DSS DESIGN: A SYSTEMIC VIEW OF DECISION SUPPORT

A systemic view of DSS can provide a concrete framework for effective design of DSS and can also serve as a basis for accumulating DSS research results.

GAD ARIAV and MICHAEL J. GINZBERG

DSS studies have typically focused on a single set of related issues. While some studies have dealt extensively with the nature of decision situations and with the type of services provided by a DSS [14, 22], others have examined components, tools, and technologies that are needed to provide decision support services [7], and still another group has emphasized the processes of DSS design, implementation, and use [17, 24]. Yet, there has been no effective integration of these diverse elements to date, and this lack of a unified approach to the subject has hampered efforts to develop a solid basis for DSS design. Research focusing solely on components or resources can offer only limited help in the design of DSS. Recent books on DSS design [4, 6, 27] have proposed a basic set of DSS components, but do not make explicit the necessary contingent relationships between the structural aspect of a system and the services it is expected to provide or the characteristics of the environment in which it will operate.

The premise of the systemic view of DSS is that in order to understand these systems the following five aspects must be considered simultaneously: environment, role, components, arrangement of components, and the resources required to support the system [9]. A meaningful DSS design must explicitly link these five aspects so that characteristics of the system's environment and role will be reflected in its components and their arrangement.

As the name DSS implies, the objects we are discussing are indeed systems, and yet this perspective has been lost in much of the DSS literature. The purpose of this article is to present a comprehensive view of DSS using the systemic framework as an organizing concept. No new terminology will be introduced. Instead, we will use the concepts of systems theory to integrate the disparate perspectives in the DSS literature into a consistent and coherent body of knowledge. This integrated view will then help to provide new insights for DSS design, research, and curricula.

THE ASPECTS OF A SYSTEM

The fundamental premise of systems theory or the systems approach is that systems, regardless of their specific context, share a common set of elements [9]. Briefly, the system elements are the following:

- The *environment* is the set of entities and conditions outside of the system boundary that affects the system or is affected by it. The entities in the environment may be affected by the system, but are not controlled by it.
- The *role*, function, or objective of a system represents its intended impact on its environment. It specifies

which services the system is supposed to deliver and what its goals are. It also provides the basis for evaluating the system and thus should be specified in terms that are amenable to measurement.

- The *components* of the system are the identifiable elements within the system boundary. Typically these components represent the functional building blocks of the system. Two common bases for component definition are division of labor and specialization by environmental segment. The former relates to the ability to perform effectively a particular, necessary task, while the latter relates to the ability to interface with a particular aspect of the environment.
- *Arrangement* concerns the links among the system components and between them and the environmental elements. The fundamental concern in arranging components is the balance between coordination and autonomy [10]. Generally it is preferable to have the minimum of interdependence among components, which still allows the system as a whole to serve its function. The two key dimensions of arrangement are the configuration or layout of the links among components, and the nature of these links.
- System *resources* are the elements that are used or consumed in building and operating the system. Like the environment, resources exist outside the system boundary. Resources are differentiated from the environment in that they are partially controllable—if not in total quantity available, at least in the mix that will be used. Resources may include people, raw materials, capital, tools and techniques, etc.

Systems thought emphasizes the need to take a holistic view in order to explain why an object is structured as it is, or how it should be structured. Thus, the system study starts from the outside, identifying the environment in which the object exists and the way it impacts that environment—that is, its role. Only after these external aspects of the system have been studied does it make sense to consider the components and their arrangement.

An intelligent selection of resources cannot be made until the other four system aspects have been considered. Resources are differentiated from system components and should be discussed last to ensure that the functional composition of the system is not unduly biased by the perceived availability of resources. Design should proceed from an "ideal" system to a critical review of resource availability, and then to a feasible system [1].

Let us now examine the five system aspects in the DSS context.

DSS ENVIRONMENT

The description of a DSS environment should be design relevant; it should highlight only those environmental features that have, or should have, an impact on the system structure. There seem to be at least two important dimensions to a DSS environment—task characteristics and access pattern.

Task Characteristics

Task characteristics have been discussed frequently in the DSS literature, but a somewhat broader range of task aspects is needed.

