CHAPTER 4
Decision Support Systems

ANDREW P. SAGE
George Mason University

1. INTRODUCTION

In very general terms, a decision support system (DSS) is a system that supports technological and managerial decision making by assisting in the organization of knowledge about ill-structured, semi-structured, or unstructured issues. For our purposes, a structured issue is one that has a framework with elements and relations between them that are known and understood within a context brought by the experiential familiarity of a human assessing the issue or situation. The three primary components of a decision support system are generally described as:

- *A database management system* (DBMS)
- *A model-based management system* (MBMS)
- *A dialog generation and management system* (DGMS)

The emphasis in the use of a DSS is upon providing support to decision makers to increase the effectiveness of the decision making effort. This need should be a major factor in the definition of

 Handbook of Industrial Engineering: Technology and Operations Management, Third Edition.
Edited by Gavriel Salvendy

requirements for and subsequent development of a DSS. A DSS is generally used to support humans in the formal steps of problem solving, or systems engineering (Sage 1992, 1995; Sage and Rouse 1999a; Sage and Armstrong 2000), that involve:

- *Formulation* of alternatives
- *Analysis* of their impacts
- *Interpretation* and selection of appropriate options for implementation

Efficiency in terms of time required to evolve the decision, while important, is usually secondary to effectiveness. DSSs are intended more for use in strategic and tactical situations than in operational situations. In operational situations, which are often well structured, an expert system or an accounting system may often be gainfully employed to assist novices or support the myriads of data elements present in these situations. Those very proficient in (experientially familiar with) operational tasks generally do not require support, except perhaps for automation of some routine and repetitive chores. In many application areas the use of a decision support system is potentially promising, including management and planning, command and control, system design, health care, industrial management, and generally any area in which management has to cope with decision situations that have an unfamiliar structure.

1.1. Taxonomies for Decision Support

Numerous disciplinary areas have contributed to the development of decision support systems. One area is computer science, which provides the hardware and software tools necessary to implement decision support system design constructs. In particular, computer science provides the database design and programming support tools that are needed in a decision support system. Management science and operations research provides the theoretical framework in decision analysis that is necessary to design useful and relevant normative approaches to choice making, especially those concerned with systems analysis and model base management. Organizational behavior and behavioral and cognitive science provide rich sources of information concerning how humans and organizations process information and make judgments in a descriptive fashion. Background information from these areas is needed for the design of effective systems for dialog generation and management systems. Systems engineering is concerned with the process and technology management issues associated with the definition, development, and deployment of large systems of hardware and software, including systems for decision support.

Many attempts have been made to classify different types of decisions. Of particular interest here is the decision type taxonomy of Anthony, which describes four types of decisions (Anthony 1965; Anthony et al. 1992):

1. *Strategic planning decisions* are decisions related to choosing highest level policies and objectives, and associated resource allocations.
2. *Management control decisions* are decisions made for the purpose of ensuring effectiveness in the acquisition and use of resources.
3. *Operational control decisions* are decisions made for the purpose of ensuring effectiveness in the performance of operations.
4. *Operational performance decisions* are the day-to-day decisions made while performing operations.

Figure 1 illustrates how these decisions are related and how they normatively influence organizational learning. Generally, low-consequence decisions are made more frequently than high-consequence decisions. Also, strategic decisions are associated with higher consequences and are likely to involve more significant risk and therefore must be made on the basis of considerably less perfect information than are most operational control decisions.

A decision support system should support a number of abilities. It should support the decision maker in the formulation or framing or assessment of the decision situation in the sense of recognizing needs, identifying appropriate objectives by which to measure successful resolution of an issue, and generating alternative courses of action that will resolve the needs and satisfy objectives. It should also provide support in enhancing the decision maker's abilities to assess the possible impacts of these alternatives on the needs to be fulfilled and the objectives to be satisfied. This analysis capability must be associated with provision of capability to enhance the ability of the decision maker to provide an interpretation of these impacts in terms of objectives. This interpretation capability will lead to evaluation of the alternatives and selection of a preferred alternative option. These three steps of formulation, analysis, and interpretation are fundamental for formal analysis of difficult issues. They are the fundamental steps of systems engineering and are discussed at some length in Sage (1991,

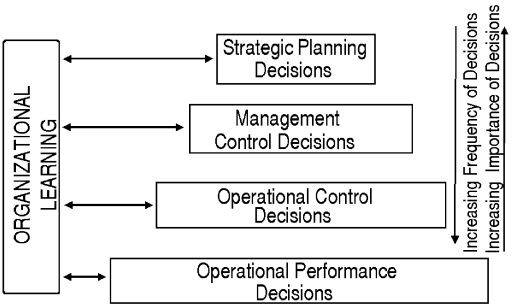

Figure 1 Organizational Information and Decision Flow.

1992, 1995), from which much of this chapter is derived. It is very important to note that the purpose of a decision support system is to support humans in the performance of primarily cognitive information processing tasks that involve decisions, judgments, and choices. Thus, the enhancement of information processing in systems and organizations (Sage 1990) is a major feature of a DSS. Even though there may be some human supervisory control of a physical system through use of these decisions (Sheridan 1992), the primary purpose of a DSS is support for cognitive activities that involve human information processing and associated judgment and choice. Associated with these three steps must be the ability to acquire, represent, and utilize information or knowledge and the ability to implement the chosen alternative course of action.

The extent to which a support system possesses the capacity to assist a person or a group to formulate, analyze, and interpret issues will depend upon whether the resulting system should be called a management information system (MIS), a predictive management information system (PMIS), or a decision support system. We can provide support to the decision maker at any of these several levels, as suggested by Figure 2. Whether we have a MIS, a PMIS, or a DSS depends upon the type of automated computer-based support that is provided to the decision maker to assist in reaching the decision. Fundamental to the notion of a decision support system is assistance provided in assessing the situation, identifying alternative courses of action and formulating the decision situation, structuring and analyzing the decision situation, and then interpreting the results of analysis of the alternatives in terms of the value system of the decision maker. In short, a decision support system provides a decision recommendation capability. A MIS or a PMIS does not, although the information provided may well support decisions.

In a classical management information system, the user inputs a request for a report concerning some question, and the MIS supplies that report. When the user is able to pose a "what if?" type question and the system is able to respond with an "if then" type of response, then we have a

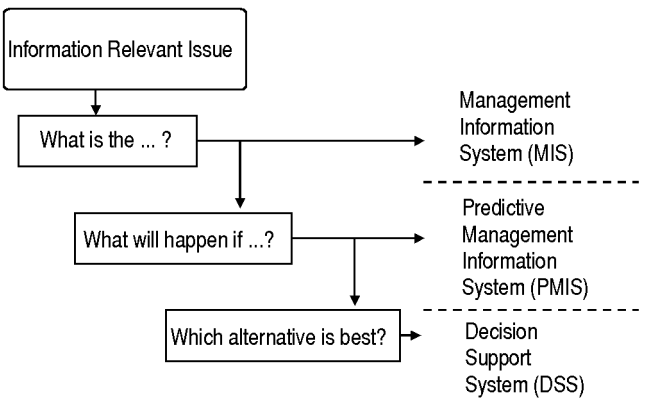

Figure 2 Conceptual Differences Between MIS, PMIS, and DSS.

predictive management information system. In each case there is some sort of formulation of the issue, and this is accompanied by some capacity for analysis. The classic MIS, which needs only to be able to respond to queries with reports, is composed of capabilities for data processing, structured data flows at an operational level, and preparation of summary reports for the system user. The predictive management system would also include an additional amount of analysis capability. This might require an intelligent database query system, or perhaps just the simple use of some sort of spreadsheet or macroeconomic model.

To obtain a decision support system, we would need to add the capability of model-based management to a MIS. But much more is needed, for example, than just the simple addition of a set of decision trees and procedures to elicit examination of decision analysis-based paradigms. We also need a system that is flexible and adaptable to changing user requirements such as to provide support for the decision styles of the decision maker as these change with task, environment, and experiential familiarity of the support system users with task and environment. We need to provide analytical support in a variety of complex situations. Most decision situations are fragmented in that there are multiple decision makers and their staffs, rather than just a single decision maker. Temporal and spatial separation elements are also involved. Further, as Mintzberg (1973) has indicated so very well, managers have many more activities than decision making to occupy themselves with, and it will be necessary for an appropriate DSS to support many of these other information-related functions as well. Thus, the principal goal of a DSS is improvement in the effectiveness of organizational knowledge users through use of information technology. This is not a simple objective to achieve as has been learned in the process of past DSS design efforts.

1.2. Frameworks for the Engineering of Decision Support Systems

An appropriate decision support system design framework will consider each of the three principal components of decision support systems—a DBMS, an MBMS, and a DGMS—and their interrelations and interactions. Figure 3 illustrates the interconnection of these three generic components and illustrates the interaction of the decision maker with the system through the DGMS. We will describe some of the other components in this figure soon.

Sprague and Carlson (1992), authors of an early, seminal book on decision support systems, have indicated that there are three technology levels at which a DSS may be considered. The first of these is the level of DSS tools themselves. This level contains the hardware and software elements that enable use of system analysis and operations research models for the model base of the DSS and the database elements that comprise the database management system. The purpose of these DSS tools is to design a *specific DSS* that is responsive to a particular task or issue. The second level is that of a decision support system generator. The third level is the specific DSS itself. The specific DSS may be designed through the use of the DSS tools only, or it may be developed through use of a generic DSS generator that may call upon elements in the generic MBMS and DBMS tool repository for use in the specific DSS.

Often the best designers of a decision support system are not the specialists primarily familiar with DSS development tools. The principal reason for this is that it is difficult for one person or small group to be very familiar with a great variety of tools as well as to be able to identify the

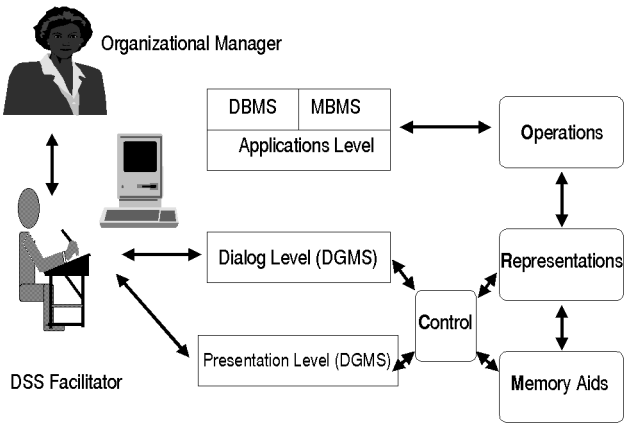

Figure 3 Generic Components in a Decision Support System, Including the ROMC Elements.

requirements needed for a specific DSS and the systems management skills needed to design a support process. This suggests that the decision support generator is a potentially very useful tool, in fact a design level, for DSS system design. The DSS generator is a set of software products, similar to a very advanced generation system development language, which enables construction of a specific DSS without the need to formally use micro-level tools from computer science and operations research and systems analysis in the initial construction of the specific DSS. These have, in effect, already been embedded in the DSS generator. A DSS generator contains an integrated set of features, such as inquiry capabilities, modeling language capabilities, financial and statistical (and perhaps other) analysis capabilities, and graphic display and report preparation capabilities. The major support provided by a DSS generator is that it allows the rapid construction of a prototype of the decision situation and allows the decision maker to experiment with this prototype and, on the basis of this, to refine the specific DSS such that it is more representative of the decision situation and more useful to the decision maker. This generally reduces, often to a considerable degree, the time required to engineer and implement a DSS for a specific application. This notion is not unlike that of software prototyping, one of the principal macro-enhancement software productivity tools (Sage and Palmer 1990) in which the process of constructing the prototype DSS through use of the DSS generator leads to a set of requirements specifications for a DSS that are then realized in efficient form using DSS tools directly.

The primary advantage of the DSS generator is that it is something that the DSS designer can use for direct interaction with the DSS user group. This eliminates, or at least minimizes, the need for DSS user interaction with the content specialists most familiar with micro-level tools of computer science, systems analysis, and operations research. Generally, a potential DSS user will seldom be able to identify or specify the requirements for a DSS initially. In such a situation, it is very advantageous to have a DSS generator that may be used by the DSS engineer, or developer, in order to obtain prototypes of the DSS. The user may then be encouraged to interact with the prototype in order to assist in identifying appropriate requirements specifications for the evolving DSS design.

The third level in this DSS design and development effort results from adding a decision support systems management capability. Often, this will take the form of the dialog generation and management subsystem referred to earlier, except perhaps at a more general level since this is a DGMS for DSS design and engineering rather than a DGMS for a specific DSS. This DSS design approach is not unlike that advocated for the systems engineering of large scale systems in general and DSS in particular.

There are many potential difficulties that affect the engineering of trustworthy systems. Among these are: inconsistent, incomplete, and otherwise imperfect system requirements specifications; system requirements that do not provide for change as user needs evolve over time, and poorly defined management structures. The major difficulties associated with the production of trustworthy systems have more to do with the organization and management of complexity than with direct technological concerns. Thus, while it is necessary to have an appropriate set of quality technological methods and tools, it is also very important that they be used within a well chosen set of lifecycle processes and set of systems management strategies that guide the execution of these processes (Sage 1995; Sage and Rouse 1999a).

Because a decision support system is intended to be used by decision makers with varying experiential familiarity and expertise with respect to a particular task and decision situation, it is especially important that a DSS design consider the variety of issue representations or frames that decision makers may use to describe issues, the operations that may be performed on these representations to enable formulation analysis and interpretation of the decision situation, the automated memory aids that support retention of the various results of operations on the representations, and the control mechanisms that assist decision makers in using these representations, operations, and memory aids. A very useful control mechanism results in the construction of heuristic procedures, perhaps in the form of a set of production rules, to enable development of efficient and effective standard operating policies to be issued as staff directives. Other control mechanisms are intended to encourage the decision maker to personally control and direct use of the DSS and also to acquire new skills and rules based on the formal reasoning-based knowledge that is called forth through use of a decision support system. This process independent approach toward development of the necessary capabilities of a specific DSS is due to Sprague and Carlson (1982) and is known as the ROMC approach (representations, operations, memory aids, and control mechanisms). Figure 3 illustrates the ROMC elements, together with the three principal components (DBMS, MBMS, and DGMS) of a DSS.

2. DATABASE MANAGEMENT SYSTEMS

As we have noted, a database management system is one of the three fundamental technological components in a decision support system. Figure 3 indicates the generic relationship among these components, or subsystems. We can consider a database management system as composed of a

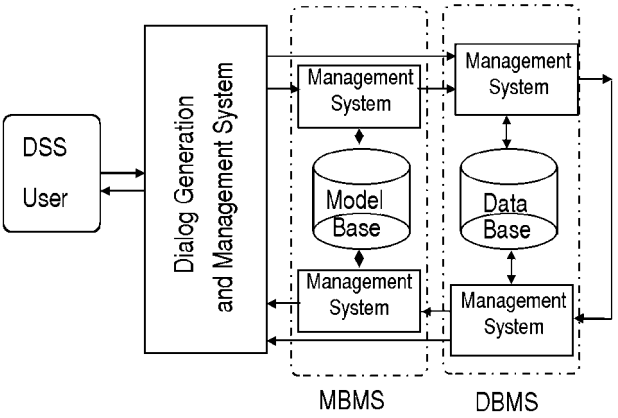

Figure 4 Expanded View of DSS Indicating Database and Management System for DBMS and Model Base and Management System for MBMS.

database (DB) and a management system (MS). This sort of expansion holds for the model base management system also. Figure 4 indicates the resulting expanded view of a DSS. The DBMS block itself can be expanded considerably, as can the other two subsystems of a DSS. Figure 5 indicates one possible expansion to indicate many of the very useful components of a database management system that we will briefly examine later in this section.

Three major objectives for a DBMS are data independence, data redundancy reduction, and data resource control, such that software to enable applications to use the DBMS and the data processed are independent. If this is not accomplished, then such simple changes to the data structure as adding four digits to the ZIP code number in an address might require rewriting many applications programs. The major advantage to data independence is that DSS developers do not need to be explicitly concerned with the details of data organization in the computer or how to access it explicitly for information processing purposes, such as query processing. Elimination or reduction of data redundancy will assist in lessening the effort required to make changes in the data in a database. It may also assist, generally greatly, in eliminating the inconsistent data problem that often results from updating data items in one part of a database but (unintentionally) not updating this same data that

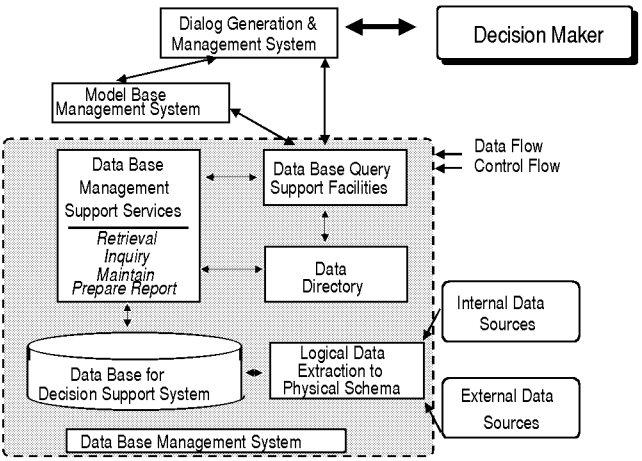

Figure 5 Generic Components in Decision Support System, with Expansion of Database Management System.

exists elsewhere in the database because of data redundancy. With many people potentially using the same database, resource control is essential. The presence of these three features in a DBMS is the major factor that differentiates it from a file management system.

The management system for a database is composed of the software that is beneficial to the creation, access, maintenance, and updating of a database. A database contains data that are of value to an individual or an organization and that an individual or an organization desires to maintain. Maintenance should be such that the data survive even when the DBMS system hardware and/or software fails. Maintenance of data is one of the important functions provided for in DBMS design.

There are many tasks that we desire to perform using a database. These five are especially important:

1. *Capturing* relevant data for use in the database
2. *Selecting* relevant data from the database
3. *Aggregating* data to form totals, averages, moments, and other items that support decision making
4. *Estimating, forecasting, and predicting* in order to obtain extrapolations of events into the future, and such other activities as
5. *Optimizing* in order to enable selection of a ''best'' alternative

We note that these database raise issues that are associated with the model base management system. In a similar way, the dialog generation and management system determines how data is viewed and is therefore also important for use of a DBMS.

2.1. DBMS Design, Selection, and Systems Integration

As with any information systems engineering-based activity, DBMS design should start with an identification of the DBMS design situation and the user requirements. From this, identification of the logical or conceptual data requirements follows specification of a logical database structure in accordance with what is known as a data definition language (DDL). After this, the physical database structure is identified. This structure must be very concerned with specific computer hardware characteristics and efficiency considerations. Given these design specifications, an operational DBMS is constructed and DBMS operation and maintenance efforts begin. The logical database design and physical database design efforts can each be disaggregated into a number of related activities. The typical database management system lifecycle will generally follow one of the systems engineering development life cycles discussed in the literature (Sage 1992, 1995; Sage and Rouse 1999a).

The three important DBMS requirements—data independence, redundancy reduction, and increased data resource control—are generally applicable both to logical data and to physical data. It is highly desirable, for example, to be able to change the structure of the physical database without affecting other portions of the DBMS. This is called physical data independence. In a similar way, logical data independence denotes the ability of software to function using a given applications-oriented perspective on the database even though changes in other parts of the logical structure have been made. The requirements specification, conceptual design, logical design, and physical design phases of the DBMS development life cycle are specifically concerned with satisfaction of these requirements.

