TABLES SIGNAL BETTER COMMUNICATION

By Burton Grad, Manager
Systems Engineering Services
IBM

TABLES SIGNAL BETTER COMMUNICATI ON

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 occurrences 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 a different machine at its 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, 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 which will 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 accept 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.

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, <u>not</u> that they can describe systems that cannot also be described in other ways.

Given the state name, determine the name of the capital.

Fig. 1

STATE	Alabama	Alaska		Wyoming
CAPITAL	Montgomery	Juneau		Cheyenne

In this example State appears above the heavy 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.

Fig. 2

Health / Age	EXCELLENT	GOOD	FAIR	POOR
≧25 <35	1.27	1.52	1.98	2.73
≧35 <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

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 shows a table with four independent and three dependent factors where clarity, interrelationship and comprehensiveness have been maintained.

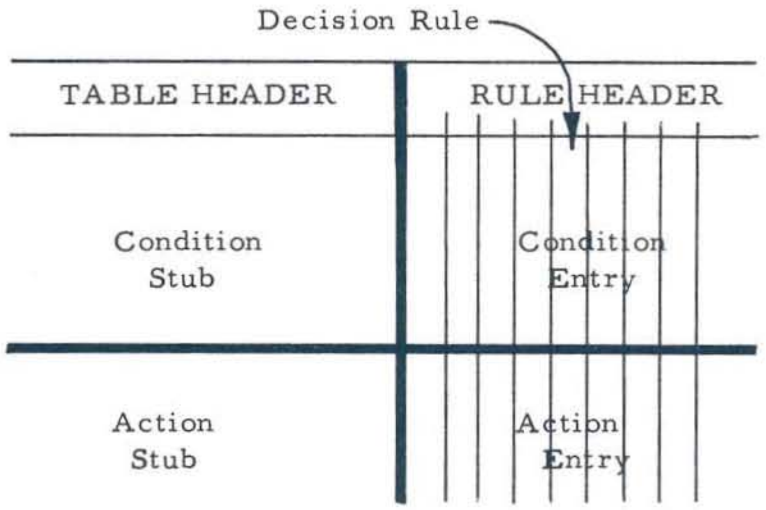
Fig. 3.

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		9.82
Policy Limit	200,000	100,000		10,000
Type of Policy	A	B		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. 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.

Fig. 4



The heavy lines serve as demarcation: CONDITIONS are shown above the heavy horizontal line, ACTIONS below. The STUB is to the left of the heavy vertical 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.

Fig. 5.

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 in terms of 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.

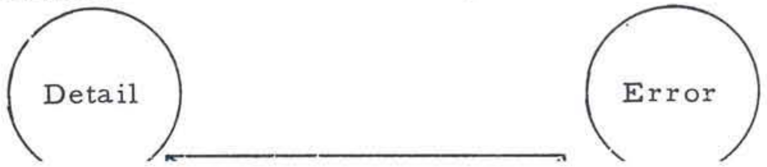
While I was project leader for General Electric's Integrated Systems Project, the potential application of tables to a wide variety of problems was explored including its use for product design, operation planning, cost determination, factory scheduling, etc. The results certainly revealed the opportunity 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 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. Evans' 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 demonstrated 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 benefited 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.

Fig. 6.



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 is one of two tables prepared to perform this job.

Fig. 7.

	Rule #							
TABLE: Update	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 vs. Master					<	>	=	
Detail is 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	Up-date	End	Up-date	Chg.	Chg.	Up-date	Chg.	Up-date

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 [illegible]

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; control returns to the beginning of the Update Table. 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;

Fig. 8

TABLE: Update	RULES							
	1	2	3	4	5	6	7	8
Detail vs. Master	>	=	<					Else
Detail is an "addition"			Y		Y			
End of Detail	N	N	N	Y	N		Y	
End of Master	N	N	N	N	Y		Y	
Start	N	N	N	N	N	Y	N	
Do Error Routine								x
Set Addition Switch On			x		x			
Move Detail to New Master			x		x			
Move Master to New Master		x						
Write Master	x			x				
Read Master	x			x		x		
Read Detail			x		x	x		x
Go to Table	Up-date	Chg.	Chg.	Up-date	Chg.	Up-date	End	Up-date

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 (Figure detail and end of master. It seems evident that the comparison of detail to master would be the most discriminating criterion 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 narrative following it. 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 (decision) 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 (including decision tables) will

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 that 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 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.

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 touch of creativity to make practical breakthroughs. While current table methodology does not yet provide the 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

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.