The task characteristic most frequently associated with DSS is the degree to which the decision maker can apply predefined rules and procedures—that is, task structurability [12]. Gorry and Scott-Morton [14] used task structure as a key concept for defining the appropriate environment for DSS. More recent work has suggested that there is no inherent structure to the task itself, only structure as perceived by individuals [24]. Therefore, instead of task structure, the characteristic of interest should be task structurability, the possibility of bringing structure to bear on a task, which is dependent upon both the individual performing the task and the task itself.

A second key task characteristic is its level—that is, operational control, management control, or strategic planning [3]. This characteristic has also been discussed in much of the "classic" DSS literature [14], and the suggestion has often been made that DSS might be appropriate only at higher levels. More recent DSS literature claims that DSS may be appropriate at any level [11, 13, 27].

The third characteristic is the decision-process phase—intelligence, design, or choice [26]. Most DSS have been used at the solution-selection and choice phase, or, to a lesser extent, the intelligence and problem-definition phase of the decision-making process. DSS, however, are also applicable to the design and alternative-generation phase, and the pattern of use to date mainly reflects the difficulty of designing systems to support this phase.

The fourth is the functional area of the application—for example, finance, marketing, and production. The differences in the demands and constraints that different functional areas place on a DSS may be as significant as those resulting from any of the other task dimensions.

Access Pattern

The access pattern encompasses three major concerns in DSS design. The first of these is the mode of user interaction. The early DSS literature assumed that use would always be highly interactive; that the decision maker would interact with the DSS as a decision was being made [14]. Although some DSS are used in this manner, it has become clear that at least as many others are not [12, 16]. User interaction with a DSS can range from true on-line dialogue to intermittent batch use.

The access pattern also captures the salient "dimensions" of the user community, including the number of people who use the system, their expertise in computer usage or the problem area, and their role in the decision process. One dimension of the user community, which has been discussed in the literature but which we do not include at this time, is cognitive style. We agree with Huber's assessment [15] that not enough is

known about the impact of cognitive style to make design-relevant prescriptions.

Finally there is the relationship to "neighboring" information systems. A DSS may interface with other computer-based systems to acquire its source data or dispose of its output. Most early DSS were entirely freestanding and had no direct interactions with any other systems, but this pattern has begun to change, and many DSS now require access to large operational databases [5, 21]. This trend is likely to continue as more complex DSS drawing on common data and model bases are developed. Each system in such a network, in addition to producing data that other systems could also use, might also require data produced by other systems.

DSS ROLE AND FUNCTION

The objective or purpose of DSS has long been defined as support of a decision-making process. There are, however, many ways to support decision making. Three important dimensions of support are level, decision range, and process.

Alter identifies several DSS types that provide users with different levels of support [2], ranging from retrieving and displaying raw data to suggesting or selecting solutions. There is a hierarchical relationship among the levels of decision support with each higher level containing and adding to the previous levels.

Decision range refers to the degree to which support is generalized versus particularized. DSS can provide support that is tailored to a particular problem, or even to a specific individual's view of a particular problem. At the other extreme, DSS can provide general analytic capabilities that will support multiple decision makers in related problem contexts.

A third dimension of support concerns the process that is supported. Most DSS literature focuses on supporting individual cognitive processing capabilities [11] or on facilitating learning [18]. Other decision-related processes can be supported, including communication or coordination among parties involved in the decision process, and the exercising of control or influence over the outcome of a decision process [12].

DSS COMPONENTS

Components represent a functional breakdown of the system and should not be confused with modules or formal subsystems. The assignment of system functions to specific software modules is mainly a question of arrangement and resource allocation.

The DSS design literature [7, 27] identifies three major functions or conceptual components necessary for a DSS: management of the dialogue between the user and the system, management of data, and management of models (see Figure 1).

The most pervasive and fundamental aspect of the DSS environment is people, and the dialogue management component embodies the specialized functionality necessary to handle the system's interaction with its users. The data management component