A number of questions need to be asked and answered successfully in order to design an effective DBMS. Among these are the following (Arden 1980):

1. Are there data models that are appropriate across a variety of applications?
2. What DBMS designs that enable data models to support logical data independence?
3. What DBMS designs enable data models to support logical data independence, and what are the associated physical data transformations and manipulations?
4. What features of a data description language will enable a DBMS designer to control independently both the logical and physical properties of data?
5. What features need to be incorporated into a data description language in order to enable errors to be detected at the earliest possible time such that users will not be affected by errors that occur at a time prior to their personal use of the DBMS?
6. What are the relationships between data models and database security?
7. What are the relationships between data models and errors that may possibly be caused by concurrent use of the database by many users?
8. What are design principles that will enable a DBMS to support a number of users having diverse and changing perspectives?
9. What are the appropriate design questions such that applications programmers, technical users, and DBMS operational users are each able to function effectively and efficiently?

The bottom line question that summarizes all of these is, How does one design a data model and data description language to enable efficient and effective data acquisition, storage, and use? There are many related questions; one concerns the design of what are called standard query languages (SQLs) such that it is possible to design a specific DBMS for a given application.

It is very important that, whenever possible, a specific DBMS be selected prior to design of the rest of the DSS. There are a variety of reasons why this is quite desirable. The collection and maintenance of the data through the DSS are simplified if there is a specified single DBMS structure and architecture. The simplest situation of all occurs when all data collection (and maintenance) is accomplished prior to use of the DSS. The DBMS is not then used in an interactive manner as part of DSS operation. The set of database functions that the DSS needs to support is controlled when we have the freedom to select a single DBMS structure and architecture prior to design of the rest of the DSS. The resulting design of the DSS is therefore simplified. Further, the opportunities for data sharing among potentially distributed databases are increased when the interoperability of databases is guaranteed. Many difficulties result when this is not possible. Data in a DBMS may be classified as *internal data,* stored in an internal database, and *external data,* stored in an external database. Every individual and organization will necessarily have an internal database. While no problem may exist in ensuring DBMS structure and architecture compatibility for internal data, this may be very difficult to do for external data. If both of these kinds of data can be collected and maintained prior to use of a DSS, then there will generally be no data integration needs. Often, however, this will not be possible. Because we will often be unable to control the data structure and architecture of data obtained externally, the difficulties we cite will often be real, especially in what are commonly called real-time, interactive environments. When we have DBMSs that are different in structure and architecture, data sharing across databases is generally difficult, and it is often then necessary to maintain redundant data. This can lead to a number of difficulties in the systems integration that is undertaken to ensure compatibility across different databases.

In this chapter, as well as in much DSS and information system design in practice, we may generally assume that the DBMS is preselected prior to design of the rest of the DSS and that the same DBMS structure and architecture are used for multiple databases that may potentially be used in the DSS. This is often appropriate, for purposes of DSS design, since DBMS design technology (Rob and Coronel 1997; Kroenke 1998; Date 1999; Purba 1999; Atzeni et al. 2000) is now relatively mature in contrast to MBMS and DGMS design. This does not suggest that evolution and improvements to DBMS designs are not continuing. It is usually possible, and generally desirable, to select a DBMS, based on criteria we will soon discuss, and then design the MBMS and DGMS based on the existing DBMS. This will often, but surely not always, allow selection of commercial off-the-shelf (COTS) DBMS software. The alternate approach of designing a MBMS and DGMS first and then specifying the requirements for a DBMS based on these is possible and may in some cases be needed. Given the comparatively developed state of DBMS software development as contrasted with MBMS and DGMS software, this approach will usually be less desirable from the point of view of design economy in engineering the DSS.

2.2. Data Models and Database Architectures

We will now expand very briefly on our introductory comments concerning data model representations and associated architectures for database management systems. Some definitions are appropriate. A data model defines the types of data objects that may be manipulated or referenced within a DBMS. The concept of a logical record is a central one in all DBMS designs. Some DBMS designs are based on mathematical relations, and the associated physical database is a collection of consistent tables in which every row is a record of a given type. This is an informal description of a relational database management system, a DBMS type that is a clear outgrowth of the file management system philosophy. Other DBMS may be based on hierarchical or network structures, which resemble the appearance of the data in the user's world and may contain extensive graphical and interactive support characteristics.

There are several approaches that we might take to describing data models. Date (1983, 1999), for example, discusses a three-level representation in which the three levels of data models are:

1. An external model, which represents a data model at the level of the user's application and is the data model level closest and most familiar to users of a DBMS or DSS
2. A conceptual model, which is an aggregation model that envelopes several external models
3. An internal model, which is a technical-level model that describes how the conceptual model is actually represented in computer storage

Figure 6 is a generic diagram that indicates the mappings and data translations needed to accommodate the different levels of data models and architectures. The relations between the various levels, called mappings. These specify and describe the transformations that are needed in order to obtain one model from another. The user supplies specifications for the source and target data structures in

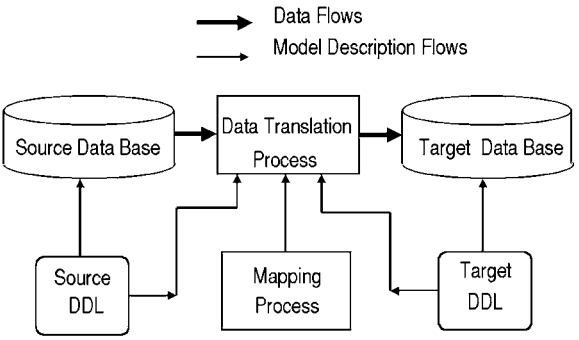

Figure 6 Data Transforms and Model Mappings.

a data description language (DDL) and also describes the mapping that is desired between source and target data. Figure 7 represents this general notion of data transformation. This could, for example, represent target data in the form of a table that is obtained from a source model composed of a set of lists.

Figure 7 is a very simple illustration of the more general problem of mapping between various schemas. Simply stated, a schema is an image used for comparison purposes. A schema can also be described as a data structure for representing generic concepts. Thus, schemas represent knowledge about concepts and are structurally organized around some theme. In terms appropriate here, the user of a database must interpret the real world that is outside of the database in terms of real-world objects, or entities and relationships between entities, and activities that exist and that involve these objects. The database user will interact with the database in order to obtain needed data for use or, alternatively, to store obtained data for possible later use. But a DBMS cannot function in terms of real objects and operations. Instead, a DBMS must use data objects, composed of data elements and relations between them, and operations on data objects. Thus, a DBMS user must perform some sort of mapping or transformation from perceived real objects and actions to those objects' and actions' representations that will be used in the physical database.

The single-level data model, which is conceptually illustrated in Figure 7, represents the nature of the data objects and operations that the user understands when receiving data from the database. In this fashion the DBMS user models the perceived real world. To model some action sequence, or the impact of an action sequence on the world, the user maps these actions to a sequence of operations that are allowed by the specific data model. It is the data manipulation language (DML) that provides the basis for the operations submitted to the DBMS as a sequence of queries or programs. The development of these schemas, which represent logical data, results in a DBMS architecture or DBMS framework, which describes the types of schemas that are allowable and the way in which these schemas are related through various mappings. We could, for example, have a two-schema framework or a three-schema framework, which appears to be the most popular representation at this time. Here

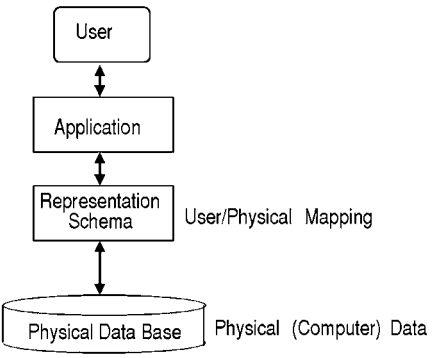

Figure 7 Single-Level Model of Data Schema.

the external schemas define the logical subset of the database that may be presented to a specific DBMS user. The conceptual schemas define conceptual models as represented in the database. The internal schema describes physical data structures in the database. These data structures provide support to the conceptual models. The mappings used establish correspondence between various objects and operations at the three levels. If consistency of the conceptual models is to be ensured, we must be able to map the internal, or physical, and external, or logical, schemas to the conceptual schemas.

Useful documentation about the database structure and architecture is provided through the various schemas, which represent explicit data declarations. These declarations represent data about data. The central repository in which these declarations are kept is called a data dictionary or data directory, and is often represented by the symbol DD. The data directory is a central repository for the definitions of the schemas and the mappings between schemas. Generally, a data dictionary can be queried in the same manner as can the database, thereby enhancing the ability of the DBMS user to pose questions about the availability and structure of data. It is often possible to query a data directory with a high-level, or fourth-generation, query language.

A data dictionary is able to tell us what the records in the data dictionary consist of. It also contains information about the logical relationships that pertain to each of the particular elements in the database. The development of a data dictionary generally begins with formation of lists of desired data items or fields that have been grouped in accordance with the entities that they are to represent. A name is assigned for each of these lists, and a brief statement of the meaning of each is provided for later use. Next, the relationships between data elements should be described and any index keys or pointers should be determined. Finally, the data dictionary is implemented within the DBMS. In many ways, a data dictionary is the central portion of a DBMS. It performs the critical role of retaining high-level information relative to the various applications of the DBMS and thereby enables specification, control, review, and management of the data of value for decision making relative to specific applications.

For very large system designs, it is necessary that the data dictionary development process be automated. A typical data dictionary for a large system may include several thousand entries. It is physically impossible to manually maintain a dictionary of this size or to retain consistent and unambiguous terms for each data element or composite of data elements. Therefore, automated tools are needed for efficient and effective development and maintenance of a data dictionary. These are provided in contemporary DBMSs.

This ability to specify database structures, through use of schemas, enables efficient and effective management of data resources. This is necessary for access control. The use of one or more subschemas generally simplifies access to databases. It may also provide for database security and integrity for authorized users of the DBMS. Since complex physical (computer) structures for data may be specified, independent of the logical structure of the situation that is perceived by DBMS users, it is thereby possible to improve the performance of an existing and operational database without altering the user interface to the database. This provides for simplicity in use of the DBMS.

The data model gives us the constructs that provide a foundation for the development and use of a database management system. It also provides the framework for such tools and techniques as user interface languages. The user of these languages is often called a database administrator (DBA). There are three types of user interface languages:

1. *Data definition languages* (DDLs) provide the basis for definition of schemas and subschemas.
2. *Data manipulation languages* (DMLs) are used to develop database applications.
3. *Data query languages* (DQLs) or simply *query languages* (QLs) are used to write queries and reports. Of course, it is possible to combine a DDL, DML, and DQL into a single database language.

In general, a data model is a paradigm for representation, storing, organizing, and otherwise managing data in a database. There are three component sets in most data models:

1. A set of data structures that define the fields and records that are allowed in the database. Examples of data structures include lists, tables, hierarchies, and networks.
2. A set of operations that define the admissible manipulations that are applied to the fields and records that comprise the data structures. Examples of operations include *retrieve, combine, subtract, add,* and *update*.
3. A set of integrity rules that define or constrain allowable or legal states or changes of state for the data structures that must be protected by the operations.

There are three generically different types of data model representations and several variants within these three representations. Each of these is applicable as an internal, external, or conceptual model. We will be primarily concerned with these three modeling representations as they affect the

external, or logical, data model. This model is the one with which the user of a specific decision support system interfaces. For use in a decision support system generator, the conceptual model is of importance because it is the model that influences the specific external model that various users of a DSS will interface with after an operational DSS is realized. This does not mean that the internal data model is unimportant. It is very important for the design of a DBMS. Through its data structures, operations, and integrity constraints, the data model controls the operation of the DBMS portion of the DSS.

The three fundamental data models for use in the external model portion of a DSS are record-based models, structurally based models, and expert system-based models. Each of these will now be briefly described.

2.2.1. Record-Based Models

We are very accustomed to using forms and reports, often prepared in a standard fashion for a particular application. Record-based models are computer implementations of these spreadsheet-like forms. Two types can be identified. The first of these, common in the early days of file processing systems (FPSs) or file management systems (FMSs), is the *individual record model*. This is little more than an electronic file drawer in which records are stored. It is useful for a great many applications. More sophisticated, however, is the relational database data model, in which mathematical relations are used to electronically ''cut and paste'' reports from a variety of files. Relational database systems have been developed to a considerable degree of sophistication, and many commercial products are available. Oracle and Informix are two leading providers.

The individual record model is surely the oldest data model representation. While the simple single-record tables characteristic of this model may appear quite appealing, the logic operations and integrity constraints that need to be associated with the data structure are often undefined and are perhaps not easily defined. Here, the data structure is simply a set of records, with each record consisting of a set of fields. When there is more than a single type of record, one field contains a value that indicates what the other fields in the record are named.

The relational model is a modification of the individual record model that limits its data structures and thereby provides a mathematical basis for operation on records. Data structures in a relational database may consist only of relations, or field sets that are related. Every relation may be considered as a table. Each row in the table is a record or tuple. Every column in each table or row is a field or attribute. Each field or attribute has a domain that defines the admissible values for that field.

Often there is only a modest difference in structure between this relational model and the individual record model. One difference is that fields in the various records of the individual record model represent relationships, whereas relationships among fields or attributes in a relational model are denoted by the name of the relation.

While the structural differences between the relational model and the individual record model are minimal, there are major differences in the way in which the integrity constraints and operations may affect the database. The operations in a relational database form a set of operations that are defined mathematically. The operations in a relational model must operate on entire relations, or tuples, rather than only on individual records. The operations in a relational database are independent of the data structures, and therefore do not depend on the specific order of the records or the fields. There is often controversy about whether or not a DBMS is truly relational. While there are very formal definitions of a relational database, a rather informal one is sufficient here. A relational database is one described by the following statements.

1. Data are presented in tabular fashion without the need for navigation links, or pointer structures, between various tables.
2. A relational algebra exists and can be used to prepare joins of logical record files automatically.
3. New fields can be added to the database without the necessity of rewriting any programs that used previous versions of the database.

If a DBMS does not satisfy these three criteria, then it is almost surely NOT a relational database.

2.2.2. Structural Models

In many instances, data are intended to be associated with a natural structural representation. A typical hierarchical structure is that of a classic organization. A more general representation than a hierarchy, or tree, is known as a *network*. It is often possible to represent a logical data model with a hierarchical data structure. In a hierarchical model, we have a number of nodes that are connected by links. All links are directed to point from ''child'' to ''parent,'' and the basic operation in a hierarchy is that of searching a tree to find items of value. When a query is posed with a hierarchical database, all branches of the hierarchy are searched and those nodes that meet the conditions posed in the query are noted and then returned to the DBMS system user in the form of a report.

Some comparisons of a hierarchical data model with a relational data model are of interest here. The structures in the hierarchical model represent the information that is contained in the fields of the relational model. In a hierarchical model, certain records must exist before other records can exist. The hierarchical model is generally required to have only one key field. In a hierarchical data model, it is necessary to repeat some data in a descendant record that need be stored only once in a relational database regardless of the number of relations. This is so because it is not possible for one record to be a descendant of more than one parent record. There are some unfortunate consequences of the mathematics involved in creating a hierarchical tree, as contrasted with relations among records. Descendants cannot be added without a root leading to them, for example. This leads to a number of undesirable characteristic properties of hierarchical models that may affect our ability to easily add, delete, and update or edit records.

A network model is quite similar to but more general than the hierarchical model. In a hierarchy, data have to be arranged such that one child has only one parent, and in many instances this is unrealistic. If we force the use of a hierarchical representation in such cases, data will have to be repeated at more than one location in the hierarchical model. This redundancy can create a number of problems. A record in a network model can participate in several relationships. This leads to two primary structural differences between hierarchical and network models. Some fields in the hierarchical model will become relationships in a network model. Further, the relationships in a network model are explicit and may be bidirectional. The navigation problem in a network data model can become severe. Because search of the database can start at several places in the network, there is added complexity in searching, as well.

While spreadsheet type relational records are very useful for many purposes, it has been observed (Kent 1979) that not all views of a situation, or human cognitive maps, can be represented by relational data models. This has led to interest in entity and object-oriented data models and to data models based on artificial intelligence techniques. The basic notion in the use of an ER model is to accomplish database design at the conceptual level, rather than at the logical and/or physical levels. ER models (Chen 1976) are relatively simple and easy to understand and use, in large part because of the easy graphical visualization of the database model structure. Also, ER modeling capability is provided by many computer aided software engineering (CASE) tools. While such limitations as lack of a very useful query language are present (Atzeni and Chen 1983), much interest in ER data models, especially at the conceptual level, exists at this time. Figure 8 illustrates the conceptual orientation of the ER modeling approach.

ER models are based on two premises: (a) information concerning entities and relationships exists as a cognitive reality, and (b) this information may be structured using entities and relationships among them as data. An entity relationship data model is a generalization of the hierarchical and network data models. It is based on well-established graph theoretic developments and is a form of structured modeling. The major advantage of the ER approach is that it provides a realistic view of the structure and form for the DBMS design and development effort. This should naturally support the subsequent development of appropriate software. In addition, the approach readily leads to the

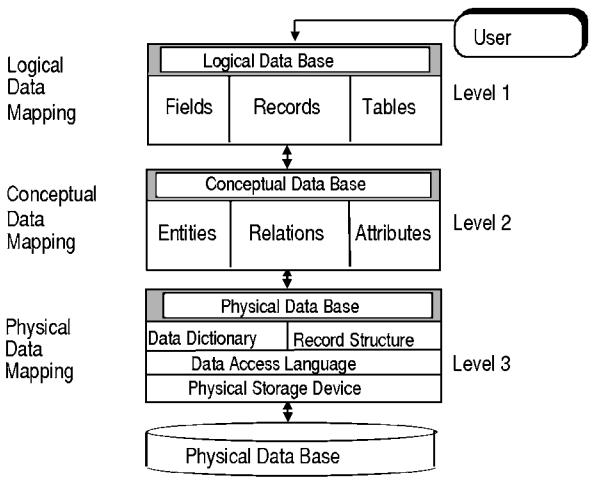

Figure 8 The Potential Role of ER Data Models at the Conceptual Data Level.

development of the data dictionary. The primary difficulties that may impede easy use of the ER method are in the need for selection of appropriate verbs for use as contextual relationships, elimination of redundancy of entities, and assurances that all of the needed ER have been determined and properly used. Often a considerable amount of time may be needed to identify an appropriate set of entities and relations and obtain an appropriate structural model in the form of an ER diagram.

2.2.3. Expert and Object-Oriented Database Models

An expert database model, as a specialized expert database system, involves the use of artificial intelligence technology. The goal in this is to provide for database functionality in more complex environments that require at least some limited form of intelligent capability. To allow this may require adaptive identification of system characteristics, or learning over time as experience with an issue and the environment into which it is embedded increases. The principal approach that has been developed to date is made up of an object-oriented database (OODB) design with any of the several knowledge representation approaches useful in expert system development. The field is relatively new compared to other database efforts and is described much more fully in Kerschberg (1987, 1989), Myloupoulos and Brodie (1989), Parsaye et al. (1989), Debenham (1998), Poe et al. (1998), and Darwen and Date (1998).

The efforts to date in the object oriented area concern what might be more appropriately named object-oriented database management systems design (OODBMS). The objective in OODBMS design is to cope with database complexity, potentially for large distributed databases, through a combination of object-oriented programming and expert systems technology. Object-oriented design approaches generally involve notions of separating internal computer representations of elements from the external realities that lead to the elements. A primary reason for using object-oriented language is that it naturally enables semantic representations of knowledge through use of 10 very useful characteristics of object-oriented approaches (Parsaye et al. 1989): information hiding, data abstraction, dynamic binding and object identity, inheritance, message handling, object oriented graphical interfaces, transaction management, reusability, partitioning or dividing or disaggregating an issue into parts, and projecting or describing a system from multiple perspectives or viewpoints. These could include, for example, social, political, legal, and technoeconomic perspectives (Sage 1992).

