characteristics. There should also be lower quality costs through the integrated planning of quality control along with engineering design and manufacturing operation planning.

Cost accounting offers another opportunity. The objective was to determine planned product costs for quotation work or for cost standards to be used for comparison with actual costs. They find again that cost standards can be automatically developed and that a computer properly programmed can also be used for establishing work-in-process inventory value.

Through this cost work, it should be possible to obtain better cost analyses by having all the facts at our finger tips when they are needed. It would be far easier to maintain up-to-date costs because of the potential simplicity in storing the cost information. There should be reduced cost determination expense through the use of a computer to replace human effort.

The next area of production control is particularly intriguing. Each of the previous steps in the computer portion of the integrated Main Line System have all dealt with tangible product characteristics: what to build from engineering, how to build from operation planning, etc. In contrast, production control, the final element of the computer portion of the Main Line System, develops a fourth dimension by determining the time and sequence in which main line activities take place.

Production control is interested in when things happen. It has the responsibility of actually carrying out at the right time the data processing and decision-making calculations necessary to support each function. Production control is concerned with the time inter-relationships of all customer orders. It is responsible for economically satisfying these customers' requirements considering the actual status of the shop.

Production control provides the scheduled release of the factory's action documents:

- purchase requests
- punched tape for automatic machine programs
- operator instructions to make and assemble products
- quality instructions for inspecting and testing
- shipping papers to deliver the customer's product

In this integrated system, the computer should daily schedule shop operations, specify operation release dates and due dates, specify specific order quantities, review inventory stock levels and issue customer promises. These orders should not be released prematurely. One key element in computerized manufacturing control is frequent feedback coupled with frequent scheduling for close shop control; using today's performance to guide tomorrow's shop decisions. In the past, a major obstacle to such tight shop control has been the mass of detailed data which had to be gathered and interpreted before any meaningful results could be obtained. Manual and even punched card techniques often sagged under this burden; but electronic computers offer the high speed, low cost calculating ability necessary to cope with this problem.

The Integrated Main Line System has daily feedback of completions for shop control. This information will be digested by the computer each night and recognized in the releases to be prepared for the following day. The parts to be started the next day will depend upon the exact status of each of the areas of the shop; whether they be behind schedule or ahead of schedule, what their status is on rush jobs and related information.

The result is a flexible system prepared to respond quickly and accurately to changes. Time delays in handling information are avoided and corrective actions can be initiated immediately throughout the Integrated Main Line System.

Developing production control rules presents some special difficulties. For example, product performance can be proven in the laboratory, operation time standards can be checked by a stopwatch, but how can you pretest a rule for customer promising? General Electric has been instrumental in applying simulation techniques to similar business problems involving many interdependent activities that change with time. The heart of shop simulation is a computer model which realistically duplicates the behavior of the shop as it processes customer orders, making allowances for set-up and processing times, absenteeism, machine breakdowns, and the like. The specific computer model developed for the Integrated Systems Project compressed four months of shop experience into a fifteen minute computer run. As a result, it was possible to test how well various proposed sets of production control rules would meet due dates and planned cycle times without actually trying them in the shop. In addition, inventory levels, employment stability and man-machine utilization could also be evaluated and compared.

Integrated production control offers several benefits. For example, it now seems quite practical to obtain a shorter main line information cycle -- actually less than one day. Similarly, electronic computers can be expected to lower paperwork costs. Shorter cycles in the office and factory, as well as improved scheduling techniques, will permit substantially lower inventories. These improvements should lead to shorter customer promises, improved service and potentially higher sales. Somewhat unexpectedly, indications are that these gains can be achieved while improving employment stability -- and without a sacrifice in promises kept and equipment utilization.

Of course, the only reason for all of this information is to procure the parts that are needed, on schedule, at optimum cost; and to direct the machines and operators in the factory to transform the raw materials into the right finished parts at the right time. This leads directly to the concept of flexible factory automation.

In the physical processing system, rather than visualizing automation as a long line of highly specialized machines and transfer devices, it may well be the important aspect of automation will be the ability of machines to switch from one task to another at little or no extra cost. The inherent flexibility of the individual machine or group of machines will be a determining factor in the effectiveness and usefulness of these automatic systems concepts. With numerically controlled machines, such as are now available, the set-up cost is generally reduced to practically zero. Hence, flexible factory automation permits direct response to the external, customer oriented requirements and not such heavy consideration to the internal shop.

This flexible factory automation will lower direct labor costs per unit through replacing human activities, where desirable, by machine operations. Machine accuracy and set-up flexibility will reduce both scrap and rework. Integrated planning and control with the right tempo will result in shorter manufacturing cycles.

The fundamental concept in carrying out this project was the idea of vertical integration. Integration is currently a by-word, but most new work has been concerned with automating common activities like payroll, inventory control or requisition processing across many product lines or the whole business. This might be called horizontal integration. However, true integration should probably follow lines of information flow; it should cut vertically through all functions in a product line. By having all the information processes linked together inside a computer, it is unnecessary for each function to duplicate the other's files. For example, cost will no longer have to maintain independent files of material lists, blueprints and planning records for every part and assembly. This elimination of file redundancy will be felt in many indirect labor activities.

Further, vertical integration of effort also has a major effect on reducing the information and physical processing time cycle. Since all of the decision-making logic needed to completely process an order is in the computer, it is reasonable to expect overnight data processing and, by having dynamic control of the whole physical process from purchasing through parts making and assembly, it is possible to reduce significantly the actual "make" cycle. This type of control should result in lower inventories, higher promises kept and better indirect labor efficiency.

A second principle is the need for discovering a logical structure or pattern which formally displays and relates the various decisions such as those in product design, facilities operation and factory scheduling. In manufacturing planning, for example, by focusing attention on each variation in method or elemental time standard, cause and effect relations can be spotlighted to aid in making improvements. By organizing the multitude of detail into a clear, easy-to-understand framework, it shows what design characteristics control the various manufacturing process elements making clear the simplification and standardization opportunities. The use of logical decision patterns in a business should reduce direct labor and direct material through the powerful analytical insights they make possible.

Another basic concept was to design the system with the computer in mind. Although computers and humans perform many of the same tasks, their relative efficiencies and economic advantages are quite different. To arbitrarily make the computer follow the same routines, the same steps, the same processes as humans is illogical. Rather, the basic system should be reconceived and redirected to obtain maximum performance from the electronic computing equipment.

It was also quite an insight to think about the system as being directed solely toward the ultimate user, ignoring all the intermediate functional outputs that have so commonly become identified with our data processing system. The only purpose of having any operating outputs from a system is to cause someone to take action, to cause a buyer to purchase materials, an operator to make parts, etc. The intermediate transformations and hence the intermediate outputs are not essential systems elements but are only a reflection of the particular data processing techniques currently in use.

BG:gms
3/30/60

April 15, 1960

TO: Messrs. John Backus, IBM
Carl Byham, Southern Railway
Les Calkins, U.S. Steel
Ken Foster, Sylvania
Dave Nelson, Lockheed
Sol Pollack, RAND
Pete Sheridan, IBM
John Smith, Aeronutronics

Mal Smith, Sperry-Rand
Jack Strong, North American Aviation
Dick Utman, Ramo-Wooldridge

In developing a systems engineering language it is desirable to introduce a wide variety of techniques for problem description while restricting the form in which the description can be stated so as to have simplicity of learning and ease of communication. It is with this underlying concept in mind that a proposed formulation of Data Element and Data Set has been prepared. I should also like to make it clear that the ideas contained resulted from various suggestions and recommendations at the CODASYL Systems Meeting of April 4 - 6, 1960, and that I am essentially serving only as a reporter and summarizer.

Because of time problems I have only enclosed the write-up on Data Elements (Section A). A proposal on Data Sets (Section B) will follow in about 1 - 2 weeks. I have also enclosed (Exhibit 1) a copy of a table prepared by Perry Crawford of IBM in October, 1957. I believe that this table shows more general concepts of tabular systems description than does the Orren Evans' work. I do not claim to know which formulation is more useful, but I do believe it would be worthwhile to establish a basic structure for tabular presentation so that each particular approach could be seen in terms of a more general picture. To this end I plan to send some tentative notes concerning tabular format which may suggest various techniques to be explored in the description of data transformation. This will be Section C.

Burton Grad
Programming Systems, IBM

cc: Orren Evans
Perry Crawford

April 25, 1960

TO: Messrs. John Backus, IBM
Carl Byham, Southern Railway
Les Calkins, U. S. Steel
Orren Evans, Hunt Foods
Ken Foster, Sylvania
Dave Nelson, Lockhead
Sol Pollack, RAND
Pete Sheridan, IBM

John Smith, Aeronutronics
Mal Smith, Sperry-Rand
Jack Strong, North American Aviation
Dick Utman, Ramo-Wooldridge

This attached material is a follow-up to my notes of April 15, 1960. Included are a few additional notes for Section A - Data Element. This is followed by an initial attempt to specify Data Set construction in Section B with a few examples and one possible form. After this comes a few preliminary comments on Data Transformation (Section C) which I will add to within the next two weeks. I am also planning to forward some considerations in regards to particular forms descriptive techniques. These may be required when the systems engineer has to specify a particular report or record format or a certain representation scheme or medium.

I shall be looking forward to further comments and improvements when you've had a chance to critically review these very rough notes.

Burton Grad
Programming Systems, IBM

11 July 1960

To: CODASYL Systems Group

In accordance with the assignments made at the San Francisco meeting, I should like to submit a proposal for a Data Transformation concept which would incorporate the Evans Table Technique. This is, of course, far from complete or even thoroughly thought-through, but it may provide a basis for further discussion and recommendations. This has been numbered so as to be attached to the previous material which I forwarded prior to the San Francisco meeting.

I have also attached a personal critique and recommendation on the Data Element - Data Set area. These notes are directed specifically to Sol Pollack, Dick Utman, and Kendall Wright who are working on this area. The comments noted have specifically resulted from certain criticisms and suggestions made by Gertrude McKay and Tom Glans, both from IBM, and recognizes the comments and thoughts expressed by Ken Foster and Sol Pollack.

I'm looking forward to our next meeting.

Burton Grad

Burton Grad
IBM, Logical
Systems Standards

BG/js

Attachment

Proposals for a Systems Engineering Language

Section A. Data Element

A Data Element entry defines the structure and or content of a related series of symbols. A Data Element name may refer to any series of symbols which has a unified meaning to the systems engineer in the context of the application system.

A Data Element may be described by any combination of the methods noted below:

(1) Pictorial: direct indication of the possible range of symbols on a position by position basis. This might include certain editing considerations.

(2) Literal: direct indication of the actual symbol for each position.

(3) Elemental: indirect indication through specifying positions of other Data Elements.

(4) Conditional: indirect indication through specifying the value of positions of other Data Elements.

A Data Element then is defined at the convenience of the systems engineer in the most useful way to him. There is an unlimited hierarchy of Data Elements and there may be any degree of overlap and nesting. However it should be remembered that a change in one Data Element may have repercussions on other Data Elements and therefore careful cross referencing (in the sense of where used information) is a necessity for accurate systems design.

It is suggested that a form like the one shown as figure 1 (next page) might prove quite useful. There are a series of examples shown on this form for which a brief explanation follows:

(1) This illustrates a straight pictorial representation (P under Type) of the data element called ACCOUNT-NUMBER. It is seven positions long (Initial Position - Final Position designation or by actual symbol count) and each position can have a symbol from the range of symbols 0 through 9 (which is what is meant by N).

(2) Example 2 shows a literal description of a data element named P1. It is seven positions long and the specific symbol for each position is shown.

DATA ELEMENT DEFINITION

	ELEMENT			DESCRIPTION					COMMENTS →
	NAME	INIT POS	FIN POS	TYPE	REFERENCE	INIT POS	FIN POS	VALUE	
(1)	ACCOUNT-NUMBER	1	7	P	N N N N N N N				
(2)	PI	1	7	L	3.14159				
(3)	FICA-LIMIT	1	6	L	4800V00				
(4)	ICC-NUMBER	1	7	P	N				
		8	10	L	-46				
(5)	CHECK-AMOUNT	1	7	P	$bbb.NN				
	CHECK-DOLLARS	1	3	E	CHECK-AMOUNT	2	4		
(6)	PAY-AMOUNT			E	CHECK-AMOUNT				
(7)	DAY	1	2	P	NN				
	MONTH	1	2	P	NN				
	YEAR	1	2	P	NN				
	DATE	1	2	E	DAY				
		3	4	E	MONTH				
		5		E	YEAR	2			
(8)	MARITAL-STATUS	1	1	P	N				
	MARRIED			C	MARITAL-STATUS	1		1	
	SINGLE			C				2	
	DIVORCED			C				3	
(9)	MILEPOST	1	5	P	NNNNVN				
	CITY	1	20	P	A				
	CHICAGO-CODE			C	CITY	1	3	CHI	

pg. 12

(figure 1)

(3) Another example of a literal is given by the definition of FICA - LIMIT which is only 6 positions long. It has the specific symbol values shown except that v is used to represent an implied point which does not actually occupy space in the data element.

(4) The fourth example defines the data element ICC-NUMBER. It actually has a split definition. The first seven positions are each occupied by a symbol within the range of N (0 - 9). This is a short hand scheme which avoids the necessity of repeating the N seven times. In addition, however, positions 8 - 10 consist of the literal symbols " - 46." This might indicate a particular code used by the ICC for this company. Without repeating ICC-NUMBER on the second line we have implied that the name is the same as the one in the line directly above.

(5) CHECK-AM⊙UNT is defined pictorially to show that there are 4 different symbol ranges possible in different positions. For example, we might mean "$" to indicate just that symbol ($) and no other; "b" might mean blank and 0 - 9; "." might indicate only a point; and "N" could mean 0 - 9. CHECK-D⊙LLARS is described in terms of CHECK-AM⊙UNT; this is an elemental description (E under Type). It is defined to be a three position data element which is the same as the second through fourth position of the data element CHECK-AM⊙UNT.

(6) This shows a short hand way of indicating a synonym. PAY-AM⊙UNT is identified as being the same as CHECK-AM⊙UNT. Since the data element positions are not listed it is assumed that they are equal and the descriptive data element (in this case CHECK-AM⊙UNT) will govern. The systems engineer may, for precision or as a cross-check, specify either or both of the lengths.

(7) This example first indicates a straightforward Pictorial designation of three data elements. Then another data element DATE is shown to be composed of these first three elements. The inclusion of DAY and M⊙NTH is quite in following with the previous examples (position has been omitted since the entire data element is being used), but the use of YEAR is a little different. If the final position is the same as the initial position, we are defining a one position data element.

(8) Example 8 shows the use of a conditional definition of a data element. MARITAL-STATUS is a one position data element which can take on a value from 0 - 9. MARRIED is a data element which has a value of either true or false depending on the value of MARITAL-STATUS. If MARITAL-STATUS has the value 1 then MARRIED is true, otherwise it is false. Similarly for SINGLE and DIV⊙RCED. Repetition is implied for MARITAL-STATUS and for Initial and Final position. We will have to establish a maximum number of literal positions for value. I would suggest that it be relatively small (around two or three positions).

(9) There are two more data elements defined that we may want to use in later examples of data sets. MILEP⊙ST is a five position data element with an implied point between the fourth and fifth positions. CITY is a 20 position data element consisting of symbols chosen from the range: blank and A through Z. CHICAG⊙-C⊙DE is a data element which is true when the first three positions of CITY are "CHI".

There are certain observations we can now make about data elements.

(a) No data element that is not defined properly may be used in a Data Transformation. This, of course, does not preclude defining various general data elements like TEMPORARY, CONSTANT or INDEX.

(b) Every data element must eventually (through the appropriate hierarchy) be defined in terms of a pictorial or literal for each position.

(c) A data element may have a name or a value of any length up to the maximum number of positions definable. Continuation may be indicated through indenting by one space in the name and/or the reference column.

(d) Since lengthy literals can be introduced it would be possible to consider the contents of an element as a list or table of values. It is also possible to introduce a list of values through giving each value a definite name.

(e) By itself, a data element cannot be subscripted or indexed. It is only through membership in a data set that multiple data elements can be directly considered.

(f) The individual positions of a data element can be referenced in a Data Transformation through mentioning the data element name and the initial and final position. While the exact form of this reference has not been determined it might be something like this:

ACCOUNT-NUMBER (3-5)

(g) A data element is defined independently of data set membership. Hence DATE might simply designate a 5 digit data element regardless of where it is used or how many different sets it is used in. If we wanted to deal with a 6 digit data element (including two positions of YEAR) we would have to give it a different name like DATEF.

(h) This proposal does not directly handle the difference between a floating-point and fixed point number. One operational solution would be to have special floating-point operators to work with floating-point data elements and leave responsibility for proper operator selection to the systems engineer. Another solution would be define a floating-point number as really two separate data elements (the mantissa and the exponent). A third solution would be to introduce a symbol (f) to mean an implied floating-point separator; on that basis NNNNfNN would be a 6 position data element with an implied floating-point separator between the fourth and fifth position. The mantissa would be four positions long and the exponent 2 positions long. This problem obviously requires further study.

(i) A question was raised concerning handling signs. If there is an implied sign associated with a data element it can be introduced by permitting the use of a symbol (like S) in the first pictorial position to represent that this data element may have either a plus or minus value. For explicit signs a symbol code will be needed to represent the various possibilities and it should be placed in the proper actual position.

(j) Various kinds of special editing can be indicated through symbol range representation. For instance, if * was used to mean (blank, *, 0 - 9) and appropriate rules were defined then $***. NN could indicate zero suppression with asterisk replacement. If D meant ($, blank, 0 - 9) with proper rules DDDD.NN could be used to indicate a floating dollar sign. There would seem to be a basic question as to whether editing should be handled through data element definition or through special editing operators.