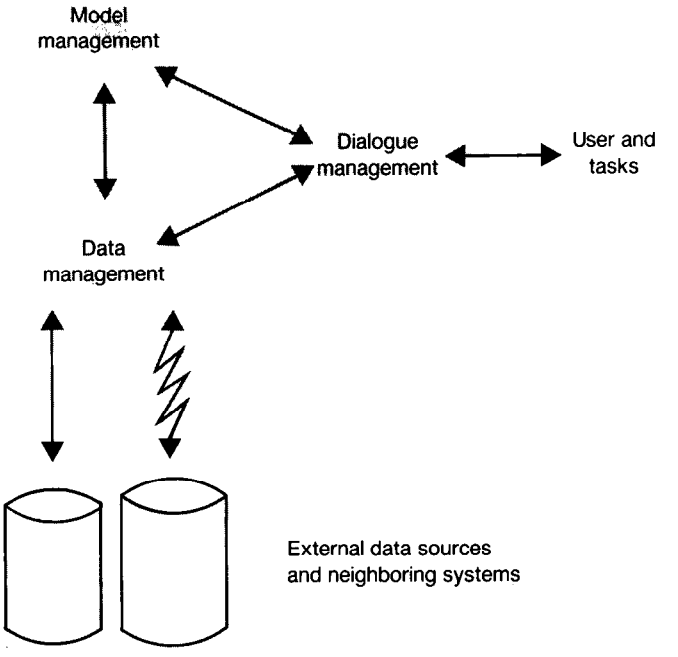

FIGURE 1. DSS Design: Functional Components

reflects a fundamental aspect of the role of DSS—all levels of decision support are based on access to a set of data. The need for a model management functionality is derived from the nature of the tasks to which DSS are applied, that is, tasks that are only partially structurable and consequently require the manipulation of underspecified, evolving models.

While there is general agreement about the three major functional components, there is less agreement about their specific content. Our view is closer to that of Sprague and Carlson [27], although it does incorporate some features of Bonczek et al. [7]. Later in this article we will examine major relationships between the various components and other aspects of the system, especially environmental conditions or constraints.

The dialogue between the user and the system establishes the framework in which outputs are presented as well as the context for user inputs. This suggests three necessary dialogue management capabilities (see Figure 2):

1. a *user interface* to handle the syntactic aspects of the interaction (e.g., the devices, the physical view, and the style of interaction);
2. a *dialogue-control* function to determine the basic semantics of interactions and maintain the interaction context, which could range between strictly system defined or loosely "user driven";

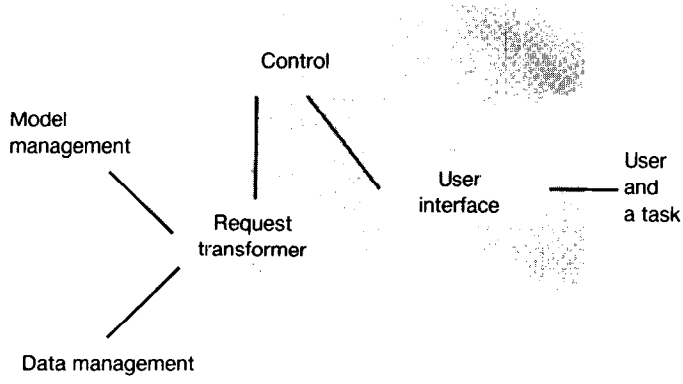

FIGURE 2. Dialogue Management

3. a *request transformer* to provide the necessary (two-way) translations between users' vocabulary and the system's internal modeling and data access vocabulary.

The management of data—that is, the ability to store, retrieve, and manipulate data—is fundamental to any service that a DSS provides. The data management component maintains the factual basis (including possible links and associations) of the DSS. The specific capabilities required (see Figure 3) for the management of data in a DSS are

1. a *database* and a *database management system* (DBMS) to provide an access mechanism to data in it;
2. a *data directory* to maintain data definitions and descriptions of the types and sources of data in the system;
3. a *query facility* to interpret requests for data (from either of the other major components), determine how these requests could be filled (consulting, possibly, the data directory), formulate the DBMS-specific requests for data, and, finally, return the results to the issuer of the original request;
4. a *staging and extraction* function for accessing external sources of (especially historic) data, and connecting the DSS with its relevant neighboring systems (databases or possibly multiple other DSS, either personal or centralized).

The mechanism for explicit management of models and modeling activity is what distinguishes DSS from more traditional information-processing systems. The ability to invoke, run, change, combine, and inspect models is a key capability in DSS and therefore a core service. Any support beyond direct access to raw data requires the application of a model. In particular, inferential retrieval of data from the database [7] is achieved through a model-driven process. The state of development of model management functionality still lags far behind the other two components, but is an active research area [20, 23].

