DECISION SUPPORT SYSTEMS:

A RESEARCH PERSPECTIVE

Peter G. W. Keen

March 1980

CISR No. 54
Sloan WP No. 1117-80

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

DECISION SUPPORT SYSTEMS:

A RESEARCH PERSPECTIVE

Peter G. W. Keen

March 1980

CISR No. 54
Sloan WP No. 1117-80

Many of the ideas expressed in this paper belong as much to my colleagues at the Wharton School, 1978-79, as to myself. In particular, the concepts of task representation were developed by Gerry Hurst, Dick Hackathorn and myself, and extended through discussions with John Henderson and Tom Gambino. The research framework owes much to a seminar on DSS run by Dick Hackathorn.

CONTENTS

1. Introduction

Decision Support Systems (DSS) represent a concept of the role of computers within the decision making process. The term has become a rallying cry for researchers, practicioners and managers concerned that Management Science and the Management Information Systems fields have become unnecessarily narrow in focus. As with many rallying cries, the term is not well-defined. For some writers, DSS simply mean interactive systems for use by managers. For others, the key issue is support, rather than system. They focus on understanding and improving the decision process; a DSS is then designed using any available and suitable technology. Some researchers view DSS as a subfield of MIS, while others regard it as an extension of Management Science techniques. The former define Decision Support as providing managers with access to data and the latter as giving them access to analytic models.

Research on DSS gained momentum around 1974. Only within the past two years has it reached a critical mass and expanded beyond a fairly narrow circle. By 1979, almost thirty fairly detailed case studies of DSS had been published. As the concept has become fashionable it has been used in looser and looser ways. Last year's article on "interactive marketing models" is cut-and-pasted and resubmitted with "decision support systems" sno-paked in instead. It may well be that DSS are more important as a liberating rallying cry than as a theoretical concept. However, the published case studies and conceptual proposals imply a coherent framework that makes DSS a meaningful discipline for both research and practice.

This paper presents a formal definition of DSS. It aims at answering two key questions:

(1) Is the term really necessary?

(2) If so, what are the research issues it implies?

The key argument is that the term DSS is relevant to situations where a "final" system can be developed only through an adaptive process of learning and evolution. The design strategy must then focus on getting finished; this is very different from the Management Science and Data Processing approaches. The research issues for DSS center around adaptation and evolution; they include managerial learning, representation of tasks and user behavior, design architecture and strategies for getting started.

2. Definitions of DSS

Most work on DSS adopts one of the following conceptions, even if only implicitly:

(1) A DSS is defined in terms of the structure of the task it addresses.

(2) DSS require a distinctive design strategy based on evolution and "middle-out" techniques.

(3) DSS support the cognitive processes of individual decision makers; decision research provides descriptive insights into management problem-solving and normative theories for defining how to improve its effectiveness.

(4) DSS reflect an implementation strategy for making computers useful to managers; this strategy is based on the use of skilled intermediates, responsive service and "humanized" software interfaces.

None of these conceptions necessarily implies interactive computer systems; a DSS is defined in terms of context and use. There is no technical conception for which one cannot readily generate counterexamples. For instance, the design architecture, mode of use and available functions of an airline reservation system are virtually the same as those in many data-based "DSS". If a given DSS is identical to, say, a standard interactive model, there seems no value whatsoever in using a new label. DSS become a meaningful research topic only if the term can be shown to be a necessary concept. In the pragmatic context of information systems development and analytic techniques, calling a system a DSS must lead to some actions, by the designers or users, that would not have occurred otherwise; the actions should contribute to the effective development of the system or its effective use.

A potential strength of the DSS movement has been that it has at least tried to link theory and practice. It describes real systems used in real organizations by real problem-solvers, not experiments involving captive students. At the same time, since it explicitly argues that DSS are different from traditional systems, the better empirical work addresses conceptual issues, if only assertively. The available studies of DSS thus often provide illustrations, extensions or counterexamples that can be used to test and extend their authors' conceptual assumptions.

3. Case-Based Studies of DSS

This is not a survey paper, but many of the ideas expressed in it come from a detailed analysis of 30 articles or chapters in books that describe particular "DSS" in detail.[1] (Appendix 1 provides the necessary references.) Some clear and general conclusions can be drawn from the studies:

(1) The actual uses of the DSS are almost invariably different from the intended ones; indeed, many of the most valued and innovative uses could not have been predicted when the system was designed.

(2) Usage is personalized; whether a system is recently operational or has been in place for some time, there are wide variations among individuals in how they use its functions.

(3) DSS evolve; case studies frequently state that key factors explaining successful development are a flexible design architecture that permits fast modification and extension and a phased approach to implementation.

(4) The functions DSS provide are generally not elaborate; complex systems are evolved from simple components.

(5) While the orthodox (academic) faith views DSS as tools for individual decision makers,[2] users regard the concept as more relevant to

systems that support organizational processes. They also feel they do not really use DSS for decision making.

(6) Major benefits identified by users are flexibility, improved communications (of, for example, the logic of an analysis), insight and learning.

(7) DSS are frequently used by managers through intermediaries and chauffeurs; while an interactive computer system is essential for ease of access, there is little interactive problem-solving.

