DECISION SUPPORT SYSTEMS:
ISSUES AND PERSPECTIVES

Michael J. Ginzberg
and
Edward A. Stohr

Center for Research on Information Systems
Computer Applications and Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #27

GBA #82-12(CR)

I. Introduction

The past few years have witnessed the emergence of Decision Support Systems (DSS) as an area of great activity within the information systems field. The pages of MIS and MS journals are filling with articles about DSS. Several conferences on DSS have already taken place and more are scheduled. One major publisher has started a series of books on DSS, which currently includes five volumes. The advertising of DP service providers extols the virtues of their wares as DSS or as components from which DSS can be built. Several academics have been seen criss-crossing the country (and occassionally venturing overseas) trying to win converts to the DSS faith.

What is the basis for all of this action? What is it that DSS does that has not been done before? More fundamentally, what is DSS, and how does it differ from the other computer-based systems organizations have been using for years?

Claims about the benefits and capabilities of DSS are substantial. They will make managers more effective. They will improve managerial decision making, especially in relatively unstructured tasks. They will extend managers' cognitive capabilities, while leaving the manager free to exercise his or her judgement where that is needed. The book is not yet written on what DSS will actually do. Certainly, some DSS have had the types of impacts claimed. Others have not yet shown such a major impact, nor are they ever likely to.

In part, the range of DSS impacts which have been observed stems from the variety of systems which have been labeled DSS. There is at present little consensus about what qualifies a system as a DSS. This paper will begin by examining some of the definitions that have been suggested. It will then examine the implications of these different definitions, focusing on the issues highlighted by and ignored by each definition. The paper will then suggest a definition for DSS which highlights those issues we believe are most central to developing and implementing more effective DSS. The remainder of the paper will explore those issues and attempt to outline the areas where further research over the next few years could be most fruitful.

II. DSS Definitions

The earliest definitions of DSS (e.g., Gorry and Scott Morton [1971]) identify DSS as systems to support managerial decision makers in unstructured or semi-structured decision situations. Two key concepts in this definition are support and unstructured. First, these systems were meant to be an adjunct to the decision maker, to extend his capabilities but not to replace his judgement. Second, they were aimed at supporting the manager in those decisions where judgement was required, decisions that could not be completely specified as an algorithm and turned over to the computer. Not specifically stated in, but implied by, the early definitions was that the system would be computer-based, would operate on-line, and preferably would have graphics output capabilities.

A refinement of these early definitions is provided by John Little [1970] in his definition of a "decision calculus." He defines this as a "model-based set of procedures for processing data and judgements to assist a manager in his decision making" (p. B470). He argues that in order to be successful, such a system must be (1) simple, (2) robust, (3) easy to control, (4) adaptive, (5) complete on important issues, and (6) easy to communicate with. Implicit in this definition, too, is the assumption that the system will be computer-based and that it will serve as an extension to the user's probem-solving capabilities.

Throughout most of the 1970's, definitions of DSS like those presented above were accepted by practitioners and researchers who wrote about DSS. By the end of the decade, however, new definitions began to emerge. Alter [1980] defines DSS by contrasting them to traditional EDP systems on five dimensions:

1) Use: active (DSS) vs. passive (EDP)

2) User: line, staff & management (DSS) vs. clerk (EDP)

3) Goal: overall effectiveness (DSS) vs. mechanical efficiency (EDP)

4) Time horizon: present & future (DSS) vs. past (EDP)

5) Objective: flexibility (DSS) vs. consistency (EDP)

Three other recent definitions of DSS are offered by Moore & Chang [1980], Bonczek, Holsapple & Whinston [1980], and Keen [1980]. Moore & Chang argue that the "structuredness" concept, so much a part of early DSS definitions, is not meaningful in general; that a problem can be described as structured or unstructured only with respect to a particular decision maker or group of decision makers. Thus, they define DSS as (1) extensible systems, (2) capable of supporting ad hoc data analysis and decision

modeling, (3) oriented towards future planning, and (4) used at irregular, unplanned intervals.

Bonczek, Holsapple & Whinston [1980] define a DSS as a computer-based system consisting of three interacting components. Those components are (1) a Language System -- a mechanism to provide communication between the user and other components of the DSS, (2) a Knowledge System -- the repository of problem domain knowledge embodied in the DSS, either as data or procedures, and (3) a Problem Processing System -- the link between the other two components, containing one or more of the general problem manipulation capabilities required for decision making.

Finally, Keen [1980] applies the term DSS "to situations where a 'final' system can be developed only through an adaptive process of learning and evolution" (p. 15). Thus, he defines DSS as the product of a development process in which the DSS user, the DSS builder, and the DSS itself are all capable of influencing one another and resulting in evolution of the system and the pattern of its use.

These definitions can be contrasted by examining the types of concepts each employs to define DSS. This contrast is shown in Exhibit 1. It should be apparent that the basis for defining DSS has been migrating from an explicit statement of what a DSS does (i.e., support decision making in unstructured problems) to some ideas about how the DSS's objective can be accomplished (i.e., what components are required? what usage pattern is appropriate? what development process is necessary?).

Concepts Underlying DSS Definitions

Source	DSS defined in terms of:
Gorry & Morton [1971]	problem type, system function (support)
Little [1970]	system function, interface characteristics
Alter [1980]	usage pattern, system objectives
Moore & Chang [1980]	usage pattern, system capabilities
Bonczek et al. [1980]	system components
Keen [1980]	development process

Exhibit 1.

One result of this migration is a narrowing of the population of systems that each author would identify as DSS -- e.g., Keen would exclude any systems which can be built without following an evolutionary strategy, and Moore & Chang would exclude systems which are used at regular, planned intervals to support decisions about current operations. This type of narrowing of a population is indeed a proper function of a definition. By dealing with a smaller population of objects, we can identify those characteristics which the members of the population have in common as well as those characteristics which differentiate one population from another. This helps to focus attention on those problems where research is most needed and is likely to be most fruitful.

Unfortunately, the most recently offered definitions of DSS do not provide a consistent focus, since each tries to narrow the population in a different way. We can consider the types of questions each definition leads us to ask. Following Moore & Chang we would ask, how can you build extensible systems or systems to support analyses which have not been prespecified? Bonczek et al. would lead us to ask how knowledge can be represented in a system and how to provide various problem processing capabilities. Keen's definition would cause us to ask how the development process can be structured to assure that the feedback loops among user, builder, and system are in place and functioning.