The ideal model management facility (see Figure 4) should provide

1. a *model-base management system* (MBMS) to generate, retrieve, and update parameters, to restructure models, and to include a "model directory" for maintaining information about available models;
2. *model execution* to control the actual running of the model and to link models together when integration is needed;
3. a *modeling command processor* to accept and interpret modeling instructions as they flow out of the dialogue component, and to route them to the MBMS or the model-execution function;
4. a *database interface* to retrieve data items from the database for running models, and, eventually, to store model outputs in the database for further processing, perusal, or as input to other models.

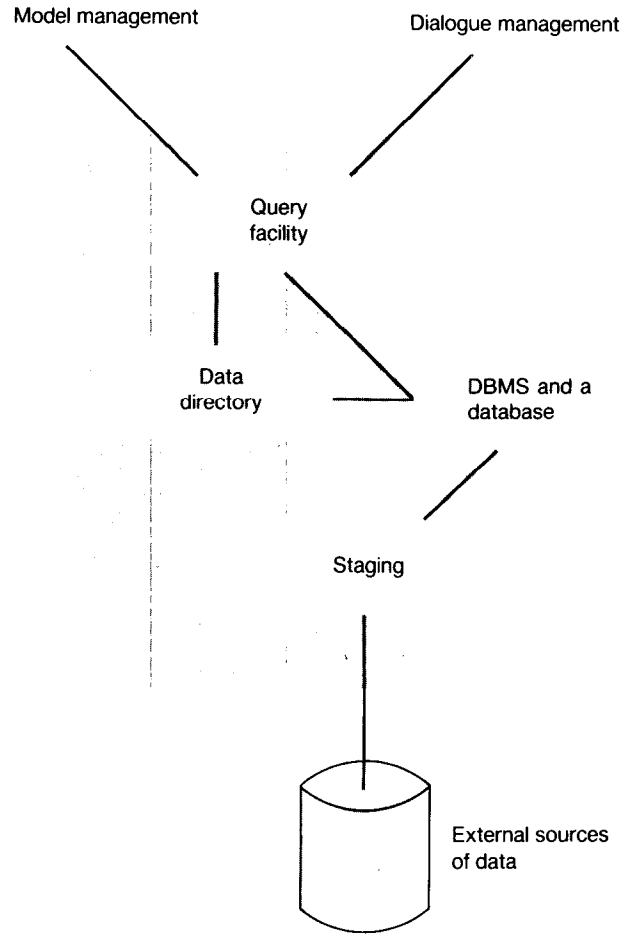

FIGURE 3. Data Management

ARRANGEMENT OF DSS COMPONENTS

The systemic view of DSS suggests that architecture (component arrangement), like components, has to be justified in the broader context of environment and role. Moreover, architecture cannot be independent of the available resources. For example, some DBMS products already contain a dialogue management function, eliminating the explicit interaction between such separate components.

In general the linkages among the DSS components, the nature of these linkages, and especially the justification for them in terms of environment and role have been treated in an extremely limited fashion. Sprague and Carlson introduce four generic architectures: the "Network," the "Bridge," the "Sandwich," and the "Tower" [27], but their discussion of these architectures completely fails to embed them in the overall context of the DSS, and the appropriateness of each arrangement in various decision situations is never analyzed. In addition, the advantages and drawbacks of the four architectures are related solely to ease of construction and to the internally oriented concerns of system engineering. In fact, two of these "architectures," the Bridge and the Network, are merely specific implementations of the generic DSS structure shown in Figure 1. Treating these as distinct architectures serves only to confuse the notion of components (needed functionality) and resources (the way functionality is provided).

The section below on DSS design includes a discussion of some major links between DSS environment, role, and architecture.

DSS RESOURCES

It is only after the DSS has been designed—after the components and their "ideal" arrangement have been selected—that resources should be considered. The key questions are, How can the proposed DSS best be realized? How close to that ideal can a feasible system come? Which resources should be employed to build and support the DSS?

The resources available for DSS fall into four major categories: hardware, software, people, and data. Hardware includes processors, terminals, storage media, and communication networks. None of these is unique to DSS. Some linkages, however, can be made between DSS environment and role, and hardware configuration. These are discussed in the section on DSS design.

There are four types of DSS software: general-purpose programming languages, DSS tools, DSS generators, and generalized DSS. Ultimately, all DSS software is built upon general-purpose programming languages, and any DSS can be written using only this type of language. However, general-purpose programming languages, or even very high-level languages like APL, provide only limited leverage for the development of DSS.

