ones who would put various interpretations on data and thus support various solutions with the same data, on a second dimension. Witkin has defined individuals as field independents who have the ability to deliniate and structure a given whole, or as field dependents who show a global point of view and for whom the organization of a field as a whole dictates the way its parts are experienced. Witkin has experimentally shown that field independents have more analytical capability than the field dependents.

The findings of the experimental studies on the impact of cognitive styles coupled with the differences hypothesized in the theoretical models would tend to support Mason and Mitroff's hypothesis that the cognitive style of the decision maker should be considered in DSS design. Broad generalizations on the impact of cognitive styles are hard to derive due to the limited number of experimental results and the various test instruments used to measure cognitive styles.

Nevertheless, some suggestions on DSS can be summarized, always keeping in mind that further research is still necessary. Analytic decision makers prefer to work with decision aids and are more likely to accept reports which are supported by mathematical or quantitative arguments. Heuristics like to rely more on the historical reporting system; they want to analyze more information than analytics, thus preferring disaggregated information. Heuristics should be allowed to search and simulate for a variety of alternate solutions. The kind of decision aid which would be suitable for heuristics is one where the user should be able to create an arbitrary order of processing to develop an approach as feedback from the environment as obtained, should be able to explore scenarios to generate cues or test trial solutions, should be able to shift between levels of detail and generality and should be able to control the format of the output and have means for flexible inputs. [Botkin (1973)].

With respect to data collection and usage, preceptive individuals should be given aids to summarize and filter data, such as exception reports and routines for determining patterns of given attributes over time. Receptive and maximal data user types should have a data-retrieval system with flexible and timely search capabilities to large data bases.

How do the systems designers go about implementing these suggestions? Should they (a) create a flexible system and let decision makers choose the type of model and format of reports which they think will best fit their needs, (b) ask the manager what models and report formats he wants and only provide him with those or (c) give each decision maker only the models and reports which best fit the decision maker's needs based on a psychological (cognitive style) test? There is no simple answer.

Option (c) which I prefer is supported by Ackoff's (1967) contention that the manager does not know the information he needs. To implement this option, the designer would keep a cognitive style profile on each decision maker based on an initial test and design all future systems to support the decision maker according to this profile. To effectively implement option (c), we need more studies to clarify the impact of cognitive styles and provide the designer a pool of knowledge to work with.

Option (a) lets the decision maker decide, but this may involve extra time and unnecessary searches on the part of the decision maker. Option (b) assumes the manager will be able to choose and know what he wants. The questions on implementation, together with the many unresearched topics in cognitive styles will, we hope, interest DSS researchers in more studies to clarify this promising area which could potentially improve DSS design.

References

1. Ackoff, R.L., "Management Misinformation Systems," *Management Science,* Vol. 14 (1967), pp. 147–156.
2. Botkin, J.W., "An Intuitive Computer System: A Cognitive Approach to the Management Learning Process," unpublished doctoral dissertation, Harvard Business School (1973).
3. Driver, Michael and Theodore Mock, "Human Information Processing, Decision Style Theory and Accounting Information Systems," *The Accounting Review,* Vol. L, No. 3 (1975).
4. Huysmans, J.H.B.M., *The Implementation of Operations Research,* New York: Wiley-Interscience (1970).
5. Keen, P.W.G., "The Implications of Cognitive Style for Individual Decision Making," D.B.A. Thesis, Harvard University (1973).
6. Mason, R.O. and I.I. Mitroff, "A Program for Research on Management Information Systems," *Management Science,* Vol. 19, No. 5 (1973), pp. 475–487.
7. Schroeder, H.M., N.J. Driver and S. Streufert, *Human Information Processing,* New York: Holt, Rinehart and Winston, Inc. (1967).
8. Witkin, H.A.,*The Role of Cognitive Style in Academic Performance and in Teacher-Student Relations,* (1972).

A STATISTICAL EVALUATION OF TACTICS USAGE

By Edward Gainer
Department of Management
Michigan State University
&
Edward Kimball
and Alfred Maley
COMSHARE, Inc.
&
Alan Kortesoja
Manufacturing Data Systems, Inc.