There are at least two approaches that we might use in modeling a complex large-scale system: (1) functional decomposition and structuring and (2) purposeful or object decomposition and structuring. Both approaches are appropriate and may potentially result in useful models. Most models of real-world phenomena tend to be purposeful. Most conventional high-level programming languages are functionally, or procedurally, oriented. To use them, we must write statements that correspond to the functions that we wish to provide in order to solve the problem that has been posed. An advantage of object decomposition and structuring is that it enables us to relate more easily the structure of a database model to the structure of the real system. This is the case if we accomplish our decomposition and structuring such that each module in the system or issue model represents an object or a class of objects in the real issue or problem space. Objects in object-oriented methodology are not unlike elements or nodes in graph theory and structural modeling. It is possible to use one or more contextual relations to relate elements together in a structural model. An object may be defined as a collection of information and those operations that can be performed upon it. We request an object to perform one of its allowable operations by instructing it with a message.

Figure 9 illustrates the major conceptual difference between using a conventional programming approach and using an object-oriented approach for construction of a DBMS. In the conventional approach, procedures are at the nexus and procedures update and otherwise manipulate data and return values. In the object-oriented approach, the collection of independent objects are at the nexus and communicate with each other through messages or procedures. Objects investigate requests and behave according to these messages. Object-oriented design often provides a clear and concise interface to the problem domain in that the only way to interact with an object is through the operations or messages to the object. These messages call for operations that may result in a change of state in the object in question. This message will affect the particular object called and only that one, as no other object is affected. This provides a high degree of modularity and increased ability to verify and validate outcomes and thus provides an increased sense of reliability in the resulting DBMS design. Object-oriented languages are, for the most part, very high-level languages used to accomplish precisely the same results as high-level languages. By focusing upon the entities of objects and the relationships between objects, they often provide a simpler way to describe precisely those elements needed in detailed design procedures.

Object-oriented DBMS design is based on appropriate linking of objects, or data items or entities, and the operations on these, such that the information and processing is concentrated on the object classes, attributes, and messages that transfer between them. The features that set object-oriented approaches from other approaches are the capability of object-oriented languages to support abstraction, information hiding, and modularity. The items used in object-oriented DBMS design are objects,

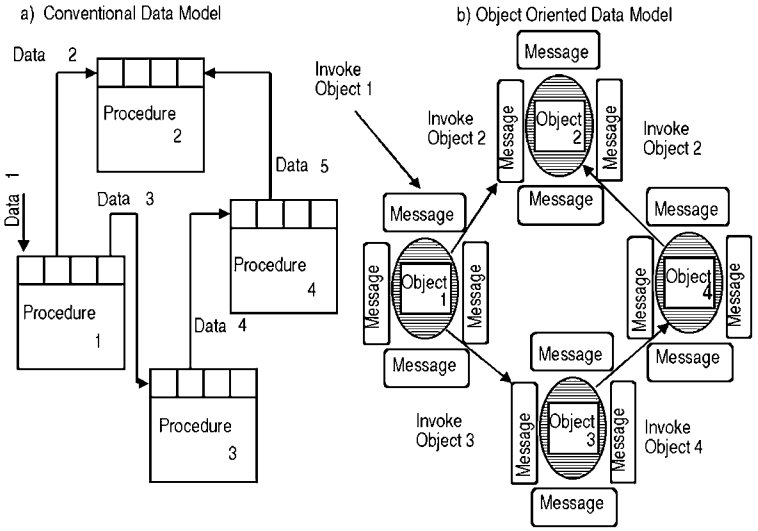

Figure 9 Conventional and Object-Oriented Data Models.

abstracts or physical entities that pertain to the domain of the system; attributes, or characteristics of objects; operations, or dynamic entities that may change with time; and messages, or requests to an object to perform an operation. Objects represent such real-world entities as machines, files, products, signals, persons, things, and places. They may be either physical entities or abstract representations. The attributes are characteristics that describe the properties ascribed to objects. Each attribute may be dynamically associated with a numerical value, and the combination of these values together with the description of the object in the problem domain presents the state of the object. Thus, the attributes are related to the object as subobjects. Attributes include such things as the name of the object, the number of specific objects in a file, or the name for the place, and the like. The basic attribute, one that may not be further decomposed, is the primitive type of object or subobject. Attributes may also be nonnumeric. Operations consist of processes and data structures that apply to the object to which it is directed. These are generally dynamic entities whose value may change over time. Each object may be subjected to a number of operations that provide information relative to control and procedural constructs that are applied to the object. Defining an object such as to have a private part and then assigning a message to address the appropriate processing operation enables us to achieve information hiding. Messages are passed between objects in order to change the state of the object, address the potentially hidden data parts of an object, or otherwise modify an object.

Coad and Yourdon (1990) identify five steps associated with implementing an object-oriented approach:

1. Identify objects, typically by examining the objects in the real world.
2. Identify structures, generally through various abstracting and partitioning approaches that result in a classification structure, an assembly structure, or a combination of these.
3. Identify subjects through examining objects and their structures to obtain this more complex abstract view of the issue. Each classification structure and each assembly structure will comprise one subject.
4. Identify attributes that impart important aspects of the objects that need to be retained for all instances of an object.
5. Identify services such that we are aware of occurrence services, creation or modification of instances of an object, calculation services, or monitoring of other processes for critical events or conditions.

Steps such as these should generally be accomplished in an iterative manner because there is no unique way to accomplish specification of a set of objects, or structural relations between objects.

A major result from using object-oriented approaches is that we obtain the sort of knowledge representation structures that are commonly found in many expert systems. Thus, object-oriented

design techniques provide a natural interface to expert system-based techniques for DBMS design. This is so because objects, in object-oriented design, are provided with the ability to store information such that learning over time can occur; process information by initiating actions in response to messages; compute and communicate by sending messages between objects; and generate new information as a result of communications and computations. Object-oriented design also lends itself to parallel processing and distributed environments. Object-oriented design and analysis is a major subject in contemporary computer and information system subject areas and in many application domains as well (Firesmith 1993; Sullo 1994; Jacobson et al. 1995; Zeigler 1997).

2.3. Distributed and Cooperative Databases

We have identified the three major objectives for a DBMS as data independence, data redundancy reduction, and data resource control. These objectives are important for a single database. When there are multiple databases potentially located in a distributed geographic fashion, and potentially many users of one or more databases, additional objectives arise, including:

1. Location independence or transparency to enable DBMS users to access applications across distributed information bases without the need to be explicitly concerned with where specific data is located
2. Advanced data models, to enable DBMS users to access potentially nonconventional forms of data such as multidimensional data, graphic data, spatial data, and imprecise data
3. Extensible data models, which will allow new data types to be added to the DBMS, perhaps in an interactive real-time manner, as required by specific applications.

An important feature of a distributed database management systems (DDBMSs) is that provisions for database management are distributed. Only then can we obtain the needed "fail-safe" reliability and "availability" even when a portion of the system breaks down. There are a number of reasons why distributed databases may be desirable. These include and are primarily related to distributed users and cost savings. Some additional costs are also involved, and these must be justified.

A distributed database management system will generally look much like replicated versions of a more conventional single-location database management system. We can thus imagine replicated versions of Figure 7. For simplicity, we show only the physical database, the database access interface mechanism, and the data dictionary for each database. Figure 10 indicates one possible conceptual

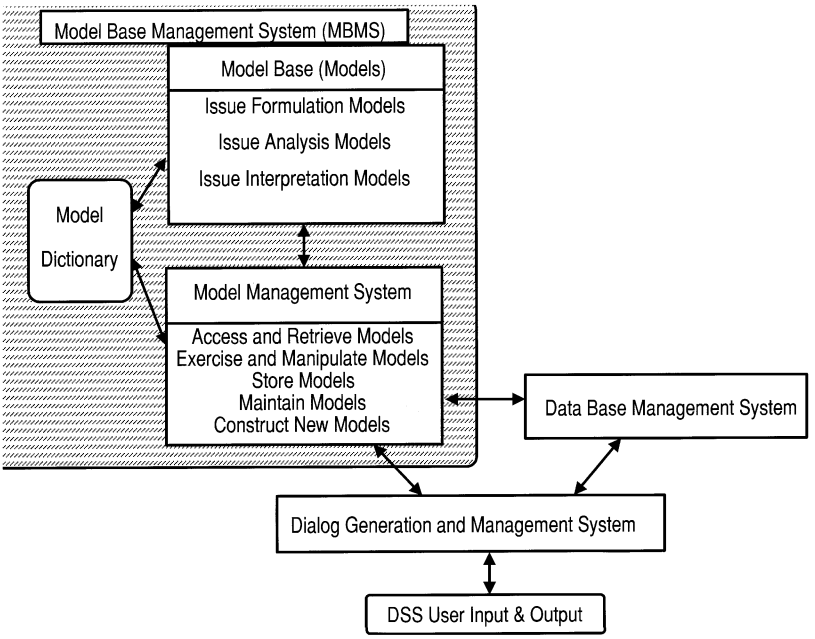

Figure 10 Prototypical Structure of Model Management System.

architecture of a distributed database in which there are two requests for data being simultaneously submitted. Through use of protocols, all requests for data are entered as if those data were stored in the database of the requesting user. The data dictionaries of the distributed system are responsible for finding the requested data elements and delivering them to the requesting user. All of these auxiliary requests are accomplished in a manner that is transparent to the requesting database user. Data transmission rate concerns must be resolved in order for this to be done satisfactorily. Potentially, however, real-time access to all of the data in the entire distributed system can be provided to any local user.

It is important to note that completion of the above steps does not transfer the requested data into the database of the requesting user, except perhaps in some temporary storage fashion. The very good reason why this is not done is that it would result in database redundancy and also increase database security difficulties.

An alternate approach to this distributing of data is to use what is often called cooperative processing or cooperative database management. This involves central database and distributed database concepts, blended to produce better results than can be had with either approach. Central to this are various approaches for the distributing and coordinating of data. We could use function, geography, or organizational level as the basis for distributing and cooperating, and possible system configurations may be based upon background communications, microcomputer-based front-end processing for host applications, and peer-to-peer cooperative processing.

The database component of a DSS provides the knowledge repository that is needed for decision support. Some sort of management system must be associated with a database in order to provide the intelligent access to data that is needed. In this section we have examined a number of existing constructs for database management systems. We now turn to the model-based portion of a DSS.

3. MODEL BASE MANAGEMENT SYSTEMS

There are four primary objectives of a DBMS:

1. To manage a large quantity of data in physical storage
2. To provide logical data structures that interact with humans and are independent of the structure used for physical data storage
3. To reduce data redundancy and maintenance needs and increase flexibility of use of the data, by provision of independence between the data and the applications programs that use it
4. To provide effective and efficient access to data by users who are not necessarily very sophisticated in the microlevel details of computer science.

Many support facilities will typically be provided with a DBMS to enable achievement of these purposes. These include data dictionaries to aid in internal housekeeping and information query, retrieval, and report generation facilities to support external use needs.

The functions of a model base management system (MBMS), a structure for which is illustrated in Figure 10, are quite analogous to those of a DBMS. The primary functions of a DBMS are separation of system users, in terms of independence of the application, from the physical aspects of database structure and processing. In a similar way, a MBMS is intended to provide independence between the specific models that are used in a DSS and the applications that use them. The purpose of a MBMS is to transform data from the DBMS into information that is useful for decision making. An auxiliary purpose might also include representation of information as data such that it can later be recalled and used.

The term *model management system* was apparently first used over 15 years ago (Will 1975). Soon thereafter, the MBMS usage was adopted in Sprague and Carlson (1982) and the purposes of an MBMS were defined to include creation, storage, access, and manipulation of models. Objectives for a MBMS include:

1. To provide for effective and efficient creation of new models for use in specific applications
2. To support maintenance of a wide range of models that support the formulation, analysis, and interpretation stages of issue resolution
3. To provide for model access and integration, within models themselves as well as with the DBMS
4. To centralize model base management in a manner analogous to and compatible with database management
5. To ensure integrity, currency, consistency, and security of models.

Just as we have physical data and logical data processing in a DBMS, so also do we have two types of processing efforts in a MBMS: model processing and decision processing (Applegate et al.

1986). A DSS user would interact directly with a decision processing MBMS, whereas the model processing MBMS would be more concerned with provision of consistency, security, currency, and other technical modeling issues. Each of these supports the notion of appropriate formal use of models that support relevant aspects of human judgment and choice.

Several ingredients are necessary for understanding MBMS concepts and methods. The first of these is a study of formal analytical methods of operations research and systems engineering that support the construction of models that are useful in issue formulation, analysis, and interpretation. Because presenting even a small fraction of the analytical methods and associated models in current use would be a mammoth undertaking, we will discuss models in a somewhat general context. Many discussions of decision relevant models can be found elsewhere in this Handbook, most notably in Chapters 83 through 102. There are also many texts in this area, as well as two recent handbooks (Sage and Rouse 1999a; Gass and Harris 2000).

3.1. Models and Modeling

In this section, we present a brief description of a number of models and methods that can be used as part of a systems engineering-based approach to problem solution or issue resolution. Systems engineering (Sage 1992, 1995; Sage and Rouse 1999a; Sage and Armstrong 2000) involves the application of a general set of guidelines and methods useful to assist clients in the resolution of issues and problems, often through the definition, development, and deployment of trustworthy systems. These may be product systems or service systems, and users may often deploy systems to support process-related efforts. Three fundamental steps may be distinguished in a formal systems-based approach that is associated with each of these three basic phases of system engineering or problem solving:

1. Problem or issue formulation
2. Problem or issue analysis
3. Interpretation of analysis results, including evaluation and selection of alternatives, and implementation of the chosen alternatives

These steps are conducted at a number of phases throughout a systems life cycle. As we indicated earlier, this life cycle begins with definition of requirements for a system through a phase where the system is developed to a final phase where deployment, or installation, and ultimate system maintenance and retrofit occur. Practically useful life cycle processes generally involve many more than three phases, although this larger number can generally be aggregated into the three basic phases of definition, development, and deployment. The actual engineering of a DSS follows these three phases of definition of user needs, development of the DSS, and deployment in an operational setting. Within each of these three phases, we exercise the basic steps of formulation, analysis, and interpretation. Figure 11 illustrates these three steps and phases and much more detail is presented in Sage (1992, 1995), Sage and Rouse (1999a), Sage and Armstrong (2000), and in references contained therein.

3.1.1. Issue, or Problem, Formulation Models

The first part of a systems effort for problem or issue resolution is typically concerned with problem or issue formulation, including identification of problem elements and characteristics. The first step in issue formulation is generally that of definition of the problem or issue to be resolved. Problem

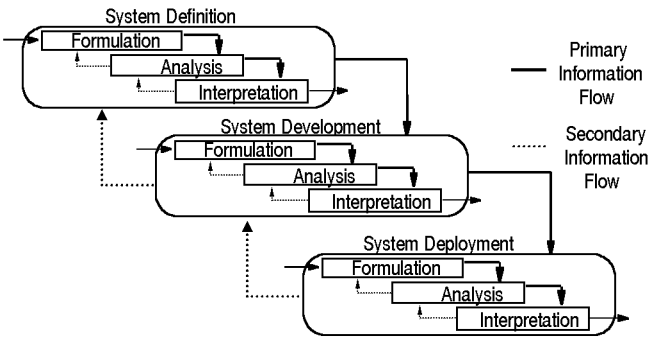

Figure 11 Basic Phases and Steps in Systems Engineering.

definition is generally an outscoping activity because it enlarges the scope of what was originally thought to be the problem. Problem or issue definition will ordinarily be a group activity involving those familiar with or impacted by the issue or the problem. It seeks to determine the needs, constraints, alterables, and social or organizational sectors affecting a particular problem and relationships among these elements.

Of particular importance are the identification and structuring of objectives for the policy or alternative that will ultimately be chosen. This is often referred to as *value system design,* a term apparently first used by Hall (1969) in one of his seminal works in systems engineering. Generation of options (Keller and Ho 1990) or alternative courses of action is a very important and often neglected portion of a problem solving effort. This option generation, or system or alternative synthesis, step of issue formulation is concerned primarily with the answers to three questions:

1. What are the alternative approaches for attaining objectives?
2. How is each alternative approach described?
3. How do we measure attainment of each alternative approach?

The answers to these three questions lead to a series of alternative activities or policies and a set of activities measures.

Several of the methods that are particularly helpful in the identification of issue formulation elements are based on principles of collective inquiry (McGrath 1984). The term *collective inquiry* refers to the fact that a group of interested and motivated people is brought together in the hope that they will stimulate each other's creativity in generating elements. We may distinguish two groups of collective inquiry methods here, depending upon whether or not the group is physically present at the same physical location.

3.1.1.1. Brainstorming, Synectics, and Nominal Group Techniques These approaches typically require a few hours of time, a group of knowledgeable people gathered in one place, and a group leader or facilitator. The nominal group technique is typically better than brainstorming in reducing the influence of dominant individuals. Both methods can be very productive: 50–150 ideas or elements may be generated in less than one hour. Synectics, based on problem analogies, might be very appropriate if there is a need for truly unconventional, innovative ideas. Considerable experience with the method is a requirement, however, particularly for the group leader. The nominal group technique is based on a sequence of idea generation, discussion, and prioritization. It can be very useful when an initial screening of a large number of ideas or elements is needed. Synectics and brainstorming are directly interactive group methods, whereas nominal group efforts are nominally interactive in that the members of the group do not directly communicate.

3.1.1.2. Questionnaires, Survey, and Delphi Techniques These three methods of collective inquiry do not require the group of participants to gather at one place and time, but they typically take more time to achieve results than the methods above. In most questionnaires and surveys a large number of participants is asked, on an individual basis, to generate ideas or opinions that are then processed to achieve an overall result. Generally no interaction is allowed among participants in this effort. Delphi usually provides for written anonymous interaction among participants in several rounds. Results of previous rounds are fed back to participants, and they are asked to comment, revise their views as desired, and so on. The results of using the Delphi technique can be very enlightening, but it usually takes several weeks or months to complete the effort.

Use of some of the many structuring methods, in addition to leading to greater clarity of the problem formulation elements, will typically lead also to identification of new elements and revision of element definitions. Most structuring methods contain an analytical component and may therefore be more properly labeled as analysis methods. The following element structuring aids are among the many modeling aids available that are particularly suitable for the issue formulation step.

There are many approaches to problem formulation (Volkema 1990). In general, these approaches assume that ''asking'' will be the predominant approach used to obtain issue formulation elements. Asking is often the simplest approach. Valuable information can often be obtained from observation of an existing and evolving system or from study of plans and other prescriptive documents. When these three approaches fail, it may be necessary to construct a ''trial'' system and determine issue formulation elements through experimentation and iteration with the trial system. These four methods (asking, study of an existing system, study of a normative systems, experimentation with a prototype system) are each very useful for information and system requirements determination (Davis 1982).

3.1.2. Models for Issue Analysis

The analysis portion of a DSS effort typically consists of two steps. First, the options or alternatives defined in issue formulation are analyzed to assess their expected impacts on needs and objectives. This is often called impact assessment or impact analysis. Second, a refinement or optimization effort

is often desirable. This is directed towards refinement or fine-tuning a potentially viable alternative through adjustment of the parameters within that alternative so as to obtain maximum performance in terms of needs satisfaction, subject to the given constraints.