(k) The handling of value ranges and value lists was also mentioned. I believe that this is most pertinent in conjunction with conditionally defined data elements. For instance, MARRIED might mean that MARITAL - STATUS equaled 1 or 5 or 7. We could show this simply by repetitive definition of MARRIED for each assigned value of MARITAL - STATUS. A more difficult question is suppose DIVORCED was to be used if MARITAL - STATUS was greater than or equal to 3 but less than 8. If we are to avoid stating each possible value in the range then we will have to introduce a notation along with value to show greater than, equal or less than. I am not at all sure this is desirable since from the systems engineer's standpoint I don't believe that we are generally concerned about the range of values once we have assigned an adequate number of positions. I feel that any restriction on value range or value set should be introduced through the Data Transformation.

B. Grad
4/14/60

Cont'd Section A

(l) A literally specified element has its precise value defined in the data element description. Because of this, a rule should probably be established that such an element cannot have its value modified by the Data Transformation. This will have to be explored further since it may be desirable to be able to set initial values for a variable in this way. If we adopt the suggested rule then we are limiting a literal element name to the representation of a constant only. Another question on literals is in regard to how we will indicate that the value represented is really to a base different from 10. For instance, if a data element named SWITCH-PATTERN was defined as 101100, we might not be able to operate with it properly unless the systems engineer identified it as being to the base 2. I believe that we should avoid indicating base 10 and only indicate base for numbers which are to other than 10. In other words, if a systems engineer is sophisticated enough to use non-10 base literals, we should permit him to do so.

(m) A pictorial description defines an externally useful form for a variable; it does not specify the manner in which the value will be represented internally. The actual value will be assigned through the Data Transformation or by association through a Data Set with a particular input form. One problem which has been raised is whether it is necessary to explicitly state the class of value for a variable or whether it can be implied by the symbol classes chosen for each position and its usage in the Data Transformation. I would be interested in any thoughts you may have as to the need for differentiating in the Data Element description among integers, fixed-point numbers, floating-point numbers, symbol strings, etc. This analysis might also consider the question raised by Ken Foster concerning the usage of the element - whether as classifying, quantitative, pictorial, etc.

Another problem to be considered is that of data elements which may have variable length and symbol makeup. Suppose we have three different types of part identification numbers: NNNANNN, NNNNNNN-NNN, or NNNNNAA-ANN. It may be desirable to have only one element called IDENT-NUMBER. To show its construction we have to indicate a maximum or worst case like NNNBNBB-BNN where B indicates (0-9, A-Z). However, in any particular case the value of the element may not require this extended definition. Therefore, we might see if there is any simple way to show and use multiple formats for the same data element. In line with this we might consider the item description on a purchase order; except for specific forms restrictions there is no inherent limit on the length of this data element. How then would we go about describing its construction? If we carefully think through the various kinds of data elements and how they might be used, we may find that we want to allow a somewhat simpler data element description with some way of indicating to the programmer that this is really a variable length element and should be stored and processed accordingly.

(n) An elemental definition is a means of conveying synonyms not construction. Any name defined through other elements refers to the same values as the descriptive element. There is no shorthand for defining same construction except through subscripting a variable by Data Set membership. It's necessary that any element name used in a Data Set be defined in a Data Element description. This cannot be done through the Data Set itself. This point is quite important in that certain Data Transformation operators will only work on data elements and cannot be used on Data Sets. Actually an elemental definition is simply the name of a particular extractive and/or combinatorial function.

(o) A conditionally defined data element is the name of a simple Boolean (or logical) function. We should be careful not to depend on too much scope in these elements since this type of information should probably be picked up in the Data Transformation. What we might permit is a simple definition of an element which allows it to take on only the values True and False. Then through the Data Transformation we can assign to this variable an appropriate code based on the value of some conditional expression.

Burton Grad/bm
4/27/60

SHIPPING SCHEDULE DETERMINATION

CONDITIONS:

Stockage	Delivery	Availability	Further Conditions	Action	Shipping Schedule Date
S	DI or DN	QA		Ship at once	Today
		QN		Back order	DRO
	DD	Not applicable	OH-QRP $\geq$ QO	Defer order without reserving	DD
		Not applicable	OH-QRP $<$ QO and DD $\geq$ DRO	Defer order without reserving	DD
		Not applicable	OH-QRP $<$ QO and DD $<$ DRO	Defer order without reserving	DRO
NS	DI or DN	QA	QA $\geq$ 1/4 QO	Ship at once	Today
			QA $<$ 1/4 QO	Defer order and reserve	Today + SLT
		QN		Suspend order and order replenishment	Today + SLT
	DD	QA	QA $\geq$ QO	Defer order and reserve	DD
			DD $\geq$ Today + SLT and QA $<$ QO	Defer order and reserve	DD
			DD $<$ Today + SLT and QA $<$ QO	Defer order and reserve	Today + SLT
		QN	DD $\geq$ Today + SLT	Defer order	DD
			DD $<$ Today + SLT	Defer order	Today + SLT

Exhibit 1

Sample Tabular Presentation

by Perry Crawford, IBM

October, 1957

Section B - Data Set

A Data Set is defined by a series of entries each of which consists of the name of a Data Element or a Data Set and the number of times this Element or Set is repeated within the Set. The Data Set will itself be given a name which can be used to make up other entries. There is an ordering of the entries through an entry reference number though the particular ordering is arbitrary.

Every Data Set must eventually (through the steps in the hierarchy) be defined in terms of Data Elements. There are two degrees of freedom in every Data Set in that it may contain a number of different Elements as well as multiple repetitions of any Data Element. Each of these repetitions represents a potentially different value.

To adequately convey the use of a Data Set Definition a few examples are shown in figure 2 (next page). They are explained below:

(1) The Data Set INVENTORY-RECORD is defined by five different entries. The first entry (1 in Reference Position) is the Data Element IDENTIFICATION. There is but one IDENTIFICATION element per INVENTORY-RECORD. The same reasoning holds for the next three entries (2-4). Entry 5 however refers to a Data Element called MONTHLY-USAGE which will appear 12 times. It is through repetition that subscripting can be introduced. If we refer to INVENTORY-RECORD MONTHLY-USAGE we need to indicate a subscript or index to denote which one. MONTHLY-USAGE (5) means the fifth appearance of this Data Element.

We can think of this representation as being a shorthand for designating a list of names. We could have defined MONTHLY-USAGE (1) etc. by a Data Element Definition and then indicated each of these as separate entries. However, it is obviously convenient to use an abbreviated method.

(2) INVENTORY-FILE is a Data Set consisting of 500 repetitions of the Data Set INVENTORY-RECORD. We can then refer to INVENTORY-FILE INVENTORY-RECORD (20) to indicate the 20th such set. This implies that all element of the set INVENTORY-RECORD are also members of the set INVENTORY-FILE.

INVENTORY-FILE INVENTORY-RECORD MONTHLY-USAGE

DATA SET DEFINITION

	SET NAME	ENTRY NUMBER	TYPE	DESCRIPTION REFERENCE	REPETITION	COMMENTS
(1)	INVENTORY-RECORD	1	E	IDENTIFICATION	1	
		2	E	PART-NAME	1	
		3	E	ON-HAND-QUANTITY	1	
		4	E	REORDER-POINT	1	
		5	E	MONTHLY-USAGE	12	
(2)	INVENTORY-FILE	1	S	INVENTORY-RECORD	500	
(3)	SALES-REPORT	1	E	DISTRICT-NUMBER	V225	
		2	E	UNIT-SALES	V225	
		3	E	YTD-SALES	V225	
		4	E	BUDGET-SALES	V225	
		5	E	REALIZATION-PERCENT	V225	
		6	S	TOTAL-SALES	1	
	TOTAL-SALES	1	E	TOTAL-UNIT-SALES	1	
		2	E	TOTAL-YTD-SALES	1	
		3	E	TOTAL-R-PERCENT	1	
(4)	SALES-REPORT	7	E	SALES-REPORT-HEADING	1	
(5)	OSR	1	I	SALES-REPORT	1	

Figure (2)

(20, 5) would mean the fifth month of the twentieth inventory item. You will notice that this does not provide any explicit means for indicating precise order or format between elements. Order will have to be determined through the Data Transformation and format details through a special set of forms descriptive techniques. One possible forms descriptive technique is discussed in Section D.

(3) SALES-REPORT is a Data Set described by six entries. The first five entries refer to Data Elements each of which may recur up to 225 times but this is not a fixed value (the V stands for variable). Omission of the V implies that the repetition is fixed for some reasonable period. The exact use of the V will be at the discretion of the systems engineer and permits him to convey information to the "programmer" which may help in the preparation of a more efficient program. The sixth entry refers to another Data Set TOTAL-SALES which then happens to be designated next.

(4) By proper sequencing of Data Set entries it is possible to achieve any relative reference layout for set hierarchy. For example SALES-REPORT entry 7 could be incorporated after the description of TOTAL-SALES. This could have been continued to any number of levels as long as care was exercised in using the proper set name with the entry. This also indicates that if an omission occurs it can be readily corrected simply by using the next entry number for the Data Set and then filling out the entry. This entry also shows reference to a Data Element which could be literally defined as a Data Element.

(5) This example shows how a synonym can be established for any Data Set name. OSR may now be used interchangeably with SALES-REPORT as indicated by I (for Identity) under Type. This can be a major convenience in abbreviating Data Transformation references. The only requirement is that a synonym have one and only one meaning just as any Data Set name.

With these examples as a foundation we can make some general observations about Data Sets:

(a) The specific values to be assigned to Data Elements will be determined through input form association or through the Data Transformation. This implies that every Data Element must be referred to in terms of set membership except where it is an absolutely unique element which will not ever have a different value as a function of Data Set membership. This indicates clearly the need for careful exploration of abbreviation schemes for set indication. It will be necessary to determine whether every step in the set hierarchy will have to be indicated or whether it will only be required to designate up to a unique set. Intuitively it seems that it would be adequate to specify only the top level set and the element name except where the same name appears more than once within the Data Set. However, I haven't really explored this at all.

(b) At the present stage of development, a fixed format is needed for each Data Element as a member of a Data Set. There is no basic constraint from a set standpoint of permitting a Data Set to contain a variable length Data Element. Element to element order is not really defined by the Data Set description. It must be supplied by the Forms Description or the Data Transformation. Element to element order can be introduced through various Data Set operators used in the Data Transformation. Rearrangement of a Data Element requires either a special Data Element definition or the use of string manipulative operators in the Data Transformation.

(c) A Data Element may belong to many Data Sets. We may want to use the same value for a particular Data Element in many Data Sets like CURRENT-DATE. In other cases we do not want to use the same value. It is probably desirable to identify set membership not simply as E but rather to denote a special type as F to indicate that this refers to the fixed value of the element named. Operationally this would mean that if we assigned a new value to CURRENT-DATE, every set to which it belonged by an F type relation would automatically be assigned that same value while those to which it belonged by an E type relation could not have the value changed except through a name qualified reference; e.g. TRANSACTION CURRENT-DATE. Your thoughts on this problem would be most helpful. A Data Element can be modified or used independently of any sets of which it is a member; however, its value as a set member will not be changed by such a change unless it has used an F type relationship.

(d) A Data Set cannot be defined as incorporating part of another set except through directly using the subsets or elements wanted. The reference position (entry number) on the Data Set definition form does not imply an actual ordering of the elements or subsets; it is meaningful only in terms of orderliness, and addition and deletion of entries.

(e) If we wish to assign the same value to two elements which have the same name but belong to different sets (by an E relation), then the Data Transformation will have to provide for the assignment, e. g. TRANSACTION QUANTITY = ERROR QUANTITY.

(f) We can now generalize on the concept of subscripting. Any element may be subscripted as a function of its membership in a particular set. SALES DISTRICT (25) and CREDIT DISTRICT (25) refer to different Data Elements. DISTRICT (25) has no defined meaning unless that Data Element appears in a Data Element definition. The mention of a particular Data Set name in conjunction with a particular Data Element implies all intermediate sets. INVENTORY-FILE MONTHLY-USAGE (20, 5) states that this reference is to the 5th repetition of the element USAGE in the implied subset INVENTORY-RECORD for the 20th repetition of that subset in the set INVENTORY-FILE. There are no fixed number of levels in the Data Set hierarchy nor does each level require or permit subscripts. The subscript notation is such as to line up with an adjective—type hierarchy: highest-level, next-highest-level . . . . , element. The first subscript refers to the particular next-highest-level member in the highest-level set. The last subscript refers to the particular element in the lowest-level set.

(g) Tables can be introduced through Data Set membership, however, it would probably be done somewhat differently from the usual row-column method. Suppose we have a Data Element called RATE. It is the only element of a Data Set called JOB-CLASS. As such it is repeated 10 times, once for each experience class. JOB-CLASS is a member of JOB-RATE-TABLE and is itself repeated 15 times, once for each different value of JOB-CLASS. The JOB-RATE-TABLE is a two-dimensional matrix 15 x 10. JOB-RATE-TABLE RATE (12, 8) would refer to the 8th RATE within the 12th JOB-CLASS. If we also wish to define EXPERIENCE-CLASS we may do so by indicating that RATE is a member of this set and is repeated 15 times, once for each job-class.

(h) Specialized lists of value names can be identified through Data Set and Data Element definition. For instance a Data Set called TEMPORARY might consist of 100 repetitions of the element TEMP which was a 20 position numeric element. We could therefore use TEMP (1), TEMP (10), etc. directly without further element definition. This also indicates the ability to use the element name by itself without denoting set membership if we are certain that the element will not be a member of any other set. This problem needs to be explored further since such specialized gimmicks could make a systems program difficult to modify.

(i) It will be necessary to develop a means for assigning set-element names to input values and correspondingly to assign the values of set-element names to designated output forms positions. This whole area of specific association of names and input and output values and forms is covered in some detail in Section D. I believe that it may well be the critical bridge

between the practicalities of real business operation and the concepts of logical systems design. In this regard it would be meaningless to associate an input value with a Data Set Element which had been defined as a literal. In effect, this would be permitting the input data to change a constant. However, the assignment of a literal element to an output Data Set may often be desirable particularly in terms of headings or various fixed information. With a literal element of course it is never necessary to indicate Data Set membership in referring to that element in the Data Transformation.

(j) There is an interesting analogy to be drawn between Data Element and Data Set definition and the use of blueprint and bills of material for product descriptions. I would like to discuss this separately, at a later time, since I feel it might be advantageous to clearly understand the similarities and differences between product description and information system description.

BGrad/bm
4/29/60

Additonal Material for Sections A and B

After having tried to use the Data Element and Data Set forms on a few sample problems it is my belief that we need to make certain changes and that we should recognize certain cases which were unfortunately overlooked on the first pass. Suggestions for correcting these deficiencies are described below:

(1) The purpose of both forms is to name certain classes of "informational objects" and to associate form and/or value with these names. The major reason for differentiating sets from elements was to indicate that different functional operators might be used on sets (like Merge, Sort) than would be used on elements (+,-,<).

Since the dividing line definitionally is so vague I should like to recommend that an effort be made to combine the two forms and that the type of definition be expanded to indicate whether the item being defined is an element or a set. Among the models which might be examined for good ideas are FACT and the new Commercial Translator Data Description (a copy is enclosed).

(2) Another major problem which will be alleviated by this combination of forms is the desire in many situations to define a Data Element only in terms of a particular Data Set. This comes into play when we have a number of specialized elements where it's a nuisance to have separate sheets to fill out or where it's convenient to apply the same name to an element which has a variety of forms depending on the particular set. For instance, Employee-name might be pictured as 2 initials and a 15 character last name in one place, last name only in a second place, and full name in a third. Whereas before we would have to give different names to each of these occurrences, now we can use the same name and just provide a means of indicating (as was discussed for Data Elements at the last meeting) whether this is a form definition or a value definition. In other words a value definition says that this name represents a specific value (and the same value) for all Data Sets to which it belongs. The name can therefore be used at any time without set qualification. This does not mean that the value must be stated in the Data Description, but only that it is defined as a unique name. Examples might be employee-number, shop-order, account-number.

In contrast, a form definition states that this name represents a unique pictorial though the values will vary with different sets. Some examples are date, signature, form-number. For both of these cases we should have a code means of indicating that this element name is supposed to represent a unique form or unique value and we should also have a short cut approach to avoid needless repetition.

(3) It has also been suggested that we give a pictorial definition to each fixed-value (constant) element to avoid ambiguity and confusion.

(4) Each element or set name must be identified as representing a particular value class: Quantity, True-False, String.

B.G./lsd
7/19/60

Section C - Data Transformation

From the system engineering viewpoint, we wish to describe the logic required to transform the input Data Elements into the output Data Elements and to present these values in appropriate reports, signals, or records. We are not concerned with the specific procedure adopted to carry out this transformation logic.

Regardless of the form of expressing the Data Transformation logic, it is necessary to conceive of the fundamental purpose that this part of the systems description performs. Basically we have certain elements in which we are interested whose value can be obtained from the outside world. For example an employee's name can be obtained as an input. If all elements were of this nature then all that would be necessary would be to have a means of mapping inputs onto outputs. However, most information that we wish to examine on output requires definition in terms of special operators acting on various input and intermediate values.

This leads to the conclusion that we have three ways of defining the value of an element in which we are interested:

(1) As a literal value in the Data Element definition.

(2) As an input value through the Data Set definition and Forms Description assignment.

(3) As a "computed" value through the Data Transformation logic.