DSS tools are single-function building blocks that can be used to construct DSS; that is, they address only one of the major DSS components/functions. Four key types of DSS tools are DBMS, model management systems, dialogue management systems, and arrangement packages. Arrangement packages, often referred to as "software environments" or "windowing packages" [8], address the interfaces among the other subsystems, but not the content of any of them.

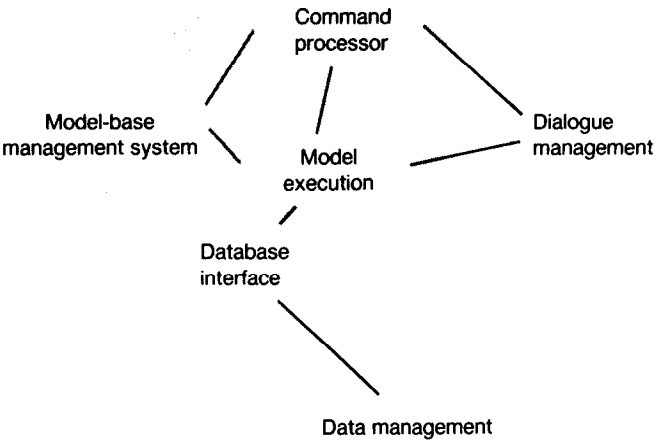

FIGURE 4. Model Management

DSS generators are, in essence, a collection of DSS tools. They are software packages that address all three DSS functionalities (at least to some extent) and that can be used to construct DSS tailored to specific problems or problem situations. A typical example of a DSS generator is a modeling package (such as EXPRESS, EMPIRE, or IFPS) that includes some data management capabilities as well as dialogue/presentation facilities.

Generalized DSS provide support for a class of problems. For example, a generalized DSS for scheduling and assignment is conceivable in which professors could be assigned to classes, operators to shifts, or specialists to projects. All of these scheduling decisions share a similar structure and, more importantly, a model base. Generalized DSS (e.g., PERT/CPM systems) fall between DSS generators and specific DSS in terms of generality and the amount of effort required to apply them to specific problems. They provide capabilities that can be applied directly to a decision problem, and do not require the extensive customization and development needed with a DSS generator. Their generality, however, means that some of the linkages and transitions that would be built into a specific DSS must be performed manually. Generalized DSS are becoming an increasingly popular mechanism for providing decision support as more and more people want decision-support tools, but specifically tailored DSS either are not readily available or are too expensive in resource-constrained situations.

For a DSS, the people resource is conceptually differ-

ent from the user-community dimension of the system's environment. This resource provides one of the clearest examples of the substitutability of alternative resources in DSS design. It is almost always necessary to acquire data from other systems for the DSS. This staging mechanism can be a hardware bridge, a hardware/software bridge, or a human bridge. The issue is to recognize that there are many roles in developing and operating a DSS that can, but need not, be played by humans; the choice should depend on the availability and relative cost of both human and other resources.

Data resources include the various sources of data available to the DSS. This includes internal, operational data as well as externally available data. The latter could come from generally accessible databases or may be the result of special studies.

DSS DESIGN: LINKING THE SYSTEM ASPECTS

In this section we will discuss some of the major contingencies between a DSS's internal structure and its role, environmental conditions, and resource availability. More exhaustive identification and validation of such linkages are clearly needed.

Task Structurability

Task structurability directly impacts the model component of the DSS. Models can be maintained in the MBMS in a variety of forms, ranging from subroutines (where the model base is a software library), to more abstract and manipulable forms, where models are treated like data [19]. When tasks are highly structurable, procedural model specification (e.g., as subroutines) is appropriate, whereas low structurability suggests a more flexible, declarative, and "open-ended" form of model definition (e.g., production rules).

Low structurability also suggests that the model-execution function should interact directly with the dialogue component, both for eliciting user-provided parameters and for handling user interventions (e.g., intermittent perusal of variable values). A related issue is the design of the dialogue-control function: "System-prompted" dialogue fits situations with relatively high levels of structure, whereas "user-driven" dialogue is appropriate where no predefined sequence of activities can be specified.

Task Level

One obvious impact of task level is on the needed data resources and on the facilities for accessing them—that is, on the staging function in the data management component. Operational control tasks are likely to require access to current operational data. The DSS, then, must provide an on-line linkage to data files maintained by operational systems. Strategic planning tasks, on the other hand, may require access to external data sources, but probably not on a continuously on-line basis. These DSS must include a mechanism for remote access and for selective extracting of data from extra-organizational sources.