Simulation and modeling methods are based on the conceptualization and use of an abstraction, or model, that hopefully behaves in a similar way as the real system. Impacts of alternative courses of action are studied through use of the model, something that often cannot easily be done through experimentation with the real system. Models are, of necessity, dependent on the value system and the purpose behind utilization of a model. We want to be able to determine the correctness of predictions based on usage of a model and thus be able to validate the model. There are three essential steps in constructing a model:

1. Determine those issue formulation elements that are most relevant to a particular problem.
2. Determine the structural relationships among these elements.
3. Determine parametric coefficients within the structure.

We should interpret the word ''model'' here as an abstract generalization of an object or system. Any set of rules and relationships that describes something is a model of that thing. The MBMS of a DSS will typically contain formal models that have been stored into the model base of the support system. Much has been published in the area of simulation realization of models, including a recent handbook (Banks 1998).

Gaming methods are basically modeling methods in which the real system is simulated by people who take on the roles of real-world actors. The approach may be very appropriate for studying situations in which the reactions of people to each others actions are of great importance, such as competition between individuals or groups for limited resources. It is also a very appropriate learning method. Conflict analysis (Fraser and Hipel 1984; Fang et al. 1993) is an interesting and appropriate game theory-based approach that may result in models that are particularly suitable for inclusion into the model base of a MBMS. A wealth of literature concerning formal approaches to mathematical games (Dutta 1999).

Trend extrapolation or time series forecasting models, or methods, are particularly useful when sufficient data about past and present developments are available, but there is little theory about underlying mechanisms causing change. The method is based on the identification of a mathematical description or structure that will be capable of reproducing the data. Then this description is used to extend the data series into the future, typically over the short to medium term. The primary concern is with input–output matching of observed input data and results of model use. Often little attention is devoted to assuring process realism, and this may create difficulties affecting model validity. While such models may be functionally valid, they may not be purposefully or structurally valid.

Continuous-time dynamic-simulation models, or methods, are generally based on postulation and qualification of a causal structure underlying change over time. A computer is used to explore long-range behavior as it follows from the postulated causal structure. The method can be very useful as a learning and qualitative forecasting device. Often it is expensive and time consuming to create realistic dynamic simulation models. Continuous-time dynamic models are quite common in the physical sciences and in much of engineering (Sage and Armstrong 2000).

Input-output analysis models are especially designed for study of equilibrium situations and requirements in economic systems in which many industries are interdependent. Many economic data formats are directly suited for the method. It is relatively simple conceptually, and can cope with many details. Input–output models are often very large.

Econometrics or macroeconomic models are primarily applied to economic description and forecasting problems. They are based on both theory and data. Emphasis is placed an specification of structural relations, based upon economic theory, and the identification of unknown parameters, using available data, in the behavioral equations. The method requires expertise in economics, statistics, and computer use. It can be quite expensive and time consuming. Macroeconomic models have been widely used for short- to medium-term economic analysis and forecasting.

Queuing theory and discrete event simulation models are often used to study, analyze, and forecast the behavior of systems in which probabilistic phenomena, such as waiting lines, are of importance. Queuing theory is a mathematical approach, while discrete-event simulation generally refers to computer simulation of queuing theory type models. The two methods are widely used in the analysis and design of systems such as toll booths, communication networks, service facilities, shipping terminals, and scheduling.

Regression analysis models and estimation theory models are very useful for the identification of mathematical relations and parameter values in these relations from sets of data or measurements. Regression and estimation methods are used frequently in conjunction with mathematical modeling, in particular with trend extrapolation and time series forecasting, and with econometrics. These methods are often also used to validate models. Often these approaches are called system identifi-

cation approaches when the goal is to identify the parameters of a system, within an assumed structure, such as to minimize a function of the error between observed data and the model response.

Mathematical programming models are used extensively in operations research systems analysis and management science practice for resource allocation under constraints, planning or scheduling, and similar applications. It is particularly useful when the best equilibrium or one-time setting has to be determined for a given policy or system. Many analysis issues can be cast as mathematical programming problems. A very significant number of mathematical programming models have been developed, including linear programming, nonlinear programming, integer programming, and dynamic programming. Many appropriate reference texts, including Hillier and Lieberman (1990, 1994), discuss this important class of modeling and analysis tools.

3.1.3. Issue Interpretation Models

The third step in a decision support systems use effort starts with evaluation and comparison of alternatives, using the information gained by analysis. Subsequently, one or more alternatives are selected and a plan for their implementation is designed. Thus, an MBMS must provide models for interpretation, including evaluation, of alternatives.

It is important to note that there is a clear and distinct difference between the refinement of individual alternatives, or optimization step of analysis, and the evaluation and interpretation of the sets of refined alternatives that result from the analysis step. In some few cases, refinement or optimization of individual alternative decision policies may not be needed in the analysis step. More than one alternative course of action or decision must be available; if there is but a single policy alternative, then there really is no decision to be taken at all. Evaluation of alternatives is always needed. It is especially important to avoid a large number of cognitive biases in evaluation and decision making. Clearly, the efforts involved in the interpretation step of evaluation and decision making interact most strongly with the efforts in the other steps of the systems process. A number of methods for evaluation and choice making are of importance. A few will be described briefly here.

Decision analysis (Raiffa 1968) is a very general approach to option evaluation and selection. It involves identification of action alternatives and possible consequences, identification of the probabilities of these consequences, identification of the valuation placed by the decision maker upon these consequences, computation of the expected value of the consequences, and aggregation or summarization of these values for all consequences of each action. In doing this we obtain an evaluation of each alternative act, and the one with the highest value is the most preferred action or option.

Multiple-attribute utility theory (Keeney and Raiffa 1976) has been designed to facilitate comparison and ranking of alternatives with many attributes or characteristics. The relevant attributes are identified and structured and a weight or relative utility is assigned by the decision maker to each basic attribute. The attribute measurements for each alternative are used to compute an overall worth or utility for each alternative. Multiple attribute utility theory allows for various types of worth structures and for the explicit recognition and incorporation of the decision maker's attitude towards risk in the utility computation.

Policy Capture (or *Social Judgment*) *Theory* (Hammond et al. 1980) has also been designed to assist decision makers in making values explicit and known. It is basically a descriptive approach toward identification of values and attribute weights. Knowing these, one can generally make decisions that are consistent with values. In policy capture, the decision maker is asked to rank order a set of alternatives. Then alternative attributes and their attribute measures or scores are determined by elicitation from the decision maker for each alternative. A mathematical procedure involving regression analysis is used to determine that relative importance, or weight, of each attribute that will lead to a ranking as specified by the decision maker. The result is fed back to the decision maker, who typically will express the view that some of his or her values, in terms of the weights associated with the attributes, are different. In an iterative learning process, preference weights and/or overall rankings are modified until the decision maker is satisfied with both the weights and the overall alternative ranking.

Many efforts have been made to translate the theoretical findings in decision analysis to practice. Klein (1998), Matheson and Matheson (1998), and Hammond et al. (1999) each provide different perspectives relative to these efforts.

3.2. Model Base Management

As we have noted, an effective model base management system (MBMS) will make the structural and algorithmic aspects of model organization and associated data processing transparent to users of the MBMS. Such tasks as specifying explicit relationships between models to indicate formats for models and which model outputs are input to other models are not placed directly on the user of a MBMS but handled directly by the system. Figure 11 presents a generic illustration of a MBMS. It shows a collection of models or model base, a model base manager, a model dictionary, and connections to the DBMS and the DGMS.

A number of capabilities should be provided by an integrated and shared MBMS of a DSS (Barbosa and Herko 1980; Liang 1985) construction, model maintenance, model storage, model manipulation, and model access (Applegate et al. 1986). These involve control, flexibility, feedback, interface, redundancy reduction, and increased consistency:

1. *Control:* The DSS user should be provided with a spectrum of control. The system should support both fully automated and manual selection of models that seem most useful to the user for an intended application. This will enable the user to proceed at the problem-solving pace that is most comfortable given the user's experiential familiarity with the task at hand. It should be possible for the user to introduce subjective information and not have to provide full information. Also, the control mechanism should be such that the DSS user can obtain a recommendation for action with this partial information at essentially any point in the problem-solving process.
2. *Flexibility:* The DSS user should be able to develop part of the solution to the task at hand using one approach and then be able to switch to another modeling approach if this appears preferable. Any change or modification in the model base will be made available to all DSS users.
3. *Feedback:* The MBMS of the DSS should provide sufficient feedback to enable the user to be aware of the state of the problem-solving process at any point in time.
4. *Interface:* The DSS user should feel comfortable with the specific model from the MBMS that is in use at any given time. The user should not have to supply inputs laboriously when he or she does not wish to do this.
5. *Redundancy reduction:* This should occur through use of shared models and associated elimination of redundant storage that would otherwise be needed.
6. *Increased consistency:* This should result from the ability of multiple decision makers to use the same model and the associated reduction of inconsistency that would have resulted from use of different data or different versions of a model.

In order to provide these capabilities, it appears that a MBMS design must allow the DSS user to:

1. Access and retrieve existing models
2. Exercise and manipulate existing models, including model instantiation, model selection, and model synthesis, and the provision of appropriate model outputs
3. Store existing models, including model representation, model abstraction, and physical and logical model storage
4. Maintain existing models as appropriate for changing conditions
5. Construct new models with reasonable effort when they are needed, usually by building new models by using existing models as building blocks

A number of auxiliary requirements must be achieved in order to provide these five capabilities. For example, there must be appropriate communication and data changes among models that have been combined. It must also be possible to locate appropriate data from the DBMS and transmit it to the models that will use it.

It must also be possible to analyze and interpret the results obtained from using a model. This can be accomplished in a number of ways. In this section, we will examine two of them: relational MBMS and expert system control of an MBMS. The objective is to provide an appropriate set of models for the model base and appropriate software to manage the models in the model base; integration of the MBMS with the DBMS; and integration of the MBMS with the DGMS. We can expand further on each of these needs. Many of the technical capabilities needed for a MBMS will be analogous to those needed for a DBMS. These include model generators that will allow rapid building of specific models, model modification tools that will enable a model to be restructured easily on the basis of changes in the task to be accomplished, update capability that will enable changes in data to be input to the model, and report generators that will enable rapid preparation of results from using the system in a form appropriate for human use.

Like a relational view of data, a relational view of models is based on a mathematical theory of relations. Thus, a model is viewed as a virtual file or virtual relation. It is a subset of the Cartesian product of the domain set that corresponds to these input and output attributes. This virtual file is created, ideally, through exercising the model with a wide spectrum of inputs. These values of inputs and the associated outputs become records in the virtual file. The input data become key attributes and the model output data become content attributes.

Model base structuring and organization is very important for appropriate relational model management. Records in the virtual file of a model base are not individually updated, however, as they

are in a relational database. When a model change is made, all of the records that comprise the virtual file are changed. Nevertheless, processing anomalies are possible in relational model management. Transitive dependencies in a relation, in the form of functional dependencies that affect only the output attributes, do occur and are eliminated by being projected into an appropriate normal form.

Another issue of considerable importance relates to the contemporary need for usable model base query languages and to needs within such languages for relational completeness. The implementation of joins is of concern in relational model base management just as it is in relational database management. A relational model join is simply the result of using the output of one model as the input to another model. Thus, joins will normally be implemented as part of the normal operation of software, and a MBMS user will often not be aware that they are occurring. However, there can be cycles, since the output from a first model may be the input to a second model, and this may become the input to the first model. Cycles such as this do not occur in relational DBMS.

Expert system applications in MBMS represent another attractive possibility. Four different potential opportunities exist. It might be possible to use expert system technology to considerable advantage in the construction of models (Hwang 1985; Murphy and Stohr 1986), including decisions with respect to whether or not to construct models in terms of the cost and benefits associated with this decision. AI and expert system technology may potentially be used to integrate models. This model integration is needed to join models. AI and expert system technology might be potentially useful in the validation of models. Also, this technology might find potential use in the interpretation of the output of models. This would especially seem to be needed for large-scale models, such as large linear programming models. While MBMS approaches based on a relational theory of models and expert systems technology are new as of this writing, they offer much potential for implementing model management notions in an effective manner. As has been noted, they offer the prospect of data as models (Dolk 1986) that may well prove much more useful than the conventional information systems perspective of "models as data."

4. DIALOG GENERATION AND MANAGEMENT SYSTEMS

In our efforts in this chapter, we envision a basic DSS structure of the form shown in Figure 12. This figure also shows many of the operational functions of the database management system (DBMS) and the model base management system (MBMS). The primary purpose of the dialog generation and management system (DGMS) is to enhance the propensity and ability of the system user to use and benefit from the DSS. There are doubtless few users of a DSS who use it because of necessity. Most uses of a DSS are optional, and the decision maker must have some motivation and desire to use a DSS or it will likely remain unused. DSS use can occur in a variety of ways. In all uses of a DGMS, it is the DGMS that the user interacts with. In an early seminal text, Bennett (1983) posed three questions to indicate this centrality:

1. What presentations is the user able to *see* at the DSS display terminal?
2. What must the user know about what is seen at the display terminal in order to use the DSS?
3. What can the DSS user *do* with the systems that will aid in accomplishing the intended purpose?

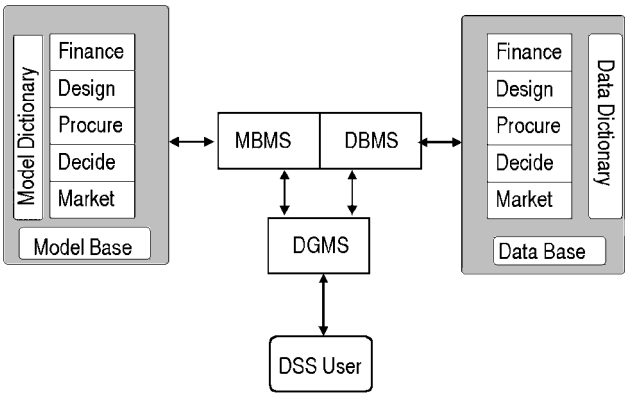

Figure 12 The DGMS as the User-Focused System in a DSS.

Bennett refers to these elements as the *presentation language,* the *knowledge base,* and the *action language*.

It is generally felt that there are three types of languages or modes of human communications: *words, mathematics,* and *graphics*. The presentation and action languages, and the knowledge base in general, many contain any or all of these three language types. The mix of these that is appropriate for any given DSS task will be a function of the task itself, the environment into which the task is embedded, and the nature of the experiential familiarity of the person performing the task with the task and with the environment. This is often called the contingency task structure. The DSS, when one is used, becomes a fourth ingredient, although it is really much more of a vehicle supporting effective use of words, mathematics, and graphics than it is a separate fourth ingredient.

Notions of DGMS design are relatively new, especially as a separately identified portion of the overall design effort. To be sure, user interface design is not at all new. However, the usual practice has been to assign the task of user interface design to the design engineers responsible for the entire system. In the past, user interfaces were not given special attention. They were merely viewed as another hardware and software component in the system. System designers were often not particularly familiar with, and perhaps not even especially interested in, the user-oriented design perspectives necessary to produce a successful interface design. As a result, many user interface designs have provided more what the designer wanted than what the user wanted and needed. Notions of dialog generation and dialog management extend far beyond interface issues, although the interface is a central concern in dialog generation and dialog management. A number of discussions of user interface issues are contained in Part IIIB of this Handbook.

Figure 14 illustrates an attribute tree for interface design based on the work of Smith and Mosier (1986). This attribute tree can be used, in conjunction with the evaluation methods of decision analysis, to evaluate the effectiveness of interface designs. There are other interface descriptions, some of which are less capable of instrumental measurement than these. On the basis of a thorough study of much of the human–computer interface and dialog design literature, Schneiderman (1987) has identified eight primary objectives, often called the ''golden rules'' for dialog design:

1. Strive for consistency of terminology, menus, prompts, commands, and help screens.
2. Enable frequent users to use shortcuts that take advantage of their experiential familiarity with the computer system.
3. Offer informative feedback for every operator action that is proportional to the significance of the action.
4. Design dialogs to yield closure such that the system user is aware that specific actions have been concluded and that planning for the next set of activities may now take place.
5. Offer simple error handling such that, to the extent possible, the user is unable to make a mistake. Even when mistakes are made, the user should not have to, for example, retype an entire command entry line. Instead, the user should be able to just edit the portion that is incorrect.
6. Permit easy reversal of action such that the user is able to interrupt and then cancel wrong commands rather than having to wait for them to be fully executed.

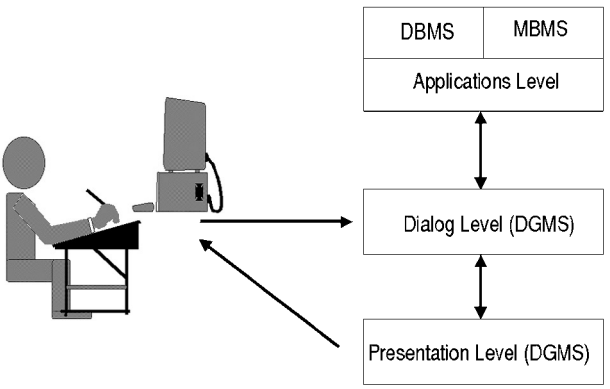

Figure 13 Dialog Generation and Management System Architecture.

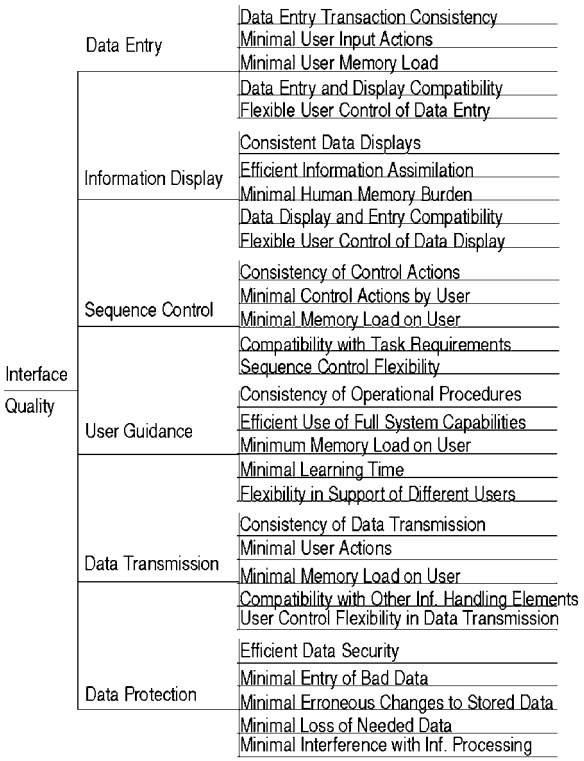

Figure 14 Attribute Tree of Smith and Mosier Elements for Interface Design Evaluation.

7. Support internal locus of control such that users are always the initiators of actions rather than the reactors to computer actions.
8. Reduce short-term memory load such that users are able to master the dialog activities that they perform in a comfortable and convenient manner.

Clearly, all of these will have specific interpretations in different DGMS environments and need to be sustained in and capable of extension for a variety of environments.

Human–computer interaction and associated interface design issues are of major contemporary importance, so it is not surprising that there have been a number of approaches to them. Some of these approaches are almost totally empirical. Some involve almost totally theoretical and formal models (Harrison and Thimbleby 1990). Others attempt approximate predictive models that are potentially useful for design purposes (Card et al. 1983). One word that has appeared often in these discussions is "consistency." This is Schneiderman's first golden rule of dialog design, and many other authors advocate it as well. A notable exception to this advocacy of consistency comes from Grudin (1989), who argues that issues associated with consistency should be placed in a very broad context. He defines three types of consistency:

1. *Internal consistency:* The physical and graphic layout of the computer system, including such characteristics as those associated with command naming and use and dialog forms, are consistent if these internal features of the interface are the same across applications.
2. *External consistency:* If an interface has unchanging use features when compared to another interface with which the user is familiar, it is said to be externally consistent.
3. *Applications consistency:* If the user interface uses metaphors or analogous representations of objects that correspond to those of the real-world application, then the interface may be said to correspond to experientially familiar features of the world and to be applications consistent.