While all of these questions are interesting, they collectively ignore the central issue in DSS; that is, support of decision making. There seems to have been a retreat from consideration of outputs, the dependent variable, and a focus on the inputs instead. A very likely reason for this change in emphasis is the difficulty of measuring the outputs of a DSS

(i.e., decision quality). While such measurement difficulties no doubt exist, they must not be used as an excuse for ignoring what should be our central concern.

Supporting and improving decision making is the issue in DSS. Definitions which attempt to narrow the field, to focus research along some other dimension are missing the point. Indeed, this is the reason why recent DSS definitions have been so inconsistent with one another and have not developed a clear notion of DSS.

We propose that, for now at least, a definition of DSS quite close to the early definitions of Gorry & Scott Morton and Little be adopted. That is, a DSS is a computer-based information system used to support decision making activities in situations where it is not possible or not desirable to have an automated system perform the entire decision process. The remainder of this paper will explore some of the implications of this definition: What characteristics are common to all (or most) DSS? What characteristics can differ? What development process (or processes) is appropriate for DSS? What usage pattern (or patterns) is appropriate for DSS? What research is needed to enable us to build better DSS? Perhaps once these questions have been answered it will be possible to draw narrower boundaries around the field, to more clearly define DSS.

III. The Anatomy, Physiology and Ontogeny of DSS

The elements that characterize DSS can be separated into three major groups: (1) the underlying technological components from which DSS are built, (2) the ways in which DSS are used, and (3) the processes by which

DSS are designed and implemented. These three groups represent the anatomy, physiology, and ontogeny of DSS, and a full understanding of them will provide a relatively complete picture of DSS.

III.A. Anatomy: DSS Technology

In earlier work, Keen and Scott Morton [1978] argued that technological issues were secondary considerations in DSS. Advances in hardware and software had made DSS possible, and the important considerations in DSS design involved human decision making. Since that time, research has continued on discovering problem situations where DSS can be applied, on the man-machine interface, on the impact of DSS on individuals and groups, and on the behavioral aspects of implementation. Other research, however, has stressed the hardware and software aspects of DSS, seeking to broaden the domain of application of computer support, to speed the development process, and to make the resulting system more adaptable to the changing needs of decision makers.

In our view, both areas of research are vital to the success of the DSS concept. Technological progress determines what can be done; behavioral research determines what should be done and how best to apply technology to serve organizational goals. Ideally, the latter should drive the former, but it seems that technology has a momentum of its own.

In this section we review DSS technology, the elements which make up a DSS. To be complete, our review should consider both hardware and software. DSS hardware, however, differs little from that for any modern computer-based information system. No special hardware is required for

DSS, in our view, nor are there any necessary hardware requirements to qualify a system as a DSS. There are some interesting potential impacts of hardware trends for future DSS, and we shall address these in the final section of this paper.

Turning to DSS software, we first consider some suggested taxonomies. The first (Alter [1977]), divides DSS software into seven types based on function performed. Three of the types are data-oriented, performing data retrieval and/or data analysis. The remaining four types are model-oriented, providing either a simulation capability, optimization or computations that 'suggest an answer'. The IRIS system (Berger and Edelman [1977]) which utilizes a high level interactive query language and data base techniques provides an excellent example of a data-oriented DSS. On the other hand Hax and Meal's [1977] 'hierarchical' production planning and scheduling system which combines optimization and heuristics provides an example of a model-oriented system. Some DSS, however, (e.g., Holsapple and Whinston, [1976]) seem equally oriented to both data retrieval and modelling while others (GADS, Carlson et al [1974]) are graphics-oriented. Neither system fits easily in this classification scheme.

Donovan and Madnick [1977], differentiate DSS based on the nature of the decision situation they are designed to support. Institutional DSS deal with decisions of a recurring nature. An example is the Portfolio Management System (PMS) which has been used by several large banks to support investment managers (Gerrity, [1977]). Another example is the comprehensive system being developed by AT&T and described by Jeske in this volume. Institutional DSS may be developed and refined over a number of years. Ad hoc DSS deal with specific problems that are neither anticipated

nor recurring. To support this kind of situation requires general-purpose software for information retrieval, data analysis and modelling that can quickly be customized to a specific application. Donovan [1976] describes the development and use of the Generalized Management Information System (GMIS) to support ad hoc decision-making.

A classification scheme with similar implications is proposed by Sprague [1980]. Here the criterion is flexibility and transportability across decision situations. Specific DSS are built to support a particular organization and task. An early example is the Potlatch Forests system (Boulden and Buffa, [1970]) which provided an interactive planning system containing a model of the company's operations embedded in FORTRAN code. DSS generators, on the other hand, provide more general purpose retrieval and modelling facilities and can be quickly tailored to a specific problem. A survey by Naylor and Schauland [1976] documents a marked growth in the use of DSS generators for the projection of financial statements and the development of more general corporate planning models. Current commercial DSS generators of this type include SIMPLAN (Mayo, [1979]) and IFPS (EXECUCOM, [1979]).

A final classification scheme is based on the degree of non-procedurality of the data retrieval and modelling languages provided by the DSS (Bonczek et al., [1980]). Procedural languages require a step-by-step specification of how data is to be retreived or a computation performed. Non-procedural languages require the user to specify only what is required. At an intermediate level of procedurality are systems that utilize a command language allowing the user to specify the name of a prespecified report or model. The cartesian product of these three levels

of procedurality for data and model-oriented interfaces provides nine different possible classes of DSS. DSS systems have progressed from systems where both data retrieval and modelling is achieved using procedural languages to DSS generators that provide intermediate levels of non-procedurality. An objective of DSS development is to provide easy-to-use non-procedural languages for both the data and modelling interfaces.

The various approaches to classifying DSS are summarized in Exhibit 2. Of these schemes we favor the last. It provides an historical perspective on DSS development and objectives, and focuses on the user interface as a key issue in DSS design.

Are there characteristics of DSS software that distinguish it from other software systems? Not necessarily. We have already argued for a broad definition of DSS which would not exclude a batch-oriented interface. A data base system with a high level query language can be the basis for decision support. Office automation systems with their emphasis on the user interface and possessing such features as high-level languages for defining and processing forms and graphic representations of 'in-baskets', 'out-baskets' and files possess many of the attributes of advanced DSS software.