Examples of all these points are shown in Appendix 2. They add up to a fairly clear picture of DSS development that differs from the orthodox faith in important details. In the first place, they suggest that the term Decision Support System is too broad and the cognitive focus of much of the research too narrow. Keen and Hackathorn argue that a distinction should be made between

(a) Personal Support Systems (PSS), for use by individuals in tasks which involve no interdependencies, so that the user can indeed make a decision;

(b) Group Support Systems (GSS), for tasks with "pooled" interdependencies which thus require substantial face-to-face discussion and communication;

(c) Organizational Support Systems (OSS), for tasks involving "sequential" interdependencies.

A PSS may thus support a manager's own budget decision, a GSS support the budget negotiation, and an OSS support the organizational budget process.

Several writers have been uneasy with the D in DSS. It largely reflects the cognitive focus -- even bias -- in the early DSS research, which draws on Simon's theories of individual decision making and concepts of cognitive style and cognitive complexity. Organizational Support Systems far outnumber PSS in the published case studies and require a very different theoretical base, which is so far lacking.

4. Middle-Out Design

The studies strongly support the concept of middle-out design for DSS.[3] Almost all the descriptions of DSS implementation highlight careful use of prototypes, continued incremental development, and response to users' changing demands. Writers such as Ness, Courbon, Grajew and Tolovi, and Berger and Edelman make a strong implicit case for viewing X Support Systems (where X may stand for Decision, Management, Personal, Organizational, Interactive, or whatever) as an adaptive design strategy.

The obvious question is: Is the strategy a general one for interactive systems or needed only for particular situations? Middle-out design differs most from traditional techniques in that it explicitly proceeds without functional specifications. Data Processing (DP) has learned, through vicarious trial-and-error learning and occasional reflection, that systems development requires planning before programming. Brooks' brilliant and somewhat rueful review of software engineering, The Mythical Man-Month, established that coding is only 10% of the total effort in the system's development life cycle. Standard textbooks generally recommend that around 40% of the effort go to analysis and specifications, 10% to coding, 30% to testing, and 20% to installation (and another 100% - 300% to maintenance). The vocabulary of DP is full of terms like "signing-off", "functional specifications" and making a system "operational".

The DSS case studies, including those in which the design strategy was not based on middle-out, contradict the recommendations underlying the systems development life cycle. This clearly implies that defining a system as a DSS, rather than, say, an interactive information retrieval system, does make a difference. It shifts the development process from a focus on delaying coding to getting going on it as fast as possible, from aiming towards a clearly-defined "final" system to implementing an initial one that can then be firmed-up, modified and evolved. The systems development life cycle is a strategy for getting finished; adaptive design (this term captures all the middle-out, incremental and evolutionary techniques scattered throughout the case studies) is a method for getting started.

5. "Semi-Structured" Tasks

Viewing DSS in terms of the design process is not enough to integrate all the conclusions from the case studies. It also sidesteps key conceptual issues raised by the decision research and task-centered conceptions of DSS. Gorry and Scott Morton's A Framework for

Management Information Systems (1971) was a seminal paper for DSS. It built on Simon's concept of programmed and non-programmed tasks and identified "semi-structured" tasks as those requiring special treatment. Structured tasks can be automated or routinized, thus replacing judgment, while unstructured ones entirely involve judgment and defy computerization. Semi-structured tasks permit a synthesis of human judgment and the computer's capabilities.

There are several problems with this argument. The terms "structured" and "unstructured" point to a spectrum of tasks, but there is no real operationalization of "semi-structured". More importantly, it is unclear if structure is perceptual or intrinsic to the task. Stabell also points out that organizations often decide to treat an unstructured task as if it were structured; the degree of structure is then socially defined, as well as perceptual.

The Gorry-Morton framework is not a complete or convincing theoretical statement. The range of applications, technologies and mode of use of the DSS described in the case studies are too broad to fit into it. (This applies also to Gorry and Morton's use of Anthony's distinction between strategic planning, management control and operational control. Morton (1971) suggests that DSS apply to the first two areas, but Berger and Edelman give striking examples of a DSS for operational control.)

Despite the looseness of its definition and the lack of comprehensive supporting evidence in the case studies, Gorry and Morton's notion of semi-structured tasks is intuitively convincing. Keen and Scott Morton rely on it in explaining the concept of support, rather than replacement, of managerial judgment. Any effort to define how a DSS helps improve effectiveness in decision making, and not just efficiency, has to introduce some similar notion of the relationship between task structure and process (Stabell, Carlson and Sutton).

6. DSS Redefined

A central argument of this paper is that what Gorry and Morton present, and Gerrity, Morton, Stabell, and Keen and Morton later extend, is not the general case but a special one. The following definition of Support Systems meshes the task-centered perspective into that of adaptive design and also picks up on the most interesting finding from the case studies, the unpredictability of DSS usage:

> The label "Support System" is meaningful only in situations where the "final" system must emerge through an adaptive process of design and usage.
>
> This process may be needed for a variety of reasons:
>
> (1) The designer or user cannot provide functional specifications or is unwilling to do so. A "semi-structured" task is such an instance; we either lack the necessary knowledge to lay out procedures and requirements (i.e., the degree of structure is perceptual) or feel that such a statement can never be made (i.e., the lack of structure is instrinsic to the task).
>
> (2) Users do not know what they want and the de-

signers do not understand what they need or can accept; an initial system must be built to give users something concrete to react to (this is the assumption underlying middle-out).