Task level may interact with the type of support to impact the selection of hardware resources. A stand-alone microcomputer-based DSS may be an appropriate support system for senior managers in a strategic decision situation (e.g., new product selection)—little data are needed, the data do not come from the operational stream, and the system is meant primarily to provide cognitive support to individual managers. On the other hand, providing support for real-time operational decisions (e.g., production control) in a complex production environment would best be accomplished on a mainframe-based DSS with good data communications capabilities. This type of support requires a substantial quantity of quite volatile, operational data, and the primary purpose is to assure coordination among the multiple parties involved in the decision.

Decision Process Phase

The decision-process phase can impact the need for a directory. The data directory in a DSS provides the basis for answering questions about the availability of data items, their sources, and their exact meanings. As such it is most important in systems that support the intelligence phase of the decision-making process, where data exploration is of paramount concern.

Functional Area

The functional area determines the set of verbs and objects useful in the specific problem-solving situation. Both the request transformer and the dialogue control should reflect this user vocabulary.

Modes of User Interaction

The desired interaction mode has obvious implications for both resources and the arrangement of components. On-line interaction, for example, limits the selection of employable resources and calls for tighter intermodule linkages (immediate exchange of messages is necessary), while batch-oriented design can use data files on secondary storage as a medium for message exchange.

User Community

The design of the user interface should be determined by the nature of the user community. Criteria such as the proficiency of users and frequency of use should be the basis for selecting the interaction style the interface will accommodate (e.g., menu-driven or Q/A-driven dialogue).

Relationship to Other Computer-Based Systems

Neighboring systems have the greatest effect on the staging function—the DSS structure should directly reflect the nature of the data sources in its environment. For example, if remote databases are included, some data-communication facility is necessary. Likewise, the selection of a data model—the data structures in the

database and valid operations on it—is contingent upon the structure of the available external data.

Level of Support

As noted earlier, access to data underlies all DSS roles. Levels of support differ, however, in the degree of model intensity they imply. Consequently, the arrangement of DSS components—the nature and layout of linkages among components—should be contingent upon the model intensity of the service to be provided. The Sandwich architecture [27] forces all access to data to be mediated by a model by removing the linkage between dialogue management and data management (c.f. Figure 1). This is appropriate in the higher levels in Alter's classification of decision support [2] where models are more dominant and, in fact, drive the decision process. Less model-intensive support is better provided by a different architecture in which the direct linkage between model management and data management is removed. This architecture, which we call "Exploratory," requires the user to direct all data movements between models and the database.

IMPLICATIONS OF THE SYSTEMIC VIEW

The fact that a number of interesting issues are indeed raised demonstrates the usefulness of the systemic view. Perhaps the clearest implication of the systemic view for DSS design is the necessity of taking an "outside-in" approach. The selection of components and their arrangement (the "inside") must follow from an understanding of the environment and role (the "outside"). Unfortunately, recent books on DSS design have provided only limited guidance for relating environmental conditions to the specific system components or to their arrangement. The focus of early DSS literature [14] on the system's environment and role was critical. These elements are central to explaining why DSS are different from traditional information systems and why they will probably be built differently. More recent DSS literature has focused on resources, components, and architecture (arrangement) [6, 27]. It is only when all of these elements are considered in the context of environment and role that true understanding and insight can be developed. A change from "coming up with designs" to explicit derivation and justification of components and architectures is needed in DSS research strategy. Such a change does not imply that the study of single system elements should be avoided, but it does suggest that in any study we must first understand the context.

Future DSS research should move in the direction of gaining a better understanding of the range of DSS environments and roles. Although the practical use of DSS has permeated a substantial array of decision environments—from labor negotiations to military combat situations—the major share of DSS research to date has focused on a limited set of environmental conditions, typically, the intelligence and choice phases of organizationally isolated, financial decisions. The discussion in the preceding section of this article is a first step toward an explicit DSS design theory in which design is contingent on comprehensive notions of environment and role. Further work is needed, however, to develop a design-relevant taxonomy of DSS environments, identifying key environmental characteristics and the constraints they place on the choice of DSS components, their arrangement, and the resources they require.