Nevertheless distinctive characteristics do emerge when we consider general purpose DSS software and especially software incorporating modelling capabilities. Before discussing these features we note several important objectives. DSS generators exist because they provide a means for satisfying ad hoc decision-making support. They speed-up the development process for specific DSS applications. To this end they must

DSS Software Classification Schemes

Source	Classification Scheme
Alter [1977]	Data-oriented vs. model-oriented
Donovan and Madnick [1977]	Ad-hoc vs. institutional
Sprague [1980]	Specific DSS vs. DSS generators
Bonczek, et al. [1980]	Procedural vs. non-procedural

Exhibit 2.

be adaptable to different decision-making situations and easy to use. Beyond this, the generated systems should possess the same qualities as other good software -- accuracy, reliability, maintainability, modifiability and so on.

The major components of DSS software incorporating a modelling capability are depicted in Figure 1, which is based on Sprague and Watson [1976], Bonczek et al. [1980], Haseman and Kellner [1977], and Blin et al. [1978]. A similar architecture was evident as early as 1970 in the Potlatch system cited earlier. The major change in modern design concepts is the attempt to give each component the properties of independence, generality of function and intelligence. A similar evolution occurred in the development of data retrieval facilities from operating system access methods activated by calls from procedural languages to file management systems and finally to database management systems (DBMS).

The purpose of the Data Extraction System is to load the DSS data base with external data and data generated internally to the organization by the MIS system. Although it is generally accepted that a DSS requires less detailed and timely data than operational systems (Keen and Scott-Morton, [1978]) this can be a major implementation problem. Since the DBMS used by the DSS may be different from that of the operational MIS and since the data will generally be structured differently, quite complex data conversion operations may be required.

The use of a DBMS in a DSS where the major purpose is data retrieval is direct and obvious. One only needs to define the 'schema' for the data base and perhaps provide specially-tailored user interfaces if the data retrieval language associated with the data base is inadequate to the task.

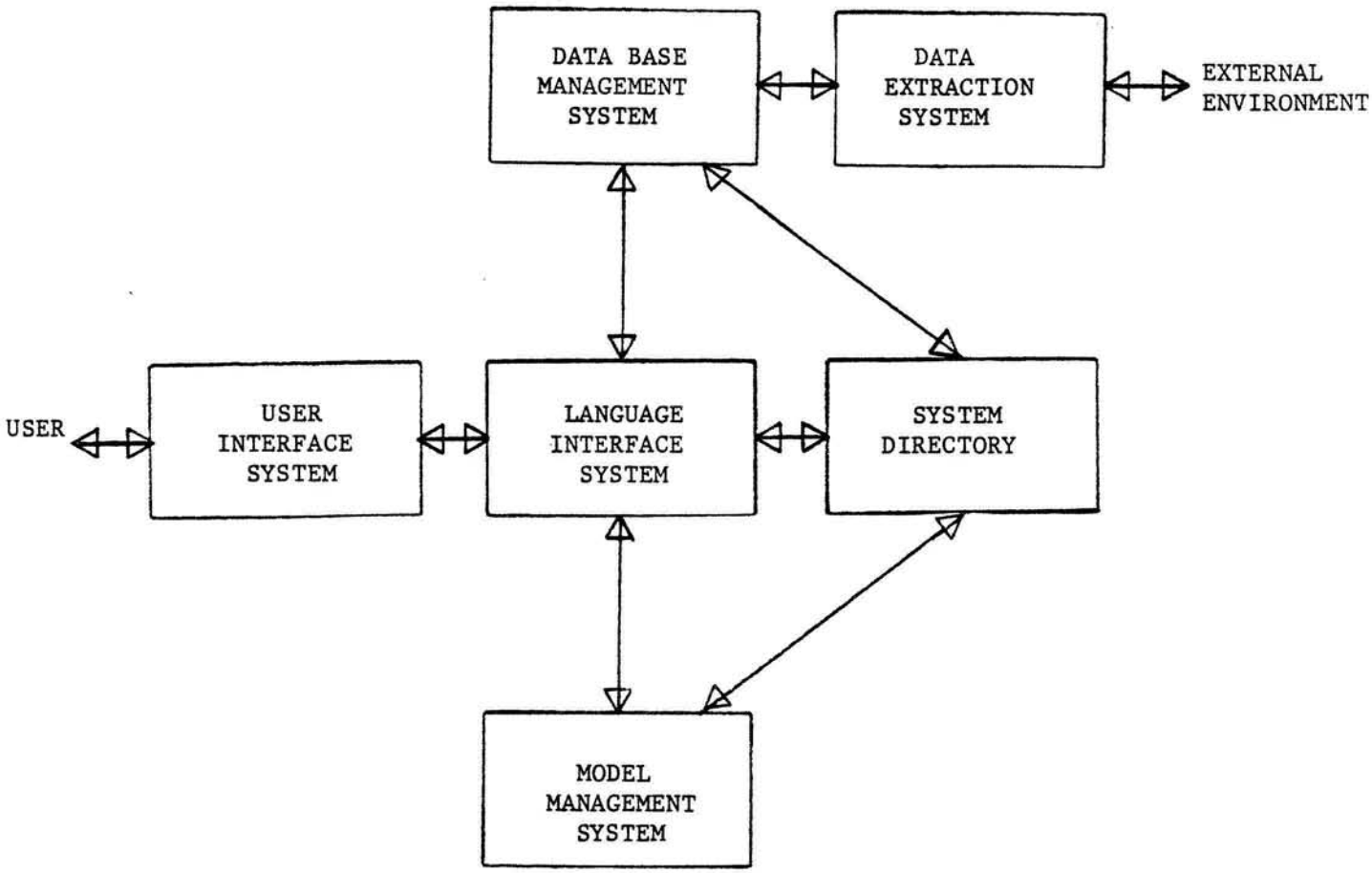

Figure 1

MAJOR COMPONENTS OF DSS GENERATORS

In model-oriented DSS there are several additional functions that might be performed by the DBMS:

1. management of both the inputs and outputs of the models

2. storage and access of the models themselves

3. storage and access of a 'knowledge base' of meta-data concerning the semantics of the models and the structure of their constituent processes.