Therefore we do not need to think about the operation of adding or subtracting or moving information but rather only of being able to use certain functions or operators which may be useful to us in defining element values.

It's my feeling that there are a relatively limited number of generally useful element and set operators but a much larger number of specialized macro-operators which will be useful to a particular class of business or application. For instance, we'll surely need the general arithmetic operators of Sum, Difference, Product, Quotient as well as logical operators like Intersection, Smaller, Larger, Identity. In addition, generalized set operators like Order, Extract, Match, Merge will be needed. To illustrate specialized operators, certain applications will need mathematical functions like Sine, Power, Minimum; Specialized logical operators like Inclusion, Dual Implication may also be required. There will also be many specialized set operators like Deduction, Post, Explode.

Each operator will need to be precisely defined in terms of positional line-up, truncation, rounding, etc. It will also be necessary to indicate the classes of elements or sets to which the operator is pertinent. Similarly we will probably have to provide a straight forward way to allow a systems engineer to define any specialized macro-operators which he finds useful.

There are certain other properties which we may be looking for in the language regardless of the particular form that is chosen such as the ability to use literals directly, the opportunity to use readable terms for element names, the capability of handling a variable level of user sophistication, etc; nevertheless with these general comments out of the way we find ourselves right back against the format problem.

It is apparently the feeling of the CODASYL Systems Committee that some sort of tabular presentation of the systems logic would be highly advantageous. Therefore it would seem to be reasonable to examine the structure of the Orren Evans tabular form and the Perry Crawford example to see if there are certain underlying principles which can help us obtain a better understanding of the concepts underlying tabular format.

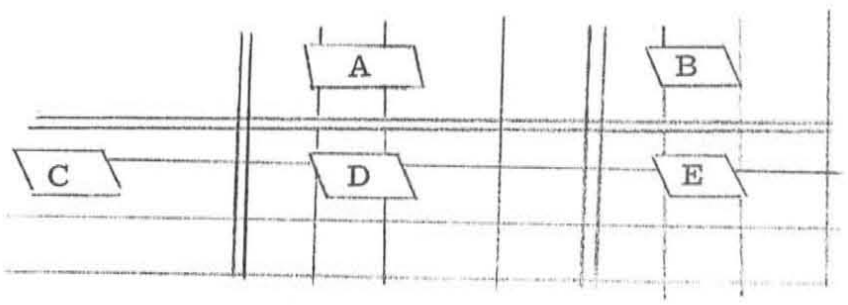
A portion of one of Orren Evans' sample tables is shown as Figure 3 on the next page.

If we represent a question (is, does) by "q" and an action by "a" then we can structure the first rule in this table as follows:

Rule No.

001 [If] q_1 [=] Y [and] q_2 [=] Y [Then] Y [for] a_1 [and] Y [for] a_2

The words and symbols in square brackets, [ ] , are implied by the tabular form. In addition there are, of course, various rules concerning the placement of symbols and expressions within the table. For instance, we can think of the table as having five segments as shown below.



Rule No.	Is Credit - Limit > In - Process - Balance (+) Accounts - Receivable (+) Order - Amount?	Is Customer - Past - Due - Balance = Zero?	Does the Customer have Prompt - Payment Experience?	Approve Credit on Order	Send Order to Stock-Control	Send Order to Sales
001	Y	Y		Y	Y	
002	Y	N	Y	Y	Y	
003	Y	N	N			Y
004	N		Y	Y	Y	
005	N		N			Y

Figure 3

In seqment A only "q" type expressions (conditions) can be shown. In segment B only "a" type statements (action) can be made. Segment C includes rule number, frequency and prior rule requirements. Segment D allows the entry of a Y, N or blank. Y serves to indicate that if the conditional expression in that column is true then this part of the rule is satisfied. An N shows that if the conditional expression is false then this part of the rule is satisfied. A blank denotes that the truth value of this conditional expression is not pertinent to the data rule. In segment E either a Y or a blank may be entered. Y indicates that the action in that column should be executed; a blank indicates that the action is not executed for that data rule.

A properly formed test requires consideration of both the conditional expression and the appropriate table entry (Y, N or blank). Similarly, to determine the inclusion of an action we need to see if there is a "Y" in the proper box. This means that a complete data rule requires knowledge of (and relationship between) the column heading and the intersection contents.

With but minor changes, the Evans questions would be perfectly acceptable COBOL conditional expressions; however the actions as stated do not as readily fall within the COBOL framework.

In the Evans work the order of testing columns and the order of testing rules is not significant in determining the result that will be achieved. However in executing actions there may well be an implicit or explicit order involved. This is not to say that efficiency will not be affected by the order of testing, but rather that the logic will not be affected.

In the Perry Crawford table (Exhibit sent previously) there are some interesting differences. Instead of simply noting a Y, N, or blank in the intersection he has inserted certain of the information which would normally appear in the column heading in an Evans table. He has also used an "or" between possible values within a single intersection. He has, in the fourth column, made another major change; the whole question appears in the intersection and the column heading has no operational significance. Similarly in column five each action is defined in the intersection and the heading is meaningless. In column six the action is described in two segments: the heading and the intersection, with, in four cases, a mathematical expression (Today + SLT) appearing in the intersection.

Between the two examples we find a variety of cases which permit anything from a meaningless column heading to a fully stated question (but not the answer) and a fully stated action (but not whether it is to be carried out). In an intersection we find everything from a simple y, n, or blank up to a complete question (with implied inclusion) or a complete action

(with implied execution). It is clear that full clarity requires the dual consideration of the appropriate column heading and the proper row intersection. Generally speaking we could describe a tabular technique that permitted any type of expression in either a column heading or a row intersection but the usefulness of this degree of freedom may well be questionable. Correspondingly we can define various simple structures which are well suited to various classes of problems.

Among areas which require further exploration in regard to tabular format are things such as:

(1) table to table transfers

(2) row independence

(3) heading format

(4) intersection format

(5) operator and relationship indicators

(6) Two-state (Y and blank) versus three-state (Y, N and blank) logic for "truth tables".

I will try some real examples so as to gain a better understanding of the scope of the Evans table format and an indication of areas which can be improved. Any illustrations which have been worked out might prove helpful to our experimentation.

BGrad/bm
4/29/60

Cont'd Section C - Data Transformation

I would like to propose a generalized concept for tabular representation. I believe that this form of description incorporates the Evans and Crawford work as special cases.

(1) A Table is defined as a two dimensional form structured as follows:

CS_{11}		CS_{1n}
	CS_{ij}	
CS_{m1}		CS_{mn}
AS_{11}		AS_{1n}
	AS_{kj}	
AS_{r1}		AS_{rn}

Above the double line there appears various Conditions (CS); below the double line there are certain Actions (AS). A CS may only take on a True or False value though it may be composed of any number of expressions. Formation of a CS will be described later. An AS represents an action and as such it has no inherent value; it may be used for procedural control (like go to or do) or for value assignment. Formation of an AS will also be discussed later in this Section.

(2) A Table should be interpreted as noted below:

if CS_{11} is true and CS_{21} is true and ....CS_{m1} is true

then execute AS_{11} and AS_{21} and....AS_{r1}

if CS_{12} is true and...CS_{m2} is true

then execute AS_{12} and ...AS_{r2} ;

if CS_{1n} is true and...CS_{mn} is true

then execute AS_{1n} and ...AS_{rn}

Mathematically this can be stated:

if $\{CS_{ij}$ is true$\}$ for i = 1 to m

then execute $\{AS_{kj}\}$ k = 1 to r

It should be noted that this formulation is not directly equivalent to the McCarthy LISP work where the first "true" set of conditions determines the actions and precludes the execution of the other actions. As of this point there is no solution implication other than that success within a column depends on "true" answers for all Conditions in that column.

(3) It is desirable to define various items which will be used in describing Conditions and Actions. Many of these definitions have been "borrowed" from the ALGOL - 58 and ALGOL - 60 write-ups.

The notation is quite straightforward:

:: = means "is defined as".

| (vertical line) means "or".

(a) There are three value classes which describe the values that a Data Element can assume:

(1) Quantity

(2) True-False

(3) String

(a.1) Quantity :: = NNN—NN.NNN—N

N :: = 0|1|2|3|4|5|6|7|8|9

Examples: 357; 1010; 35.17; .826

Quantity is the classification used for all values which represent measurable phenomena or those which have numeric (e.g. quantitative) significance. For instance an identification number, though composed entirely of numbers, would <u>not</u> normally be a quantity. Codes, in general, though often numeric in nature, are not quantities in the strictest sense in that they are not usually interpretable in arithmetic terms or in conjunction with the usual arithmetic functional operators (+, -, *, /).

A quantity will consist of the figures (0-9) together with point location (explicit or implicit), base notation and/or floating point designation.

(a.2) True-False :: = B

B :: = T|F

Examples: T; F

We are using T and F to represent "true" and "false" in a two-valued logic system. Obviously any other symbols would be as satisfactory (like 1,0, or Y,N), but for consistency the T,F notation will be maintained.

(a.3) String :: = SSS ———S

S :: 0|1|———|9|A|B|———|Z|+|-|*|/|

Examples: AR12; ABC + (259)

As of now the set of symbols permitted in a string should be open-ended. Obviously, for any given data processing machine there will be a limited number of different character codes which may be used in a single character position, however the symbol concept does not necessarily limit itself to single character representations. A symbol could consist of multiple characters. This emphasizes the inherent extendability and generality of strings and symbol manipulation.

(b) There are three ways to represent a particular value:

Literal

Name

Expression

(b.1) A literal is a quantity value, true-false value or string value which represents itself. We may speak of quantity value literals, true-false value literals and string value literals. Literals will be shown inside square brackets so that they will not be confused with values represented by names or expressions. Square brackets should be read as "the literal value...".

Examples: [357] ; [1010] ; [35.17] ; [.826]

[T] ; [F]

[AR12] ; [ABC + (259)]

(b.2) A name is itself a string which may be used to represent a value. When a string is used as a name it will never be shown inside square brackets; e.g., AR12 means the value represented by the name AR12. The value class represented by a name cannot be deduced from the string value of the name itself. It can only be determined from an appropriate data element definition.

Examples: JONES ; 1357 ; TTTFF

We may speak of quantity value names, true-false value names and string value names depending not upon the nature of the symbol string used to construct the name but rather on the value class represented by that name.

(b.3) An expression is a properly formed string consisting of literals, name represented values, expression represented values and functional operators. The precise construction depends on the formation rules associated with the particular functional operators used. The functional operators will determine the possible number of operands and their associated value classes; they will also establish the value class represented by the expression. The three kinds of expression are identified by the value class of the defined result: quantity value expression, true-false value expression and string value expression. An expression cannot use a name or another expression except as they may be consisdered to be literal strings; otherwise it must explicitly deal with their values. For consistency we will designate the value of an expression by not enclosing the expression in square brackets; a + b + c means the quantity value of the quantity value expression a+ b + c. Since an expression may itself include the value of an expression we have a recursive definition.

Examples: a<b ; avb;

$$[32] + \text{PAYAMT};\ (a+b)/c > [27];$$

$$a > [5] \equiv [T]$$

Each functional operator must be well defined over its class of permissible values so that it may produce a consistent result value given consistent input values. For the time - being all functions (a functional operator with its associated operands) will result in a single result value though it may have multiple arguments (operands).

(c) A numeric string value literal and an integer quantity value literal cannot be differentiated by examination, therefore we will adopt the convention that the functional operator with which they are used will determine their value class. Hence the expression a + [13] would be interpreted as a quantity value expression and 13 as a quantity value literal because of the use of the arithmentic functional operator +. Similarly the expression Concatenate (a, [13]) would be considered a string value expression and +13 a string value literal because of the use of the string functional operator, concatenate.

A similar possible confusion exists between a string value literal T or F and a true-false value literal T or F. Again, however, functional operator use should adequately distinguish these; e.g.

FIRST SYMBOL (EMP-NAME) $\equiv$ [T]

is obviously referring to the string value literal, T, while

a<b - [T]

is using the true-false value literal, T.

Therefore it is not necessary to explicity denote the value class of a literal since it can be determined through expression content.

(4) A Condition (CS) is the true-false value of a name, the true-false value of an expression or a true-false literal.

CS :: = [True-False LITERAL] | NAME | EXPRESSION

Any condition may be restated in any of the ways shown below:

CS :: = CS ≡ [T]

CS :: = [T] ≡ CS

CS :: = CA, R, CB where R represents any functional operator that defines a true-false value, CA and CB represent any legitimate operands that may be associated with R.

In header-trailer Tabular form this might be used in four ways:

header	trailer
CS ≡	[T]
[T] ≡	CS
CA, R	CB
CA	R, CB

The precise manner in which the transformation logic should be shown depends only on the system engineer's convenience.

The following sample table (Conditional portion only) shows many of the possible forms:

Married ≡	[T]	[F]	[T]
[T] ≡	SENIOR	JUNIOR	SOPHOMORE OR FRESHMAN
OLD AGE >	[4800.00]	[3600.00]	HEALTH AGE
TAX DEDUCT	< [7]	≥ [5]	= DEPENDENTS * 600
A + [35]	< B + C	< MIN (A,B)	≠ SQRT (B)

(a) Some of the functional operators which use quantity value operands and develop a quantity value result are +, -, *, /, **, SIN, MIN, MAX, ABS. (b) Some of the true-false value functional operators are: ∧ (and), V (inclusive or), ¬ (not). (c) Some of the string value functional operators may be: concatenate, extract, code, etc.

(d) There are relational functional operators which deal with quantity value operands: $<, \leq, =, \geq, >, \neq$. ⊙ther relational functional operators deal with true-false value operands: $\equiv, \not\equiv, \supset$(implies), etc. There may also be a well developed group of relational operators for string value operands, but this will require the development of a symbol hierarchy. $\equiv, \not\equiv$ do not depend on this hierarchy but $<, >$ do depend on it. All relational functional operators result in a true-false value result.

(e) Because of the header-trailer type construction it is necessary to define a third logical state (beside True and False). For instance suppose a particular CS is used for the header and in the third trailer column it doesn't matter whether this CS is True or False. We could show this by writing [T] ⊙R [F] , but this seems somewhat lengthy. Another alternative would be to leave the box blank, but this might lead to errors of omission. It therefore seems desirable to predefine a special function name to represent a generalized operand which will always result in the value [T] regardless of the functional operator or other operands. For symmetry we could also define a function name to represent a generalized operand which will always result in the value [F] regardless of the functional operator or other operands.

This concept of a universal truth function and a universal false function should be explored much further to determine if a universal value set and a null value set can be be consistently defined so as to permit the proper use of all of the various functional operators (R) which might be used between a CA and CB.

(5) An Action (AS) does not have a value since it simply designates carrying out the instruction of a procedural operation. There appear to be four major types of procedural operators:

Assignment

Procedure Control

Communication

Definition

(a) The Assignment operator associates a particular value with a name. Using assign.... as to mean "is assigned the value of" we can show:

assign N as [literal] | name | expression

The assignment establishes a value as of that point in the procedure. Further procedural operators may modify or change this value assignment.

The following examples show a few possible forms of Assign actions:

assign NUMBER-⊙F-DEPENDENTS as [7]

assign SALARY as GR⊙SS-PAY

assign DEDUCTI⊙NS as [600] * EXEMPTI⊙NS

assign SENIOR as YEARS > [3]

assign ANSWER as SQUARE-ROOT ([2] * U/I)

The value class of the name must match the value class of the literal, name or expression generating the value. Assignment has the implication of execution.

(b) A Procedure Control operator involves the next instruction (or Table) to be carried out. The basic operator is GO TO which designates next following (succeeding). There is some question concerning the use of DO where it could be stated that we are merely using a complex assign operator. However, for now, we can certainly think of DO in the sense of GO TO and COME BACK. Other Procedure Control operators are START and STOP. The basic form is:

go to PROCEDURE-NAME.

(c) A Communication operator involves fixing information in a particular form or at a particular value for later use. Examples include READ, WRITE, PUNCH, etc.

read TRANSACTION - RECORD

write ERROR - MESSAGE

punch INSTRUCTION

(d) The Define operator is closely akin to Assign. The form is: define NAME as VALUE. Value may be represented by a literal, a name or an expression, but it is inherently non-executable. This is a substitutive operator; the name may be replaced by the value term wherever its value is being used. Examples are:

define EXEMPTION as [600]

define CODE as EMPLOYEE-NUMBER

define AGE-GROUP as (AGE- [20]) / [5]

Define is one major aspect of the open-endedness of the systems language that we are developing. In spite of the many compiling difficulties that may be encountered in using a Define operator it permits the introduction of short-cuts and changes at the Data Transformation level without having to modify the Data Description. It would seem logical therefore to extend the use of Define to include the possibility of making a formal element or set definition including value, class, form, etc. This should certainly be investigated further to determine its reasonableness and usefulness.

(e) As in the Conditional portion of the Table it is also desirable in the Action portion to provide for a header-trailer type of construction. The following statements are aimed at accomplishing this:

AS::= AS, $\mathcal{E}$ where $\mathcal{E}$ represents "Execute"

<null> ::= $\mathcal{N}$ where $\mathcal{N}$ represents "Not Execute"

for Assign and Define Actions, AS::= AN,R,AV

where AN represents the name, R the Assign or Define operator and AV the value term.

for Procedure Control Actions, AS::= AP,P

where AP represents the procedure-name and P the Procedure Control operator.

for Communication Actions, AS::= AT, C

where AT represents the set name and C the Communication operator.