Two of Grudin's observations relevant to interface consistency are that ease of learning can conflict with ease of use, especially as experiential familiarity with the interface grows, and that consistency can work against both ease of use and learning. On the basis of some experiments illustrating these hypotheses, he establishes the three appropriate dimensions for consistency above.

A number of characteristics are desirable for user interfaces. Roberts and Moran (1982) identify the most important attributes of text editors as functionality of the editor, learning time required, time required to perform tasks, and errors committed. To this might be added the cost of evaluation. Harrison and Hix (1989) identify usability, completeness, extensibility, escapability, integration, locality of definition, structured guidance, and direct manipulation as well in their more general study of user interfaces. They also note a number of tools useful for interface development, as does Lee (1990).

Ravden and Johnson (1989) evaluate usability of human computer interfaces. They identify nine top-level attributes: visual clarity, consistency, compatibility, informative feedback, explicitness, appropriate functionality, flexibility and control, error prevention and correction, and user guidance and support. They disaggregate each into a number of more measurable attributes. These attributes can be used as part of a standard multiple-attribute evaluation.

A goal in DGMS design is to define an abstract user interface that can be implemented on specific operating systems in different ways. The purpose of this is to allow for device independence such that, for example, switching from a command line interface to a mouse-driven pull-down-menu interface can be easily accomplished. Separating the application from the user interface should do much towards ensuring portability across operating systems and hardware platforms without modifying the MBMS and the DBMS, which together comprise the applications software portions of the DSS.

5. GROUP AND ORGANIZATIONAL DECISION SUPPORT SYSTEMS

A group decision support system (GDSS) is an information technology-based support system designed to provide decision making support to groups and/or organizations. This could refer to a group meeting at one physical location at which judgments and decisions are made that affect an organization or group. Alternatively, it could refer to a spectrum of meetings of one or more individuals, distributed in location, time, or both. GDSSs are often called *organizational decision support systems,* and other terms are often used, including *executive support systems* (ESSs), which are information technology-based systems designed to support executives and managers, and *command and control systems,* which is a term often used in the military for a decision support system. We will generally use GDSS to describe all of these.

Managers and other knowledge workers spend much time in meetings. Much research into meeting effectiveness suggests that it is low, and proposals have been made to increase this through information technology support (Johansen 1988). Specific components of this information technology-based support might include computer hardware and software, audio and video technology, and communications media. There are three fundamental ingredients in this support concept: technological support facilities, the support processes provided, and the environment in which they are embedded. Kraemer and King (1988) provide a noteworthy commentary on the need for group efforts in their overview of GDSS efforts. They suggest that group activities are economically necessary, efficient as a means of production, and reinforcing of democratic values.

There are a number of predecessors for group decision support technology. Decision rooms, or situation rooms, where managers and boards meet to select from alternative plans or courses of action, are very common. The first computer-based decision support facility for group use is attributed to Douglas C. Engelbart, the inventor of the (computer) mouse, at Stanford in the 1960s. A discussion of this and other early support facilities is contained in Johansen (1988).

Engelbart's electronic boardroom-type design is acknowledged to be the first type of information technology-based GDSS. The electronic format was, however, preceded by a number of nonelectronic formats. The Cabinet war room of Winston Churchill is perhaps the most famous of these. Maps placed on the wall and tables for military decision makers were the primary ingredients of this room. The early 1970s saw the introduction of a number of simple computer-based support aids into situation rooms. The first system that resembles the GDSS in use today is often attributed to Gerald Wagner, the Chief Executive Officer of Execucom, who implemented a planning laboratory made up of a U-shaped table around which people sat, a projection TV system for use as a public viewing screen, individual small terminals and keyboards available to participants, and a minicomputer to which the terminals and keyboards were connected. This enabled participants to vote and to conduct simple spreadsheet-like exercises. Figure 15 illustrates the essential features of this concept. Most present-day GDSS centralized facilities look much like the conceptual illustration of a support room, or situation room, shown in this Figure.

As with a single-user DSS, appropriate questions for a GDSS that have major implications for design concern the perceptions and insights that the group obtains through use of the GDSS and the

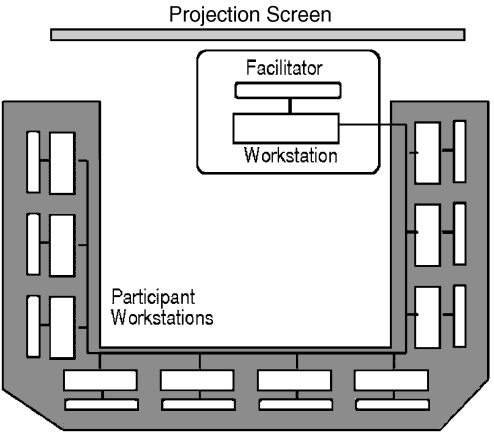

Figure 15 Early Group Decision Support System (GDSS) Situation Room.

activities that can be carried out through its use. Also, additional concerns arise regarding the public screen, interactions between the public screen and individual screens, the characteristics of individual work screens, and contingency task structural variables associated with the individuals in the group using the GDSS (Gray and Olfman 1989).

Huber (1982) has indicated both the needs for GDSS and how an appropriately designed GDSS can meet these needs. He identifies four interacting and complicating concerns:

1. Effective decision making requires not only obtaining an appropriate decision, but also ensuring that participants are happy with the process used to reach the decision and will be willing to meet and work cooperatively in the future.
2. Productivity losses occur because of dominant individuals and group pressures that lead to conformity of thought, or groupthink.
3. Miscommunications are common in group situations.
4. Insufficient time is often spent in situation assessment, problem exploration, and generation of alternative courses of action.

Huber further indicates that a GDSS can help improve the unaided decision situation, which often suffers from imperfect information processing and suboptimal decision selection.

A GDSS is made up of:

1. *Technological components,* in terms of computer hardware and software, and communication equipment
2. *Environmental components,* in terms of the people involved, their locations in time and space, and their experiential familiarity with the task at hand
3. *Process components,* or variables made up of the conventions used to support task performance and enable the other components of decision making to function appropriately.

We have already described the technological design features of DSSs. Thus, our commentary on technological design features of a GDSS will be brief. We do need to provide a perspective on groups and organizations, and we will do this in our next two sections. Then we will turn to some architectural considerations specifically relevant to GDSS.

5.1. Information Needs for Group and Organizational Decision Making

The nature of the decisions and the type of information required differ across each of these four levels identified in Figure 1. Generally, operational activities occur much more frequently than strategic planning activities. Also, there is a difference in the degree to which the knowledge required for each of these levels is structured. In 1960, Herbert Simon (Simon 1960) described decisions as structured or unstructured, depending upon whether the decision making process can be explicitly described prior to the time when it is necessary to make a decision. This taxonomy would seem to

lead directly to that in which expert skills (holistic reasoning), rules (heuristics), or formal reasoning (holistic analysis) are normatively used for judgment. Generally, operational performance decisions are much more likely than strategic planning decisions to be prestructured. This gives rise to a number of questions concerning efficiency and effectiveness tradeoffs between training and aiding (Rouse 1991) that occur at these levels.

There are a number of human abilities that a GDSS should augment.

1. It should help the decision maker to formulate, frame, or assess the decision situation. This includes identifying the salient features of the environment, recognizing needs, identifying appropriate objectives by which we are able to measure successful resolution of an issue, and generating alternative courses of action that will resolve the needs and satisfy objectives.
2. It should provide support in enhancing the abilities of the decision maker to obtain and analyze the possible impacts of the alternative courses of action.
3. It should have the capability to enhance the decision maker's ability to interpret these impacts in terms of objectives. This interpretation capability will lead to evaluation of the alternatives and selection of a preferred alternative option.

Associated with each of these three formal steps of formulation, analysis, and interpretation must be the ability to acquire, represent, and utilize information and associated knowledge and the ability to implement the chosen alternative course of action.

Many attributes will affect the quality and usefulness of the information that is obtained, or should be obtained, relative to any given decision situation. These variables are very clearly contingency task dependent. Among these attributes are the following (Keen and Scott Morton 1978).

- *Inherent and required accuracy of available information:* Operational control and performance situations will often deal with information that is relatively accurate. The information in strategic planning and management control situations is often inaccurate.
- *Inherent precision of available information:* Generally, information available for operational control and operational performance decisions is very imprecise.
- *Inherent relevancy of available information:* Operational control and performance situations will often deal with information that is fairly relevant to the task at hand because it has been prepared that way by management. The information in strategic planning and management control situations is often obtained from the external environment and may be irrelevant to the strategic tasks at hand, although it may not initially appear this way.
- *Inherent and required completeness of available information:* Operational control and performance situations will often deal with information that is relatively complete and sufficient for operational performance. The information in strategic planning and management control situations is often very incomplete and insufficient to enable great confidence in strategic planning and management control.
- *Inherent and required verifiability of available information:* Operational control and performance situations will often deal with information that is relatively verifiable to determine correctness for the intended purpose. The information in strategic planning and management control situations is often unverifiable, or relatively so, and this gives rise to a potential lack of confidence in strategic planning and management control.
- *Inherent and required consistency and coherency of available information:* Operational control and performance situations will often deal with information that is relatively consistent and coherent. The information in strategic planning and management control situations is often inconsistent and perhaps even contradictory or incoherent, especially when it comes from multiple external sources.
- *Information scope:* Generally, but not always, operational decisions are made on the basis of relatively narrow scope information related to well-defined events that are internal to the organization. Strategic decisions are generally based upon broad-scope information and a wide range of factors that often cannot be fully anticipated prior to the need for the decision.
- *Information quantifiability:* In strategic planning, information is very likely to be highly qualitative, at least initially. For operational decisions, the available information is often highly quantified.
- *Information currency:* In strategic planning, information is often rather old, and it is often difficult to obtain current information about the external environment. For operational control decisions, very current information is often needed and present.
- *Needed level of detail:* Often very detailed information is needed for operational decisions. Highly aggregated information is often desired for strategic decisions. There are many difficulties associated with information summarization that need attention.

- *Time horizon for information needed:* Operational decisions are typically based on information over a short time horizon, and the nature of the control may change very frequently. Strategic decisions are based on information and predictions based on a long time horizon.
- *Frequency of use:* Strategic decisions are made infrequently, although they are perhaps refined fairly often. Operational decisions are made quite frequently and are relatively easily changed.
- *Internal or external information source:* Operational decisions are often based upon information that is available internal to the organization, whereas strategic decisions are much more likely to be dependent upon information content that can only be obtained external to the organization.

These attributes, and others, could be used to form the basis for an evaluation of information quality in a decision support system.

Information is used in a DSS for a variety of purposes. In general, information is equivalent to, or may be used as, evidence in situations in which it is relevant. Often information is used directly as a basis for testing an hypothesis. Sometimes it is used indirectly for this purpose. There are three different conditions for describing hypotheses (Schum 1987, 1994):

1. Different alternative hypotheses or assessments are possible if evidence is imperfect in any way. A hypothesis may be imperfect if it is based on imperfect information. Imperfect information refers to information that is incomplete, inconclusive, unreliable, inconsistent, or uncertain. Any or all of these alternate hypotheses may or may not be true.
2. Hypotheses may refer to past, present, or future events.
3. Hypotheses may be sharp (specific) or diffuse (unspecified). Sharp hypotheses are usually based on specific evidence rather than earlier diffuse hypotheses. An overly sharp hypothesis may contain irrelevant detail and invite invalidation by disconfirming evidence on a single issue in the hypothesis. An overly diffuse hypothesis may be judged too vague and too uninteresting by those who must make a decision based upon the hypothesis, even though the hypothesis might have been described in a more cogent manner.

The support for any hypothesis can always be improved by either revising a portion of the hypothesis to accommodate new evidence or gathering more evidence that infers the hypothesis. Hypotheses can be potentially significant for four uses:

1. *Explanations:* An explanation usually involves a model, which can be elaborate or simple. The explanation consists of the rationale for why certain events occurred.
2. *Event predictions:* In this case the hypothesis is proposed for a possible future event. It may include the date or period when the possible event will occur.
3. *Forecasting and estimation:* This involves the generation of a hypothesis based on data that does not exist or that is inaccessible.
4. *Categorization:* Sometimes it is useful to place persons, objects, or events into certain categories based upon inconclusive evidence linking the persons, objects, or events to these categories. In this case the categories represent hypotheses about category membership.

Assessment of the validity of a given hypothesis is inductive in nature. The generation of hypotheses and determination of evidence relevant to these hypotheses involve deductive and abductive reasoning. Hypotheses may be generated on the basis of the experience and prior knowledge that leads to analogous representations and recognitional decision making, as noted by Klein (1990, 1998).

Although no theory has been widely accepted on how to quantify the value of evidence, it is important to be able to support a hypothesis in some logical manner. Usually there is a major hypothesis that is inferred by supporting hypotheses, and each of these supporting hypotheses is inferred by its supporting hypothesis, and so on. Evidence is relevant to the extent that it causes one to increase or decrease the likeliness of an existing hypothesis, or modify an existing hypothesis, or create a new hypothesis. Evidence is direct if it has a straightforward bearing on the validity of the main hypothesis. Evidence is indirect if its effect on the main hypothesis is inferred through at least one other level of supporting hypothesis.

In many cases, it is necessary to acquire, represent, use, and/or communicate knowledge that is imperfect. This is especially important in group decision situations. In describing the structure of the beliefs and the statements that people make about issues that are of importance to them, the nature of the environment that surrounds them, as well as the ways in which people reason and draw conclusions about the environment and issues that are embedded into the environment, especially when there are conflicting pieces of information and opinions concerning these, people often attempt to use one or more of the forms of logical reasoning. Many important works deal with this subject. Of particular interest here is the work of Toulmin and his colleagues (Toulmin et al. 1979), who have described an explicit model of logical reasoning that is subject to analytical inquiry and computer

implementation. The model is sufficiently general that it can be used to represent logical reasoning in a number of application areas.

Toulmin assumes that whenever we make a claim, there must be some ground on which to base our conclusion. He states that our thoughts are generally directed, in an inductive manner, from the grounds to the claim, each of which are statements that may be used to express both facts and values. As a means of explaining observed patterns of stating a claim, there must be some reason that can be identified with which to connect the grounds and the claim. This connection, called the warrant, gives the grounds–claim connection its logical validity.

We say that the grounds support the claim on the basis of the existence of a warrant that explains the connection between the grounds and the claim. It is easy to relate the structure of these basic elements with the process of inference, whether inductive or deductive, in classical logic. The warrants are the set of rules of inference, and the grounds and claim are the set of well-defined propositions or statements. It will be only the sequence and procedures, as used to formulate the three basic elements and their structure in a logical fashion, that will determine the type of inference that is used.

Sometimes, in the course of reasoning about an issue, it is not enough that the warrant will be the absolute reason to believe the claim on the basis of the grounds. For that, there is a need for further backing to support the warrant. It is this backing that provides for reliability, in terms of truth, associated with the use of a warrant. The relationship here is analogous to the way in which the grounds support the claim. An argument will be valid and will give the claim solid support only if the warrant is relied upon and is relevant to the particular case under examination. The concept of logical validity of an argument seems to imply that we can make a claim only when both the warrant and the grounds are certain. However, imprecision and uncertainty in the form of exceptions to the rules or low degree of certainty in both the grounds and the warrant do not prevent us on occasion from making a *hedge* or, in other words, a vague claim. Often we must arrive at conclusions on the basis of something less than perfect evidence, and we put those claims forward not with absolute and irrefutable truth but with some doubt or degree of speculation.

To allow for these cases, modal qualifiers and possible rebuttals may be added to this framework for logical reasoning. Modal qualifiers refer to the strength or weakness with which a claim is made. In essence, every argument has a certain modality. Its place in the structure presented so far must reflect the generality of the warrants in connecting the grounds to the claim. Possible rebuttals, on the other hand, are exceptions to the rules. Although modal qualifiers serve the purpose of weakening or strengthening the validity of a claim, there may still be conditions that invalidate either the grounds or the warrants, and this will result in deactivating the link between the claim and the grounds. These cases are represented by the possible rebuttals.

The resulting structure of logical reasoning provides a very useful framework for the study of human information processing activities. The order in which the six elements of logical reasoning have been presented serves only the purpose of illustrating their function and interdependence in the structure of an argument about a specific issue. It does not represent any normative pattern of argument formation. In fact, due to the dynamic nature of human reasoning, the concept formation and framing that result in a particular structure may occur in different ways. The six-element model of logical reasoning is shown in Figure 16.

Computer-based implementations of Figure 16 may assume a Bayesian inferential framework for processing information. Frameworks for Bayesian inference require probability values as primary inputs. Because most events of interest are unique or little is known about their relative frequencies of occurrence, the assessment of probability values usually requires human judgment. Substantial

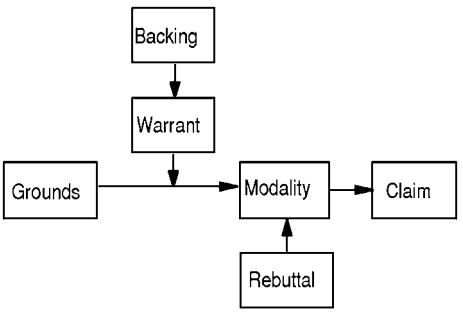

Figure 16 Representation of Toulmin Logic Structure.

psychological research has shown that people are unable to elicit probability values consistent with the rules of probabilities or to process information as they should, according to the laws of probability, in revising probability assessment when new information is obtained. For example, when people have both causal and diagnostic implications, they should weigh the causal and diagnostic impacts of the evidence. More often, however, unaided humans will reach judgments that suggest they apparently assess conditional probabilities primarily in terms of the direct causal effect of the impacts. If A is perceived to be the cause of B, for example, people will usually associate higher probabilities with $P(B|A)$ than they will with $P(A|B)$. Studies concerning the elicitation of probability values often report that individuals found it easier and showed more confidence in assessing $P(A|B)$ if B was causal to A. This strongly suggests that the choice of which form of inference to invoke depends more on the level of familiarity of the observer with the task at hand and the frame adopted to initially represent the knowledge. Again, this supports the wisdom of appropriate structuring of decision situations.

In general, grounds can be categorized by the several means in which are warranted:

- Empirical observation
- Expert judgment
- Enumerative induction (statistics)
- Experiment (hypothesis test)
- Direct fact

A decision assessment system based on this concept is described in Janssen and Sage (2000) together with application to the issues in public policy decision making associated with agricultural pest control. This system can support several user groups, who can use the support system to state arguments for or against an important policy issue and to assist in identifying and evaluating alternatives for implementation as policy decisions. The resulting decision support system assists in improving the clarity of the lines of reasoning used in specific situations; the warrants, grounds, and backings that are used to support claims and specific lines of reasoning; and the contradictions, rebuttals, and arguments surrounding each step in the reasoning process associated with evaluating a claim or counterclaim. Thus, experts and decision makers with differing views and backgrounds can better understand each other's thought processes in complex situations. The net effect is enhanced communications and understanding of the whole picture and, in many cases, consensus on decisions to be taken.

A number of human information processing capabilities and limitations interact with organizational arrangements and task requirements to strongly influence resource allocations for organizational problem solving and decision making. These needs have provided much motivation for the development of group decision support systems (GDSSs). The purpose of these GDSSs as computerized aids to planning, problem solving, and decision making include:

1. Removing a number of common communication barriers in groups and organizations
2. Providing techniques for the formulation, analysis, and interpretation of decisions
3. Systematically directing the discussion process and associated problem solving and decision making in terms of the patterns, timing, and content of the information that influences the actions that follow from decisions that have been taken

A number of variations and permutations are possible in the provision of group and organizational decision support. These are associated with specific realization or architectural format for a GDSS to support a set of GDSS performance objectives for a particular task in a particular environment.