A major research issue in the application of DBMS within DSS concerns the use of time-series and cross-sectional data in modelling. Existing modelling languages have special facilities for managing such data but are weak in general retrieval capabilities. Conversely DBMS's have powerful retrieval capabilities and can maintain complex data interrelationships; however special application programming may be required to enable them to handle multiple copies of dynamically generated variable length series data as occurs in forecasting applications. Another DBMS issue concerns the need to store intermediate and final results in the database while the DSS is being used. The structure of this data generally can not be anticipated giving rise to the need to dynamically restructure the data base schema.

The concept of a Model Management System (MMS) is an innovative product of DSS research. Its purpose is to facilitate both the development of models and their subsequent use during sensitivity analyses (Elam et al. [1980]). An example of an operating MMS is contained in Sprague and Watson [1976], and a general discussion is contained in the paper by Bonczek et al. in this volume. There are several unresolved research issues. How

can the results of sensitivity analysis be stored in the data base and related to the underlying model assumptions and data values? (Stohr and Tanniru, [1980]). Can decision aids be dynamically constructed from module components thus providing the user with a modelling language with the same level of non-procedurality as advanced data base query languages such as IBM's SEQUEL? The idea is that the user would simply specify the required data items without necessarily being aware that they were not stored in the data base; the DSS would then determine the proper sequence of operations, including models to be applied, and display the desired result. Two approaches involving the use of artificial intelligence techniques are proposed by Elam et al. [1980] and Bonczek et al. [1981]. The first involves the use of semantic inheritance nets (Findler [1979]) for knowledge representation, and the second, the use of the resolution principle of first-order predicate calculus (Nilsson, [1971]). The relative merits of these and other alternatives remain to be investigated.

The Language Interface Subsystem consists of the compilers and interpreters that translate the statements and commands used for specifying retrieval requests and defining models. The power of these languages has an important effect on the ease of use of the DSS. Attempts to avoid the complexity of step-by-step procedural specifications have already been mentioned. However it is not yet clear how much non-procedurality can or should be attained, especially with respect to modelling languages. Another approach in this area is to develop languages that will perform more computations per character of instruction and avoid some of the tedious details of the control structures and loops of conventional programming languages. A forerunner in this area is APL which has been used in many DSS (see the paper by Mattern in this volume).

Another unresolved issue concerns the use of quasi-natural, English-like languages in DSS. Are such languages truly easier to use? Can they be made precise enough? The answers to these questions are not yet clear (Schneiderman, [1980], Ch. 9). Finally, there is the possibility of including a parser-generator in the language sub-system. This would greatly simplify the process of tailoring the user interface both to the particular problem being addressed and to the needs of the users. An example of a mathematical programming generator system incorporating a parser generator is given in Mills et al., [1977].

The idea of the User Interface Subsystem as a separate layer of software lying between the end-user and the other components of the DSS has been used in the DAISY system (Buneman et al., [1977]) and AIPS (Advanced Information Presentation System, Yonke and Greenfield, [1980]). The objectives are:

(1) to eliminate the need for special applications programming to generate displays and control the user dialogue;

(2) to provide the user with a variety of information channels ('windows') and input-output media including high-level command languages, light-pens and other devices for manipulating visual displays, voice input and output, conventional reports and graphics.

The design of the user interface system provides a fertile area for DSS research. How should the user interaction be managed? When should graphical displays be used? Some guidelines are included in Schneiderman [1980, Ch. 11.], however further human factors research is required.

The System Directory has been shown as a separate component although it may often be included as part of the DBMS software (Bonczek et al. [1980]). Its function is to give the system some degree of self-knowledge. The purpose is to help the user learn the system, to reduce the need for a complete specification of problems, and to prevent misuse of the system. Outstanding research problems here include the determination of the best methods for representing knowledge -- a problem traditionally attacked by artificial intelligence research (see Findler, [1979]).

In summary, the software components of a DSS are quite diverse. No DSS that we know of contains all of the components discussed above, though most DSS contain at least some of them. Most of these components can be found in other computer-based systems as well. The Model Management System and the potential role of artificial intelligence, however, seem to be unique to DSS. A further distinctive characteristic of DSS is the attempt to integrate both human and machine decision making into one system. A good example is provided by multiple criteria decision making and other human-aided optimization techniques (see the paper by Hurst & Kohner in this volume).

III.B. Physiology: DSS Usage Patterns

Many authors have attempted to circumscribe DSS through the definition of "appropriate" DSS usage patterns. This includes who should use a DSS, for what problems, at what intervals, through what mechanism, and to what end. In this section we shall consider each of these questions about DSS use, asking whether any single pattern of use is really more appropriate for DSS than any other.

Much of the DSS literature asserts or at least implies that DSS must be used by managers, or perhaps even upper level managers. But, is such a usage pattern a necessary condition for DSS? We conclude not. While, clearly, many DSS are used by managers, many others exist (or could exist) where this is not the case. Some of the DSS which have been most widely studied and written about are, in fact, used by non-managers at least as much as by managers. The principle users of Gerrity's [1971] Portfolio Management System while often having the title of manager (e.g., Portfolio Manager) are not managers in the traditional sense. Rather, they are investment decision makers, a professional, but not truly managerial role. GADS (Carlson, Grace and Sutton, [1977]), a DSS generator which has been used in a number of real decision situations involving geographic boundaries, has had as users many different "interested parties" to those decisions; but, certainly, they were not all managers. Alter [1980] also describes several DSS whose primary users were non-managerial personnel; e.g., an insurance renewal system used by underwriters.

Note that we are not developing an artificial position by looking at the "hands-on" user of the system and asking whether he (or she) is a manager. Rather, we are considering the position of the consumer of DSS output, the person who directs what analyses the DSS is to perform. In the GADS case, for example, usage is always through a trained intermediary. Our focus, however, is on those people who instruct the intermediary; and, the evidence indicates that these people are quite likely to be non-managers.

What do these DSS users have in common if not managerial status? All are either decision makers or key "stakeholders" in the outcome of a decision, people who need to understand the implications of decision alternatives and who want to influence the choice among alternatives. Thus, rather than using formal role --i.e., managerial status -- as the criterion for defining proper DSS users, it is more sensible to use functional role -- i.e., key decision influencers. Perhaps this explains why relatively few top level managers use DSS and why we should not expect them to. Though Mintzberg [1971] describes top managers as the ultimate decision makers for their organizations, in many (if not most) cases this is a misleading description of their role. Top managers are more often decision ratifiers than decision makers. That is, their staffs present them with alternatives, recommendations, and rationales to support the recommendations. In many cases, the manager then accepts the staff recommendation and ratifies the decision which was made by the staff. Since it is the staff, not the executive, who goes through the entire decision process, one should expect the staff to be the DSS user as well.