(3) Users' concepts of the task or decision situation will be shaped by the DSS. The system stimulates learning and new insights, which in turn stimulate new uses and the need for new functions in the system. The unpredictability of DSS usage surely reflects this learning, which can be exploited only if the DSS evolves in response to it.

(4) Intended users of the system have sufficient autonomy to handle the task in a variety of ways, or differ in the way they think to a degree that prevents standardization. In this situation, any computer support must allow personalized usage and be flexible.

While (3) states that the DSS shapes the user, (4) equally suggests that the user shapes the DSS.

This conception makes DSS a necessary concept. For any given system development effort, it makes a great deal of difference whether or not the implementers view it as requiring a DSS as opposed to a marketing model, retrieval system, report generator, etc. It would be a severe mistake to rely on traditional development techniques if the final system will evolve only through the ongoing interaction of designer and user, learning, personalized use, or the evolution of new functions. Learning, adaptation and evolution are made feasible by building a "DSS" and not a "model". If these are not needed for effective development and use of a system, then one should build it as a "model" in the traditional way and the new label is not relevant.

This definition of DSS in terms of adaptive design and use provides a base for a research framework that is consistent with the empirical findings of the case studies and that integrates the conceptual issues they raise or reflect. There seem to be three overall issues for a theory of DSS:

(1) understanding the dynamics of the adaptive relationship between user, designer and technical system;

(2) analyzing tasks in relation to users' processes and criteria for system design;

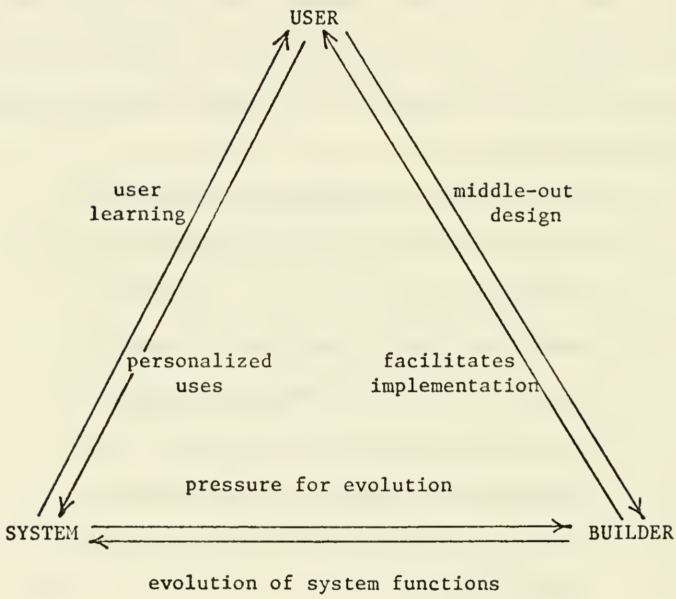
(3) developing an organizational focus to complement the cognitive perspective and thus include Organizational as well as Personal Support Systems.

7. Adaptive Development and Use

Figure 1 shows the adaptive links between the major actors involved in any DSS development and the technical system. The arrows represent a direction of influence. For example, SYSTEM ⟶ USER

FIGURE 1

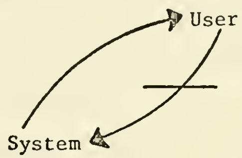
An Adaptive Framework for DSS



indicates that learning is stimulated by the DSS while USER→SYSTEM refers to the personalized, differentiated mode of use that evolves. The two adaptive processes work together: an effective DSS encourages the user to explore new alternatives and approaches to the task (S→U). This in itself stimulates new uses of the system, often unanticipated and idiosyncratic (U→S).

The arrows are not merely a convenient schematic. They help clarify whether a particular system should be called a DSS. For example, an airline reservation system is technically similar to many retrieval-based DSS. However, it is not intended to stimulate learning (S↛U), nor are there personalized modes of usage; there is a "right" way to operate the system and the user must adjust to it, not vice versa (U↛S). Similarly, an interactive planning model that is used to assess a predetermined range of alternatives is a system for solutions, not for learning. It need not be flexible and adapt to the user (U↛S).

The arrows also represent requirements for successful DSS development and use. For example, if the system forces users to follow a fixed set of procedures, learning cannot be exploited:



In effect the DSS contains its own obsolescence. It stimulates new approaches which it in turn inhibits.

The definition of DSS as applicable in situations where the final system must evolve from adaptive development and use thus implies:

(1) A system is a "DSS" only if each of the arrows is relevant to the situation.

(2) Where they are relevant, the design process must ensure they are not blocked by inflexible design structures, failure to allocate resources for implementing new functions, or lack of a direct relationship between user and designer.

(3) Each arrow represents a distinctive aspect of research and practice.

Figure 1 ignores the context of the DSS development process, especially the task to be supported and the wider organization. Before expanding it, however, it seems useful to discuss each adaptive link in relation to DSS research. There are three loops:

S ⇄ U , U ⇄ B , and S ⇄ B.