On this basis the following forms are possible in header-trailer construction.

header	trailer
AS	$\mathcal{E}$
$\mathcal{E}$	AS
AS	$\mathcal{N}$
AN,R	AV
AN	R,AV
AV,R	AN
AV	R,AN
AP	P
P	AP
AT	C
C	AT

(f) I believe that with these formats it is possible to show that any Action in an Evans table can be represented adequately, as can any Action from a Crawford Table. Therefore, further study and evaluation should probably be in terms of what is most effective from a systems engineering viewpoint rather than with a "it can't be handled" attitude.

Burton Grad/lsd
7/12/60

THE "EXPLOSION" OPERATOR AS USED IN PRODUCTION CONTROL

By Burton Grad
Systems Standards, IBM
5/15/60

The structure of one basic production control activity can be developed fairly readily. This is the function commonly known as "Explosion". The Explosion operator converts a time - demand for a particular commodity into a time - demand for a set of resources. Commonly this operator is associated with determining the parts and materials needed to support an end product. However, it would be just as valid to include the determination of machine and labor loads since this is simply the same operator applied to a different set of resources.

This paper will try to define a general formulation of the deterministic case of explosion. It will be done in the context of a specific parts determination example, but would be directly analogous for labor or machine requirements. The particular formulation may very well not be optimal for any one machine, however, it is a convenient way of describing the over-all logic.

It is necessary to define certain terms:

1. A commodity is any object desired by a customer. Each commodity has associated with it an identifier. Commodities include products, reports, service, etc.

2. A resource is any object used to satisfy a demand for a commodity. This may be a tool, jig, fixture, material, assembly, machine tool, labor, money, etc.

---

1. The ideas expressed here were developed with the aid of Mr. A. Bernstein of Math & Application, IBM.

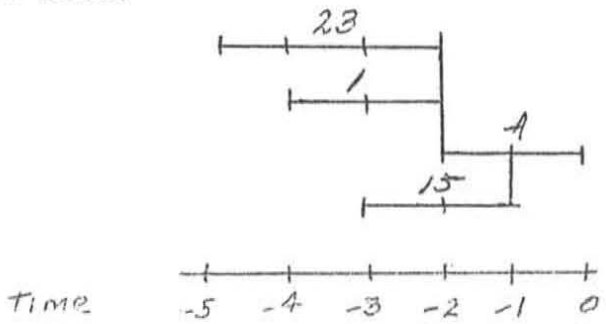
3. The unit of measure by which we describe the amount of a commodity or resource required may be pieces, pounds, gallons, minutes, dollars, etc.

4. The quantity associated with a particular commodity will always be in terms of a unit of measure. This unit of measure may be implied if it is obvious from context or consistent throughout a class of problems (e.g. $in Accounting reports).

5. Demand for a quantity of a commodity will only be meaningful if it has a "due" date (or required date) associated with it. This in no way constrains or describes the profit or loss which will result from not meeting the date. This simply establishes a point of reference without which a production control problem has little meaning. This does not preclude the possibility of implicitly defining the due date through the context of the problem.

6. The usage quantity of a resource means the amount of that resource (in the appropriate unit of measure for the resource) required per unit of the demand commodity (in its appropriate unit of measure). For instance, the usage quantity of a particular kind of paint for a particular model of automobile may be expressed as 5 (gallons)/(one car), where the two units of measure may well be implied by the problem context.

7. Finally, the setback time for a resource can be defined graphically as shown below:



Commodity A has a due date of 0. Resource 15 must be available 1 time period before this due date, hence the setback time for Resource 15 in Commodity A is 1 time period. Resource 1 must be available 2 time periods prior to the completion of commodity A, hence its setback time is 2 time periods. Similarly the setback time for Resource 23 in Commodity A is also 2 time periods.

With these definitions it is now possible to describe the operands associated with the Explosion operator:

1. There is a Commodity Demand List (D) which has the following structure:

Commodity Identifier	Demand Quantity	Due Date
A	5	11
A	7	12
A	2	15
A	1	16
B	4	13

The order of the elements of this list should not be implied by the particular order used in this example.

2. There is a Resource Usage List (U) which looks as follows:

Commodity Identifier	Resource Identifier	Usage Quantity	Setback Time
A	1	2	2
A	15	6	1
A	23	3	2
B	1	4	1
B	12	1	4

Again, the order of this list should not be implied by the way the example is written.

3. The result of using the Explosion operator on the Commodity Demand List and the Resource Usage List is to produce a Resource Demand List (R) with the individual entries having specialized meaning. Each row in the U List is examined and its commodity identifier compared for each row in the D List. Whenever the Commodity identifiers match, a new row is created in the R List, with the elements structured as follows:

R Resource Identifier = U Resource Identifier

R Demand Quantity = (D Demand Qty) X (U Usage Quantity)

R Due Date = (D Due Date - U Setback Time

This is an example of the Resource Demand List:

Resource Identifier	Demand Quantity	Due Date
1	10	9
15	30	11
23	15	9
1	14	10
15	42	11
23	21	10
1	4	13
15	12	14
23	6	13
1	2	14
15	6	15
23	3	14
1	16	12
12	4	9

The particular sequence of solution or examination is of no importance to the nature of the operator as long as every possible resource demand is determined.

It is possible therefore to state that

$$D\ (X)\ U \longrightarrow R \qquad \text{or} \qquad (X)\ (D,U) \longrightarrow R$$

Where D, U, and R represent the respective Lists and (X) represents the Explosion operator.

This description of the Explode operator permits a fully recursive definition for a multiple level process. We only need to make the following statement:

> Any resource may be considered as a commodity at a subsequent explosion level.

On this basis the R List is completely equivalent to the D List and can be treated as a D List for a subsequent use of the Explosion operator.

As can be seen, this approach does not explicitly recognize any of the common efficiency problems associated with performing an explosion procedure. We are being quite careful to distinguish between the operator which defines the transformation of certain objects. Particular ways of achieving this transformation, given different parameters for the Lists and given a specific machine, can be described separately.

TRANSFORMATION LOGIC

The following report proposes a structure for detailed analysis and formulation of the Transformation Logic used in applying Tabular Form to a precise systems language. It suggests major study topics and then divides the Table construction area into a series of specific subjects. Each of these subjects then has possible solutions described, certain problems stated and indicates the related portions of previous reports. It is our hope that this framework will provide a basis for intensive future work and for relating our work with that of the Language Structures Committee.

We feel that there are three major study topics for work in Tabular techniques:

I Table Construction and Data Description
II Applications
III Language Implementation

At the present we are not particularly concerned with Language Implementation (III), but have been concentrating on Table Construction (I) so that appropriate Applications Studies (II) may be carried out.

In subdividing the Table Construction part of the first topic we believe there are three key subjects:

A. Inside - Box Considerations
B. Within - Column Logic
C. General Table Considerations

We will discuss each of these subjects in turn indicating specific areas of work, appropriate solutions, etc.

Subject (A) Inside - Box Considerations:

(1) Operators:

We have identified certain general element operators:

(a) quantitative (+, -, *, /, power, absolute, sine, square root, etc.). These operators may require 1, 2 or more quantitative factors as input and result in a quantitative value.

(b) true-false (Andalso ($\wedge$), Andor ($\vee$), Or ($\not\vee$), Not ($\neg$)). These operators may require 1 or 2 true-false factors as input and result in a true-false value.

(c) string (Concatenate, Insert, Replace, Substring, etc.). These operators may require 1, 2 or more string factors as input and result in a string value.

(d) mixed (Transform, Convert, Count, etc.) These operators typically require either 1 string factor or else 1 quantitative factor as input and then result in one quantitative factor or else one string factor... the opposite of what was available to start.

(e) multiple (not yet explored) These operators would use single or multiple input factors to establish multiple result values.

There are also a number of Relational Element Operators:

(f) special (Exist, Defined, Non-existent, Undefined). These operators establish whether a particular element has an established value or whether the value assigned has defined meaning in terms of the Data Element Description. The input is typically a single factor and the result is a true-false value.

(g) quantitative relational (Less than, Greater than, Equal, and their negatives). These operators require 2 quantitative factors as input and result in a true-false value.

(i) string relational (Identical, Not Identical, Lower, Higher). These operators require 2 string factors as input and result in a true-false value.

There are Set manipulative operators:

(j) general (arrange, extract, join, etc.). These operators require 1, 2 or more sets as input and result in a new set.

(k) complex (explode, update, post, etc.). These operators change values of elements as well as change set membership. 1 or more sets and/or elements are required as input as either a set or an element may result.

(l) Set relational (Equivalent, Identical, Subset and their negatives). These operators require 2 sets as inputs and result in a true-false value. Specifically they will examine these two sets for identical values, identical order, element name similarity, etc. It may be necessary to differentiate the operators for unordered sets from those for ordered sets. This point requires further study.

There are other operators which may be of particular significance:

(m) Change value or set membership (Assign, Copy, Communicate, Receive, Transmit, etc).
These provide for the specific association of a particular value with an element name. It may be possible to change the value or else it may be regarded as a "permanent" value assignment.

(n) Definition (Define)
This operator provides for a substitution of some other factor for an element or set name whenever it is referred to.

(o) Sequence control (Goto, Interrupt, Stop, Perform, Come from, Prior rule, Start, etc).
These are inherently "procedural" operators which do not themselves change data element values, set memberships or establish conditional logic; however, their importance lies in their ability to break up a highly complex description into an understandable group of simpler descriptions. They provide a convenient way to subdivide system logic and a shorthand for indicating conditional repetition or order of action execution.

The operators in each category will be suitably extended and arrangements will be made so that new or specialized operators may be defined in terms of the basic operators predefined by the Language Specification. It is expected that each operator definition in the language specification will explicitly denote the representation of the operator (words, symbols, abbreviations), the type and number of input factors required and/or permitted. The type of result will be indicated and also any parameterization allowed. In other words each operator will have a full definition sheet with suitable examples of its use.

(2) Factors:

We have identified four value classes which factor value can assume.

(a) quantitative -- arithmetically manipulatable values regardless of number base, radix point or graphics. (See B. Grad, July 11, Section C, pg. 7).

(b) true-false -- boolean values. (See B. Grad, July 11, Section C, pg. 7 and 8).

(c) string -- any ordered symbols other than true-false or quantitative. If a quantitative symbol (e.g. 1, 7) is used in a string it has a different meaning from that same symbol used in a quantity. (See B. Grad, July 11, Section C, pg. 8).

(d) set -- a collection of values each of which may be of any of the three types. A set does not of itself have a value in the sense that it cannot be directly manipulated by quantitative, true-false or string operators. The values of the set elements may be associated with element names but do not have to be. There is a reference order for a set though this has nothing to do with actual physical sequence.

There are three basic ways of referring to a particular value: by a Literal, Name or Expression. The meaning and use of these terms is described in B. Grad, July 11, sect. C, pages 8 and 9. The definitions given may be extended to incorporate Set Literals, Set Names and also Set Expressions. Literals must be easily differentiated from Names or Expressions. This needs further exploration.

Rules for the formation of Expressions using various operators should be stated under the appropriate value type category. In addition each of the value type definitions and rules for using Literals and forming Names should be more clearly spelled out. Furthermore the Set-Element Description should provide a convenient means for establishing value type and also specific value to be associated with a particular Name. The development of this Description Sheet should supply effective definitions for many of the terms used under Factors.

There are other types of Factors which need to be considered. For instance we may find it convenient to name or otherwise identify tables, rows, columns, particular conditions or particular actions. The ability to name subroutines or functions, equipments, or various associated physical objects (e.g. machine tools, personnel, locations) may have a strong impact on the communicability of any systems language. The comments in ALGOL-60 in reference to Labels and the COBOL, Commercial Translator, FACT and Flowmatic discussions of names may prove of value to us. Certainly we must explore set names, name qualification of elements and even the possibility of using jargon names.

For example is it possible that we can communicate at the jargon level with data processing machines -- can a machine (intuitively) understand the difference between a "report" and a"graph"?

(3) Conditions:

Certain Boxes will only be able to accept Conditions as their content. These will be called Condition Boxes. A Condition is defined as a properly constructed group of operators and factors which can be determined to be either satisfied or unsatisfied, i.e., whose condition value can be determined. The comments in COBOL, April 1960 page V-2 (except for the last paragraph) are appropriate. Evaluating a Condition does not change the value of any factor involved in such an evaluation.

A Condition Box consists of three things: Condition Operators, Factors, and Condition syntax or structure.

There are only certain Condition Operators. We identify particularly the various relational operators, (f, g, h, i, l).

One Factor is worthy of special mention; this is the Condition Name which represents a Conditional Expression. This Name is used as though the Expression were substituted for the Name. The Name representation itself cannot be described in a Condition Box; it must be done with a Define Operator either in a Define Box or in the Data Description.

The internal structure of a Condition Box permits a great deal of experimentation. It may, however, be desirable initially to limit the variations in the interest of simplicity and clarity. It will certainly be possible later on to add further sophistications to the Condition Box Structure. The following comments and examples are intended to provide only a fundamental structure.

An evaluated Condition will be either satisfied or unsatisfied. If a particular Condition Box statement is met or is "Not Pertinent" to a Decision Rule then the Condition Box is said to be satisfied. If the statement is not met or if undefined values are related to defined values then the Condition Box is said to be unsatisfied.

A Condition Box statement must always be totally satisfied or else it is unsatisfied. A simple Condition is one which consists of a single Condition Operator and the appropriate number of factors. Simple Conditions can only be compounded within a Box by the proper use of connectives between legitimate simple Conditions. Implied repetition of any factor or operator is <u>not</u> permitted within a Box. True-false factors do not receive special treatment within a Condition Box.

The entire issue of Connectives is quite up in the air. They are not the same as true-false operators, nor do they exhibit the same properties. However the English language equivalents for the true-false operators happen to be essentially the same as the primary Connectives which we would like to use. One solution would be to restrict the true-false operators to symbolic representation (e.g. $\wedge, \vee$, etc.) and reserve the English words for Connectives; an alternative would be to use special punctuation symbols like comma, semi-colon, etc. to represent the Connectives and keep the English words for true-false operators. This will obviously require further work before any firm proposal can be made.

The various artifices suggested in B. Grad, July 11, Section C, page 10 should be ignored in terms of Inside-Box Condition construction. These problems will be discussed under General Table Considerations. Reference is made in this context to D. Nelson, August 17.

Examples of valid Simple Conditions are:

X = Y

2 + 2 = 4

PAY. CODE = 6

MARRIED (e.g. MARITAL. STATUS = 3)

X $\leq$ (A + 4)

3 > 2

(A andor X) = Y

((A / X) + 3) = PAY. CODE

DATE = 123160

NAME not = "GEORGE"

(MONTH concatenate DAY) higher than 0228

"WHITE" is subset of SNOW

FILE. A is equivalent to FILE. C

(4) Actions:

Action Boxes serve to change values. There are apparently two major types of special Action Operators: Assign and Communicate. Assign is described in B. Grad, July 11, Section C, page 11. It is executable and provides that a Name will retain the assigned value until it is changed by a new Assign. The sequence of Assign Boxes may be pertinent to the use of the values. Assign also permits Actions of the form:

Assign J as J + 1, where a Name is given a value as a function of a previously assigned value for that same Name. Assign provides for quantitative, true-false or string value manipulation as well as the establishment of appropriate set values.

The question is raised as to whether there should be a special Assign operator for modifying factors inside a Box. There are many pros and cons to this issue which should be explored in depth before making even a tentative decision.

The second type of special Action Operator is Communicate. One solution to this is described by Mal Smith, July 1960 where he suggests the use of Receive and Transmit as the particular Operators in this class.

Action Names would be used to represent functions or tables. It is probably desirable that we permit parameterization of Action Names. This could also be interpreted so as to permit a function which generates multiple values rather than just one value. However, opening this loophole would automatically indicate resolution of the question as to whether a single Action Box can be used to establish the value of more than one factor Name.

The construction of Actions seems quite straightforward. The rules for using Action Operators are such that we consider that a value will be established for a Name by evaluating an expression, named function or other factor. Basic questions have to do with whether to allow multiple value assignment within an Action Box. This also leads to consideration of what connectives to use to indicate independent versus non-independent value assignments. For instance, we might specify that a series of Actions separated by a semi-colon must be executed in the order specified while those separated by periods may be in any sequence. This entire issue of independence and dependence will be discussed in the Between-Box portion of this paper. However, under the category of Action Boxes we need to resolve the question of permitting compound Actions.

Examples of valid Simple Actions are:

assign PAY. CODE as 336611

assign WEEK. PAY as HOURS * RATE

assign MARRIED as MARITAL. STATUS identical 1

assign LINE. A as SET. B

receive PAY. DATA from TIME. CARD

transmit CHECK. INFO to PAY. CHECK

(5) Definitions:

Definition Boxes serve to associate a Name with a value generating factor. The basic Operator is DEFINE which is not executable per se. Rather it imputes a substitution. Whenever a Defined Name is used in an Action it will have the Definition substituted for it. This can also be viewed from the standpoint that a Defined Name will be evaluated in terms of the underlying or "root" values each time it is used. Suppose we have the following Boxes:

define K as P $\neq$ [7]

assign R as K * R

The Define Statement would indicate that K is not to be evaluated except when it is used; and when it is evaluated it is in terms of the then current value of P. In a Define statement the same Name cannot appear in both the "subject" and "predicate". A definition is persistent throughout a system description, but the value is not. In contrast, an Assignment results in a particular value for a Name which can only be modified by another Assignment. Definitions can be nested so that P might in turn be defined in terms of Q, and so on. The point at which a definition appears is of no importance. It is always treated as though it occurred at the very beginning. Definitions should occur only in Unconditional Tables since otherwise there would be the possibility of their being overlooked in a particular solution path. The problem of compound define statements does not seem to arise except in conjunction with multiple synonyms.