The early DSS literature clearly defines the types of problems for which DSS are appropriate as those which are semi-structured, that is not completely structured at one or more of the problem solving phases -- intelligence, design, or choice (Gorry and Morton, [1971]). Further, it is suggested that DSS are most appropriate to strategic planning, rather than control, problems, a theme which has been reiterated recently (Moore and Chang, [1980]). To what extent should the characteristics of problem structure and problem level (planning vs. control) constrain the definition of DSS?

Moore and Chang [1980] argue that problem structuredness cannot be defined in absolute terms, hence they dismiss it as a meaningful concept for defining DSS. While their premise is no doubt true, we cannot accept their conclusion. Degree of problem structure, even if it can only be discussed with reference to particular decision makers, is central to DSS. At one extreme, if a problem can be completely structured to the satisfaction of some decision maker, an algorithm can be written to replace the human decision maker. If no judgement is required -- i.e., the complete decision process can be specified -- decision support is not an issue. At the other extreme, if no structure can be brought to the problem, that is if none of the data requirements nor any of the necessary processing at any problem solving stage can be specified, decision support is impossible. It is only between these extremes that DSS is relevant, and we agree with Moore and Chang that the location of the extreme points can vary across decision makers.

What does it mean to say that a problem is semi-structured? In essence, it means that it is possible to bring some structure to bear on the problem; that a decision maker is willing to accept a certain data set or certan processing routines as relevant to problem solution. Indeed, in a recent paper Alter [1981] points out that a good DSS brings as much structure as possible to the problem. While we would change this to as much structure as the system user will accept, we are fundamentally in agreement with Alter. Thus, rather than defining DSS as systems appropriate to partially structured problems, they are better defined as systems applicable to problems which are at least partially, but not completely, structurable.

Does it matter to what phase of the problem solving process this structure is applied? We conclude not. Structure should be brought to bear on any phase of the process where it is appropriate -- i.e., definable and acceptable to the decision maker. Indeed, the DSS described in the literature show examples of support for all phases -- intelligence or problem definition, design of alternative solutions, evaluation of and choice among the alternatives, as well as monitoring and control of implementation of the chosen solution. Of course, the type of support will vary from phase to phase since the nature of the activities varies. And, some phases are more easily supportable than others; hence, they are more likely to receive support. In particular, support for the design phase typically requires much more problem domain knowledge embedded in the system than is the case for the other stages. Consequently, relatively few DSS support design. Reitman (in this volume) discusses the application of artificial intelligence techniques to alternative generation, i.e., the design phase.

As stated earlier, many authors view DSS as appropriate only to future oriented, planning problems, not to current control problems. We find this distinction hard to accept. In what way is the decision process for (1) planning next year's operations of a complex manufacturing facility fundamentally different from that for (2) analyzing last month's sub-standard performance of the facility in order to design a corrective course of action? Both require completion of the entire intelligence - design - choice - implementation cycle. Neither is likely to be totally structurable, nor is either likely to be completely unstructurable. In other words, each would appear to be a candidate for a DSS.

One of the more recent trends in delimiting DSS is to state that they are systems whose usage patterns will evolve (e.g., Keen [1980], Moore and Chang [1980], Sprague [1980]). In a sense, this is undoubtedly true. Usage patterns for all systems, not just DSS, evolve. Additional data items are placed in an employee master file and the payroll system becomes a personnel system. New reports are added. On-line inquiry and update are added. Evolutionary usage is not unique to DSS.

The argument made by those who claim evolutionary usage as a hallmark of DSS is that use will lead to learning which will lead to new demands on the system, which will lead to refinement of the system, which will lead to new usage patterns, and so forth. There seems to be some confusion here between DSS and novelty. Any system -- transaction processing, word processing, DSS, or what have you -- that represents an initial effort in an area for the user organization is likely to evolve, both in form and in usage pattern. Perhaps some DSS will evolve more rapidly than other systems because they start with less structuring and thus have more "room" for change. This will not necessarily be the case for all DSS. In some, the initial structuring may be all that can be accomplished for quite some time; thus, the usage pattern may be relatively stable for a protracted period. In summary, evolutionary usage does not appear to be a particularly useful way to characterize DSS. It fails to distinguish DSS from other computer-based systems, and ascribes to all DSS something which is characteristic only of some.

Several other characteristics of DSS usage patterns are frequently suggested. These are: (1) voluntary usage (e.g., Lucas [1978]), (2) interactive usage (e.g., Scott Morton [1971]), and (3) unplanned usage

(e.g., Moore & Chang [1980]). In general, DSS users do have greater discretion about both type and amount of system use than do users of more conventional computer-based systems, e.g., transaction processing systems. Two caveats are necessary, however. First, voluntary usage does not distinguish DSS from other innovative systems, e.g., office automation systems. And second, what appears to be voluntary usage of a DSS may not always be so. In some cases, a DSS is the only available source of information a user needs to do his job. Thus, while use may in theory be voluntary, some amount of use becomes mandatory.

Next to use in support of semi-structured problems, perhaps the most frequently mentioned characteristic of DSS is interactive usage. Initially it was argued that interactive usage was necessary so that the decision maker could carry on an uninterrupted dialog with the DSS (see e.g., Scott Morton [1971], Carroll [1967]). History has made it apparent that few decision makers want to have on-line dialogs with their DSS. Many DSS are used through an intermediary. In those DSS where the decision maker is the hands-on user, he will, as likely as not, use the system in an "intermittent" mode -- executing a few functions, stepping back (perhaps for several hours or even days) to study the result, and then returning to the terminal. Thus, interactiveness is not a good characterization of most DSS. Moreover, technological advance has resulted in many non-DSS (e.g., real-time order entry systems) being converted to an interactive usage mode.

Perhaps more relevant than interactiveness as a characteristic of DSS usage is controllability. This includes availability of the system when the decision maker wants to use it, which often implies an on-line system.