7.1 <u>The System-User Link</u>

S ⇄ U: this, in the context of Personal Support Systems, may be termed the <u>cognitive loop</u>. (The issue of organizational support will be discussed separately.) The link S → U concerns managerial learning and U → S the individuals' exploitation of the DSS capabilities and/or their own learning. The cognitive loop helps explain the consistent finding in the case studies that individuals use a given DSS in very different ways and that uses are so often unintended and unpredicted. This seems a natural outcome of the sequence $S \rightarrow U \rightarrow S \rightarrow U \ldots$

Much early DSS research explored aspects of the cognitive loop, particularly characteristics of individual problem-solving that influence the use of a DSS. This was a fairly static analysis and it seems essential to examine the managerial (and organizational) learning process in more detail. Doing so requires richer theoretical models; the early research drew on limited concepts of cognitive style and cognitive structure, that were at too high a level of analysis to track the learning process. They focussed on general aspects of the psychology of individual differences (Stabell, Carlisle, Grochow).

7.2 The User-Builder Link

The link $U \rightleftarrows B$ is the implementation loop.

(a) $U \rightarrow B$ highlights a key aspect of adaptive design, discussed by Ness and Courbon, *et al*. Ackoff long ago pointed out that users do not know what they need. The middle-out approach relies on the quick delivery of an initial system to which users can respond and thus clarify what they really want. Middle-out design is the means by which the *designer* learns from the *user*; it also ensures that the user drives the design process.

(b) $B \rightarrow U$: this link has been explored in

> studies of DSS implementation that examine the role of the "integrating agent" (Bennett), intermediary (Keen), chauffeur (Grace), and change agent (Ginzberg). DSS are a service rather than a product and require that the designer understand the users' perspective and processes, build credibility and be responsive to their evolving needs.

The implementation loop is both well-researched and well understood. The empirical work of Courbon, Grajew and Tolovi is an exhaustive and precise test of the concepts of adaptive design. The more diffuse discussions of implementation are less operational (Bennett, Keen, Ginzberg and Scott Morton).

7.3 The System-Builders Link

This evolution loop ($S \rightleftarrows B$) is less easy to label than the others. While the case studies show again and again that DSS evolve and much of the conceptual work relevant to DSS recommends evolutionary development (Urban), there are few detailed, longitudinal studies or theoretical models. It is perhaps easiest to view the links in relation to the other loops. Managerial learning ($S \rightarrow U$) and personalized uses ($U \rightarrow S$) put strain on the existing system. This builds pressure for evolution ($S \rightarrow B$). New functions are then provided ($B \rightarrow S$). The case studies imply that this is not a continued, evenly-paced process, but occurs in discrete phases (see also Andreoli and Steadman). Users explore the initial system for a while and gradually become confident with it. At a certain point, it becomes apparent that a new function needs to be added to the system. Quite often, usage does not really take off until this extension is provided; the "new" system leads to very different volumes and modes of use than the earlier one (Andreoli and Steadman).

The S $\rightarrow$ B link needs research. Keen and Gambino have employed the common device of a data trap to track individuals' use of a DSS[4] (see also Stabell, Andreoli and Steadman), in terms of emerging patterns and "command sequences". The argument is that users initially use the commands of the DSS as single words (e.g., 'LIST', 'REGRESS'), but later develop, largely via the adaptive processes of the cognitive loop, what are effectively sentences; they use consistent sequences of commands and build up their own analytic routines. This process is easy to identify; the hypothesis is that it triggers demand for or readiness to use new commands.

The other link, B $\rightarrow$ S, is easier to explain. It simply involves the designer adding new capabilities to the DSS. This obviously is feasible only

(a) if the design architecture is modular, flexible and easily modified;

(b) the programmer can implement new functions cheaply and quickly;

(c) the designer maintains ongoing contact with the users.

The advocates of APL as "the" language for DSS (Contreras, Keen), of end-user languages (Keen and Wagner), and "command-driven" interfaces all emphasize the need for program structures and programming methods to facilitate evolution. The case studies indicate that the success of a DSS often depends on its evolution rather than its initial use, and on fast, responsive implementation.

Discussions of DSS evolution focus on new functions and commands. There is relatively little exploration of the evolution of data and data structures.[5] Model-based DSS seem both easier to build and evolve than do data-based ones. DSS research currently lacks a focus on handling data management issues.

7.4 Summary

There is not room here to discuss each adaptive link in Figure 1 in any detail. The preceding summary covers only a few issues relevant to research. Hopefully, Figure 1 constitutes a definition of DSS development that clarifies what a DSS is and is not, and what actions and processes it involves. Each arrow represents a clear research area relevant to DSS practice.

8. The Task Context

Figure 1 ignores the task to be supported. Obviously, a DSS can be built only if the designer understands the task at a level of detail sufficient to:

(a) relate the task to the users' processes;

(b) design the functions of the DSS;

(c) by extension, relate the users' processes to the DSS functions.

At present, methodologies for describing tasks, user processes and system functions are at too high a level to integrate the three components.[6] For example, one may classify an individual in terms of cognitive style (e.g., intuitive versus systematic), classify the task as semi-structured and the system as an interactive retrieval system. This provides no link between task characteristics and detailed design criteria and user behavior. DSS research needs to find a level and method of representing tasks that permit this link. Such a method does not yet exist. Hackathorn's and Meldman's use of network models comes closest, but is not intended as a general methodology for DSS.