TACTICS is a commercial decision support system that includes extensive data manipulation facilities servicing a wide variety of multivariate model-building techniques. Business and government users have employed the system's statistical modeling tools in such diverse areas as federal government revenue sharing, antitrust litigation support, salary administration and equal opportunity reporting.

TACTICS, like all software systems, is imperfect. Unlike many systems however, TACTICS contains several facilities which allow us to measure its ease of use, efficiency and reliability. The most effective of these is a logging file maintained by the controlling program. An analysis of the logging records for over 100,000 commands has yielded significant information about good syntax, programming errors and user behavior.

TACTICS consists of a controller which starts execution of a program module after recognizing a command. Each module executes one command and returns. The controller logs the outcome as successful completion, attention interrupt (break) or error termination.

A successful completion means that no error was detected; the overall success rate for all TACTICS commands was 85.5 percent. The error terminations were classified as syntax, semantic or programming errors. Since the controller logs the actual command in the case of an error termination, it was possible to examine the failures to take corrective action.

The success rate suggests that the TACTICS syntax is easy to use. The syntax is partly positional, partly keyword-oriented. Each command consists of an imperative followed by a list of variables, if appropriate. Elements of a list are separated by commas. Keywords may then follow, in any order. The use of a syntax analyzer and an automatic parsing program left us free to design reasonably intuitive constructs. Typical commands might be:

```
READ SALES, PROFITS OBSERVATIONS 1-20
FORMAT (2F10.0) ENCODE LATITUDE INTO
REGION RANGES LT30, 30 TO 60, GE 60
CROSSTAB RACE, SEX WEIGHT SALARY OMIT
FREQUENCY TRANSFORM Y = 2*SIN(X) + 14.3
```

Commands which originally had syntax error rates as high as 50 percent were scrapped or revised; troublesome keywords were made optional or dropped. As an example, the complex CUTS function in TRANSFORM was broken out as the ENCODE command using keywords to preface each argument. Noise punctuation as in:

```
OBSERVATIONS: 1—20
SIGNIFICANCE =.95
```

was found to cause a surprising number of problems; its use was dropped wherever a blank would suffice.

Generally, simpler commands with no punctuation were more successful (87 to 92 percent) than complex commands. Input/output commands were the least successful (68 to 75 percent). The most heavily used command, TRANSFORM, was 95 percent successful, in spite of a syntax which allows nearly any FORTRAN expression. Simple expressions are apparently the rule.

The operating system and the FORTRAN library routines detect some errors such as an illegal memory address, division by zero and so on. These automatically return control to the controller for appropriate action. To prevent overwriting an array bound, TACTICS uses a variable storage allocator which places a check sum by each area it dispenses to a module. The storage allocator also examines the last area allocated to see that its check sum is intact. When a module terminates, all check sums are examined. This "idiot check" has discovered the lion's share of the programming errors present in TACTICS. Lastly, when any module finds itself at a section of code

TACTICS Command Statistics

Module	*Total Number of Commands*	*Percentage Successful*
TRANSFORM	15338	95
PRINT	9261	67
REG/MULTIPLE	8873	90
READ	7799	75
NWAY/CROSS	7019	93
ASSIGN	5894	91
GRAPH	5879	83
FETCH	4414	87
REG/REG	4310	76
ANALYZE	3547	91
SELECT	3520	88
REG/STEPWISE	3292	76
MAKE	2626	93
CODE	2252	91
UPDATE	2193	79
IDENTIFY	1931	86
C7READ	1759	68
REG/DISPLAY	1720	88
REG/PREDICT	1628	88
CORRELATE	1620	87
REG/STORE	1524	89
RESET	1492	97
PARTITION	1481	79
TITLE	1416	98
HYPO/ANOVA	1246	89
NWAY/NWAY	1171	81
ENCODE	795	83
RELEASE	734	91
HYPO/HYPO	727	90
REG/MCR	714	80
HISTOGRAM	706	83
MISSING	614	87
FIT	587	93
EXHIBIT	391	75
C7WRITE	312	76
MPUNCH	292	91
HYPO/TEST	288	68
GET	273	90
NWAY/STORE	268	98
CLA/CLUSTER	199	56
ORDER	186	82
MFA/FACTOR	164	62
CLA/CLA	144	75
MFA/MFA	110	68
RANK	110	78
NWAY/DISPLAY	29	59
MFA/DISPLAY	17	41
HYPO/DISPLAY	15	33
MFA/ROTATE	15	53
MFA/STORE	4	100