But, it goes farther, to what Turner [1980] calls two-way communication. In systems with two-way communication, the user can react to "intermediate" processing results, and direct further processing on the basis of these results. One-way communication, on the other hand, implies that the user can do little to alter the course of processing once it has begun. This two-way communication can take place during a single session at a terminal, over multiple terminal sessions spread out in time, or even with a batch system. This type of controllability is a far more meaningful way to characterize DSS than is interactive usage.

Unplanned usage means different things to different people. To some it means that system outputs cannot be planned in advance. To others it means that usage is aperiodic and cannot be prescheduled. While both are characteristic of many DSS -- distinctly more so than they are of conventional systems -- neither seems a necessary DSS characteristic. Many decisions recur with substantial regularity in form, in timing, or in both. In the first case, at least some parts of the output can be prespecified; indeed, this is the point of bringing structure to the decision process. In the second case, scheduled, periodic usage of the DSS should be possible. Surely, neither of these circumstances should disqualify a system as DSS.

One final question about DSS usage patterns concerns purpose of use. By far, the bulk of the DSS literature views the purpose of DSS as enhancing an individual decision maker's cognitive capabilities (e.g., Gerrity [1971], Keen and Hackathorn [1979], Stabell [1977]). This view ignores the fact that DSS are used in organizational settings, and do not simply support lone decision makers. Alter [1976] describes the

DSS Usage Patterns

WHO: - Decision influencers

WHAT: - Partially structurable decisions

- Any/all decision process phases
- Planning and control

HOW: - Directly or through intermediary

- Evolutionary, but with widely varying time frames
- Largely voluntary
- Controllable, though not necessarily interactive
- Scheduled and unscheduled
- Partially prespecified and ad hoc

WHY: - Cognitive enhancement, communication, and control

Exhibit 3.

"offensive" use of DSS as tools to bolster an individual's position on a contested issue, to provide the "weight of evidence" to enable him to prevail. Ginzberg [1980] notes that DSS are often used to coordinate decision making activities among the multiple, interdependent participants in a decision. In general, DSS are used to exert control or influence, achieve coordination, as well as enhance cognitive capabilities.

Exhibit 3 summarizes the who, what, how, and why of DSS usage. This summary makes it clear that very few of the distinctions suggested in the literature hold up under close analysis. DSS usage patterns are widely varied. The major commonalities in usage which help define DSS as something unique are (1) largely voluntary, (2) controllable (3) use by decision influencers (4) in partially structurable decisions.

III.C. Ontogeny: DSS Development Patterns

Much less has been written about DSS development patterns than about usage patterns. Still, some authors have proposed certain development patterns as appropriate to DSS and others as inappropriate. The most common prescriptions for DSS development are that (1) it must include normative decision modeling (e.g., Gerrity [1971], Keen and Scott Morton [1978]), (2) it must be participative (e.g., Schultz and Slevin [1975], Ginzberg [1978]), and (3) it must be evolutionary (e.g., Keen [1980], Moore and Chang [1980], Sprague [1980]). Several other issues about DSS development which should be considered are (1) its focus -- on an individual, an organizational role, or a problem, and (2) its orientation towards change.

Normative modeling is the mechanism by which additional structure is brought to unstructured decision situations. That is, the normative model specifies how a decision (or part of a decision) should be made. Thus, if one purpose of DSS is to bring structure to decision making, normative decision modeling is a necessary part of the DSS development process.

The call for user participation in system development is hardly unique to DSS. Indeed, this is one of the most commonly heard prescriptions for developing any type of computer-based system. Since the general case for participation has been amply discussed elsewhere, we will not repeat it here. It should be noted, however, that user involvement in DSS development is perhaps more important than in other, less innovative computer-based systems. DSS often are less well defined, imply greater change, and require more training than many other systems. As a result, user involvement is needed to help resolve design uncertainties and to prepare the users for the new system. It should be noted that while we strongly believe user involvement in DSS development is important, we do not suggest that this should be a criterion for identifying or defining DSS. In some cases, gaining user involvement during DSS development is difficult -- e.g., because of a large number of users, or because development is being conducted by an entrepreneur who will later attempt to sell the system. Nonetheless, such systems can be DSS, and Alter [1978] has shown that they are often quite successful.

Recent DSS literature has argued strongly the need for evolutionary design, and Keen [1980] goes as far as saying that an evolutionary design process is a prerequisite for calling a system a DSS. Moore and Chang [1980] state well why evolutionary design is often necessary: the user's

problem or problem view changes, hence the system must evolve to remain relevant and useful. However, as stated in the discussion of DSS usage patterns, not all DSS will experience rapid evolution in usage, while some non-DSS will. Further, Henderson and Ingraham (in this volume) raise some serious questions about the efficacy of evolutionary design for DSS. Rather than requiring an evolutionary design process for DSS, it makes more sense to require a flexible process. Where substantial uncertainty exists about user needs or probable system usage patterns, evolutionary design may well be appropriate. If uncertainty is somewhat less, prototyping may be the best approach to design. And, where little uncertainty exists and a fairly stable usage pattern can be projected, a more traditional, structured approach to design is appropriate. The key is to match the design approach to the needs of the situation (see Alter [1981] for additional comments on this issue).

DSS focus refers to the orientation of the system: towards a particular individual, an organizational role or set or roles, or a specific problem or set of problems. As such, it has implications for system content and usage patterns, but its strongest implications are for the development process. A DSS oriented towards a particular individual should be designed with the needs and preferences of that individual in mind. This includes his/her view of which decision(s) should be supported, how they should be supported (i.e., what models and data are appropriate or necessary), and how data should be presented. That is, the potential user's cognitive style and view of his job are key constraints on system design. DSS oriented towards specific roles are designed with much less attention to individual user preferences. Rather, they attempt to support an organizational definition of appropriate decision making behavior for

people holding certain positions. Problem focussed DSS are also organizationally defined, but the concern here is with how certain problems should be solved, regardless of who is doing the problem solving. The principal implication of these differences in focus is the source of the models and data which are used to design the system. While normative decision modeling is a part of the development process for any DSS, these normative models must be tempered by the needs of the specific setting. The focus of the system determines where we must look to define those needs.