(6) Sequence Control:

The final area of Inside - Box discussion is concerned with expressing sequence control. This may be concerned with column selection, Table selection, etc. Reference is made to M. K. Hawes letter August 16, 1960 in which she speaks of Goto, Perform, Halt and Stop. In this class should also be considered Prior Rule. A Sequence Control statement does not change any values, but it is of critical significance to the effective description of the logic of a system.

To support Sequence Control it is necessary to be able to name certain control points in the transformation logic description. In Tabular form this requires identifying a Table Name (numeric, mnemonic or descriptive) or even a particular entry point in a Table; it also may require explicit designation of individual Boxes (probably through a column-row numeric code).

The simplest Sequence Control Operator is Goto, which states unequivocally that logic control should next proceed to the designated named location. It can be used as a short-hand way of indicating the repetition of certain Conditions. This same effect can be produced by the use of a Prior Rule designation. Implicitly, this subsumes (or repeats) all previous conditions preceding that particular branch. This is a highly important convenience because without it we would have to go to very large, highly qualified Tables; with this sequence control ability we can break a complex logic into a series of smaller problems. It is of course essential to the systems planner that he keep careful track of all logical conditions indicated by the chain of Goto's or Prior Rules. This in itself can be a complex problem and may also require the use of Tabular form to maintain logical understanding and control.

Two other Sequence Control Operators are quite simple; these are Halt and Stop. In the first case we provide for a planned interruption for manual intervention--maybe to add a new factor or make a non-formalized decision. This would provide for operational control as in a Man-Machine Simulation of a system. Stop concludes the process and indicates that the transformation logic has been fully defined.

The fourth Sequence Control Operator is Perform. It is used to represent the idea of "Goto and Return". There is a serious question as to the need for this device in a systems planner's Transformation Logic. Basically it is a device for cascading levels of Tables. In other words, in a Box we can indicate by a Perform statement that a whole set of Conditions and Actions are to be carried out. This then is a convenience device designed

to avoid complex elaboration of a particular path within a certain Table or else a means for "repeating" a standard subroutine in many Tables. These are the two major concepts -- (1) an in - line part of the Transformation Logic which could have been connected by a Goto the indicated sub-area and then a Goto back to the main - line and (2) a sub-routine (parameterized or not) which could only be handled by the main - line Table having preset a return instruction before using the Goto. There is, however, an alternative made available by the combined use of the Define and Assign Operators. The Subroutine can be Named and appropriately Defined. Then through a multiple Assign (with or without parameterization) the subroutine can be executed with provision built-in for automatic continuation of the main-line table. With this alternative there is a reasonable likelihood that the Perform Operator will not be needed at all.

(7) Summary

Inside-Box Criteria require an understanding of the various types of boxes which the following have been identified: Condition, Action, Definition and Sequence Control. Each of these boxes may contain a suitable statement consisting of appropriate Operators and Factors as required for that type of box. Operators are a means of transforming Factors: Sets and Elements. A Factor may consist of Literal, Name or Expression. Careful consideration will have to be given to each of these identified topics and adequate definitions of each specific item will be needed.

Subject (B) Within-Column Considerations:

Certainly one of the most potent reasons for using a tabular form for recording transformation logic is the ability to relate one box to another visually; it permits almost automatic implication of certain "noise" words that are required in normal sentence construction. Further, the basic value of the table form comes from the ability to readily associate conditions with actions and to compare alternative sets of conditions or alternative courses of action. So while there are some advances in the Inside-Box concepts mentioned under Section A, we begin to see the basic advantages of table form as we examine Within-Column possibilities.

Essentially, the table form permits ready visual aggregation of conditions or actions and visual relationship between groups of conditions and groups of actions. In every example seen to date the evaluation of a group of conditions (in a single row or column) directly indicates whether or not to carry out a group of actions (in that same row or column).

For convenience, we will talk about a decision rule being expressed in a column rather than a row, though of course the two forms are equivalent. To organize the further discussion on this topic we will sub-divide the subject as follows: (1) Among Condition Boxes; (2) Among Action Boxes; (3) Between Conditions and Actions; (4) Among Conditions, Actions, Definitions and Sequence Control.

(1) Among Condition Boxes

Each Condition Box can be individually evaluated and each will be satisfied or not satisfied as a result of this evaluation. We can therefore relate each box to the whole and indicate whether the whole group of conditions is satisfied. In the simplest case we speak of independent, required Condition Boxes. Independent means that no condition is a function of another; more simply, evaluating one condition has no effect on any other condition. Required refers to the fact that satisfaction of all Conditions is necessary; this can be thought of as "if C_1 and also C_2 and also C_3", where all three Conditions must be satisfied to satisfy the whole.

We can make this more complex by permitting other logical connectives between Conditions like "Andor", or "Or". We could also develop "best 2 out of 3" or "at least 3 satisfied" rules. In the simplest case (if...and also...) the conditions can be examined in any order without influencing whether the group of conditions will be satisfied. In more complex cases this is no longer true and the exact sequence of testing could affect the decision. Although it is logically and technically possible to handle these more complex cases it is recommended that initial consideration be given solely to independent, required Conditions and that only after this is clarified should we be concerned with more advanced approaches. There is a sound reason for this recommendation: The basic power of table form rests on visual relationships; Complex patterns are not visually easy to follow or conceive and hence may well destroy the original reason for going to tabular form. Because of the common need to handle "exclusive or" it should be noted that its incorporation in the Inside-Box concept would provide for the necessary flexibility; e.g., (Inside a Condition Box) MARITAL STATUS is MARRIED or MARITAL STATUS is HEAD.OF.FAMILY. It may even be desirable to offer a short-hand notation for this one special case if it occurs frequently enough; e.g., MARITAL. STATUS is MARRIED or is HEAD OR FAMILY.

(2) Among Action Boxes

Implicit in handling a group of actions is the connective "and then". There must often be a stated sequence of action performance since it is likely that a particular action may affect the value of a factor which is used in a subsequent action. The simplest case is achieved when written sequence (top to bottom) is maintained. Given any series of actions it is clear that they can be written in proper sequence from top to bottom. Questions about explicit sequence indication arise because of attempts at redundancy elimination or for row association. These will be discussed under General Table Considerations.

The problems which arise among actions are concerned not with indication of implicit sequence, but rather with those actions which need not be done in order, but are automatically sequenced in a column approach. This same objection is valid for flow charts, narrative languages and machine-oriented languages. The question can be stated: how can we use sequential language and still indicate that two or more items need not follow each other. No solution is suggested here though certain possibilities are explored briefly. We could have a rule that between action boxes there was no sequential relation except as indicated by common use of a certain factor, in which case written sequence would hold. We would have to explore whether action to action (necessary) sequence can always be logically determined; i.e., the need for sequence, no what the proper sequence is. We should also investigate the usefulness of explicit sequence indication. Another approach is to provide a precedence matrix or precedence chart as shown by Barankin's work on Precedence Matrices and Salveson's paper on Assembly Line Balancing. A third approach might be to use some special line weight or symbol to indicate that the preceding actions must be performed prior to carrying out the following action(s).

It is recommended at this time that we only deal with tables with the following property: all actions in a column are to be performed if any actions in that column are to be performed.

One other interesting idea is the possibility of compacting actions by allowing within an action box a complete loop statement like assign X as X / I for I from 1 to 10 by 1. This would allow a summarization to be explicitly indicated within a single box and would avoid certain types of Goto and condition testing.

(3) Between Conditions and Actions

In virtually all illustrative tables the connection between conditions and actions has been simply, "if...then...". If the stated conditions are satisfied, then execute then actions specified. This may well be the standard and most significant need for tables, but at least one other logical possibility should be explored: "if...then do not do...". This would provide for editing and error correction since as long as various conditions are satisfied, no special action need be taken.

One recommendation is that present studies concentrate on tables where satisfying the various conditions in column will always result in that column's actions being carried out. We should also try to work with "cause and effect" relationships or with functional relationships. Reliance upon incidental or happenstance relations is highly suspect in that it may make table acceptance and maintenance unnecessarily difficult.

As far as deciding whether to execute a group of actions, the Among-Conditions control determines if the Condition-column is satisfied. The knowledge of whether a Condition-column is satisfied determines whether the corresponding action-column is to be executed.

An interesting development of Condition-Action relations can be observed through the following formula.

let N = Total number of conditions in a particular column

and S = number of satisfied conditions in that column after evaluation.

then there seems to be 5 possible situations

(1) if $S = N$ then...

(2) if $S \geq 1$ then...

(3) if $N - S \geq 1$ then...

(4) if $N - S = r$ then...

(5) if $p \leq (N - S) \leq r$ then...for $p \neq r$

These can be reformulated as follows:

General Case:

$$\text{if } p \leqq (N - S) \leqq r \text{ for } p \leqq r \geq 0$$

(1) $p = r = 0$

$$0 \leqq N - S \leqq 0$$

(2) $p = 0 \; ; \; r = N - 1$

$$0 \leqq N - S \leqq N - 1$$

(3) $p = 1 \; ; \; r = N$

$$1 \leqq N - S \leqq N$$

(4) $p = r > 1$

$$r \leqq N - S \leqq r$$

(5) $p \neq r \; ; \; 0 \leqq p \leqq N \; ; \; 1 < r \leqq N$

$$p \leqq N - S \leqq r$$

What this reformulation means is that for each table (or even for each column in a table) we could specify a "p" and an "r" and this would fully describe the logic for carrying out the actions in that column.

(4) Among Conditions, Actions, Definitions and Sequence Control.

The first point we might recognize is the usefulness of an "unconditional" column. This says that the actions, definitions, etc. in that column are to be carried out always. This can be done in a one column Table (the "degenerate" form) or by its use as the "last" column in a table (an "all others" column) or in a table with multiple sets of actions permitted it could be any column (a "must" column).

In general Goto should occur after all other actions have been specified, hence it may be desirable to require Goto operations in the last row of a Table. Prior Rule operations should occur as the first row in a Table. The question of sequence control will be discussed more fully under Subject (C) General Table Considerations. We only want to note that a column will need to be able to distinguish at least these basic relationships: must come directly from; must come from one of these; must previously have been considered (and of course the go to complements).

Burton Grad
December 29, 1960

A MODEL OF AN ELEMENTARY INDUSTRIAL PROCESSING SYSTEM

Burton Grad
Programming Systems
International Business Machines
March 1, 1960

Prepared for presentation at the AIIE 1960 National Conference,

Dallas, Texas, May 12, 1960.

A Model of an Elementary Industrial Processing System

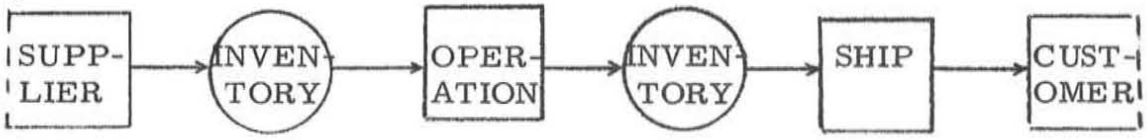
by BURTON GRAD
Programming Systems, IBM
March 1, 1960

The design of an industrial information processing system involves an ability to understand the structure of the physical system being controlled. This type of understanding is essential if reasonable and effective control rules are to be determined and applied. To this end it seems pertinent to be able to model or simulate the behavior of a particular proposed information processing system to see if it will perform in the way that you wish. However, to simulate the performance of an information processing system implies a terminology and format for expressing the control rules and describing the physical system being controlled.

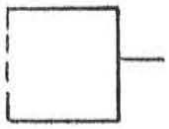
This paper will present a proposed terminology and format for such a representation and will illustrate the use of these ideas in the context of a simple example. As a clear corollary it would seem logical that a generalized simulation language structured in this way might prove highly beneficial to those charged with the information systems design.

Suppose we try to describe a simple manufacturing business which has but one operation, one product model, a single source of supply and just one customer. A flow chart of the physical flow in such a business might appear as follows in Fig. 1.

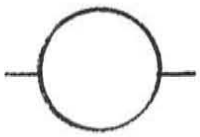
Fig. 1



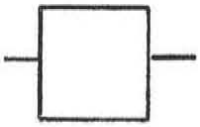
The flow of material is from left to right. There are six Stations in the system with each of the different types of stations being represented by a different symbol:



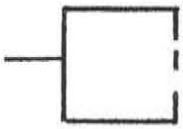
represents a material source external to the controlled system.



represents an inventory of a product model or a material.



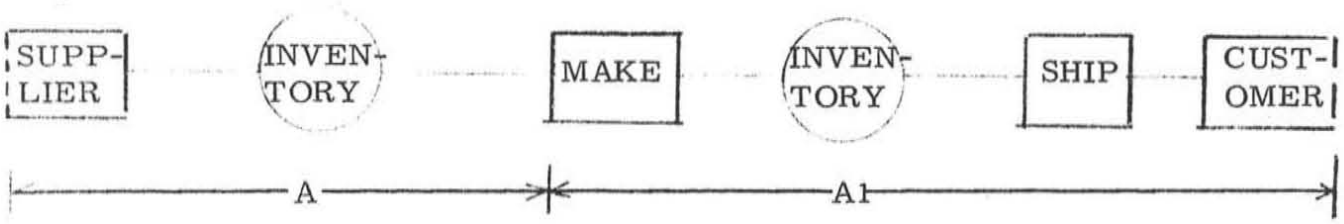
represents a processing station which in some way transforms material.



represents a customer who is essentially a product "sink" and is external to the controlled system.

The status of the system, from a purely physical standpoint, can be described at any point in time by defining the number of product or material units at each station within the controlled system. To aid in expressing this we have numbered the stations as shown on the next page in Fig. 2, and identified the "product" which, of course, changes identity after the "make" operation.

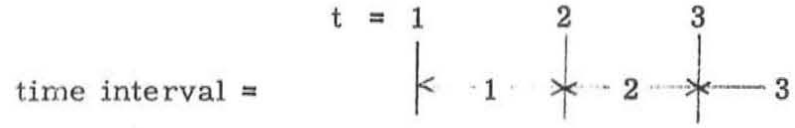
Fig. 2



We can speak of the Raw Inventory at the Inventory Stations and the In-process Inventory at the Processing Stations. Hence, the values I^2_A, I^3_{A1}, I^4_{A1}, I^5_{A1} are required to define the current status of the physical system.

I^s_m means the Inventory at Station s of product model m. To introduce the time concept I^s_m is always defined at a particular point in time, represented by t. Hence, $I^s_{m,t}$ would be the general format for describing the physical status of the business at the beginning of time interval t.

The first step beyond this static representation is to introduce the factors which will transform the system from its state at t = 1 to its state at t = 2. Essentially, we need a record of all movement between stations during time interval 1.



This can be done by defining a set of shipments to represent this movement. Using H to stand for shipments (S is used later for another factor). $H^2_{A,1}$ means the shipment from Station 2 of product A during time interval 1.

Hence $H^1_{A,1}$, $H^2_{A,1}$, $H^3_{A1,1}$, $H^4_{A1,1}$, $H^5_{A1,1}$ are needed to completely itemize all physical events which take place during time interval 1. The general form is $H^s_{m,t}$.

On this basis we can define

$$I^2_{A,2} = I^2_{A,1} + H^1_{A,1} - H^2_{A,1}$$

To generalize this relationship we need to introduce the succeed-precede concept.

Station 1 is said to precede Station 2 if its output passes to that station either directly or indirectly. In our simple business, Station 1 precedes Station 2 and Station 3. Succeed is the opposite of precede. In other words Station 3 succeeds Station 2 and Station 1. These relationships are summarized in Fig. 3.

Fig. 3

Precede (-)
Succeed (+)
Relationships

Station no.	1	2	3	4	5	6
1	0	-1	-2	-3	-4	-5
2	+1	0	-1	-2	-3	-4
3	+2	+1	0	-1	-2	-3
4	+3	+2	+1	0	-1	-2
5	+4	+3	+2	+1	0	-1
6	+5	+4	+3	+2	+1	0

The number in the intersection indicates the relationship between the two stations. For example the first row indicates that Station 1 precedes Station 2 by 1 stage.

$H^{s:-1}_{m,t}$ would be interpreted to mean the shipments of model m during time interval t from a station 1 stage before (preceding) Station s.

The Inventory equation would then have the following form:

$$I^s_{m,t+1} = I^s_{m,t} + H^{s:-1}_{m,t} - H^s_{m,t}$$

This is satisfactory except for the case of transforming a material into a product model which still needs to be explained. This can be shown by a "Gozinto"Table like Fig. 4.

Fig. 4

Gozinto Table

Station	1	2	3	4	5	6
"Gozinto" Material - (g)	A	A	A	A1	A1	A1
Output Product - (m)	A	A	A1	A1	A1	A1

In the Gozinto Table, m is identified as the "output" model at the station under consideration. The Gozinto material (g) may be the same model or it may be a different input material. For example, at station 3 the model identity is A1 but the "Gozinto" material identity is A. Therefore the general equation should be:

$$I^{s}_{m,\,t+1} = I^{s}_{m,\,t} + H^{s:1}_{m:g,\,t} - H^{s}_{m,\,t}$$

Where m:g means the "Gozinto" material g for model m at succeeding station s. To illustrate

$$I^{3}_{A1,\,2} = I^{3}_{A1,\,1} + H^{2}_{A,\,1} - H^{3}_{A1,\,1}$$

This fully defines a manner for computing the physical state of a system given the initial state ($I^{s}_{m,\,t}$) and the shipments between stations during the time interval ($H^{s}_{m,\,t}$).