The ideas presented below require a major research effort[7] before they can be validated and made operational. In a way, they pick up on Gorry and Morton's discussion of "semi-structured" tasks, at a more molecular level:

(1) The tasks a DSS addresses involve some degree of discretion, inference and selection of information; if this is not so, there are no adaptive links between user and system $(U \rightleftarrows S)$. A whole task is composed of subtasks. The whole task may be the university admissions decision, portfolio management, media selection, etc., etc. The subtasks are discrete intellectual operations, such as

calculating a sum,

searching for a value,

or comparing two variables on a graph.

(2) The subtasks identify the potential functions for the DSS, e.g., CALC, FIND, COMPARE.

(3) User behavior and user learning can be described in terms of the sequence of and change in subtasks.

(4) Use of the DSS can be tracked in relation to the functions.

This level of representation has several practical and conceptual merits. It also suggests that DSS should be command-driven. Keen and Alter argue that the commands correspond to the users' verbs (e.g., 'list', 'graph'). Keen adds that if a function in a system does not relate directly to some concept in the users' mind, it really cannot be used. Carrying out a task involves a sequence of verb-related subtasks (Do this...then this...). Using a DSS involves invoking a sequence of verb-based commands. Evolving it means adding new commands. Learning is identifiable only in terms of use of new commands or redefinition of existing ones, and personalized use is apparent from the choice of commands.

In a structured task, the subtasks can be clearly specified and the sequence in which they are invoked predicted. A "semi-structured" task is thus one where either not all the subtasks can be represented and hence translated into DSS commands or they can be represented but the sequence not predicted. Focussing on subtasks, rather than whole tasks, retains the intuitive appeal of the Gorry-Morton framework but eliminates its problems of definition. In addition, doing so addresses Stabell's point, that a whole task is often socially-defined; two universities may handle the admissions process -- the whole task -- very differently, but it will have common subtasks.

Keen and Morton, building on Gerrity and Stabell, discuss DSS design as a balance between descriptive and prescriptive understanding of the decision process. Supporting users implies providing commands that correspond to their existing (or at least desired) verbs. Improving the effectiveness of their decision making means identifying new commands and stimulating their use through the adaptive processes described by Figure 1.

A number of DSS researchers share this focus on subtasks. Blanning outlines the equivalent of a generative grammar for DSS that goes beyond verbs. Keen and Gambino suggest that most whole tasks require a common set of verbs; almost any DSS needs such functions as Graph, List, Select and Describe (provide descriptive statistics). Henderson et al. have designed a set of experiments on DSS use which track user behavior at the command and subtask level.[7]

The ideas presented above are tentative and a proposal for research rather than a conclusion from it. The central postulate is that adaptive design and use of DSS, DSS evolution, managerial learning, etc., require a decision research process where the level of analysis is at the subtask level. Much of the vagueness of DSS concepts disappears when this is provided. Of course, the research issue is how to represent the subtasks. Contreras, following on Berry, argues that they are linguistically at the level of APL functions, which can be further broken down into primitives. Blanning adopts a similar perspective.

Figure 2 adds the task dimension to the adaptive loops. Whatever methodology or theoretical model of task structure and performance is used, it is obvious that the representation can only be at the subtask level if it is to translate an understanding of how the users think, and an assessment of how their performance can be made more effective into specific functions in a DSS.

9. Contextual Issues in DSS Development

Figure 3 expands Figure 2 to include contextual forces. The additional links are not so much adaptive influences, as limiting ones. For example, organizational procedures may constrain user discretion and behavior (O--->U). In several of the case studies, DSS were not effectively used because the organization's control, communication and reward systems provided no incentive. Clearly, the extent to which organizational procedures affect individuals in a given task determines whether the situation requires a Personal, Group or Organizational Support System. In turn, the extent to which the user(s) can influence the procedures (U--->O) limits the organizational learning a DSS can stimulate.

In a similar fashion, the DSS itself is constrained by the organization's available technology (T--->S). This includes data as well as computer power, and the base of reliable operational systems and technical expertise on which a DSS capability is built. The case studies mainly describe successful systems but there are several suggestions that DSS will not take root in an organization that has

FIGURE 2.