DSS differ from most conventional computer-based systems in their orientation towards change. Conventional systems, for the most part, attempt to avoid change, to maintain the status quo in the organization. DSS, on the other hand, are change inducing; they attempt to alter the way people or organizations define and solve problems. As a result, substantial attention must be paid to defining the organizational changes which are required and to assuring that these changes in fact occur. This implies a very different role for the DSS designer from that common to designers of conventional systems. Keen and Scott Morton [1978] illustrate this difference by contrasting two designer behavior patterns -- change agent vs. technician. The substantial change requirements also imply a need for more comprehensive training activities than are normally provided in conventional system development efforts (see Ginzberg [1978]).

Exhibit 4 summarizes these attributes of DSS development. Two characteristics, normative modeling and change induction, do differentiate the DSS development process from that for conventional systems. The requirement of user involvement is similar for DSS and other systems. And,

DSS DEVELOPMENT PATTERNS

- Normative decision modeling
- User involvement (to the extent possible)
- Flexible design process
- Individual, role, or problem focused
- Change inducing

Exhibit 4.

while both type and focus of the design process are important aspects of DSS development, both can vary substantially; thus, neither provides a way to uniquely characterize all DSS.

IV. Summary, Trends, and Directions for Research

This paper has presented a rather broad definition of DSS, one that does not attempt to limit the range of such systems by requiring that specific components be included nor that specific usage or development patterns be followed. We feel this broad definition is appropriate, since it leads us to focus on the central issue in DSS -- the decision process and how that process can be supported.

In this final section we shall review some developing trends in computer technology and examine their likely impacts on DSS. Finally, we shall identify some of the key directions for research that is needed to improve the quality of future DSS.

First, we turn to technological trends. Continued advances in all aspects of hardware, including higher CPU speeds, will help extend the range of structurable decision situations by making more sophisticated heuristics and, in particular, artificial intelligence applications feasible. However, optimal solutions to certain 'difficult' management science problems are likely to remain computationally infeasible. Communications technology -- international, national and local networks, distributed data bases, distributed processing, electronic mail, teleconferencing -- will increase the opportunities for coordination of geographically dispersed activities and for collaborative decision making. An example of the use of communications capability in a DSS is provided by

the Hertz system (see the article by Edelstein and Melnyk in this volume).

The trend towards the automation of office activities brings the man-machine interface closer to general management. The initial emphasis seems to be in providing clerical support using form-driven systems (de Jong, [1980]). This could be followed by systems that support management by providing an 'electronic file cabinet', meeting schedulers, reminders, electronic mail and telephone messages, etc. (Wohl, [1980]). From here it is but a short step to providing the capability for decision support as defined in this paper.

Another hardware trend is the development of cheap and powerful microcomputers and their rapid acceptance by both large and small businesses. This greatly increases the availability of systems that can support decision making. An example is the popular VISICALC system, available on both APPLE and TRS-80 microcomputers, which supports spread-sheet accounting and performs functions similar to DSS financial planning generators. Finally, we should note the increasing availability of devices that support the user interface - graphics terminals, voice recognition, and voice synthesis.

In summary, technological advances will increase the effectiveness of DSS. Computational power will be more readily available and will migrate away from the central DP shop towards the locus of decision making, increasing user familiarity with computers and providing more opportunities for the application of DSS.

These hardware advances will become available to DSS developers in the next few years, but by themselves, they will have little impact on the quality of DSS. Two equally important areas are advances in software and in our understanding of decisions and decision making. While the hardware advances will be made largely outside the DSS community, many of the needed gains in software capability and decision process understanding will have to be made by DSS researchers and practitioners.

In the software area, three types of development seem particularly important. The first is Model Management Systems. These systems are in an early stage of development, and substantial progress in this area seems both necessary and likely. The second is the incorporation of artificial intelligence techniques in DSS software. These techniques have a number of potential uses within DSS, including making the user interface "smarter" and more flexible, and providing better support for the intelligence and design phases of problem solving. The final area where software development is needed is in the Data Base Management System, providing facilities for managing highly dynamic data bases.

We conclude this paper very near where we began, by turning again to the decision making process. Our understanding of decisions and decision making remains quite limited. We need better models of specific decision situations and taxonomies which explain in a meaningful way the similarities and differences across decisions. We need measures of decision effectiveness. Ultimately, our progress in developing better DSS will be limited by how well we understand the needs of decision makers.

REFERENCES

1. Alter, S.L., "How Effective Managers Use Information Systems," Harvard Business Review, Vol. 54, No. 6 (Nov. - Dec., 1976), pp. 97-104.

2. _____, "A Taxonomy of Decision Support Systems," Sloan Management Review, Vol. 19, No. 1 (Fall 1977), pp. 39-56.

3. _____, "Development Patterns for Decision Support Systems," MIS Quarterly, Vol. 2 , No. 3 (September, 1978), pp. 33-42.

4. _____, Decision Support Systems: Current Practices and Continuing Challenges, Reading, Massachusetts: Addison-Wesley, 1980.

5. _____, "Transforming DSS Jargon into Principles for DSS Success," presented to DSS-81, Atlanta, Georgia, June 8-10, 1981.

6. Berger, P. and F. Edelman, "IRIS: A Transaction-Based DSS for Human Resources Management," Data Base, Vol. 8, No. 3, (Winter 1977), pp. 22-29.

7. Blin, J.M., E.A. Stohr and M. Tanniru, "A Structure for Computer-Aided Corporate Planning," Policy Analysis and Information Systems, Vol. 2, No. 2 (June, 1978), pp. 111-139.

8. Bonczek, R.H., C.W. Holsapple and A.B. Whinston, "The Evolving Roles of Models in Decision Support Systems," Decision Sciences, Vol. 11, No. 2 (1980), pp. 339-356.

9. _____, _____ and _____, "Representing Modeling Knowledge with First Order Predicate Calculus," Operations Research, (forthcoming).

10. Boulden, J.B. and E.S. Buffa, "Corporate Models: On-Line Real-Time Systems," Harvard Business Review, Vol. 48 (July-August, 1970), pp. 143-154.

11. Buneman, O.P., H.L. Morgan and M.D. Zisman, "Display Facilities for DSS Support: The Daisy Approach," Database, Vol. 8, No. 1 (Winter 1977), pp. 46-50.

12. Carlson, E.D., J. Bennet, G. Giddings and P. Mantrey, "The Design and Evaluation of an Interactive Geo-Data Analysis and Display System," IFIP Congress 74 (1974), pp. 1057-1061.