These values, however, have certain built-in constraints. For example the amount shipped from a station can obviously not exceed the inventory at that station at the beginning of the time interval, assuming that any receipts during the period would not be in time to be operated on. We might express the first part of this constraint by stating that:

$$H^{s}_{m,\,t} \leq I^{s}_{m,\,t}$$

This establishes the limitation of initial inventory available but it does not give recognition to the possibility of immediate access to more input material. This can be done by defining the cycle time required to provide a shipment from a station.

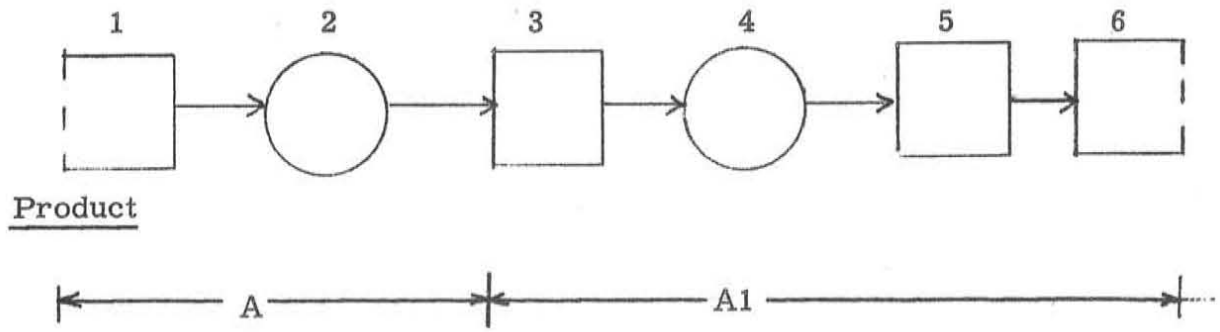
C^{s}_{m} = Cycle time at Station s for model m.

If s is an inventory station then C^{s}_{m} can be O, so that the next Station (s:+1) may receive additional input material in time to be of direct use during that period. This implies a negligible information cycle time which is the time it takes to communicate a material requirement to the preceding station (s:-1). So we might repeat the constraint in this way:

$$H^{s}_{m,t} \leq I^{s}_{m,t} + I^{s:-1}_{m:g,t}$$

So far we have just described the operation of a physical system with no thought of product demands, decision rules or even an explicit recognition of the role of information processing. Let's modify the original flow chart to bring into play certain of these features as in Fig. 5:

Fig. 5



This "phantom" information system has certain obvious properties. The information flow is essentially the reverse of the physical product flow. There is a one to one mapping of stations on decision elements. There are only two types of decision elements: Order and Schedule. A Processing Station and a Source become Schedule decision elements; the other two become Order elements. For the sake of clarity, we will use the same station code numbers to identify the decision elements as were used for the physical stations themselves. The meaning will be obvious from the context.

The first decision element is that of Order. This function is concerned with determining the quantity desired from the preceding station. Without getting involved in the structure of a particular rule, we can see that $R^{s}_{m:g,t}$ (the order quantity from station s for the gozinto material g for model m ordered during time interval t) will depend on present inventory status, current orders and unfilled requests from the succeeding stations, forecast of future requirements, preceding station cycle time, etc. Each station has the opportunity of placing an order on any directly preceding station (s:-1) during the preparatory phase of a time interval.

To examine the meaning of the second decision element, Schedule, it will be helpful to review the sequence of events in the system.

At the very beginning of a time interval (t) we know the inventory status of each station is ($I^{s}_{m,t}$). Then an order is received by Station 5 from Station 6 ($R^{6}_{A1,t}$). Station 5 reviews its own inventory and that of Station 4 to determine the maximum quantity of the requested model which it can process. For simplicity, let's assume that the cycle for shipping is just one time interval so $C^{5}_{A1} = 1$. We can also assume that there is a trivially simple scheduling rule:

$$S^{5}_{A1,t} = \left\{ \min \quad R^{6}_{A1,t'} \quad (I^{4}_{A1,t} + I^{5}_{A1,t}) \right\}$$

where $S^{5}_{A1,t}$ represents the scheduled quantity at Station 5 for period t and min stands for minimum of.

This rule implies no consideration of unfilled orders about which more later. For now we can imagine the customer reordering any unfilled requirements at the beginning of the next time interval (t+1). We can also simplify the equations by precluding a Processing Station from having inventory not assigned to specific orders. On this basis $I^{5}_{A1,t} = 0$; there is no available inventory. For instance, under this restriction there would not even be any actual inventory remaining at the beginning of the next period unless

$C^s_m > 1$

With this approach the general scheduling equation could be reduced to:

$$S^s_{m,t} = \min \left\{ R^{s:+1}_{m,t} \; , \; I^{s:-1}_{m:g,t} \right\}$$

We can also assume that:

$$H^s_{m,t} = S^s_{m,t} \quad \text{for } C^s_m = 1.$$

This simply means that shipments at the end of a period will always equal the quantity scheduled at the beginning of a period if the cycle time is one time interval. We have a deterministic, perfectly performing facility.

So Station 5 decides to make $S^5_{A1,t}$ based on orders received and the prior station's inventory status. It, of course, needs immediate delivery of the gozinto material if it is to make shipment by the end of the time interval. The quantity needed at Station 5 ($R^5_{A1,t}$) is ordered from the preceding inventory Station 4.

Station 4 supplies the material right away ($H^4_{A1,t}$) and then makes its own ordering decision. Here again let's envision a trivially simple ordering rule:

$$R^s_{m,t} = H^s_{m,t} \quad \text{where for the case of } C^s_m = 0 \text{ and } C^{s:+1}_m = 1$$

$$H^s_{m:g,t} = S^{s:+1}_{m,t}$$

In later papers we will try to eliminate these highly restrictive assumptions, but for the sake of explanation and understanding, I believe these simple rules are most desirable.

Station 4 will transmit its order ($R^4_{A1,t}$) to Station 3 which will schedule the required quantity. If $C^3_{A1} = 2$ this would mean that Station 4 would not receive delivery until the end of the t + 1st period. In other words, we'll have to make sure that Station 4 has a properly determined safety stock to be able to cover its expected needs at the beginning of period t + 1.

But Station 3 may not even be able to supply the entire order for Station 4 since it is itself restricted by the inventory at Station 2. Conversely it may have some material half-way through the process that it started during period t-1. We have made the statement that no quantities are scheduled on order speculation so we can assume that any $S^3_{A1,t-1}$ would have been in response to an $R^4_{A1,t-1}$

If we define $S^{s}_{m,t}$ to mean the quantity to be started at the beginning of interval t then we can say (assuming perfect performance):

$$H^{s}_{m,\,t+c-1} = S^{s}_{m,\,t} \quad \text{where } c = C^{s}_{m}$$

The quantity shipped at the end of the time interval t + c - 1 will equal the quantity started at the beginning of t. If c = 1 then this degenerates into:

$$H^{s}_{m,\,t} = S^{s}_{m,\,t} \quad \text{as for Station 5}$$

If $C^{3}_{A1} = 2$, then $H^{3}_{A1,\ t+1} = S^{3}_{A1,\,t}$

We can then say that:

$$R^{3}_{A,\,t} = S^{3}_{A1,\,t} = \min \left\{ R^{4}_{A1,\,t},\ 1^{2}_{A,\,t} \right\}$$

This order request ($R^{3}_{A,\,t}$) is immediately transmitted to the preceding inventory Station 2 which responds by delivering the required material:

$$H^{2}_{A,\,t} = R^{3}_{A,\,t}$$

This brings out a key aspect of the definition. An inventory station does no processing and therefore requires no time. Where there is a function to be performed it is necessary to distinguish between the inventory station and a processing station.

Next, Station 2 goes through its ordering procedure: we could think of $R^{2}_{A,\,t} = H^{2}_{A,\,t}$ with appropriate provision having been made for safety stock to cushion the demand variance over the lead time interval.

Let's now introduce the idea of an information lag. Suppose the order from Station 2 takes 1 time interval to arrive at Station 1 and suppose Station 1 will take 2 time periods to process the material and 1 time period to transport it to Station 2. If we simply add these factors together (1+2+1) we obtain a lead time of 4 time periods.

We need to represent separately the ideas of information cycle, processing cycle, transportation cycle and lead time since each is used in a somewhat special way in describing an operating system. This can be done by

adding a second subscript to the C^s_m term making it:

$C^s_{m,\,i}$ = Information Cycle

$C^s_{m,\,p}$ = Processing Cycle

$C^s_{m,\,r}$ = Transportation Cycle

$C^s_{m,\,d}$ = Lead Time

In the example described,

$C^1_{A,\,i} = 1$

$C^1_{A,\,p} = 2$

$C^1_{A,\,r} = 1$

$C^1_{A,\,d} = 4$

This enables us to think of $S^1_{A,\,t+1}$ being in response to $R^2_{A,\,t}$ and $H^1_{A,\,t+2}$ resulting from $S^1_{A,\,t+1}$.

Because of the transportation cycle, we should now differentiate between a shipment and a receipt so it is useful to define a receipt as an Arrival $A^s_{m:g,\,t}$. In this case $A^2_{A,\,t+3}$ should reflect $H^1_{A,\,t+2}$. The generalized formulas would now appear for the deterministic case as:

$$S^s_{m:g,\ t+c(i)} = f\left(R^{s:+1}_{m:g,\,t}\right)$$

where $c(i) = C^s_{m:g,\,i}$

$$H^s_{m,\,t+c(p)-1} = S^s_{m,\,t}$$

where $c(p) = C^s_{m,\,p}$

$$A^{s:+1}_{m:g,\,t+c(r)} = H^s_{m:g,\,t}$$

where $c(r) = C^s_{m:g,\,r}$

$$A^{s}_{m:g,\ t+c(d)-1} = f\left(R^{s}_{m:g,\ t}\right)$$

$$\text{where } c(d) = C^{s}_{m:g,\ d}$$

$$c(d) = c(i) + c(p) + c(r)$$

Station 1 may have its own problems in terms of raw materials inventory, scheduling capacity, and so on, but we will treat the supplier, for now, as though it were an infinite, deterministic source so that:

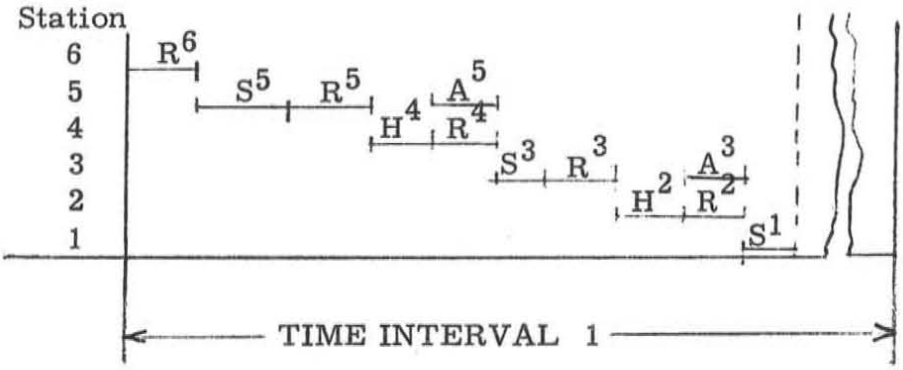
$$S^{1}_{A,\ t+1} = R^{2}_{A,\ t}$$ (to take into account the information lag)

and

$$A^{2}_{A,\ t+3} = R^{2}_{A,\ t}$$

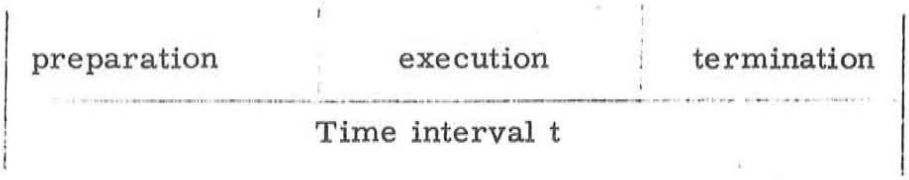
Diagrammatically the preparation period would appear as shown in Fig. 6.

Fig. 6



Where S^{1} refers to scheduling the order placed by Station 2 during the previous time interval.

At the other end of the time interval, the termination period, there is no meaningful sequence of shipments and arrivals, but there is an order of calculation. Each non-inventory station ships the quantity it had scheduled to complete by the end of that time interval. Each station next computes its arrivals based on shipments and transportation cycle. Then each inventory station calculates the inventory status which it will have at the beginning of the next time interval. The three portions of a time interval are shown in Fig. 7.



During the execution period each station carries out its assigned task.

One information concept still needed is the unfilled order logic. Let $U^{s}_{m,\,t}$ stand for the unfilled orders at station s for model m at the beginning of time interval t. So,

$$U^{5}_{A1,\,2} = U^{5}_{A1,\,1} - H^{5}_{A1,\,1} + R^{6}_{A1,\,1}$$

In effect, the unfilled orders are an "inventory" of open orders to which the "receipts" are the orders from the succeeding station and the "withdrawals" are the shipments to that station. We can consider the unfilled orders being calculated during the termination period of a time interval after the shipments have been made and arrivals computed.

To illustrate further, since we have considered that each processing station transmits an order $R^{s}_{m:g,\,t} = S^{s}_{m,\,t}$, and each inventory station makes immediate shipment, an inventory station could calculate its unfilled orders during the preparation period of the time interval. This can be quite significant in that an inventory station's ordering rules can, in this way, be made a direct function of unfilled orders.

The concept of unfilled orders brings in another factor: the ordering station's acceptance of a delivery delay. For this discussion we will simplify by assuming full willingness to accept any delivery delay for the order placed, though such delays may have their influence upon future orders.

Unfilled orders are calculated as viewed by the supplying station. The converse would be for the ordering station to keep track of its unfilled requests.

$$V^{s}_{m:g,\,t+1} = V^{s}_{m:g,\,t} + R^{s}_{m:g,\,t} - A^{s}_{m:g,\,t}$$

which is directly analogous to the unfilled orders approach. Unfilled requests must be calculated after determining shipments and arrivals. As a rule, inventory will be used to satisfy unfilled requests first, at the beginning of the next time interval; any remaining inventory will then be applied to new orders received.

The unfilled order equation can be generalized further if we withdraw the implied assumption that the information cycle time is zero. This requires defining another factor, orders received.

$O^{s}_{m,t}$ are the orders received at station s for product m during the preparation period of time interval t.

$$O^{s}_{m,\,t+c(i)} = R^{s:+1}_{m,\,t}$$

where c (i) again means $C^{s}_{m,\,i}$; we can now restate the unfilled order formula:

$$U^{s}_{m,\,t+1} = U^{s}_{m,\,t} + O^{s}_{m,\,t} - H^{s}_{m,\,t}$$

We can also consider the unfilled orders which have <u>not</u> yet been scheduled. This would be:

$$U^{s}_{m,\,t} - \sum_{a=t}^{a=t-c(p)+1} S^{s}_{m,\,a}$$

<u>Summary</u>

To review the development in this paper, a set of terms has been established which describes the changes that occur during a time interval from both a physical and an information viewpoint. These systems change factors include:

$H^{s}_{m,\,t}$ = Shipments from Station s

$A^{s}_{m,\,t}$ = Arrivals at Station s

$R^{s}_{m,\,t}$ = Orders placed by Station s

$S^{s}_{m,\,t}$ = Amount Scheduled at Station s

$O^{s}_{m,\,t}$ = Orders Received by Station s

$H^{s}_{m,\,t}$, $A^{s}_{m,\,t}$ and $O^{s}_{m,\,t}$ result from the operation of the system and, though influenced by the various decisions, are not themselves decision elements. The other two are decision elements except $R^{s}_{m,\,t}$ for s:+1 non-existent (e.g. a customer station).

Secondly, there are certain systems "constants":

$C^s_{m,i}$ = Information Cycle Time

$C^s_{m,p}$ = Processing Cycle Time

$C^s_{m,r}$ = Transportation Cycle Time

Third, there are terms which enable us to express both the physical and information status of a system at the beginning of a time interval. These systems status variables are:

$I^s_{m,t}$ = Inventory at Station s

$U^s_{m,t}$ = Unfilled Orders (received by Station s)

$V^s_{m,t}$ = Open Requests (placed by Station s)

There are also 3 key relationship concepts which have been expressed:

(1) Precedence

(2) Gozinto

(3) Time interval breakdown into preparation, execution, termination

A variety of techniques are being developed to overcome the simplifications in this model. They will be discussed in later papers. Included will be consideration of:

(1) Assembly structure including multiple supply and multiple destination stations.

(2) Alternate routing

(3) Non-stock inventory stations

(4) Quality variance including spoilage

(5) Non-infinite sources, multiple suppliers for the same product.

(6) Variable cycle and lead time

(7) Order aggregration

(8) Multiple period scheduling intervals

(9) Various scheduling rules including consideration of capacity limitations, employment stability

(10) Various ordering rules

(11) Safety stock considerations

(12) Variable processing times including setups

(13) Transportation stations

(14) Effect of various dispatching rules

(15) Stochastic production and supply rates

This paper just illustrates the initial work in an area which requires extensive further research and development.

Tables Signal Better Communication

Talk Given by Burton Grad, Manager IBM Systems Engineering Development

The pilot is preparing to land his single engine plane at the airport; it is late at night and his fuel supply is low. He calls to the radio tower and asks for landing instructions. All he hears in return is a babble in a foreign language which he can't understand.

The executive has spent the last hour of his day dictating an important speech; the next morning he comes in and wants to review the material. His secretary is out ill. The other girls in the office all read Gregg, not Pitman.

A design engineer has carefully prepared a number of complex Boolean equations to explain the operation of a new computer circuit. He shows these to the manufacturing engineer to give an indication of what needs to be constructed. The manufacturing engineer says, "I don't understand Boolean algebra."