The same maladies that affect individual decision making and problem solving behavior, as well as many others, can result from group and organizational limitations. A considerable body of knowledge, generally qualitative, exists relative to organizational structure, effectiveness, and decision making in organizations. The majority of these studies suggest that a bounded rationality or satisficing perspective, often heavily influenced by bureaucratic political considerations, will generally be the decision perspective adopted in actual decision making practice in organizations. To cope with this effectively requires the ability to deal concurrently with technological, environmental, and process concerns as they each, separately and collectively, motivate group and organizational problem solving issues.

The influencers of decision and decision process quality are particularly important in this. We should sound a note of caution regarding some possibly overly simplistic notions relative to this. Welch (1989) identifies a number of potential imperfections in organizational decision making and discusses their relationship to decision process quality. In part, these are based on an application of seven symptoms identified in Herek et al. (1987) to the Cuban Missile Crisis of 1962. These potential imperfections include:

1. Omissions in surveying alternative courses of action
2. Omissions in surveying objectives
3. Failure to examine major costs and risks of the selected course of action (COA)
4. Poor information search, resulting in imperfect information
5. Selective bias in processing available information
6. Failure to reconsider alternatives initially rejected, potentially by discounting favorable information and overweighing unfavorable information
7. Failure to work out detailed implementation, monitoring, and contingency plans

The central thrust of this study is that the relationship between the quality of the decision making process and the quality of the outcome is difficult to establish. This strongly suggests the usefulness of the contingency task structural model construct and the need for approaches that evaluate the quality of processes, as well as decisions and outcomes, and that consider the inherent embedding of outcomes and decisions within processes that lead to these.

Organizational ambiguity is a major reason why much of the observed ''bounded rationality'' behavior is so pervasive. March (1983) and March and Wessinger-Baylon (1986) show that this is very often the case, even in situations when formal rational thought or ''vigilant information processing'' (Janis and Mann 1977) might be thought to be a preferred decision style. March (1983) indicates that there are at least four kinds of opaqueness or equivocality in organizations: *ambiguity of intention, ambiguity of understanding, ambiguity of history,* and *ambiguity of human participation.* These four ambiguities relate to an organization's structure, function, and purpose, as well as to the perception of these decision making agents in an organization. They influence the information that is communicated in an organization and generally introduce one or more forms of information imperfection. The notions of organizational management and organizational information processing are indeed inseparable. In the context of human information processing, it would not be incorrect to define the central purpose of management as development of a consensual grammar to ameliorate the effects of equivocality or ambiguity. This is the perspective taken by Karl Weick (1979, 1985) in his noteworthy efforts concerning organizations.

Starbuck (1985) notes that much direct action is a form of deliberation. He indicates that action should often be introduced earlier in the process of deliberation than it usually is and that action and thought should be integrated and interspersed with one another. The basis of support for this argument is that probative actions generate information and tangible results that modify potential thoughts. Of course, any approach that involves ''act now, think later'' behavior should be applied with considerable caution.

Much of the discussion to be found in the judgment, choice, and decision literature concentrates on what may be called formal reasoning and decision selection efforts that involve the issue resolution efforts that follow as part of the problem solving efforts of issue formulation, analysis, and interpretation that we have discussed here. There are other decision making activities, or decision-associated activities, as well. Very important among these are activities that allow perception, framing, editing and interpretation of the effects of actions upon the internal and external environments of a decision situation. These might be called information selection activities. There will also exist information retention activities that allow admission, rejection, and modification of the set of selected information or knowledge such as to result in short-term learning and long-term learning. Short-term learning results from reduction of incongruities, and long-term learning results from acquisition of new information that reflects enhanced understanding of an issue. Although the basic GDSS design effort may well be concerned with the short-term effects of various problem solving, decision making, and information presentation formats, the actual knowledge that a person brings to bear on a given problem is a function of the accumulated experience that the person possesses, and thus long-term effects need to be considered, at least as a matter of secondary importance.

It was remarked above that a major purpose of a GDSS is to enhance the value of information and, through this, to enhance group and organizational decision making. Three attributes of information appear dominant in the discussion thus far relative to value for problem solving purposes and in the literature in general:

1. *Task relevance:* Information must be relevant to the task at hand. It must allow the decision maker to know what needs to be known in order to make an effective and efficient decision. This is not as trivial a statement as might initially be suspected. Relevance varies considerably across individuals, as a function of the contingency task structure, and in time as well.
2. *Representational appropriateness:* In addition to the need that information be relevant to the task at hand, the person who needs the information must receive it in a form that is appropriate for use.

3. *Equivocality reduction:* It is generally accepted that high-quality information may reduce the imperfection or equivocality that might otherwise be present. This equivocality generally takes the form of uncertainty, imprecision, inconsistency, or incompleteness. It is very important to note that it is neither necessary nor desirable to obtain decision information that is unequivocal or totally ''perfect.'' Information need only be sufficiently unequivocal or unambiguous for the task at hand. To make it better may well be a waste of resources!

Each of these top-level attributes may be decomposed into attributes at a lower level. Each is needed as fundamental metrics for valuation of information quality. We have indicated that some of the components of equivocality or imperfection are uncertainty, imprecision, inconsistency, and incompleteness. A few of the attributes of representational appropriateness include naturalness, transformability to naturalness, and concision. These attributes of information presentation system effectiveness relate strongly to overall value of information concerns and should be measured as a part of the DSS and GDSS evaluation effort even though any one of them may appear to be a secondary theme.

We can characterize information in many ways. Among attributes that we noted earlier and might use are accuracy, precision, completeness, sufficiency, understandability, relevancy, reliability, redundancy, verifiability, consistency, freedom from bias, frequency of use, age, timeliness, and uncertainty. Our concerns with information involve at least five desiderata (Sage 1987):

1. Information should be presented in very clear and very familiar ways, such as to enable rapid comprehension.
2. Information should be such as to improve the precision of understanding of the task situation.
3. Information that contains an advice or decision recommendation component should contain an explication facility that enables the user to determine how and why results and advice are obtained.
4. Information needs should be based upon identification of the information requirements for the particular situation.
5. Information presentations and all other associated management control aspects of the support process should be such that the decision maker, rather than a computerized support system, guides the process of judgment and choice.

It will generally be necessary to evaluate a GDSS to determine the extent to which these information quality relevant characteristics are present.

5.2. The Engineering of Group Decision Support Systems

There are two fundamental types of decision making in an organization: individual decisions, made by a single person, and group or organizational decisions, made by a collection of two or more people. It is, of course, possible to disaggregate this still further. An individual decision may, for example, be based on the value system of one or more people and the individual making the decision may or may not have his or her values included. In a multistage decision process, different people may make the various decisions. Some authors differentiate between group and organizational decisions (King and Star 1992), but we see no need for this here, even though it may be warranted in some contexts. There can be no doubt at all, however, that a GDSS needs to be carefully matched to an organization that may use it.

Often groups make decisions differently from the way an individual does. Groups need protocols that allow effective inputs by individuals in the group or organization, a method for mediating a discussion of issues and inputs, and algorithms for resolving disagreements and reaching a group consensus. Acquisition and elicitation of inputs and the mediation of issues are usually local to the specific group, informed of personalities, status, and contingencies of the members of the group. Members of the group are usually desirous of cooperating in reaching a consensus on conflicting issues or preferences. The support for individual vs. group decisions is different and hence DSSs and GDSSs may require different designs. Because members of a group have different personalities, motivations, and experiential familiarities with the situation at hand, a GDSS must assist in supporting a wide range of judgment and choice perspectives.

It is important to note that the group of people may be centralized at one spot or decentralized in space and/or time. Also, the decision considered by each individual in a decision making group may or may not be the *ultimate* decision. The decision being considered may be sequential over time and may involve many component decisions. Alternatively, or in addition, many members in a decision making group may be formulating and/or analyzing options and preparing a short list of these for review by a person with greater authority or responsibility over a different portion of the decision making effort.

Thus, the number of possible types of GDSS may be relatively extensive. Johansen (1988) has identified no less than 17 approaches for computer support in groups in his discussion of groupware.

1. *Face-to-face meeting facilitation services:* This is little more than office automation support in the preparation of reports, overheads, videos, and the like that will be used in a group meeting. The person making the presentation is called a ''facilitator'' or ''chauffeur.''
2. *Group decision support systems:* By this, Johansen essentially infers the GDSS structure shown in Figure 15 with the exception that there is but a single video monitor under the control of a facilitator or chauffeur.
3. *Computer-based extensions of telephony for use by work groups:* This involves use of either commercial telephone services or private branch exchanges (PBXs). These services exist now, and there are several present examples of conference calling services.
4. *Presentation support software:* This approach is not unlike that of approach 1, except that computer software is used to enable the presentation to be contained within a computer. Often those who will present it prepare the presentation material, and this may be done in an interactive manner to the group receiving the presentation.
5. *Project management software:* This is software that is receptive to presentation team input over time and that has capabilities to organize and structure the tasks associated with the group, often in the form of a Gantt chart. This is very specialized software and would be potentially useful for a team interested primarily in obtaining typical project management results in terms of PERT charts and the like.
6. *Calendar management for groups:* Often individuals in a group need to coordinate times with one another. They indicate times that are available, potentially with weights to indicate schedule adjustment flexibility in the event that it is not possible to determine an acceptable meeting time.
7. *Group authoring software:* This allows members of a group to suggest changes in a document stored in the system without changing the original. A lead person can then make document revisions. It is also possible for the group to view alternative revisions to drafts. The overall objective is to encourage, and improve the quality and efficiency of, group writing. It seems very clear that there needs to be overall structuring and format guidance, which, while possibly group determined, must be agreed upon prior to filling out the structure with report details.
8. *Computer-supported face-to-face meetings:* Here, individual members of the group work directly with a workstation and monitor, rather than having just a single computer system and monitor. A large screen video may, however, be included. This is the sort of DSS envisioned in Figure 14. Although there are a number of such systems in existence, the Colab system at Xerox Palo Alto Research Center (Stefik et al. 1987) was one of the earliest and most sophisticated. A simple sketch of a generic facility for this purpose might appear somewhat as shown in Figure 17. Generally, both public and private information are contained in these systems. The public information is shared, and the private information, or a portion of it, may be converted to public programs. The private screens normally start with a menu screen from which participants can select activities in which they engage, potentially under the direction of a facilitator.
9. *Screen sharing software:* This software enables one member of a group to selectively share screens with other group members. There are clearly advantages and pitfalls in this. The primary advantage to this approach is that information can be shared with those who have a reason to know specific information without having to bother others who do not need it. The disadvantage is just this also, and it may lead to a feeling of ganging up by one subgroup on another subgroup.
10. *Computer conferencing systems:* This is the group version of electronic mail. Basically, what we have is a collection of DSSs with some means of communication among the individuals that comprise the group. This form of communication might be regarded as a product hierarchy in which people communicate.
11. *Text filtering software:* This allows system users to search normal or semistructured text through the specification of search criteria that are used by the filtering software to select relevant portions of text. The original system to accomplish this was Electronic Mail Filter (Malone et al. 1987). A variety of alternative approaches are also being emphasized now.
12. *Computer-supported audio or video conferences:* This is simply the standard telephone or video conferencing, as augmented by each participant having access to a computer and appropriate software.
13. *Conversational structuring:* This involves identification and use of a structure for conversations that is presumably in close relationship to the task, environment, and experiential fa-

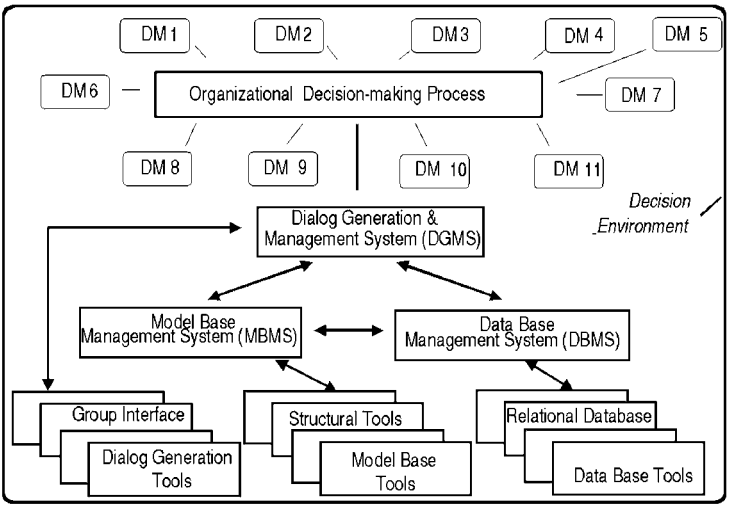

Figure 17 Level II (No Facilitator) and Level III (with Facilitator) GDSS.

miliarity of the group with the issues under consideration (Winograd and Flores 1986). For these group participants, structured conversations should often provide for enhanced efficiency and effectiveness or there may be a perception of unwarranted intrusions that may defeat the possible advantages of conversational structuring.

14. *Group memory management:* This refers to the provision of support between group meetings such that individual members of a group can search a computer memory in personally preferred ways through the use of very flexible indexing structures. The term *hypertext* (Nielson 1989) is generally given to this flexible information storage and retrieval. One potential difficulty with hypertext is the need for a good theory of how to prepare the text and associated index such that it can be indexed and used as we now use a thesaurus. An extension of hypertext to include other than textual material is known as hypermedia.
15. *Computer-supported spontaneous interaction:* The purpose of these systems is to encourage the sort of impromptu and extemporaneous interaction that often occurs at unscheduled meetings between colleagues in informal setting, such as a hallway. The need for this could occur, for example, when it is necessary for two physically separated groups to communicate relative to some detailed design issue that needs to be resolved in order to continue product development.
16. *Comprehensive work team support:* This refers to integrated and comprehensive support, such as perhaps might be achieved through use of the comprehensive DSS design philosophy described above.
17. *Nonhuman participants in team meetings:* This refers to the use of unfacilitated DSS and expert systems that automate some aspects of the process of decision making.

According to Johansen (1988), the order in which these are described above also represents the order of increasing difficulty of implementation and successful use. These scenarios of support for decision making are also characterized in terms of support for face-to-face meetings (1, 2, 4, 8); support for electronic meetings (3, 9, 10, 11, 12); and support between meetings (5, 6, 7, 13, 14, 15, 16). There is much interest in groupware as a focused subject within modern information technology developments (Shapiro et al. 1996; Chaffey 1998; Smith 1999). A number of groupware products are available, Lotus Notes arguably being the best known.

A GDSS may and doubtless will influence the process of group decision making, perhaps strongly. A GDSS has the potential for changing the information-processing characteristics of individuals in the group. This is one reason why organizational structure and authority concerns are important ingredients in GDSS designs. In one study of the use of a GDSS to facilitate group consensus (Watson et al. 1988), it was found that:

1. GDSS use tended to reduce face-to-face interpersonal communication in the decision making group.

2. GDSS use posed an intellectual challenge to the group and made accomplishment of the purpose of their decision making activity more difficult than for groups without the GDSS.
3. The groups using the GDSS became more process oriented and less specific-issue oriented than the groups not using the GDSS.

Support may be accomplished at any or all of the three levels for group decision support identified by DeSanctis and Gallupe (1987) in their definitive study of GDSS. A GDSS provides a mechanism for group interaction. It may impose any of various structured processes on individuals in the group, such as a particular voting scheme. A GDSS may impose any of several management control processes on the individuals on the group, such as that of imposing or removing the effects of a dominant personality. The design of the GDSS and the way in which it is used are the primary determinants of these.

DeSanctis and Gallupe have developed a taxonomy of GDSS. A Level I GDSS would simply be a medium for enhanced information interchange that might lead ultimately to a decision. Electronic mail, large video screen displays that can be viewed by a group, or a decision room that contains these features could represent a Level I GDSS. A Level I GDSS provides only a mechanism for group interaction. It might contain such facilities as a group scratchpad, support for meeting agenda development, and idea generation and voting software.

A Level II GDSS would provide various decision structuring and other analytic tools that could act to reduce information imperfection. A decision room that contained software that could be used for problem solution would represent a Level II GDSS. Thus, spreadsheets would primarily represent a Level II DSS. A Level II GDSS would also have to have some means of enabling group communication. Figure 17 represents a Level II GDSS. It is simply a communications medium that has been augmented with some tools for problem structuring and solution with no prescribed management control of the use of these tools.

A Level III GDSS also includes the notion of management control of the decision process. Thus, a notion of facilitation of the process is present, either through the direct intervention of a human in the process or through some rule-based specifications of the management control process that is inherent in Level III GDSS. Clearly, there is no sharp transition line between one level and the next, and it may not always be easy to identify at what level a GDSS is operating. The DSS generator, such as discussed in our preceding section, would generally appear to produce a form of Level III DSS. In fact, most of the DSSs that we have been discussing in this book are either Level II or Level III DSSs or GDSSs. The GDSS of Figure 17, for example, becomes a Level III GDSS if is supported by a facilitator.

DeSanctis and Gallupe (1987) identify four recommended approaches:

1. Decision room for small group face-to-face meetings
2. Legislative sessions for large group face-to-face meetings
3. Local area decision networks for small dispersed groups
4. Computer-mediated conferencing for large groups that are dispersed

DeSanctis and Gallupe discuss the design of facilities to enable this, as well as techniques whereby the quality of efforts such as generation of ideas and actions, choosing from among alternative courses of action, and negotiating conflicts may be enhanced. On the basis of this, they recommend six areas as very promising for additional study: GDSS design methodologies; patterns of information exchange; mediation of the effects of participation; effects of (the presence or absence of) physical proximity, interpersonal attraction, and group cohesion; effects on power and influence; and performance/satisfaction trade-offs. Each of these supports the purpose of computerized aids to planning, problem solving, and decision making: removing a number of common communication barriers; providing techniques for structuring decisions; and systematically directing group discussion and associated problem-solving and decision making in terms of the patterns, timing, and content of the information that influences these actions.

We have mentioned the need for a GDSS model base management system (MBMS). There are a great variety of MBMS tools. Some of the least understood are group tools that aid in the issue formulation effort. Because these may represent an integral part of a GDSS effort, it is of interest to describe GDSS issue formation here as one component of a MBMS.

Other relevant efforts and interest areas involving GDSS include the group processes in computer mediated communications study; the computer support for collaboration and problem solving in meetings study of Stefik et al. (1987); the organizational planning study of Applegate et al. (1987), and the knowledge management and intelligent information sharing systems study of Malone et al. (1987). Particularly interesting current issues surround the extent to which cognitive science and engineering studies that involve potential human information processing flaws can be effectively dealt with, in the sense of design of debiasing aids, in GDSS design. In a very major way, the purpose of a GDSS is to support enhanced organizational communications. This is a very important contem-

porary subject, as evidenced by recent efforts concerning virtual organizations (DeSanctis and Monge 1999; Jablin and Putnam 2000).

5.3. Distributed Group Decision Support Systems

A single-user DSS must provide single-user-single-model and single-user-multiple-model support, whereas a GDSS model base management system (MBMS) must support multiple-user-single-model and multiple-user-multiple-model support. A centralized GDSS induces three basic components:

1. A group model base management subsystem (GMBMS)
2. A group database management subsystem (GDBMS)
3. A group dialog generation and management subsystem (GDGMS)

These are precisely the components needed in a single-user DSS, except for the incorporation of group concerns. In the GDSS, all data are stored in the group database and models are stored in the group model base. The GMBMS controls all access to the individually owned and group-owned models in the model base.