13. _____, B.F. Grace and J.A. Sutton, "Case Studies of End User Requirements for Interactive Problem-Solving Systems," MIS Quarterly, Vol. 1, No. 1 (March, 1977), pp. 51-63.

14. Carroll, D.C., "Implications of On-Line, Real-Time Systems for Managerial Decision Making," in Science and Technology Series, Vol. 12, Tarzana, California: American Astronautical Society, 1967, pp. 345-370.

15. de Jong, S.P. and R.J. Byrd, "Intelligent Forms Creation in the System for Business Automation (SBA)," IBM Research Report, RC8599, 1980.

16. Donovan, J.J., "Data Base System Approach to Management Decision Support," Transactions on Data Base Systems, Vol. 1, No. 4 (December, 1976), pp. 344-369.

17. _____ and S.E. Madnick, "Institutional and Ad Hoc Decision Support Systems and Their Effective Use," Data Base, Vol. 8, No. 3 (Winter 1977), pp. 79-88.

18. Elam, J.J., J.C. Henderson and L.W. Miller, "Model Management Systems: An Approach to Decision Support in Complex Organizations," Proc. Conference on Informaton Systems, Philadelphia (December, 1980), pp. 98-110.

19. EXECUCOM, IFIPS Users Manual, EXECUCOM Systems Corporation, Austin, Texas, 1979.

20. Findler, N.V. (ed.), Associative Networks -- The Representation and Use of Knowledge in Computers, New York: Academic Press, 1979.

21. Gerrity, T.P., Jr., "Design of Man-Machine Decision Systems: An Application to Portfolio Management," Sloan Management Review, Vol. 12, No. 2 (Winter 1971), pp. 59-75.

22. Ginzberg, M.J., "Redesign of Managerial Tasks: A Requisite for Successful Design Support Systems," MIS Quarterly, Vol. 2, No. 7 (March, 1978), pp. 39-52.

23. _____, "An Organization Contingencies View of Accounting and Information Systems Implementation," Accounting, Organizations and Society, Vol. 5, No. 4 (1980), pp. 369-382.

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

25. Haseman, W.D. and M.I. Kellner, "Decision Support Systems: Their Nature and Structures," Modeling and Simulation, Vol. 8 (1977).

26. Hax, A.C. and H.C. Meal, "Hierarchical Integration of Production Planning and Scheduling," in Studies in the Management Sciences, vol. I, M.A. Geisler (ed.), North-Holland, 1975.

27. Henderson, J.C. and R.S. Ingraham, "Prototyping for DSS: A Critical Appraisal," Proceedings of the NYU Symposium on Decision Support Systems, New York, May 21-22, 1981.

28. Holsapple, C.W. and Whinston, A.B., "A Decision Support System for Area-wide Water Quality Planning," Socio-Economic Planning Sciences, Vol. 10, (1976).

29. Keen, P.G.W., "Adaptive Design for Decision Support Systems," Data Base, Vol. 12, Nos. 1 and 2 (Fall 1980).

30. _____ and R.D. Hackathorn, "Decision Support Systems and Personal Computing," Working Paper No. 79-01-03, The Wharton School, 1979.

31. _____ and M.S. Scott Morton, Decision Support Systems: An Organizational Perspective, Reading, Massachusetts: Addison-Wesley, 1978.

32. Little, J.D.C., "Models and Managers: The Concept of A Decision Calculus," Management Science, Vol. 16, No. 8 (April, 1970), pp. B466-B485.

33. Lucas, H.C., Jr., Information Systems Concepts for Management, New York: McGraw-Hill, 1978.

34. Mayo, R.B., Corporate Planning and Modeling with SIMPLAN, Reading, Massachusetts: Addison-Wesley, 1979.

35. Mills, R.E., R.B. Fetter and R.F. Averill, "A Computer Language for Mathematical Program Formulation," Decision Sciences, Vol. 8 (1977), pp.427-444.

36. Mintzberg, H., "Managerial Work: Analysis From Observation," Management Science, Vol. 18, No. 2 (October, 1971), pp. B97-B110.

37. Moore, J.H. and M.G. Chang, "Design of Decision Support Systems," Data Base, Vol. 12, Nos. 1 and 2 (Fall 1980), pp. 8-14.

38. Naylor, T.H. and H. Schauland, "A Survey of Users of Corporate Planning Models," Management Science, Vol. 22, No. 9 (1976), pp. 927-937.

39. Nilsson, N.J., Problem-Solving Methods in Artificial Intelligence, New York: McGraw-Hill, 1971.

40. Schneiderman, B., Software Psychology: Human Factors in Computer and Information Systems, Cambridge, Massachusetts: Winthrop, 1980.

41. Schultz, R.L. and D.P. Slevin, "A Program of Research on Implementation," in R.L. Schultz and D.P. Slevin (eds.), Implementing Operations Research/Management Science, New York: American Elsevier, 1975, pp. 31-52.

42. Scott Morton, M.S., Management Decision Systems: Computer-Based Support for Decision Making, Boston: Division of Research, Graduate School of Business Administration, Harvard University, 1971.

43. Sprague, R.H., Jr., "Characteristics of Decision Support Systems," Computing Newsletter for Schools of Business, Vol XIII, 1980.

44. _____, "A Framework for the Development of Decision Support Systems," MIS Quarterly, Vol. 4, No. 4 (December, 1980), pp. 1-26.

45. _____ and H.J. Watson, "A Decision Support System for Banks," OMEGA, Vol. 4 (1976), pp. 657-671.

46. Stabell, C.B., "On Defining and Improving Decision Making Effectiveness," Research Paper No. 289, Graduate School of Business, Stanford University, (July, 1977).

47. Stohr, E.A. and M. Tanniru, "A Data Base for Operations Research Models," Policy Analysis and Infomation Systems, Vol. 4, No. 1 (March, 1980), pp. 105-121.

48. Turner, J.A., "Computers in Bank Clerical Functions: Implications for Productivity and the Quality of Working Life," unpublished Ph.D. dissertation, Columbia University, 1980.

49. Wohl, A.D., "Replacing the Pad and Pencil," Datamation, Vol. 26, No. 6 (June, 1980), pp. 169-176.

50. Yonke, M.D. and N.R. Greenfield, "An Information Presentation System for Decision Makers," Data Base, Vol. 12, Nos. 1 and 2 (Fall 1980), pp. 26-32.