Task Context

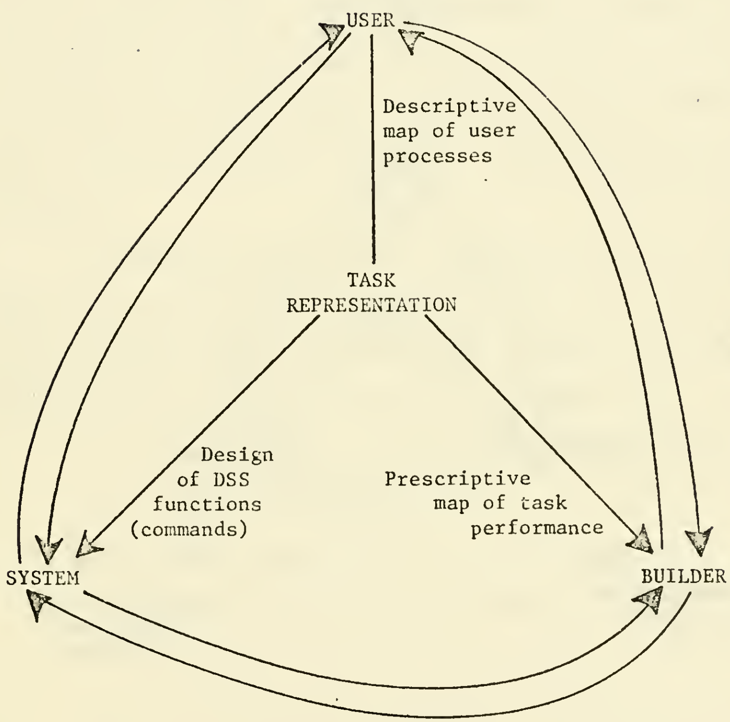
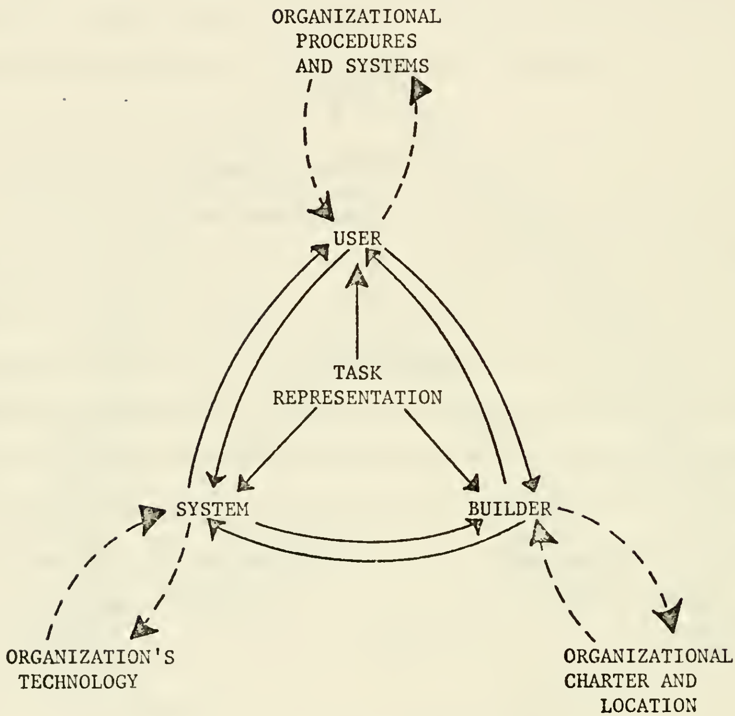


FIGURE 3.

Organizational Issues



not yet provided managers with standard data processing and reporting systems. In such situations, DSS are seen as a luxury or as irrelevant.

The link S→T is a reminder that learning and evolution can be blocked by the inability to obtain additional technology. Keen and Clark found that use of their DSS for state government policy and analysis was strongly constrained by states' existing technology and procedures for operating it. In addition, managerial learning and evolution of the DSS may require new data and structures or lead to an overloading of the organization's time-sharing computer. The whole adaptive process in Figure 3 breaks down when any influence is absent or blocked.

The final contextual issue addressed by Figure 3 is the charter for the builder. The implementation loop relies on facilitation and middle-out design. This requires a close relationship between the user and builder, which may not be feasible if:

(1) the two groups are geographically or psychologically isolated;

(2) the designers are part of a unit, such as Data Processing, with no mandate for innovation;

(3) the organization's charge-out policies and procedures for project justification discourage exploration and require "hard" benefits. Keen points out that DSS often provide mainly qualitative benefits. They "improve" a decision process and it is un-

likely that one can point in advance to a "bottom-line" payoff, especially if the value of the system is in the end determined by an adaptive, evolutionary process.

Many DSS builders are either consultants or academics, who can be brought into an organization by the user and who thus have relative freedom to define their role. A major constraint on developing a DSS capability may be the lack of a suitable organizational charter.

10. Conclusion

The additions to the earlier schema made in Figure 3 address the question of Organizational Support Systems raised earlier and left hanging. Figure 1 provides a complete research framework for Personal Support Systems. Figure 3 is far more tentative. Substantial research on organizational issues for DSS is needed, and no effort will be made here to justify or elaborate on this preliminary identification of organizational forces constraining DSS. The more important point is that Figure 1, and the definition of DSS it reflects, seem to provide a robust and adequately precise framework for DSS research. The representation of subtasks indicates a theoretical, if not yet practical, methodology for studying and building DSS.

If the framework presented here is valid, then Decision Support is a meaningful and independent discipline. The existing research base is fairly strong in certain areas, especially the implementation loop. There are more than enough case studies available to point up issues, such as the nature of managerial learning and DSS evolution, that should be explored at a more precise level, often through laboratory experiments. The major conceptual problems concern subtask representation, and a theoretical base for Organizational Support Systems. The term "Decision Support Systems" is an excellent rallying cry. "Decision Support" can be an equally outstanding field of study.