Table 1

that it should never execute, it makes a call to a routine DISABLE, which bails out in a graceful fashion. All these errors cause the controller to print the message:

```
FATAL ERROR IN MODULE
MODULE DISABLED
```

and to refuse to allow the user further access to that module, a severe measure designed to prevent incorrect answers and to insure that the problem gets reported.

Surprisingly, only 18 percent of the disablings were ever communicated to the system designers. The logging file was the only record that most of these failures occurred. Later contacts convinced us that the typical user feels the error is somehow his fault. He doesn't wish to report the problem lest it reflect upon him or his understanding of the documentation, the syntax or the analysis technique itself. He suffers from a basic belief that the computer is more likely to be correct than he is.

The logging file was also useful for determining the average cost of each command and the most frequently used techniques. In one instance a command was found to be very costly for certain data sets due to a paging problem. This was corrected by rearranging the working set. Frequency of use was a guide to making efficiency improvements and installing new features. For example, improvements in the efficiency of TRANSFORM yielded benefits for all users. Regression, cross-tabulation and analysis of variance occupied the bulk of the usage.

Conclusion

A logging file combined with error detection tools is an important means of measuring the success of a DSS, as well as maintaining and improving it. Users cannot be relied upon to perform this function. Successful results with TACTICS suggest the importance of using the simplest feasible syntax.

AN APPROACH TO AN ADMINISTRATIVE DSS

By M. Loomis, J.F. Nunamaker and B.R. Konsynski
Management Information Systems
University of Arizona

Implementing a decision support system which builds on existing data files in an established environment can be effectively accomplished by a hybrid "bottom-up with top-down control" methodology. The existing data are the basis for the bottom-up aspect of the approach, which addresses implementation; the users' ultimate requirements are the basis for the top-down aspect of the approach, which addresses design and system structure. This approach contrasts both with a pure top-down approach requiring the establishment of new data structures and with a strictly bottom-up approach which creates coordination problems.

The authors have successfully applied the methodology in development and implementation of an integrated decision support system providing information for operational and budget planning at the University of Arizona.

In the bottom-up with top-down control approach, the decision support system development project commences with iteratively refined specification of outputs, processing and required inputs. Users and analysts meet to establish system objectives and user requirements, including information needs, data sources, integrity and validation specifications. The major subsystems of the decision support system are identified.

With top-down control factors established by a complete and consistent statement of logical requirements, the analysts can proceed with bottom-up elements of the implementation, resulting in an interim system consistent with the logical design resulting from the top-down control effort and directly interfacing with existing data and operating systems. The decision support system can thus become operational with minimal effort, yet in the longer term the system can evolve according to the top-down design structure.

Systems to support policy planning and decision making by higher management in an organization are fundamentally different from systems to support daily operations, transaction processing and computer hardware. Success of a planning system is generally much more dependent upon human factors in the organization. One implication of the bottom-up with top-down control methodology is that an organization should develop its own decision support system, rather than purchasing a system developed for other institutions.

Concerning aspects of the methodology, the approach calls first for the active and informed participation in the design process by the eventual users of the system. Thus pitfalls of applying inappropriate assumptions and implications from another environment can be avoided. An information system that successfully serves the needs of a complex and socially interactive policy planning system must be tuned to the decision-making patterns of the participants. Participation of users in the design effort results not only in an improved design, but can materially reduce the education and selling effort required for acceptance of the system.

Second, all current files that could impact or relate to the planning system are analyzed. Thus a comprehensive file structure supporting the entire information and planning system and oriented toward an integrated data base can result. In general, implementation of an externally created system would introduce a layer of files in addition to the existing ones, potentially complicating problems of compatibility and extension.

Third, the major problem of data quality is directly addressed in the advocated methodology; this problem is more difficult to resolve during implementation of an imported system. Any system will fail if the data supplied to it