We could go on and on citing examples like these of events and occurences where lack of a common language for communication causes difficulties ranging all the way from the most trivial to the deadly. Systems Engineering faces communication barriers as serious as those of any profession. The systems engineer today does not have a language to communicate with management; he does not have a language to communicate with computer programmers; he does not have a language to communicate with functional specialists; he does not even have a language to communicate with other systems engineers.

Programmers who have learned one computer at the machine language level can't understand the programming of another machine at the machine language level without spending the time necessary to learn the second machine's special codes and instructions. For this reason (among others) there has been intensive effort to develop common languages like FORTRAN, Commercial Translator and COBOL which will be applicable to a number of machines. But the communication between programmer and machine is merely a small part of the total problem.

For Systems Engineering it is vital to develop tools and techniques to permit a manager to state his decision criteria and decision rules. We must find a common language so systems engineers can communicate with product engineers, accountants, and manufacturing planners, to find out their decision rules and decision logic; that is critical to determine the characteristics of the system that is going to be modelled or controlled. A method must be found for two-way communication with computer programmers to be sure that the intended decision rules are in fact being executed. A technique is needed to aid the systems engineer in establishing complete decision rules and in predetermining that these rules will accomplish the intended goals.

In the past, this problem has not been as severe. Because of the limited size of business systems problems, we could depend on the programmer to understand the particular problems well enough to be sure the logic was correct and to check the problem out thoroughly. However, as the systems we are trying to solve become larger and more complex, this expedient is no longer satisfactory. Systems engineers must take on the responsibility for designing the decision logic and for insuring that it is being executed properly. To do this systems engineers must have a professional language which will serve for effective intercommunication.

What has caused the communication void? What has caused this communication moat surrounding the systems engineer? There are at least three major factors involved:

1- The inability to clearly and concisely express decision logic and decision rules for describing business systems.

2- The inability to show cause-effect relationship between conditions and actions.

3- The inability to guarantee or even aid in achieving logical completeness in establishing decision rules.

Today, we have available a number of techniques which have been applied to solving the communication problem: we've tried to use narrative, flow charts and even logical equations. But none of these has filled the bill. Each has major drawbacks; the failure of these known techniques has led to consideration of another alternative: decision tables.

Decision Tables

Decision tables are a formal method for describing decision logic in a two-dimensional display. The layout clearly shows the cause and effect relationship between conditions and actions; it explicitly relates decision alternatives.

Decision tables use a format which is familiar to us from analytical, financial, and statistical tables. Since the days of the Babylonians, people have used tables as a means of organizing information where the relationships were complex or the amount of data great. These data tables appear to be superior to many other forms of information organization because:

1- They provide clarity and conciseness through data classification.

2- They clearly show relationship of dependent to independent variables.

3- They explicitly indicate omissions.

Decision tables use tabular format to represent dynamic situations. Where we have used flow charts, narrative, or logical equations to describe decision logic, or an operating procedure, we now find it possible to use decision tables for these same jobs. The argument in favor of tables is their relative convenience and effectiveness, not that they can describe systems that cannot also be described in other ways.

Tabular form has been used by programmers since the earliest days of computers. The most common use of tables has been to relate some function to an argument. Given the value of one factor, the table provides the value of another dependent factor. For example, a table might relate capitals to states (Figure 1). Given the state name, determine the name of the capital.

STATE	Alabama	Alaska		Wyoming
CAPITAL	Montgomery	Juneau		Cheyenne

In this example State appears above the double line and Capital below; each different state name is in a column and physically below it, the name of the corresponding capital. If the State is Alabama, then the Capital is Montgomery; if the State is Alaska, then the Capital is Juneau.

An extension of this concept is seen in Figure 2 in the use of a matrix to display the value of a particular factor as a function of multiple variables.

HEALTH / AGE	EXCELLENT	GOOD	FAIR	POOR
≥25 <30	1.27	1.52	1.98	2.73
≥30 <45	1.83	2.12	2.53	3.42
≥45 <55	2.51	2.93	3.47	5.27
≥55 <65	3.29	3.91	4.85	8.73
≥65	5.21	6.45	7.61	10.97

Insurance premium rates are shown as a function of health and age. In the example, if health is excellent and age is between 25 and 35, then the rate is $1.27. However, if health is poor and age between 55 and 65, then the rate is $8.73. Unfortunately, the visual effectiveness of a matrix is reduced when the number of independent variables exceeds two or the number of dependent variables is greater than one.

Because of the natural benefits from using tables, it seems that there should be some way to generalize tabular form so that any number of independent and dependent variables might be shown with clear visual correspondence. Figure 3 (on the next page) shows a table with four independent and three

dependent factors where clarity, interrelationship and comprehensiveness have been maintained.

Health	Excellent	Excellent		Poor
Age	25, 35	25, 35		65
Section of Country	East	East		West
Sex	Male	Female		Female
Premium Rate	1.27	1.18		0.82
Policy Limit	200,000	100,000		10,000
Type of Policy	A, B, or C	A, B, or C		R

In this example, the decision table indicates insurance premium rate, policy limit, and type of policy as a function of health, age, section of country, and sex. If the applicant is in excellent health, between 25 and 35 years of age, from the East, and is a male, his rate is $1.27, the insurance limit is $200,000, and he may be issued policy type A, B, or C. All of the alternatives are clearly set forth, one by one, across the table.

To obtain a better understanding of a decision table, let's look at its fundamental elements as shown in Figure 4.

Decision Rule

TABLE HEADER	RULE HEADER
Condition Stub	Condition Entry
Action Stub	Action Entry

The double lines serve as demarcation: CONDITIONS are shown above the horizontal double lines, ACTIONS below. The STUB is to the left of the vertical double line, ENTRIES to the right. A condition states a relationship. An action states a command.

If all the conditions in a column are satisfied then the actions in that column are executed. Each such vertical combination of conditions and actions is called a RULE. In the same column with the entries for each rule, there may be specialized data relating to that rule; this is called the RULE HEADER. Similarly, each table may have certain specialized information which is called the TABLE HEADER.

Consider another sample table which contains all the same elements, but has some different properties. This table is Figure 5.

TABLE:CREDIT	Rule 1	Rule 2	Rule 3	Rule 4
Credit limit is o.k.	Y	N	N	N
Pay experience is favorable		Y	N	N
Special clearance is obtained			Y	N
Approve order	X	X	X	
Return order to Sales				X

The first rule would be read: If credit limit is OK, then approve order. The second rule would be read: If credit limit is not OK and pay experience is favorable, then approve order. In this LIMITED ENTRY table, the entire condition or action must be written in the stub. The condition entry is limited to indicating whether the corresponding condition should be asserted, negated or ignored; the action entry indicates if the action stub should be executed or ignored.

This is in contrast, as you may note, to the table shown in Figure 3, which is called an EXTENDED ENTRY table. In this case the individual condition or action information extends from the stub into the corresponding entries. In any given table, we can, of course, mix extended and limited entry form, whichever is more convenient for a particular condition or action.

The Use of Decision Tables

To this point sample decision tables and their elements have been discussed to describe concept and structure. Now the application and use of decision tables will be presented. A number of experiments conducted over the past four years have used decision tables on a variety of problems; these will be reviewed briefly.

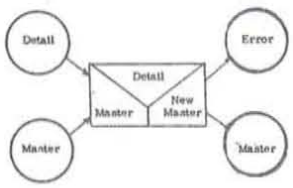
While I was project leader for General Electric's Integrated Systems Project, the potential application of tables to a wide variety of problems was explored including its use for product design, operation planning, cost determination, factory scheduling, etc. The results certainly revealed the opportunity of using decision tables as a major new tool to clarify communication among different technical specialists as well as between these specialists and computer programmers. It was stimulating to watch a manufacturing engineer suddenly grasp product design decision logic and then point out where restraints had been introduced by the product engineer that were of little value to anybody. Through this kind of examination, significant improvements might be made in the total product.

At Sutherland Company, a consulting firm in Peoria, Illinois, management decision rules have been studied with various customers and expressed in tabular form. These decision tables have been applied to Air Force logistics and various commercial situations such as accounts receivable, accounts payable, etc. From all reports, this work has permitted a more effective and comprehensive statement of the current decision logic and provided more meaningful and understandable communication between systems men and programmers.

An area of experimentation already familiar to many of you is the work done at Hunt Foods and Industries by Mr. O. Y. Evans, who is now with IBM. Mr. Evan's work was directed toward communication among different systems men, and from systems men to programmers, concerning the complex decision rules involved in stock control, sales analysis, etc. The results demonstrate that this approach was an effective formal way to state very complex logic without requiring knowledge of Boolean algebra or any other precise mathematical technique.

IBM has been working with several of its customers investigating potential applications of decision tables to a wide variety of problems. From these experiments, it seems clear that decision tables are frequently easier to prepare than comparable programming methods, and that they are an effective aid to systems analysis. In these experiments, communication between systems engineer and programmer has been substantially improved; communication between systems engineer and management has also benefitted from the common description of decision rules.

To convey how tables can be developed, let's follow the process through the significant problem of file maintenance . The block diagram in Figure 6 indicates the essential elements of the problem solution.



A detail file and a master file are the two inputs. The updated master file and an error file are the principal outputs. Within the computer, three basic areas are assigned: master, detail, and new master. The purpose of the update logic is to modify the incoming master file by the detail information to produce an updated master file containing any additions and changes and from which deleted records have been eliminated.

Figure 7 (on the following page) is one of two tables prepared to perform this job.

Rule 1 states the starting condition. At the start, one master record and one detail record are read into the corresponding memory areas. At this point, sequence control returns to the beginning of the table.

Rule 2 and all the following ones are now pertinent. Rule 2 specifically handles the end of job conditions, i.e., end of detail and end of master. In this case, control is transferred to End, a closing routine to provide for sentinels, tape marks, etc.

TABLE: Update Rule #	01	02	03	04	05	06	07	08
Start	Y	N	N	N	N	N	N	Else
End of detail		Y	Y	N	N	N	N	
End of Master		Y	N	Y	N	N	N	
Detail					Master	Master	=Master	
Detail an "addition"				Y	Y			
Do Error Routine								x
Move Master to New Master							x	
Move Detail to New Master				x	x			
Set Addition Switch				On	On		Off	
Write Master			x			x		
Read Master	x		x			x		
Read Detail	x			x	x			x
Go to Table	Update	End	Update	Change	Change	Update	Change	Update

Rule 3 describes the situation when the end of detail has been reached, but not the end of master. Since there can be no further changes, additions, or deletions to the original master, the actions are to write the updated master from the master area, read another master, and then return to the beginning of the table.

In Rule 4, the end of master has been found, but not the end of detail; the remaining details should only be additions. Therefore, the information in the detail area is moved to the new master area, the addition switch is set on, a new detail record is read, and control transferred to the Change Table.

Rules 5, 6, and 7 are concerned with cases where neither the detail nor the master file has ended. The identification number in the detail area is compared to the identification number in the master area. Rule 5 considers the event when the detail is less than the master; in this case the detail should be an addition in order to follow the same logic of Rule 4. In Rule 6 the detail is greater than the master; consequently the same logic as Rule 3 applies. Rule 7 covers the case where master and detail are equal. The information in the master area is moved to the new master area, and control is transferred to the Change Table.

The final rule, Rule 8, is the ELSE Situation. When this occurs something has gone wrong, since all legitimate possibilities have already been examined. An error routine is carried out; then another detail record is read. Rule 8 will take care of cases involving sequence errors in the master file and certain types of sequence errors in the detail file (if the out-of-sequence detail is not an addition). It will also take care of any non-matching detail which is not an addition.

The table can be rearranged to aid programming efficiency: columns with higher frequency of success should be moved to the left and those with lower frequency to the right. Rules 1 and 2 would be way over to the right since they occur only once in each program. Depending upon the particular data, Rule 6 (the column where the detail is greater than the master) will probably be the most frequent case and should be the first one considered. One recommended order is: 6, 7, 5, 3, 4, 1, 2, 8.

Another concept for improving program efficiency is to rearrange the conditions to present the most discriminating condition at the top and the least discriminating at the bottom. For example, the start condition, which is shown first, probably should be last since this only distinguishes one case out of all the thousands that will occur. A similar statement can be made about end of detail and end of master. It seems evident that the comparison of detail to master would be the most discriminating criteria and therefore placed first in the table.

The Case for Tabular Form

Look once more at Figure 7 and compare its statement of the update decision logic with that in the following narrative. Which is clearer and more concise, which shows cause-effect relationships better, which aids more in determining logical completeness.

Mr. T. F. Kavanagh speaking at the 1960 Eastern Joint Computer Conference had this to say: "the decision... table is a fundamental language concept... broadly applicable to many classes of information processing and decision making problems... tables force a step-by-step analysis of the decision... are easily understood by humans regardless of their functional background (they are) simple and straightforward (enough) that... specialists can write tables... with very little training... tables are easy to maintain (and) errors are reported at the source language level."

Mr. O. Y. Evans states of his work on tabular techniques: "The tabular approach... aids... in visualizing the numerous relationships and alternatives... (and) permits data rules to be readily reviewed for omissions and inconsistencies... (in addition it) provides flexibility in changing any portion of the analysis."

The CODASYL Systems Group, part of The Development Committee of the Conference on Data System Languages, has been looking into the use of decision tables. In a recent release the following statement was made: "Investigation... indicates that the systems analysis method discussed above (decision tables) will provide a precise and orderly method of documenting the analysis independent of the processing method. It will offer the analyst an aid in visualizing the relationships and alternatives of the problem, will provide flexibility in changing any portion of the analysis, and will establish a framework for the complete definition of the systems problem. The CODASYL Systems Group will continue to develop and experiment with these concepts."

To further indicate the potential results from use of tabular form, the following statements paraphrase various user opinions: Clarity and conciseness -- Decision tables are easy to prepare, read, and teach to others; experience shows that non-programmers can learn to prepare satisfactory tables in less than a day. The amount of writing, or number of words, lines and symbols used in describing complex decisions, is reduced by 25-50% as compared to flow charting. For certain specific cases, problem statement and programming time combined have been reduced significantly.

Meaningful Relationships -- Table structure serves to improve systems logic by aligning alternatives side by side. It also sharpens cause and effect understanding, so relationships which are accidental or incidental become clearer. Furthermore, actions based on similar or related conditions are apt to be drawn into the same table, making it easier to appreciate and consider interdependent factors.

Completeness -- Tabular form allows effective visual or deck debugging both by the analyst and the reviewer. There are fewer errors to start with since the analyst tends to catch his own mistakes; moreoever, the reviewer will typically detect a high percentage of the remaining errors by visual examination. Finally, experience shows that with this foundation and suitable test problem construction, it is easy to rapidly detect the balance of the errors during machine debugging.

The evidence quoted on the advantages of decision tables for systems analysis and computer programming is based on actual study projects. Some of these studies even tested decision tables on various data processing machines. There are many current studies which are experimenting with a variety of tabular forms.

A Plan for Action

With all its potential advantages, it is apparent that tabular form has not yet achieved full growth and stature; there are major technical and application areas still unprobed, awaiting only the touch of creativity to make practical breakthroughs. While current table methodology does not yet provide a drawbridge to cross the communications moat surrounding systems engineers, it appears to offer the greatest chance for a significant advance.

To bring these possibilities to fruition requires experimental development. Tabular form will have to be tried and used on a wide variety of applications to provide practical evaluation and determine desirable characteristics. Along with this field pre-testing, there will be a need for effective technical developments to explore new table concepts and structures.

There are many areas which need experimental and technical development:

1. Table structure
   -- multiple successes per table
   -- interspersing conditions and actions
   -- explicit control of sequence of actions
2. Relations among tables
   -- prior rule concepts
   -- use of library functions
   -- use of open and closed subroutines
3. Language considerations
   -- statement construction
   -- macro or jargon operators
   -- machine independence
4. Associated data description
   -- defining factors and expressions for man-to-man and man-to-machine use
   -- conditioned definitions
   -- input/output format
   -- preassigned values and constants
5. Implementation considerations
   -- compiling vs. interpreting
   -- sequential vs. random access to tables
   -- possibility of made-to-order processors
   -- ability to introduce specialized operators and table structures

The explosive innovations in computer hardware have not been matched by corresponding developments in systems communication. But we are on the threshold of a major breakthrough, we are on the verge of a significant advance. It's up to you and it's up to us to show equal creativity in software to that shown in hardware: To use tabular form to develop a clear, concise, meaningful, comprehensive Systems Engineering language.

TABULAR FORM IN DECISION LOGIC

by BURTON GRAD, IBM Corporation,
Thomas J. Watson Research Center,
Yorktown Heights, N.Y.

Reprinted from DATAMATION Magazine, July 1961

An F. D. Thompson Publication

Tabular form has shown promise of being an effective way to organize and present decision logic for systems analysis and computer programming. Experience to date clearly indicates the need for further exploration and development of tabular form to determine its range of application and assess its future potential. This report has the dual purpose of sketching the historical background on the development of tabular form, and indicating its possible advantages.

TABULAR FORM IN DECISION LOGIC

by BURTON GRAD, IBM Corporation,
Thomas J. Watson Research Center,
Yorktown Heights, N.Y.

Glancing around the office, I can see three young women busily engaged in the various duties of a typical work day. Let me tell you about them. Blond Marilyn is a chatterbox. Penelope and Theresa enjoy going to the movies. Marilyn is married, but the other two are single. Penelope has an attractive figure, while Marilyn is somewhat on the plump side. Theresa's quiet moods contrast to Penelope's happy ones, but they both seem to enjoy life in native Manhattan. Marilyn has dimples; Theresa may be recognized by her amber eyes and red hair. Unlike the others, Marilyn prefers Shakespeare and country living in Chappaqua.