APPENDIX 1

CASE STUDIES OF DSS

Until definitions of DSS are firmly established, it will be difficult to keep track of the literature on the topic. Three references contain most of the case-based descriptions used for the review in this paper:

(1) Keen and Scott Morton, Decision Support Systems: An Organizational Perspective, (1978). This represents the orthodox faith. It includes fairly detailed descriptions of seven DSS. It excludes (largely because it filters the world through MIT-colored glasses, but also due to the long lead time between writing and publishing), the work of Courbon, Grajew and Tolovi, and most of that done at Wharton by Ness and Hurst. It has a comprehensive bibliography.

(2) Carlson, Morgan and Morton (Editors), Proceedings of a Conference on Decision Support Systems, (1977). This contains very little conceptual material and emphasizes real-world applications. It has a strong "show-and-tell" flavor. Whereas as Keen and Scott Morton use case descriptions to illustrate the concepts of a DSS, in this volume practicioners demon-

strate their view of what aspects of concept have practical value. It seems clear that on the whole they do not share Keen and Scott Morton's emphasis on the cognitive characteristics of the individual decision maker but focus instead on organizational processes.

(3) Alter, Decision Support Systems: Current Practice and Future Challenges, (1979). This book is based on case studies of 56 systems, only a few of which are DSS. There are 7 detailed cases, some of which overlap with Keen and Scott Morton, and Carlson et al. Alter is partly concerned with sharpening the practical definitions of DSS by looking at innovative systems in general. He uses the term Decision Support System fairly loosely, mainly since his is an exploratory study which specifically asks if it is useful to identify a system as a DSS.

A fourth study, by Grajew and Tolovi describes 3 experimental DSS projects. Le Moigne criticizes Keen and Scott Morton's book as "partial et partiel" -- incomplete and limited to the US experience. He feels that French researchers are more advanced than the Americans. Certainly the work of Courbon, Grajew and Tolovi builds on earlier research imaginatively and effectively.

The best bibliographies on DSS are in Keen and Scott Morton and in Grajew and Tolovi. In the list below, the major sources of reference are identified as Keen and Scott Morton (K-M), Carlson, et al (C), and Alter (A). The major cases are:

AAIMS: An Analytic Information Management System (C and A)

BIS: Budget Information System (A)

BRANDAID: Marketing Brand Management (K-M)

CAUSE: Computer Assisted Underwriting System at Equitable (A)

CIS: Capacity Information System (K-M)

EIS: Executive Information System (C)

GADS: Geodata Analysis Display System (K-M)

GMIS: Generalized Management Information System (K-M)

GPLAN: Generalized Planning (C)

IMS: Interactive Marketing System (A)

IRIS: Industrial Relations Information System (C)

ISSPA: Interactive Support System for Policy Analysts (Keen and Gambino)

MAPP: Managerial Analysis for Profit Planning (C)

PDSS: Procurement Decision Support System (International Harvester, private paper)

PMS: Portfolio Management System (K-M and A)

PROJECTOR: Strategic Financial Planning (K-M)

REGIS: Relational Generalized Information System (C)

APPENDIX 2

1. Unanticipated Uses:

   PMS -- intended use, investment decisions tool; actual use, marketing tool and customer relations.

   MAPP -- intended use, financial planning; actual use, revealed branch bank irregularities.

   PROJECTOR -- intended use, analyzing financial data to answer preplanned questions; actual use, alerted users to new issues and unplanned questions.

2. Personalized Uses:

   GADS -- public officials (police and school system users could imagine solutions then use GADS to test hypotheses; individual users' values placed on variables led to entirely different conclusions.

   REGIS -- encouraged data browsing, discerning new relationships and questions.

   PMS -- wide variance in function combinations used by individual managers.

3. Evolution:

   BIS -- initial system modular in structure, database separate from applications programs, new programs added incrementally without upsetting data base.

   PMS -- initial prototype followed by full implementation, doubled number of programs in six months.

CAUSE -- four evolutionary versions, deliberate emphasis on phased development to build credibility and capability, routines increased from 26 to 200 during the evolutionary period.

4. Simple Functions:

AAIMS -- 60 verb like commands used, DISPLAY, PLOT, QUARTERLY, CHANGE, ...

ISSPA -- DESCRIBE, EQUITY, REGRESS, HISTO, RANK, NTILES, ...

PMS -- SCATTER, SCAN, STATUS, TABLE, GRAPH, SUMMARY, GROUP, ...

5. Organizational Support System:

CAUSE -- supports underwriting process, including data definition and collection.

PDSS -- stabilized purchasing agents' ordering system.

IRIS -- supports operations control in industrial relations applications

6. Benefits:

CAUSE -- reduced need for employer specialization, increased possibilities of internal reorganization, gave opportunity to newer employees.

PROJECTOR -- time effectiveness improved "by a factor of 20", forced consideration of related issues, "confidence-inspiring" ananysis.

MAPP -- better product definitions and costing allocation, promoted internal learning.

7. Intermediaries:

GADS -- chauffeur used as teacher and translator, used to save time to get as many possible solutions as quickly as possible.

IMS -- 50% of use by junior researcher with no decision making authority, intermediary used only to push buttons not make decisions.

PMS -- secretaries operate, managers specify desired output.