One environmental characteristic that is likely to be increasingly important for future DSS is the other systems with which the DSS must interact. There has been only a preliminary attempt to examine DSS that interface directly with other computer-based systems [27]. The trend toward interconnected systems, however, is quite clear [28], and many DSS in the future will be required to interact with other DSS, traditional information systems, etc. Research is still needed to understand both the range of potential interaction patterns and the requirements and constraints they will place on DSS design and development.

The systemic view sharpens the distinction between system components and resources. It suggests a methodology for evaluating resources and a basis for making educated decisions about the consumption of these resources. Given the popular marketing strategy of labeling every product a DSS, it is important that a more critical approach be adopted by potential users. Such an approach is not meant to discredit one product or another, but rather to identify the capabilities of each and to serve as a guideline in selecting resources and matching them with the required functionality of the DSS. The four levels of software resources discussed earlier in this article represent differing degrees of comprehensiveness and integration. Choice among them would depend, at least in part, on the availability of complementary design and development skills.

Recognition of the fact that resources constitute only one aspect of the system suggests that a modification in the focus of DSS curriculum is needed. Too many DSS courses are built around a specific tool or tools. This kind of focus does not give students an understanding of DSS or provide them with a basis for being educated consumers of DSS tools. As an alternative we have used the systemic view as an organizing framework for a DSS course. In it the analysis of the decision situation serves as the basis for the development of an "idealized" system design, against which available resources can be assessed and critically examined. Resources (available packages or generators) enter the curriculum only at its final stages, and the focus at that point is to evaluate the trade-offs between functionality and ease of construction—the core of any decision concerning the employment of resources.

Most DSS literature so far has emphasized the differences between DSS and other, more traditional computer-based systems. The systemic view helps us see the similarities. Once we have adopted this perspective, it becomes apparent that DSS differ from traditional systems in degree and not in fundamental

structure. For instance, large-scale DSS and "institutional DSS" are very similar to classical MIS both in terms of environment (e.g., anonymous user) and the constraints this places on design (e.g., run-time efficiency). This suggests that there is much that DSS researchers and developers can learn from prior experience with other types of computer-based systems, and that a cumulative experience across types of information systems is both possible and desirable.

Recognizing that DSS differ from other systems only in degree also implies that evolving types of computer-based systems might be similar to DSS and should not be treated as totally novel. A case in point is expert systems (ES). We have observed that some ES developers are treating these systems as a completely new phenomenon and are struggling afresh through the classical problems of MIS/DSS development [25]. The systemic view makes it clear that ES bear many similarities to DSS. Their environments and roles are quite similar, and it is mainly a change in the arrangement and resources that differentiates them. It is therefore inappropriate to discard the very relevant experience with DSS.

CONCLUSION

One may argue that the systemic view has been embedded in DSS all along. Although this may be true, this view has receded too far from the surface. Adopting the systemic approach means giving emphasis to the relationships among the system aspects, a subject that has received only limited attention from both the practical and the theoretical point of view, even though it is the essence of design activity.

There has been a migration of DSS definitions from an environment/role focus to a component/arrangement focus. The systemic view responds to that "tension" and brings the two under a single comprehensive framework.

Acknowledgment. The authors want to thank Gordon Davis for his many helpful comments on earlier versions of this paper.