A distributed GDSS allows each user to have an individual DSS and a GDSS. Each individual DSS consists of models and data for a particular user. The GDSS maintains the GMBMS and GDBMS, as well as controlling access to the GMBMS and GDBMS and coordination of the MBMSs of various individual DSSs (Liang 1988). During actual GDSS system use, the difference between the centralized and distributed GDSSs should be transparent to the users.

We generally use models to help define, understand, organize, study, and solve problems. These range from simple mental models to complex mathematical simulation models. An important mission of a GDSS is to assist in the use of formal and informal models by providing appropriate model management. An appropriate model base management system for a GDSS can provide the following four advantages (Hwang 1985):

1. *Reduction of redundancy,* since models can be shared
2. *Increased consistency,* since more than one decision maker will share the same model
3. *Increased flexibility,* since models can be upgraded and made available to all members of the group
4. *Improved control over the decision process,* by controlling the quality of the models adopted

A model base management system provides for at least the following five basic functions (Blanning and King 1993): construction of new models, storage of existing and new models, access to and retrieval of existing models, execution of existing models, and maintenance of existing models. MBMSs should also provide for model integration and selection. Model integration by using the existing model base as building blocks in the construction of new or integrated models is very useful when ad hoc or prototype models are desired. Model integration is needed in the production of operational MBMSs.

In this section, we have provided a very broad overview of group decision support systems that potentially support group and organizational decision making functions. Rather than concentrate on one or two specific systems, we have painted a picture of the many requirements that must be satisfied in order to produce an acceptable architecture and design for these systems. This provides much fertile ground for research in many GDSS-relevant cognitive systems engineering areas (Rasmussen et al. 1995; Andriole and Adelman 1995).

6. KNOWLEDGE MANAGEMENT FOR DECISION SUPPORT

There can be no doubt that contemporary developments in information technology have changed engineering and business practices in many ways. The information revolution has, for example, created entirely new ways of marketing and pricing such that we now see very changed relationships among producers, distributors, and customers. It has also led to changes in the ways in which organizations are managed and structured, and deal with their products and services. In particular, it creates a number of opportunities and challenges that affect the way in which data is converted into information and then into knowledge. It poses many opportunities for management of the environment for these transfers, such as enhancing the productivity of individuals and organizations. Decision support is much needed in these endeavors, and so is knowledge management. It is fitting that we conclude our discussions of decision support systems with a discussion of knowledge management and the emergence in the 21st century of integrated systems to enhance knowledge management and decision support.

Major growth in the power of computing and communicating and associated networking is fundamental to the emergence of these integrated systems and has changed relationships among people, organizations, and technology. These capabilities allow us to study much more complex issues than

was formerly possible. They provide a foundation for dramatic increases in learning and both individual and organizational effectiveness. This is due in large part to the networking capability that enables enhanced coordination and communications among humans in organizations. It is also due to the vastly increased potential availability of knowledge to support individuals and organizations in their efforts, including decision support efforts. However, information technologies need to be appropriately integrated within organizational frameworks if they are to be broadly useful. This poses a transdisciplinary challenge (Somerville and Rapport 2000) of unprecedented magnitude if we are to move from high-performance information technologies and high-performance decision support systems to high-performance organizations.

In years past, broadly available capabilities never seemed to match the visions proffered, especially in terms of the time frame of their availability. Consequently, despite these compelling predictions, traditional methods of information access and utilization continued their dominance. As a result of this, comments something like ''computers are appearing everywhere except in productivity statistics'' have often been made (Brynjolfsson and Yang 1996). In just the past few years, the pace has quickened quite substantially, and the need for integration of information technology issues with organizational issues has led to the creation of related fields of study that have as their objectives:

- Capturing human information and knowledge needs in the form of system requirements and specifications
- Developing and deploying systems that satisfy these requirements
- Supporting the role of cross-functional teams in work
- Overcoming behavioral and social impediments to the introduction of information technology systems in organizations
- Enhancing human communication and coordination for effective and efficient workflow through knowledge management

Because of the importance of information and knowledge to an organization, two related areas of study have arisen. The first is concerned with technologies associated with the effective and efficient acquisition, transmission, and use of information, or *information technology*. When associated with organizational use, this is sometimes called *organizational intelligence* or *organizational informatics*. The second area, known as *knowledge management*, refers to an organization's capacity to gather information, generate knowledge, and act effectively and in an innovative manner on the basis of that knowledge. This provides the capacity for success in the rapidly changing or highly competitive environments of knowledge organizations. Developing and leveraging organizational knowledge is a key competency and, as noted, it requires information technology as well as many other supporting capabilities. Information technology is necessary for enabling this, but it is not sufficient in itself. Organizational productivity is not necessarily enhanced unless attention is paid to the human side of developing and managing technological innovation (Katz 1997) to ensure that systems are designed for human interaction.

The human side of knowledge management is very important. Knowledge capital is sometimes used to describe the intellectual wealth of employees and is a real, demonstrable asset. Sage (1998) has used the term *systems ecology* to suggest managing organizational change to create a knowledge organization and enhance and support the resulting intellectual property for the production of sustainable products and services. Managing information and knowledge effectively to facilitate a smooth transition into the Information Age calls for this systems ecology, a body of methods for systems engineering and management (Sage 1995; Sage and Rouse, 1999a,b) that is based on analogous models of natural ecologies. Such a systems ecology would enable the modeling, simulation, and management of truly large systems of information and knowledge, technology, humans, organizations, and the environments that surround them.

The information revolution is driven by technology and market considerations and by market demand and pull for tools to support transaction processing, information warehousing, and knowledge formation. Market pull has been shown to exert a much stronger effect on the success of an emerging technology than technology push. Hardly any conclusion can be drawn other than that society shapes technology (Pool 1997) or, perhaps more accurately stated, that technology and the modern world shape each other in that only those technologies that are appropriate for society will ultimately survive.

The potential result of this mutual shaping of information technology and society is knowledge capital, and this creates needs for knowledge management. Current industrial and management efforts are strongly dependent on access to information. The world economy is in a process of globalization, and it is possible to detect several important changes. The contemporary and evolving world is much more service oriented, especially in the more developed nations. The service economy is much more information and knowledge dependent and much more competitive. Further, the necessary mix of job skills for high-level employment is changing. The geographic distance between manufacturers and

consumers and between buyers and sellers is often of little concern today. Consequently, organizations from diverse locations compete in efforts to provide products and services. Consumers potentially benefit as economies become more transnational.

Information technology-based systems may be used to support taking effective decisions. Ideally, this is accomplished through both critical attention to the information needs of humans in problem solving and decision making task and provision of technological aids, including computer-based systems of hardware and software and associated processes, to assist in these tasks. There is little question but that successful information systems strategies seek to meaningfully evolve the overall architecture of systems, the systems' interfaces with humans and organizations, and their relations with external environments. In short, they seek to enhance systems integration effectiveness (Sage and Lynch 1998).

Although information technology and information systems do indeed potentially support improvement of the designs of existing organizations and systems, they also enable fundamentally new ones, such as virtual corporations (DeSanctis and Monge 1999), and they also enable major expansions of organizational intelligence and knowledge. They do this not only by allowing for interactivity in working with clients to satisfy present needs, but also through proactivity in planning and plan execution. An ideal organizational knowledge strategy accounts for future technological, organizational, and human concerns to support the graceful evolution of products and services that aid clients. Today, we realize that human and organizational considerations are vital to success in using information technology to better support decisions. A major challenge is to ensure that people gain maximal benefit from these capabilities. This is why information technology must be strongly associated with information ecology, knowledge management, and other efforts that we discuss here and that will ultimately lead to an effective systems ecology (Sage 1998).

There are three keys to organizations prospering in this type of environment: speed, flexibility, and discretion (Rouse 1999). Speed means rapid movement in understanding a situation, such as a new product development or market opportunity; formulating a plan for pursuing this opportunity, such as an intended joint venture for the new product; and deploying this plan so as to proceed through to product availability in stores. Flexibility is crucial for reconfiguring and redesigning organizations, and consequently reallocating resources. Functional walls must be quite portable, and generally the few of them the better.

Discretion transforms flexibility into speed. Distributed organizations must be free to act. While they may have to play by the contemporary and evolutionary rules of the game, they need to be able to adapt rapidly when things are not working well. Resources can thus be deployed effectively and speedily and results monitored quickly. Resource investments that are not paying off in the anticipated time frame can be quickly redeployed elsewhere, thereby ensuring adaptive and emergent evolution of the organization.

A major determinant of these organizational abilities is the extent to which an organization possesses intellectual capital, or knowledge capital, such that it can create and use innovative ideas to produce productive results. The concept of intellectual capital has been defined in various ways (Brooking 1996; Edvisson and Malone 1997; Stewart 1997; Klein 1998; Roos et al. 1998). We would add communications to the formulation of Ulrick (1998), representing intellectual capital to yield:

$$\text{Intellectual capital} = \text{Competence} \times \text{Commitment} \times \text{Communications}$$

Other important terms, such as *collaboration* and *courage,* could be added to this generic equation.

Loosely structured organizations and the speed, flexibility, and discretion they engender in managing intellectual capital fundamentally affect knowledge management (Myers 1997; Ruggles 1997; Prusak 1997; Albert and Bradley 1997; Liebowitz and Wilcox, 1997). Knowledge workers are no longer captive and hence know-how is not ''owned'' by the organization. What matters most is the ability to make sense of market and technology trends, quickly decide how to take advantage of these trends, and act faster than other players. Sustaining competitive advantage requires redefining market-driven value propositions and quickly leading in providing value in appropriate new ways. Accomplishing this in an increasingly information-rich environment is a major challenge, both for organizations experiencing contemporary business environments (Allee 1997; Stacey 1996) and for those who devise and provide decision support systems for supporting these new ways. The major interactions involving knowledge work and intellectual capital and the communications-driven information and knowledge revolution suggest many and profound, complex, adaptive system-like changes in the economy of the 21st century (Hagel and Armstrong 1997; Shapiro and Varian 1999; Kelly 1998; Hagel and Singer 1999). In particular, this has led to the notion of virtual enterprises and virtual communities and markets where customers make the rules that enhance net gain and net worth.

All of this creates major challenges for the evolution of knowledge organizations and appropriate knowledge management. One of the major challenges is that of dealing in an appropriate manner with the interaction among humans, organizations, and technologies and the environment surrounding

these. Davenport and Prusak (1998) note that when organizations interact with environments, they absorb information and turn it into knowledge. Then they make decisions and take actions. They suggest five modes of knowledge generation:

1. Acquisition of knowledge that is new to the organization and perhaps represents newly created knowledge. Knowledge-centric organizations need to have appropriate knowledge available when it is needed. They may buy this knowledge, potentially through acquisition of another company, or generate it themselves. Knowledge can be leased or rented from a knowledge source, such as by hiring a consultant. Generally, knowledge leases or rentals are associated with knowledge transfer.
2. Dedicated knowledge resource groups may be established. Because time is required for the financial returns on research to be realized, the focus of many organizations on short-term profit may create pressures to reduce costs by reducing such expenditures. Matheson and Matheson (1998) describe a number of approaches that knowledge organizations use to create value through strategic research and development.
3. Knowledge fusion is an alternative approach to knowledge generation that brings together people with different perspectives to resolve an issue and determine a joint response. Nonaka and Takeuchi (1995) describe efforts of this sort. The result of knowledge fusion efforts may be creative chaos and a rethinking of old presumptions and methods of working. Significant time and effort are often required to enable group members to acquire sufficient shared knowledge, work effectively together, and avoid confrontational behavior.
4. Adaptation through providing internal resources and capabilities that can be utilized in new ways and being open to change in the established ways of doing business. Knowledge workers who can acquire new knowledge and skills easily are the most suitable to this approach. Knowledge workers with broad knowledge are often the most appropriate for adaptation assignments.
5. Knowledge networks may act as critical conduits for innovative reasoning. Informal networks can generate knowledge provided by a diversity of participants. This requires appropriate allocation of time and space for knowledge acquisition and creation.

In each of these efforts, as well as in much more general situations, it is critical to regard technology as a potential enabler of human effort, not as a substitute for it. There are, of course, major feedbacks here because the enabled human efforts create incentives and capabilities that lead to further enhanced technology evolution.

Knowledge management (Nonaka and Takeuchi 1995; Cordata and Woods 1999; Bukowitz and Williams, 1999) refers to management of the environment for knowledge creation, transfer, and sharing throughout the organization. It is vital in fulfilling contemporary needs in decision support. Appropriate knowledge management considers knowledge as the major organizational resource and growth through enhanced knowledge as a major organizational objective. While knowledge management is dependent to some extent upon the presence of information technology as an enabler, information technology alone cannot deliver knowledge management. This point is made by McDermott (1999), who also suggests that the major ingredient in knowledge management, leveraging knowledge, is dramatically more dependent upon the communities of people who own and use it than upon the knowledge itself.

Knowledge management is one of the major organizational efforts brought about by the realization that knowledge-enabled organizations are best posed to continue in a high state of competitive advantage. Such terms as *new organizational wealth* (Sveiby 1997), *intellectual capital* (Brooking 1996; Edvinsson and Malone 1997; Klein 1998; Roos et al. 1998), the *infinite resource* (Halal 1998), and *knowledge assets* (Boisot 1998) are used to describe the knowledge networking (Skyrme, 1999) and working knowledge (Davenport and Prusak 1998) in knowledge-enabled organizations (Liebowitz and Beckman 1998; Tobin 1998). Major objectives of knowledge management include supporting organizations in turning information into knowledge (Devin 1999) and, subsequent to this through a strategy formation and implementation process (Zack 1999), turning knowledge into action (Pfeffer and Sutton 2000). This latter accomplishment is vital because a knowledge advantage is brought to fruition only through an action advantage. Appropriate information technology and knowledge management tools (Ruggles 1997) are necessary but not at all sufficient to enable knowledge creating organizations or knowing organizations. The result of appropriate creation of knowledge in organizations (Prusak 1997) leads to a very important result, a knowing organization in which organizations use information to construct meaning, to create knowledge, and to use this knowledge in taking decisions. The term *fourth generation R&D* has been given to the efforts necessary for management of knowledge, technology, and innovation. A framework to enable this is described in Miller and Morris (1999).

The major increase in interest in knowledge management in recent years has been brought about by the reality that contemporary engineering and business success are progressively more linked to abilities associated with the management of data, information, and knowledge. Knowledge management is concerned with knowledge creation, knowledge acquisition, knowledge packaging, knowledge transfer, and knowledge use and reuse by humans and organizations. Knowledge management and decision support each support a common purpose: making decisions and actions taken on the basis of information and knowledge more effective. Thus, it is reasonable to suggest that an objective of knowledge management is to make appropriate knowledge available in a timely and cost-effective manner to decision makers. It seems very safe to predict that computer-based decision support systems will increasingly employ or be associated with various knowledge management techniques to enhance the representation and processing of information such that it can be best associated with contingency task structures to become the beneficial knowledge that is absolutely needed for effective decision support.

The development of intellectual capital such as to optimize organizational value is of particular importance in effective decision support. This is accomplished by using information to construct meaning and create knowledge and thereby enable appropriate decision and action. Such an organization has been called a ''knowing organization'' (Choo 1998). The major challenge in all of this is very often that of making sense of a wealth of opportunities concerning alternative schools and strategies. These include dramatic increases in available data, as well as an ever-growing set of potentially useful methods and tools—and the major need to convert data into information and thence into knowledge—and to manage knowledge successfully in the networked economy. Sage and Rouse (1999a,b) identifies 10 important challenges and paradigms: systems modeling, emergent and complex phenomena, uncertainties and control, access to and utilization of information and knowledge, information and knowledge requirements, information and knowledge support systems, inductive reasoning, learning organizations, planning and design, and measurement and evaluation. Ongoing trends in information technology and knowledge management pose substantial challenges for information systems frontiers. Addressing the 10 key challenges elaborated here requires a new, broader perspective on the nature of information access and utilization, as well as knowledge management. Satisfactorily addressing these 10 key challenges, will require that decision support systems engineering efforts move beyond structure-bound views of the world and the natural tendency to nail down requirements and constraints before proceeding. The current dynamics of information technology and knowledge management make such ''givens'' obsolete almost as quickly as they are envisioned. These appears to be the major decision support systems engineering challenges today, and in a very real sense they are challenges for all of engineering and engineering management. In large part, they provide a major motivation for this Handbook.

REFERENCES

Albert, S., and Bradley, K. (1997), *Managing Knowledge: Experts, Agencies, and Organizations,* Cambridge University Press, Cambridge.

Alberts, D. S., and Papp, D. S., Eds. (1997), *The Information Age: An Anthology of Its Impacts and Consequences,* National Defense University Press, Washington, DC.

Allee, V. (1997), *The Knowledge Evolution: Expanding Organizational Intelligence,* Butterworth-Heinemann, Boston.

Andriole, S., and Adelman, L. (1995), *Cognitive Systems Engineering for User–Computer Interface Design, Prototyping, and Evaluation,* Lawrence Erlbaum Associates, Hillsdale, NJ.

Anthony, R. N. (1965), *Planning and Control Systems: A Framework for Analysis*. Harvard University Press, Cambridge, MA.

Anthony, R. N., Dearden, N. J., and Govindarajan, V. (1992), *Management Control Systems,* Richard D. Irwin, Homewood, IL.

Applegate, L. M, Konsynski, B. R., and Nunamaker, J. F. (1986), ''Model Management Systems: Design for Decision Support,'' *Decision Support Systems,* Vol. 2, No. 1, pp. 81–91.

Applegate, L. M., Chen, T. T., Konsynski, B. R., and Nunamaker, J. F. (1987), ''Knowledge Management in Organizational Planning,'' *Journal of Management Information Systems,* Vol. 3, No. 4, pp. 20–38.

Arden, B. W., Ed. (1980), ''What Can Be Automated?,'' Chapter 10 in *The Computer Science and Engineering Research Study,* MIT Press, Cambridge, MA.

Atzeni, P., and Chen, P. P. (1983), ''Completeness of Query Languages for the Entity-Relationship Model,'' in *Entity-Relationship Approach to Information Modeling and Analysis,* Ed. P. P. Chen, North Holland, Amsterdam.

Atzeni, P., Ceri, S., Paraboschi, S., and Torione, R. (2000), *Database Systems: Concepts, Languages and Architectures,* McGraw-Hill, New York.

Banks, J., Ed. (1998), *Handbook of Simulation,* John Wiley & Sons, New York.

Barbosa, L. C., and Herko, R. G. (1980), "Integration of Algorithmic Aids into Decision Support Systems," *MIS Quarterly,* Vol. 4, No. 3, pp. 1–12.

Bennet, J. L., Ed. (1983), *Building Decision Support Systems*, Addison-Wesley, Reading, MA.

Blanning, R. W., and King, D. R. (1993), *Current Research in Decision Support Technology,* IEEE Computer Society Press, Los Altos, CA.

Boisot, M. H. (1998), *Knowledge Assets: Securing Competitive Advantage in the Information Economy,* Oxford University Press, New York.

Brooking, A. (1996), *Intellectual Capital: Core Asset for the Third Millennium Enterprise,* Thompson Business Press, London.

Brynjolfsson, E., and Yang, S. (1996), "Information Technology and Productivity: A Review of the Literature," in *Advances in Computers,* Vol. 43, pp. 179–214.

Bukowitz, W. R., and Williams, R. L. (1999), *The Knowledge Management Fieldbook,* Financial Times Prentice Hall, London.

Card, S.K., Moran, T. P., and Newell, A. (1993), *The Psychology of Human Computer Interaction*. Lawrence Erlbaum, Hillsdale, NJ.

Chaffey, D. (1998), *Groupware, Workflow and Intranets: Reengineering the Enterprise with Collaborative Software,* Digital Press, Boston.