REFERENCES

1. The analysis is contained in a report by Keen, "A Review of DSS Case Studies", now in draft form.

2. Keen and Scott Morton's book on DSS is mistakenly subtitled "An Organizational Perspective". The authors herewith recant.

3. The term was created by Ness, who built many of the early DSS and trained, at MIT and Wharton, many DSS designers. His working papers and case studies in Alter and Keen and Morton show the development of the middle-out concept. Courbon, Grajew and Tolovi have extended it in some brilliant empirical studies; they use the term "l'approche evolutive".

4. This work is still in progress. The DSS has been in use for less than six months, so that patterns of learning are only now becoming clear.

5. Methlie and LeMoigne have drawn attention to this and point out that Keen and Scott Morton entirely ignore data management issues.

6. Stabell has developed a range of techniques for decision research at several levels of analysis. However, they focus on task and on individual, and do not as yet include DSS design criteria.

7. Henderson, Gambino and Ghani, private communication.

Ginzberg, M.J., "A Process Approach to Management Science Implementation", Ph.D. dissertation, M.I.T., 1975.

Gorry, G.A., and M.S. Scott Morton, "A Framework for Management Information Systems", Sloan Management Review, Vol. 13, No. 1, pp. 55-70, Fall 1971.

Grace, B.F., Training Users of Decision Support Systems, San Jose, California, IBM Research Report RJ1790, May 1976.

Grochow, J.M., "Cognitive Style as a Factor in the Use of Interactive Computer Systems for Decision Support", M.I.T., 1974.

Hackathorn, R.D., "Research Issues in Personal Computing", Proceedings of the National ACM Conference, Washington, December 1978.

Keen, P.G.W., "Computer Systems for Top Managers: A Modest Proposal", Sloan Management Review, Vol. 18, No. 1, pp. 1-17, Fall 1976a.

Keen and Clark, "Simulations for School Finance, a Survey and Assessment", Research report to Ford Foundation, November 1979.

Keen and Gambino, "Mythical Man-Month Revisited", CISR working paper, January 1980.

Keen and M.S. Scott Morton, Decision Support Systems: An Organizational Perspective, Addison-Wesley, 1978.

Keen and Wagner, "Implementing Decision Support Systems Philosophy", Datamation, November 1979.

Meldman, J. "Decision Support Systems for Legal Research", Paper presented at the II Symposium Nacional de Systemas Computacionales, Monterrey, Mexico, March 15-18, 1977.

Morton, M.S. Scott, Management Decision Systems: Computer Based Support for Decision Making, Cambridge, MA: Division of Research, Harvard, 1971.

Ness, D.N. "Decision Support Systems: Theories of Design", Paper presented at the Wharton Office of Naval Research Conference on Decision Support Systems, University of Pennsylvania, Philadelphia, Pennsylvania, November 4-7, 1975.

Simon, H.A., "A Behavioral Model of Rational Choice", In H.A. Simon, Models of Man, pp. 241-260, New York: Wiley, 1957.

Stabell, C., "Decision Research: Description and Diagnosis of Decision Making in Organizations", In D. Heradstreit and O. Narvesen (eds.), Decision Making Research: Some Developments, Oslo, Norway: Norsk Utenriks politisck Institute, 1977.

BIBLIOGRAPHY

Ackoff, R.L., "Unsuccessful Case Studies and Why", Operations Research, Vol. 8, No. 4, pp. 259-263, March-April, 1960.

Alter, S.L., Decision Support Systems: Current Practice and Continuing Challenges, Addison-Wesley, 1980.

Andreoli, P., and J. Steadman, "Management Decision Support Systems: Impact on the Decision Process", Master's thesis, M.I.T., 1975.

Anthony, R.N., Planning and Control Systems: A Framework for Analysis, Cambridge, MA, Harvard University Graduate School of Business Administration, Studies in Management Control, 1965.

Bennett, J., "Integrating Users and Decision Support Systems", in J.D. White (ed.), Proceedings of the Sixth and Seventh Annual Conferences of the Society for Management Information Systems, pp. 77-86, Ann Arbor, University of Michigan, July 1976.

Berger, P., and F. Edelman, "IRIS: A Transaction-Based Decision Support System for Human Resources Management", Database, Winter 1977, Vol. 8, No. 3.

Berry, P., "The Democratization of Computing", Paper presented at Eleventh Symposium Nacional de Systemas Computacionales, Monterrey, Mexico, March 15-18, 1977.

Blanning, R., "The Decision to Adopt Strategic Planning Models, "Wharton School working papers, 1979.

Brooks, F.P., The Mythical Man-Month, Reading, MA, Addison-Wesley, 1975.

Carlisle, J., "Cognitive Factors in Interactive Decision Systems", Ph.D. dissertation, Yale University, 1974.

Carlson, E.D., and J.A. Sutton, A Case Study of Non-Programmer Interactive Problem-Solving, San Jose, California, IBM Research Report RJ1382, 1974.

Contreras, L., "Decision Support Systems and Corporate Planning", Informal Communication, 1978.

Courbon, J.C., J. Grajew, and J. Tolovi, "Design and Implementation of Interactive Decision Support Systems: A Evolutive Approach".

Gerrity, T.P., Jr., "The Design of Man-Machine Decision Systems", Ph.D. dissertation, M.I.T., 1970.