REFERENCES

1. Ackoff, R.L. *Creating the Corporate Future.* John Wiley, New York, 1981.
2. Alter, S. *Decision Support Systems: Current Practices and Continuing Challenges.* Addison-Wesley, Reading, Mass., 1980.
3. Anthony, R.N. *Planning and Control Systems: A Framework for Analysis.* Graduate School of Business Administration, Studies in Management Control, Harvard University, Cambridge, Mass., 1965.
4. Bennett, J.L., Ed. *Building Decision Support Systems.* Addison-Wesley, Reading, Mass., 1983.
5. Berger, P., and Edelman, F. IRIS: A transaction-based DSS for human resources management. *Database 8,* 3 (1977), 22–29.
6. Bonczek, R.H., Holsapple, C.W., and Whinston, A.B. *Foundations of Decision Support Systems.* Academic Press, New York, 1981.
7. Bonczek, R.H., Holsapple, C.W., and Whinston, A.B. The evolution from MIS to DSS: Extension of data management to model management. In *Decision Support Systems,* M.J. Ginzberg, W.R. Reitman, and E.A. Stohr, Eds. North-Holland, Amsterdam, 1982, pp. 61–78.
8. BusinessWeek. A fierce battle brews over the simplest software yet. *BusinessWeek* (Nov. 21, 1983), 114–115.
9. Churchman, C.W. *The System Approach.* Dell, New York, 1968.
10. Emery, J.C. *Organizational Planning and Control Systems: Theory and Technology.* Macmillan, New York, 1969.
11. Ginzberg, M.J. DSS success: Measurement and facilitation. In *Data-Base Management: Theory and Applications,* C.W. Holsapple and A.B. Whinston, Eds. Reidel, Hingham, Mass., 1983, pp. 367–387.
12. Ginzberg, M.J., and Stohr, E.A. Decision support systems: Issues and perspectives. In *Decision Support Systems,* M.J. Ginzberg, W.R. Reitman, and E.A. Stohr, Eds. North-Holland, Amsterdam, 1982, pp. 9–32.
13. Gorry, G.A., and Krumland, R.B. Artificial intelligence research and decision support systems. In *Building Decision Support Systems,* J.L. Bennett, Ed. Addison-Wesley, Reading, Mass., 1983, pp. 205–219.
14. Gorry, G.A., and Scott-Morton, M.S. A framework for management information systems. *Sloan Manage. Rev. 13,* 1 (Winter 1971), 55–70.
15. Huber, G.P. Cognitive style and DSS designs: Much ado about nothing? *Manage. Sci. 29,* 5 (May 1983), 567–579.
16. Keen, P.G.W. "Interactive" computer systems for managers: A modest proposal. *Sloan Manage. Rev.* (Fall 1976), 1–17.
17. Keen, P.G.W. Adaptive design for decision support systems. *Data Base 12,* 1–2 (Fall 1980), 31–40.
18. Keen, P.G.W., and Gambino, T.J. Building a decision support system: The mythical man-month revisited. In *Building Decision Support Systems,* J.L. Bennett, Ed. Addison-Wesley, Reading, Mass., 1983, pp. 133–172.
19. Konsynski, B. On the structure of a generalized model management system. In *Proceedings of the 14th Annual Hawaii International Conference on System Sciences* (North Hollywood, Calif., Jan.). Univ. of Hawaii, 1980, pp. 19–31.
20. Konsynski, B., and Dolk, D.R. Knowledge abstractions in model management. *DSS-82 Trans.* (June 1982), 19–31.
21. Laning, L.J., Walla, G.O., and Airaghi, L.S. A DSS Oversight—Historical Databases. *DSS-82 Trans.* (June 1982), 87–95.
22. Little, J.D.C. Models and managers: The concept of decision calculus. *Manage. Sci. 16,* 8 (Apr. 1970), B466–B485.
23. Miller, L.W., and Katz, N. Model management systems to support policy analysis. DS-WP 82-11-01, Decision Sciences Dept., Univ. of Pennsylvania, Philadelphia, Apr. 1983.
24. Moore, J.H., and Chang, M.G. Design of decision support systems. *Data Base 12,* 1–2 (Fall 1980), 8–14.
25. O'Conner, D.E. Using expert systems to manage change and complexity in manufacturing. In *Artificial Intelligence Applications for Business,* W.R. Reitman, Ed. Ablex, Norwood, N.J., 1984.
26. Simon, H.A. *The New Science of Management Decision.* Harper and Row, New York, 1960.
27. Sprague, R.H., Jr., and Carlson, E.D. *Building Effective Decision Support Systems.* Prentice-Hall, Englewood Cliffs, N.J., 1982.
28. Zmud, R.W. Large-scale interconnected information systems: Design considerations for promoting organization adaptation and organizational adaptability. In *Proceedings of the Conference on Large-Scale Interconnected Systems* (Athens, Ohio, Oct. 10–11). School of Business Administration, Univ. of North Carolina, Chapel Hill, 1983, pp. 139–149.

CR Categories and Subject Descriptors: D.2.1 [**Software Engineering**]: Requirements/Specifications; H.1.1 [**Models and Principles**]: Systems and Information Theory; H.4.2 [**Information Systems Applications**]: Types of Systems; J.1 [**Administrative Data Processing**]
General Terms: Design, Management
Additional Key Words and Phrases: decision support systems, systems design process, systems theory

Authors' Present Addresses: Gad Ariav, Computer Applications and Information Systems, Graduate School of Business Administration, New York University, 100 Trinity Place, New York, NY 10006; Michael J. Ginzberg, The Weatherhead School of Management, Case Western Reserve University, Cleveland, OH 44106.