Without looking back, can you recall all of Penelope's characteristics? Do you have a clear image of each girl and know what data is missing or where there are inconsistencies? To help answer these questions, let's rearrange the information. Displayed in tabular form, it would appear as in Figure 1:

Name	Marilyn	Penelope	Theresa
Marital Status	Married	Single	Single
Hair Color	Blond		Red
Figure	Plump	Attractive	
Enjoys Movies		Yes	Yes
Prefers Shakespeare	Yes	No	No
Residence	Chappaqua	Manhattan	Manhattan
Features	Dimples		Amber Eyes
Characteristics	Chatterbox	Happy	Quiet

Figure 1

From this illustration, some of the advantages of tables over narrative style for comparative data display can be readily appreciated: **Conciseness and clarity** is achieved by classifying data; **Completeness** is insured by revealing areas where information is missing; **Meaningful relationships** are recognized quickly and easily with the two dimensional structure.

While recognizing these advantages many will point out that tables are merely a systematic way to present static data. Do they have a worthwhile function in a more dynamic situation—that of decision making? Would tables be valuable in systems analysis and computer programming? Before we explore some preliminary answers to these questions, let's look at a brief history of tables.

universality of tables

Tables, whether statistical, financial, or analytical, have gained widespread recognition; they seem to be a natural form for expressing relationships among variable factors where there are many possible patterns for arranging the significant information. This fact is substantiated by the profusion of examples in everyday life:

> The ubiquitous government reports with ponderous breakdowns of the GNP or a simple recap on whooping crane birth rates and population.
>
> The multiplicity of financial reports showing the status and growth of businesses.
>
> The economic forecasts of things to come ranging from hula-hoop production to manned satellites in the burgeoning 60's and beyond.
>
> The daily scratch sheet, the box scores of runs, hits and errors for the latest baseball games, and the highs, lows, and closing prices for stocks – all in the local newspaper.

And the list grows.

application to computers

Since the early days of computer development, programmers have used analytical tables to convert arguments into precise functional values; they have also employed matrix structure and notation to handle common information with relatively complex structure. In the past few years, however, there has been substantial interest in probing the potential applications of tabular form for recording the decision logic itself. This exploratory work in developing decision tables has involved consideration of man-to-machine as well as man-to-man communication.

In systems analysis and computer programming, decision tables, like conventional data tables, retain a two-dimensional structure to portray significant relationships. The form, however, is considerably more elaborate to show multiple conditions and actions interlocked through position. Within a decision table any language from a business jargon to the most machine-oriented may be utilized to express the decision logic.

There are other well-known methods to describe a business system: narrative, flow charts, and logical equations. Narrative form, unfortunately, is often wordy, requiring prepositions, conjunctions, and other superfluous elements for readability; there is a certain lack of form and physical relation which may lead to inaccuracy and inconsistency if the user is not extremely careful. Flow charts require lines and connectors to show relationships; when these become too numerous, the logic may be difficult to follow and the layout may demand excessive space. Logical equations are symbolic and abstract as, for example, Boolean algebra applied to computer programming. The main limitations are the need for special skills and background to algebraically describe decision rules and the attendant difficulty in communicating equations in a business environment. Shortcomings in these well-known methods have encouraged systems analysts to take a harder look at other alternatives.

Tabular form for decision logic seems likely to satisfy this search since it compensates for many of the limitations of the other forms by providing compact expression of decision rules, visually effective display of meaningful relationships, and straightforward indication of logical correspondence. The significant difference between tabular form and other methods is not in the notational scheme used, but rather in the physical layout for recording the systems description or programs.

Let's now examine the use of decision tables. It is not intended to suggest that this form is superior to existing languages where they are appropriate for a specialized class of problems, e.g., FORTRAN for algebraic calculations, report generators for preparing output documents. Rather, the feeling is that no method today is well-designed for systems men to use for describing complex logical decisions; therefore, decision tables may well fill a current void in a total systems analysis and programming package.

extended entry tables

One type of decision table is called EXTENDED ENTRY. Figure 2 illustrates a simple application:

	Rule 1	Rule 2	Rule 30
Age	≥25 <35	≥25 <35	≥65
Health	Excellent	Excellent	Poor
Section of Country	East	West	West
Rate/1000	1.57	1.72	5.92
Policy Limit	200,000	200,000	20,000

Figure 2

The first decision rule (columns 1 and 2) can be paraphrased: If age is greater than or equal to 25 and less than 35, and health is excellent, and section of country is East, then rate per thousand is 1.57 and policy limit is 200,000. The underlined words are implied by the table layout. The other rules are alternatives to this one, so that logically, it does not matter which rule is examined first; only one rule can be satisfied in a single pass through this decision table.

As in most disciplines, a vocabulary is needed to describe the special properties and characteristics of decision tables. Fortunately, a glossary of terms for tabular form is already in existence from the statistical and financial fields; these supply an appropriate starting point.

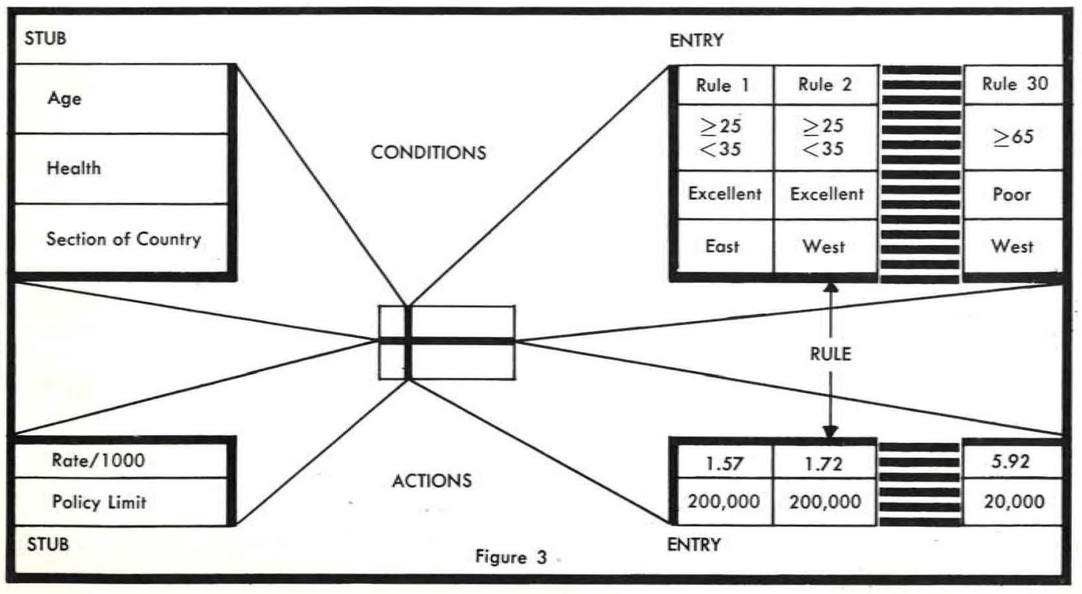


Figure 3

Using the information from the insurance example (Figure 3), the decision table is shown in an exploded view, Figure 3 to show recommended titles: (see preceding page).

The double lines serve as demarcation: CONDITIONS are shown above the horizontal double line, ACTIONS below; the STUB is to the left of the vertical double line, ENTRIES are to the right. Each vertical combination of conditions and actions is called a RULE. By adding to the elements shown a title section at the top of the table which is called a TABLE HEADER, and a RULE HEADER over the entries, the essential nomenclature is complete.

limited entry tables

LIMITED ENTRY tables offer a different approach to stating the decision logic. This type of table is shown in Figure 4:

	Credit Limit is OK	Pay Experience is Favorable	Special Clearance is Obtained	Approve Order	Return Order to Sales
Rule 1	Y			Y	
Rule 2	N	Y		Y	
Rule 3	N	N	Y	Y	
Rule 4	N	N	N		Y

Figure 4

The first rule (rows 1 and 2) is read: If credit limit is OK then approve order. Again, the underlined words are implied by the form. In limited entry tables the entire condition or action must be written in the stub; the entry is "limited" to reversing a condition or ignoring a condition or action. In contrast, extended entry tables have a part of the condition or action "extended" directly into the entry. While this decision table (Figure 4) is arranged quite differently, the same table elements are present. Structurally, the table appears as in Figure 5:

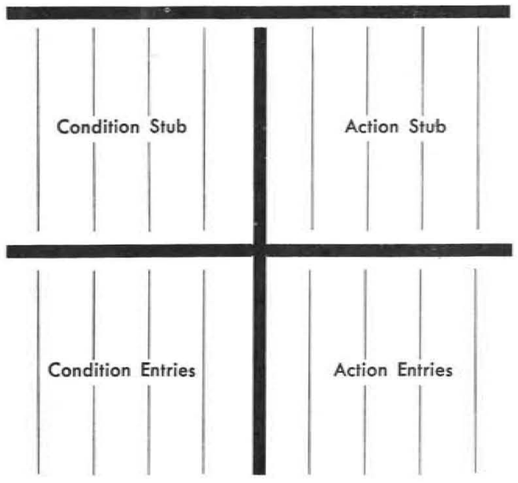


Figure 5

Limited entry permits only a few values in an entry:
Y = yes
N = no
Blank = not pertinent (e.g., condition or action need not be considered in the current rule)

business applications

Examples of successful applications of decision tables in business are as yet few in number, but some of the pioneering work can be reviewed briefly.

Initial work on the use of tabular form for recording decision logic was performed by General Electric's Integrated Systems Project from the fall of 1957 through 1959; during that period, I was the project leader. Many individuals were involved in this development work which concentrated on the use of tabular form to express the logic of product design, operation planning, cost determination, quality assurance planning, etc. This project developed extended entry decision tables for man-to-machine communication.

Mr. T. F. Kavanagh, in commenting on this work at the 1960 Eastern Joint Computer Conference,[1] noted, "the decision . . . table is a fundamental language concept . . . broadly applicable to many classes of information processing and decision making problems; . . . tables force a step-by-step analysis of the decision, . . . are easily understood by humans regardless of their functional background . . . (they are) simple and straightforward (enough) that . . . specialists can write tables . . . with very little training; . . . tables are easy to maintain (and) errors are reported at the source language level."

From late 1958 to the present time, Sutherland Company, a consulting firm in Peoria, Illinois, has been using tabular form for expressing what they call management decision rules. They have applied these techniques to a number of their clients' problems (e.g., a logistics study for Norton Air Force Base) with quite satisfactory results. In particular, they have used decision tables to record the logic for payroll, order processing, sales analysis, general ledger accounts, accounts payable, accounts receivable, and cost accounting. There has been no published material to date on the Sutherland work but available information indicates that limited entry decision tables are being used.

In 1959, Hunt Foods and Industries began experimenting with tabular form for man-to-man communication in computer systems planning. Material on this approach was the first to be released, in late 1959, describing how limited entry tables were used for systems analysis. Explorations were also carried out on complex relationships among individual decision using prior rule and sub-routine techniques. Many business systems were documented with decision tables: stock-control, credit analysis, sales analysis, and traffic.

In his report on the work at Hunt Foods, Mr. O. Y. Evans states, "The tabular approach . . . aids . . . in visualizing the numerous relationships and alternatives . . . (and) permits data rules to be readily reviewed for omissions and inconsistencies; . . . (in addition it) provides flexibility in changing any portion of the analysis."

Since early 1960, IBM has been actively engaged in exploring the value of tabular form both for systems analysis and for computer programming. The company has initiated joint projects with several customers to evaluate the effectiveness of various tabular forms, to explore alternative methods of implementation, and to investigate opportunities for incorporating these developments as an adjunct to existing languages. Since there are many different aspects of tabular form which still need to be examined, language implementing programs have not been prepared. These studies have developed and formalized mixed limited and extended entry tables, stubless tables, and unconditional decision tables.

The CODASYL Systems Group, which is part of the Development Committee of the Conference on Data Systems Languages, has been looking into the application and use of decision tables since late 1959. Their particular goal has been the creation of a systems-oriented language which would enable systems analysts to communicate their basic

decision logic either to computer programmers or to automatic program compilers. This organization contends that tabular form is one currently known technique which would aid in achieving effective mutual understanding of business decisions while maintaining machine independence. Their efforts have included research on generalizing tabular form to combine limited and extended entry format in a given table, as well as studies on more complex methods of sequence control, rule structure, and rule execution logic.

an example

To illustrate some of the possible advantages of decision tables, a composite tabular form is shown in Figure 6; these tables describe the logic of a file maintenance procedure. There are two input files (Detail and Master), each sequenced by identification number. The principal output is a similarly sequenced Master file incorporating additions and changes and omitting deleted records. The logic is based on having three internal areas: (1) Detail, (2) Master, and (3) New Master. "Read" as used here means "obtain the next record in the referenced file." "Write" means "produce an output Master record from the indicated source area." These are not detailed, precise tables for machine compilation, but rather the equivalent of a block diagram.

value of decision tables

So far, decision tables have been discussed in the light of known applications and attributed values and advantages. Though many current developments are still in the realm of "company confidential," several projects have indicated results that enable us to discuss the value of tables in concrete terms.

Recalling the three benefits mentioned previously, some studies claim that decision tables appear to be superior to other methods for representing complex decision logic in that they provide or encourage:

clarity and conciseness
completeness
meaningful relationships

To indicate the potential results from use of tabular form, the following statements paraphrase various user opinions:
Clarity and conciseness – Decision tables are easy to prepare, read, and teach to others; experience shows that nonprogrammers can learn to prepare satisfactory tables in less than a day. The amount of writing, or number of words, lines, and symbols used in describing complex decisions, is reduced by 25-50% as compared to flow charting. For certain specific cases, problem statement and programming time combined have been reduced significantly.
Completeness – Tabular form allows effective visual or desk debugging both by the analyst and the reviewer. There are fewer errors to start with since the analyst tends to catch his own mistakes; moreover, the reviewer will typically detect a high percentage of the remaining errors by visual examination. Finally, experience shows that with this foundation and suitable test problem construction, it is easy to rapidly detect the balance of the errors during machine debugging.
Meaningful relationships – Table structure serves to improve systems logic by aligning alternatives side by side. It also sharpens cause and effect understanding, so relationships which are accidental or incidental become clearer. Furthermore, actions based on similar or related conditions are apt to be drawn into the same table, making it easier to appreciate and consider dependent factors.

The evidence quoted on the advantages of decision tables for systems analysis and computer programming is based on actual study projects. Some of these studies even tested decision tables on various data processing machines. There are many current studies which are experimenting with a variety of tabular forms.

TABLE 001 — Update

Rule No.	01	02	03	04	05	06	07	08
Start	Y	N	N	N	N	N	N	ELSE
End of Detail		N	N	N	Y	Y	N	
End of Master		N	N	Y	N	Y	N	
Detail		<Master	=Master				>Master	
Detail an "Addition"		Y		Y				
Do Error Routine								X
Move Master to New Master			X					
Move Detail to New Master		X		X				
Set Addition Switch	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
Write Master					X		X	
Read Master	X				X		X	
Read Detail	X	X		X				X
GO TO TABLE	001	002	002	002	001	END	001	001

TABLE 002 — Change

Rule No.	01	02	03	04	05	06	07
Detail	<New Master	>New Master	>New Master	=New Master	=New Master	=New Master	ELSE
Addition Switch ON		Y	N		Y	N	
Detail a "Change"				Y			
Detail a "Delete"					Y	Y	
Write New Master		X	X				
Do Error Routine	X						X
Do Change Routine				X			
Do Delete Routine					X	X	
Read Master			X			X	
Read Detail	X			X	X	X	X
GO TO TABLE	002	001	001	002	001	001	002

Figure 6

future direction

With all its potential advantages, it is apparent that tabular form has not yet achieved full growth and stature; there are major technical and application areas still unprobed, awaiting only the touch of creativity to make practical breakthroughs. Current table methodology, for example, does not yet provide an effective systems-oriented language. Unable, then, to describe the decision logic in a systems-oriented language and untrained to an adequate degree in knowledge of equipment capabilities, the systems analyst often severely constrains the computer programmer.

What then of the future? Would it be desirable to directly incorporate tabular form into existing language processors such as Autocoder, FORTRAN, Commercial Translator, or COBOL, to describe complex decision procedures with decision tables? Would this approach significantly improve logical analysis? Would it simplify programming, debugging, and maintenance?

Would it be advantageous to try to create a systems-oriented language using tabluar form as a primary method for describing decision logic? Should we carefully consider the relative advantages of using interpretive rather than compiler techniques for applying tabular systems-oriented languages to computers?

We are witnessing a literal explosion in scientific technology, not the least of which is the rate of innovation in computer hardware. Laboratory shop-talk treats subjects like thin magnetic films, microminiaturization, and masers, as if they were accomplished facts; and before we realize it, they often are. Progress in language concepts, though, lags seriously behind hardware advances. Failure to keep pace can be attributed to several factors: inadequate effort, requirements for compatability with existing systems, and lack of problem recognition. Facing opportunities like automated product engineering and real-time control, we are handicapped by the limitations of current ways to describe business systems. Tabular form, one significant new tool for methods and systems people, may help to accelerate business language development and to advance systems technology.

BIBLIOGRAPHY

(1) Kavanagh, Thomas F., "TABSOL—A Fundamental Concept for Systems Oriented Languages," Proceedings of the 1960 Eastern Joint Computer Conference.

(2) Evans, Orren Y., "Advanced Analysis Method for Integrated Electronic Data Processing," IBM General Information Manual, #F20-8047.