Chen, P. P. S. (1976), "The Entity-Relationship Model: Towards a Unified View of Data," *ACM Transactions on Database Systems,* Vol. 1, pp. 9–36.

Choo, C. W. (1998), *The Knowing Organization: How Organizations Use Information to Construct Meaning, Create Knowledge, and Make Decisions,* Oxford University Press, New York.

Coad, P., and Yourdon, E. (1990), *Object-Oriented Analysis,* Prentice Hall, Englewood Cliffs, NJ.

Cordata, J. W., and Woods, J. A., Eds. (1999), *The Knowledge Management Yearbook 1999–2000,* Butterworth-Heinemann, Woburn, MA.

Darwen, H., and Date, C. J. (1998), *Foundation for Object/Relational Databases: The Third Manifesto,* Addison-Wesley, Reading MA.

Date, C. J. (1983), *Database: A Primer,* Addison-Wesley, Reading, MA.

Date, C. J. (1999), *An Introduction to Database Systems,* 7th Ed., Addison-Wesley, Reading, MA.

Davenport, T. H., and Prusak, L. (1994), *Working Knowledge: How Organizations Manage What They Know,* Harvard Business School Press, Boston.

Davis, G. B. (1982), "Strategies for Information Requirements Determination," *IBM Systems Journal*, Vol. 21, No. 1, pp. 4–30.

Debenham, J. (1998), *Knowledge Engineering: Unifying Knowledge Base and Database Design,* Springer-Verlag, Berlin.

DeSanctis, G., and Gallupe, R. B. (1987), "A Foundation for the Study of Group Decision Support Systems," *Management Science,* Vol. 33, pp. 547–588.

DeSanctis, G., and Monge, P., Eds. (1999), Special Issue: Communications Processes for Virtual Organizations, *Organizational Science,* Vol. 10, No. 6, November–December.

Devin, K. (1999), *Infosense: Turning Information into Knowledge,* W. H. Freeman & Co., New York.

Dolk, D. R. (1986), "Data as Models: An Approach to Implementing Model Management," *Decision Support Systems,* Vol. 2, No. 1, pp, 73–80.

Dutta, P. K. (1999), *Strategies and Games: Theory and Practice,* MIT Press, Cambridge, MA.

Edvinsson, L., and Malone, M. S. (1997), *Intellectual Capital: Realizing your Company's True Value by Finding Its Hidden Brainpower,* HarperCollins, New York.

Fang, L., Hipel, K. W., and Kilgour, D. M. (1993), *Interactive Decision Making: The Graph Model for Conflict Resolution,* John Wiley & Sons, New York.

Firesmith, D. G. (1993), *Object Oriented Requirements and Logical Design,* John Wiley & Sons, New York.

Fraser, N. M., and Hipel, K. W. (1984), *Conflict Analysis: Models and Resolution.* North Holland, New York.

Gass, S. I., and Harris, C. M., Eds. (2000), *Encyclopedia of Operations Research and Management Science,* Kluwer Academic Publishers, Boston.

Gray, P., and Olfman, L. (1989), "The User Interface in Group Decision Support Systems," *Decision Support Systems,* Vol. 5, No. 2, pp. 119–137.

Grudin, J. (1989), "The Case Against User Interface Consistency," *Communications of the ACM,* Vol. 32, pp. 1164–1173.

Hagel, J., and Armstrong, A. G. (1997), *Net Gain: Expanding Markets Through Virtual Communities,* Harvard Business School Press, Boston.

Hagel, J., and Singer, M. (1999), *Net Worth,* Harvard Business School Press, Boston.

Halal, W. E., Ed. (1998), *The Infinite Resource: Creating and Leading the Knowledge Enterprise* Jossey-Bass, San Francisco.

Hall, A. D. (1969), "Three Dimensional Morphology of Systems Engineering," *IEEE Transactions on Systems Science and Cybernetics,* Vol. 5, pp. 156–160.

Hammond, J. S., Keeney, R. L., and Raiffa, H. (1999), *Smart Choices: A Practical Guide to Making Better Decisions,* Harvard Business School Press, Boston.

Hammond, K. R., McClelland, G. H., and Mumpower, J. (1980), *Human Judgment and Decision Making: Theories, Methods, and Procedures,* Praeger, New York.

Harrison, H. R., and Hix, D. (1989), "Human Computer Interface Development." *ACM Computing Surveys,* Vol. 21, No. 1, pp. 5–92.

Harrison, M., and Thimbleby, H., Eds. (1990), *Formal Methods in Human–Computer Interaction,* Cambridge University Press, Cambridge.

Herek, G. M., Janis, I. L., and Hurth, P. (1987), "Decision Making During International Crises: Is Quality of Process Related to Outcome?" *Journal of Conflict Resolution,* Vol. 31, No. 2, pp. 203–226.

Hillier, F. S., and Lieberman, G. J. (1990), *Operations Research,* 5th Ed., Holden Day, San Francisco.

Hillier, F. S., and Lieberman, G. J. (1994), *Introduction to Mathematical Programming,* 2nd Ed., McGraw-Hill, New York.

Huber, G. P. (1982), "Group Decision Support Systems as Aids in the Use of Structured Group Management Techniques," in *Proceedings of the 2nd International Conference on Decision Support Systems* (San Francisco), pp. 96–108.

Hwang, S. (1985), "Automatic Model Building Systems: A Survey," in *DSS-85 Transactions,* J. J. Elam, Ed., pp. 22–32.

Jablin, F., and Putnam, L., Eds. (2000), *New Handbook of Organizational Communication,* Sage, Thousand Oaks, CA.

Jacobson, I., Ericsson, M., and Jacobson, A. (1995), *The Object Advantage: Business Process Reengineering with Object Technology*. Addison-Wesley, Reading MA.

Janis, I. J., and Mann, L. (1977), *Decision Making: A Psychological Analysis of Conflict, Choice, and Commitment,* Free Press, New York.

Janssen, T. J., and Sage, A. P. (2000), A Group Decision Support System for Science Policy Conflict Resolution," in Special Issue on Public Policy Engineering Management, E. G. Beroggi, Ed., *International Journal of Technology Management,* Vol. 19, No. 3.

Johansen, R. (1988), *Groupware: Computer Support for Business Teams,* Free Press, New York.

Katz, R., Ed. (1997), *The Human Side of Managing Technological Innovation,* Oxford University Press, New York.

Keen, P. G. W., and Scott Morton, M. S. (1978), *Decision Support Systems: An Organizational Perspective,* Addison-Wesley, Reading, MA.

Keeney, R. L., and Raiffa, H. (1976), *Decisions with Multiple Objectives,* John Wiley & Sons, New York.

Keller, L. R., and Ho, J. L. (1990), "Decision Problem Structuring, in *Concise Encyclopedia of Information Processing in Systems and Organizations,* A. P. Sage, Ed., Pergamon Press, Oxford, pp. 103–110.

Kelly, K. (1998), *New Rules for the New Economy,* Viking Press, New York.

Kent, W. (1979), "Limitation of Record Based Information Models," *ACM Transactions on Database Systems,* Vol. 4, pp. 107–131.

Kerschberg, L., Ed. (1987, 1989), *Proceedings from the First International Conference on Expert Database Systems* (Charleston, SC), Vol. 1 (1987), Vol. 2 (1989), Benjamin-Cummings, Menlo Park, CA.

King, J. L., and Star, S. L. (1992), "Organizational Decision Support Processes as an Open Systems Problem," in *Information Systems and Decision Processes,* T. Stohr and B. Konsynski, Eds., IEEE Press, Los Altos, CA, pp. 150–154.

Klein, D. A., Ed. (1998), *The Strategic Management of Intellectual Capital,* Butterworth-Heineman, Boston.

Klein, G. A. (1990), ''Information Requirements for Recognitional Decision Making,'' in *Concise Encyclopedia of Information Processing in Systems and Organizations,* A. P. Sage, Ed., Pergamon Press, Oxford, pp. 414–418.

Klein, G. A. (1998), *Sources of Power: How People Make Decisions,* MIT Press, Cambridge.

Kraemer, K. L., and King, J. L. (1988), ''Computer Based Systems for Cooperative Work and Group Decision Making,'' *ACM Computing Surveys,* Vol. 20, No. 2, pp. 115–146.

Kroenke, D. M. (1998), *Database Processing: Fundamentals, Design, and Implementation,* Prentice Hall, Englewood Cliffs, NJ.

Lee, E. (1990), ''User-Interface Development Tools,'' *IEEE Software,* Vol. 7, No. 3, pp. 31–36.

Liang, B. T. (1988), ''Model Management for Group Decision Support,'' *MIS Quarterly,* Vol. 12, pp. 667–680.

Liang, T. P. (1985), ''Integrating Model Management with Data Management in Decision Support Systems,'' *Decision Support Systems.* Vol. 1, No. 3, pp. 221–232.

Liebowitz, J., and Beckman, T. (1998), *Knowledge Organizations: What Every Manager Should Know,* CRC Press, Boca Raton, FL.

Liebowitz, J., and Wilcox, L. C., Eds. (1997), *Knowledge Management and Its Integrative Elements,* CRC Press, Boca Raton, FL.

Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A., and Cohen, M. D. (1987), ''Intelligent Information Sharing Systems,'' *Communications of the ACM,* Vol. 30, pp. 390–402.

Marakas, G. M., *Decision Support Systems in the 21st Century,* Prentice Hall, Englewood Cliffs, NJ.

March, J. G. (1983), ''Bounded Rationality, Ambiguity, and the Engineering of Choice,'' *Bell Journal of Economics,* Vol. 9, pp. 587–608.

March, J., and Wessinger-Baylon, T., Eds. (1986), *Ambiguity and Command: Organizational Perspectives on Military Decisionmaking,* Pitman, Boston.

Matheson, D., and Matheson, J. (1998), *The Smart Organization: Creating Value Through Strategic R&D,* Harvard Business School Press, Boston.

McDermott, R. (1999), ''Why Information Technology Inspired but Cannot Deliver Knowledge Management,'' *California Management Review,* Vol. 41, No. 1, pp. 103–117.

McGrath, J. E. (1984), *Groups: Interaction and Performance,* Prentice Hall, Englewood Cliffs, NJ.

Miller, W. L., and Morris, L. (1999), *Fourth Generation R&D: Managing Knowledge, Technology and Innovation,* John Wiley & Sons.

Mintzberg, H. (1973), *The Nature of Managerial Work,* Harper & Row, New York.

Murphy, F. H., and Stohr, E. A. (1986), ''An Intelligent System for Formulating Linear Programs,'' *Decision Support Systems,* Vol. 2, No. 1, pp. 39–47.

Myers, P. S. (1997), *Knowledge Management and Organizational Design,* Butterworth-Heinemann, Boston.

Mylopoulos, J., and Brodie, M. L. Eds. (1998), *Artificial Intelligence and Databases,* Morgan Kaufmann, San Mateo, CA.

Nielson, J. (1989), *Hypertext and Hypermedia,* Academic Press, San Diego.

Nonaka, I., and Takeuchi, H. (1995), *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation,* Oxford University Press, New York.

Parsaei, H. R., Kolli, S., and Handley, T. R., Eds. (1997), *Manufacturing Decision Support Systems,* Chapman & Hall, New York.

Parsaye, K., Chignell, M., Khoshafian, S., and Wong, H. (1989), *Intelligent Databases: Object-Oriented, Deductive, Hypermedia Technologies,* John Wiley & Sons, New York.

Pfeffer, J., and Sutton, R. I. (2000), *The Knowing–Doing Gap: How Smart Companies Turn Knowledge into Action,* Harvard Business School Press, Boston.

Poe, V., Klauer, P., and Brobst, S. (1998), *Building a Data Warehouse for Decision Support,* 2nd Ed., Prentice Hall, Englewood Cliffs, NJ.

Pool, R. (1997), *Beyond Engineering: How Society Shapes Technology,* Oxford University Press, New York.

Prusak, L., Ed. (1997), *Knowledge in Organizations,* Butterworth-Heinemann, Woburn, MA.

Purba, S., Ed. (1999), *Data Management Handbook,* 3rd Ed., CRC Press, Boca Raton, FL.

Raiffa, H. (1968), *Decision Analysis,* Addison-Wesley, Reading, MA.

Rasmussen, J., Pejtersen, A. M., and Goodstein, L. P. (1995), *Cognitive Systems Engineering,* John Wiley & Sons, New York.

Ravden, S., and Johnson, G. (1989), *Evaluating Usability of Human–Computer Interfaces: A Practical Method,* John Wiley & Sons, Chichester.

Rob, P., and Coronel, C. (1997), *Database Systems,* 3rd Ed., International Thomson Publishing, London.

Roberts, T. L., and Moran, T. P. (1982), "A Methodology for Evaluating Text Editors," in *Proceedings of the IEEE Conference on Human Factors in Software Development,* B. Curtis, Ed., Gaithersburg, MD.

Roos, J., Roos, G., Edvinsson, L., and Dragonetti, N. C. (1998), *Intellectual Capital: Navigating in the New Business Landscape,* New York University Press, New York.

Rouse, W. B. (1991), "Conceptual Design of a Computational Environment for Analyzing Tradeoffs Between Training and Aiding," *Information and Decision Technologies,* Vol. 17, pp. 143–152.

Rouse, W. B. (1999), "Connectivity, Creativity, and Chaos: Challenges of Loosely-Structured Organizations," *Information, Knowledge, and Systems Management,* Vol. 1, No. 2, pp. 117–131.

Ruggles, R. L., III, Ed. (1997), *Knowledge Management Tools,* Butterworth-Heinemann, Woburn, MA.

Sage, A. P. (1987), "Information Systems Engineering for Distributed Decision Making," *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 17, No. 6, pp. 920–936.

Sage, A. P. Ed. (1990), *Concise Encyclopedia of Information Processing in Systems and Organizations,* Pergamon Press, Oxford.

Sage, A. P. (1991), *Decision Support Systems Engineering,* John Wiley & Sons, New York.

Sage, A. P. (1992), *Systems Engineering,* John Wiley & Sons, New York.

Sage, A. P. (1995), *Systems Management for Information Technology and Software Engineering,* John Wiley & Sons, New York.

Sage, A. P. (1998), "Towards Systems Ecology," *IEEE Computer,* Vol. 31, No. 2, pp. 107–110.

Sage, A. P., and Armstrong, J. E. (2000), *Introduction to Systems Engineering,* John Wiley & Sons, New York.

Sage, A. P., and Lynch, C. L. (1998), "Systems Integration and Architecting: An Overview of Principles, Practices, and Perspectives," *Systems Engineering,* Vol. 1, No. 3, pp. 176–227.

Sage, A. P., and Palmer, J. D. (1990), *Software Systems Engineering,* Wiley Interscience, New York.

Sage, A. P., and Rouse, W. B., Eds. (1999a), *Handbook of Systems Engineering and Management,* John Wiley & Sons, New York.

Sage, A. P., and Rouse, W. B. (1999b), "Information System Frontiers in Knowledge Management," *Information System Frontiers,* Vol. 1, No. 3, pp. 195–204.

Schneiderman, B. (1987), *Designing the User Interface: Strategies for Effective Human Computer Interaction,* Addison-Wesley, Reading, MA.

Schum, D. A. (1987), *Evidence and Inference for the Intelligent Analyst,* 2 Vols., University Press of America, Lanham, MD.

Schum, D. A. (1994), *Evidential Foundations of Probabilistic Reasoning,* John Wiley & Sons, New York.

Shapiro, C., and Varian, H. R. (1999), *Information Rules: A Strategic Guide to the Network Economy,* Harvard Business School Press, Boston.

Shapiro, D. et al., Eds. (1996), *The Design of Computer Supported Cooperative Work and Groupware Systems,* North Holland, Amsterdam.

Sheridan, T. B. (1992), *Telerobotics, Automation, and Human Supervisory Control,* MIT Press, Cambridge, MA.

Simon, H. A. (1960), *The New Science of Management Decisions,* Harper, New York.

Skyrme, D. J. (1999), *Knowledge Networking: Creating the Collaborative Enterprise,* Butterworth-Heinemann, Boston.

Smith, D. E., Ed. (1999), *Knowledge, Groupware, and the Internet,* Butterworth-Heinemann, Boston.

Smith, S. L., and Mosier, J. N. (1986), "Guidelines for Designing User Interface Software," MITRE Corporation Technical Report MTR-10090, ESD-TR-86-278, Bedford, MA.

Somerville, M. A., and Rapport, D., Eds. (1999), *Transdisciplinarity: Re-creating Integrated Knowledge,* EOLSS, Oxford.

Sprague, R. H., Jr., and Carlson, E. D. (1982), *Building Effective Decision Support Systems,* Prentice Hall, Englewood Cliffs, NJ.

Sprague, R. H., Jr., and Watson, H. J. (1995), *Decision Support for Management,* Prentice Hall, Englewood Cliffs, NJ.

Stacey, R. D. (1996), *Complexity and Creativity in Organizations,* Berrett-Koehler, San Francisco.

Starbuck, W. E. (1985), "Acting First and Thinking Later," in *Organizational Strategy and Change,* Jossey-Bass, San Francisco.

Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. (1987), "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings," *Communications of the ACM,* Vol. 30, No. 1, pp. 32–47.

Stewart, T. A. (1997). *Intellectual Capital: The New Wealth of Organizations,* Currency Doubleday, New York.

Sullo, G. C. (1994), *Object Oriented Engineering: Designing Large-Scale Object-Oriented Systems,* John Wiley & Sons, New York.

Sveiby, K. E. (1997), *The New Organizational Wealth: Managing and Measuring Knowledge Based Assets,* Berrett-Koehler, San Francisco.

Tobin, D. R. (1998), *The Knowledge Enabled Organization: Moving from Training to Learning to Meet Business Goals,* AMACOM, New York.

Toulmin, S., Rieke, R., and Janik, A. (1979), *An Introduction to Reasoning,* Macmillan, New York.

Ulrich, D. (1998), "Intellectual Capital = Competence × Commitment," *Sloan Management Review,* Vol. 39, No. 2, pp. 15–26.

Utterback, J. M. (1994), *Mastering the Dynamics of Innovation: How Companies Can Seize Opportunities in the Face of Technological Change,* Harvard Business School Press, 1994.

Volkema, R. J. (1990), "Problem Formulation, in *Concise Encyclopedia of Information Processing in Systems and Organizations,* A. P. Sage, Ed., Pergamon Press, Oxford.

Watson, R. T. (1998), *Database Management: An Organizational Perspective,* John Wiley & Sons, New York.

Watson, R. T., DeSanctis, G., and Poole, M. S. (1988), "Using a GDSS to Facilitate Group Consensus: Some Intended and Unintended Consequences," *MIS Quarterly,* Vol. 12, pp. 463–477.

Weick, K. E. (1979), *The Social Psychology of Organizing,* Addison-Wesley, Reading, MA.

Weick, K. E. (1985), "Cosmos vs. Chaos: Sense and Nonsense in Electronic Context," *Organizational Dynamics,* Vol. 14, pp. 50–64.

Welch, D. A. (1989), "Group Decision Making Reconsidered," *Journal of Conflict Resolution,* Vol. 33, No. 3, pp. 430–445.

Will, H. J. (1975), "Model Management System," in *Information Systems and Organizational Structure,* E. Grochia and N. Szyperski, Eds., de Gruyter, Berlin, pp. 468–482.

Winograd, T., and Flores, F. (1986), *Understanding Computers and Cognition,* Ablex Press, Los Altos, CA.

Zack, M. H., Ed. (1999), *Knowledge and Strategy,* Butterworth-Heinemann, Boston.

Zeigler, B. P. (1997), *Objects and Systems,* Springer, New York.